Centurion UNIVERSITY
Shaping Lives... Empowering Communities...

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 Experiement : Blockchain in Supply Chains – Use Case Analysis

* Coding Phase: Pseudo Code / Flow Chart / Algorithm

ALGORITHM:

```
1.Start
2.Identify all participants in the supply chain (Producer, Transporter, Retailer,
Customer).
3.Assign each participant a unique blockchain address.
4.For every new product batch:
        Generate a unique Product ID.
        Record details (name, batch no., source, timestamp).
        Store transaction on blockchain.
5.During transport:
        Update location data and handling status on blockchain.
6.At retailer:
        Verify authenticity by checking the blockchain ledger.
7.Customer scans QR code to view product origin and journey details.
8.Smart contract triggers payment once delivery is confirmed.
9.Stop
```

* Software used

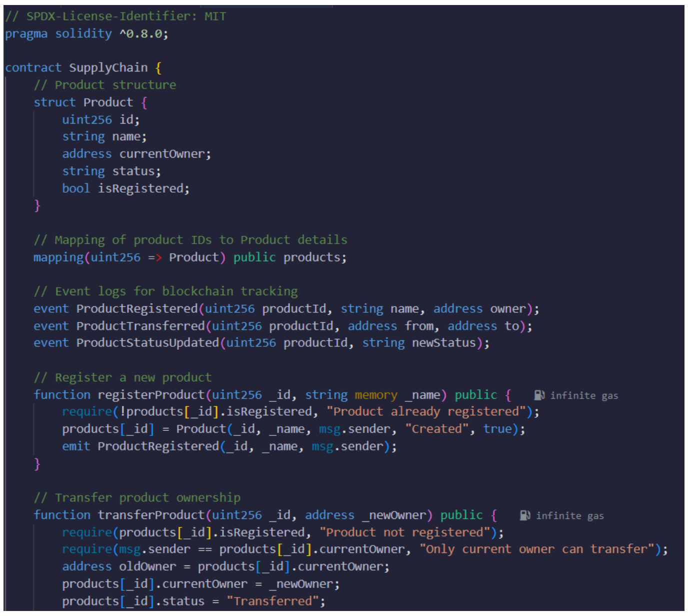
1.Remix IDE
2.MetaMask Wallet
3.Solidity Smart Contract
4.Ethereum Test Network (Sepolia)

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

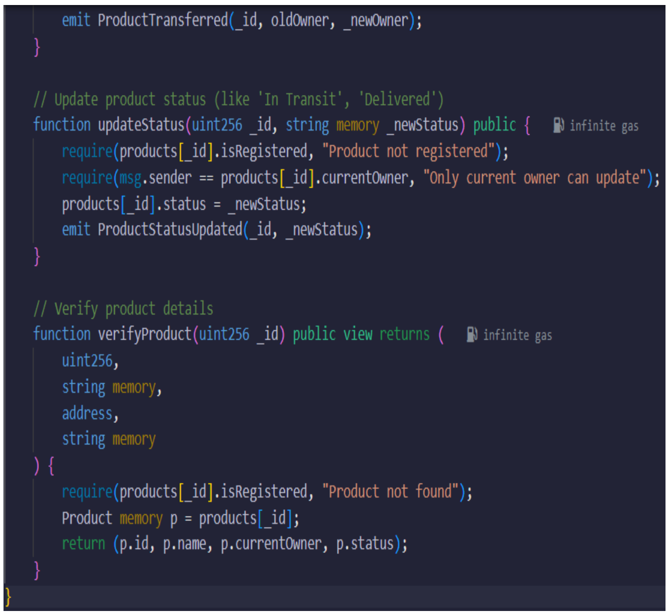
* Testing Phase: Compilation of Code (error detection)

1. Smart Contract Creation:

Open Remix IDE → create a new Solidity file SupplyChain.sol.
Write contract to register products, track transfer, and verify authenticity.



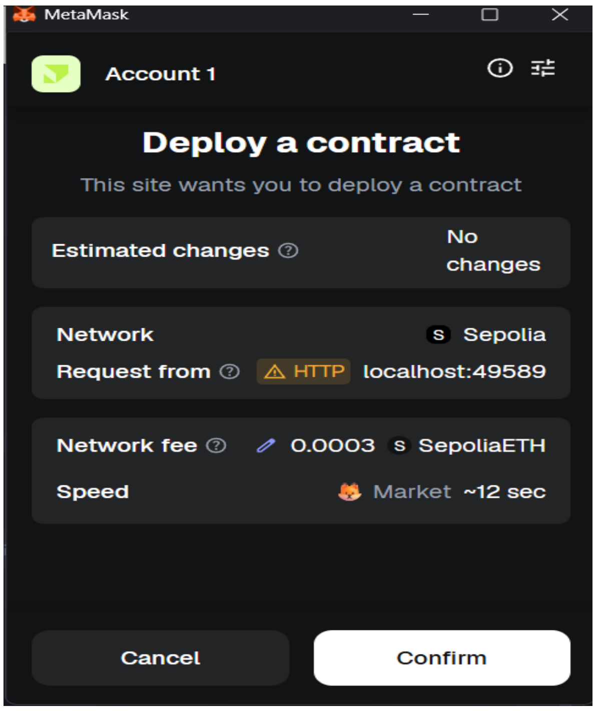
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract SupplyChain {
    // Product structure
    struct Product {
        uint256 id;
        string name;
        address currentOwner;
        string status;
        bool isRegistered;
    }

    // Mapping of product IDs to Product details
    mapping(uint256 => Product) public products;

    // Event logs for blockchain tracking
    event ProductRegistered(uint256 productId, string name, address owner);
    event ProductTransferred(uint256 productId, address from, address to);
    event ProductStatusUpdated(uint256 productId, string newStatus);

    // Register a new product
    function registerProduct(uint256 _id, string memory _name) public {    infinite gas
        require(!products[_id].isRegistered, "Product already registered");
        products[_id] = Product(_id, _name, msg.sender, "Created", true);
        emit ProductRegistered(_id, _name, msg.sender);
    }

    // Transfer product ownership
    function transferProduct(uint256 _id, address _newOwner) public {    infinite gas
        require(products[_id].isRegistered, "Product not registered");
        require(msg.sender == products[_id].currentOwner, "Only current owner can transfer");
        address oldOwner = products[_id].currentOwner;
        products[_id].currentOwner = _newOwner;
        products[_id].status = "Transferred";
```



```
        emit ProductTransferred(_id, oldOwner, _newOwner);
    }

    // Update product status (like 'In Transit', 'Delivered')
    function updateStatus(uint256 _id, string memory _newStatus) public {    infinite gas
        require(products[_id].isRegistered, "Product not registered");
        require(msg.sender == products[_id].currentOwner, "Only current owner can update");
        products[_id].status = _newStatus;
        emit ProductStatusUpdated(_id, _newStatus);
    }

    // Verify product details
    function verifyProduct(uint256 _id) public view returns (    infinite gas
        uint256,
        string memory,
        address,
        string memory
    ) {
        require(products[_id].isRegistered, "Product not found");
        Product memory p = products[_id];
        return (p.id, p.name, p.currentOwner, p.status);
    }
}
```

2.Deployment:

Deploy contract using the “Deploy & Run Transactions” tab in Remix.
Connect MetaMask to Sepolia Testnet for fake ETH gas usage.



MetaMask

Account 1

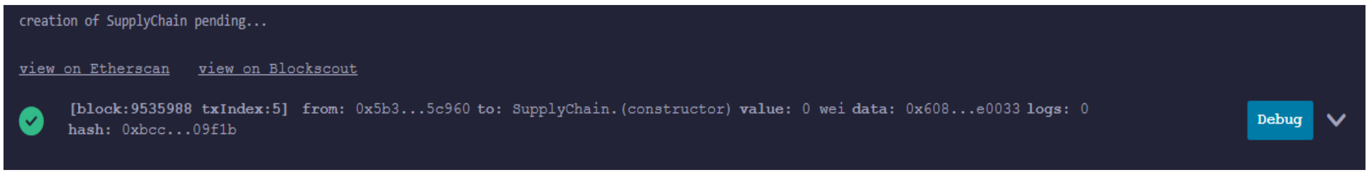
Deploy a contract

This site wants you to deploy a contract

Estimated changes: No changes

Network: Sepolia
Request from: HTTP localhost:49589

Network fee: 0.0003 SepoliaETH
Speed: Market ~12 sec



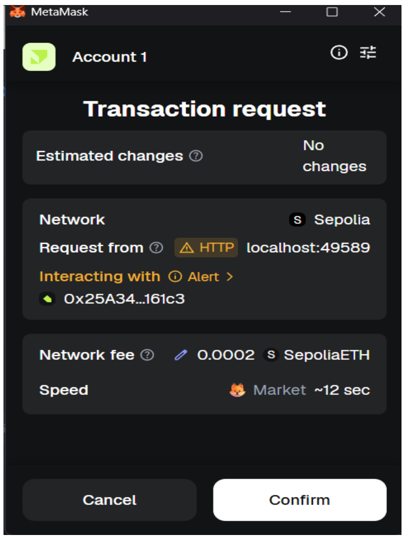
```
creation of SupplyChain pending...

view on Etherscan   view on Blockscout

[block:9535988 txIndex:5] from: 0x5b3...5c960 to: SupplyChain.(constructor) value: 0 wei data: 0x608...e0033 logs: 0
hash: 0xbcc...09f1b
```

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

* Implementation Phase: Final Output (no error)

Testing Transactions:

Call functions such as registerProduct(), transferProduct(), and verifyProduct()
Observe transaction hash and confirmation on blockchain.



MetaMask

Account 1

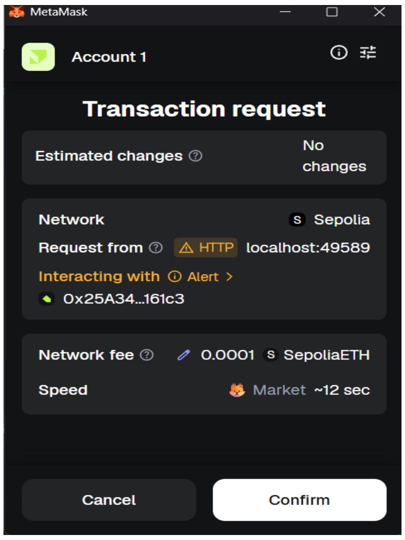
Transaction request

Estimated changes — No changes

Network — Sepolia
Request from — HTTP localhost:49589
Interacting with — Alert
0x25A34...161c3

Network fee — 0.0002 SepoliaETH
Speed — Market ~12 sec

Cancel | Confirm



MetaMask

Account 1

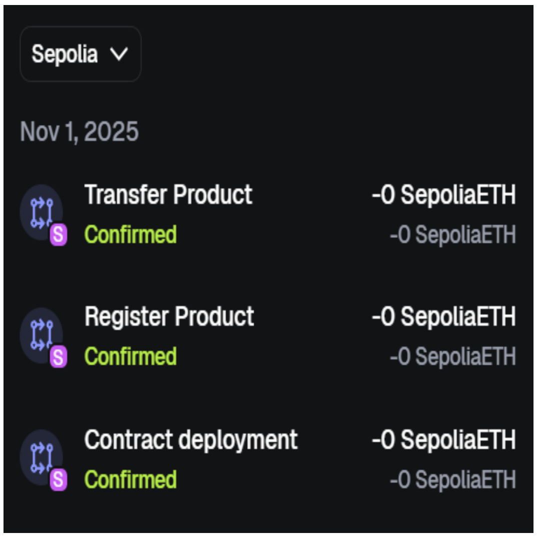
Transaction request

Estimated changes — No changes

Network — Sepolia
Request from — HTTP localhost:49589
Interacting with — Alert
0x25A34...161c3

Network fee — 0.0001 SepoliaETH
Speed — Market ~12 sec

Cancel | Confirm



Sepolia

Nov 1, 2025

Transfer Product — Confirmed — -0 SepoliaETH — -0 SepoliaETH

Register Product — Confirmed — -0 SepoliaETH — -0 SepoliaETH

Contract deployment — Confirmed — -0 SepoliaETH — -0 SepoliaETH

* Observations

1.Blockchain provides transparent and tamper-proof product tracking throughout the supply chain.

2.Smart contracts automate verification and payment processes efficiently.

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.