Anti-Counterfeit Medicine Tracking System

Leveraging Blockchain for Pharmaceutical Supply Chain Integrity

Name: Gautam Rai

Roll No: 53

Class: D17C

Batch: B

Chapter 1 / Problem & Solution

The Problem: Counterfeit Drugs

- ❖ **Counterfeit drugs pose a major global health risk** due to unverified, ineffective, or harmful ingredients.
- ❖ Manufacturers face **substantial financial losses** and damage to consumer trust.
- ❖ Existing solutions, such as **holograms** and **barcodes**, have failed to fully verify authenticity or address these challenges.
- ❖ **Current tracking systems and supply chain management** remain fragmented and lack transparency, making them difficult to secure.

The Solution: Blockchain Traceability

We propose a decentralized, immutable ledger system to track every drug batch.

Registration

Manufacturers anchor drug batches on the chain.

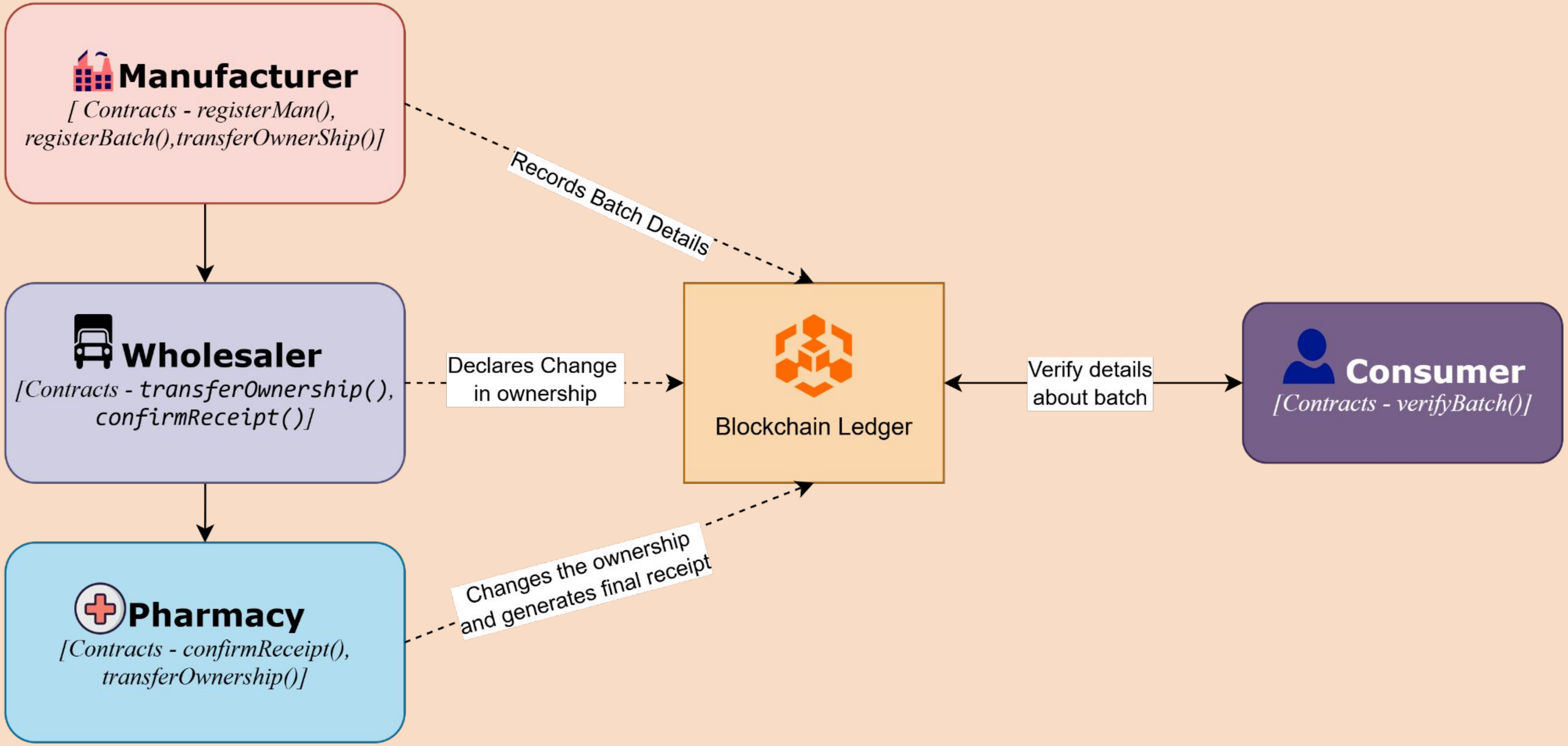
Transfer Confirmation

Wholesalers and pharmacies confirm secure handoffs.

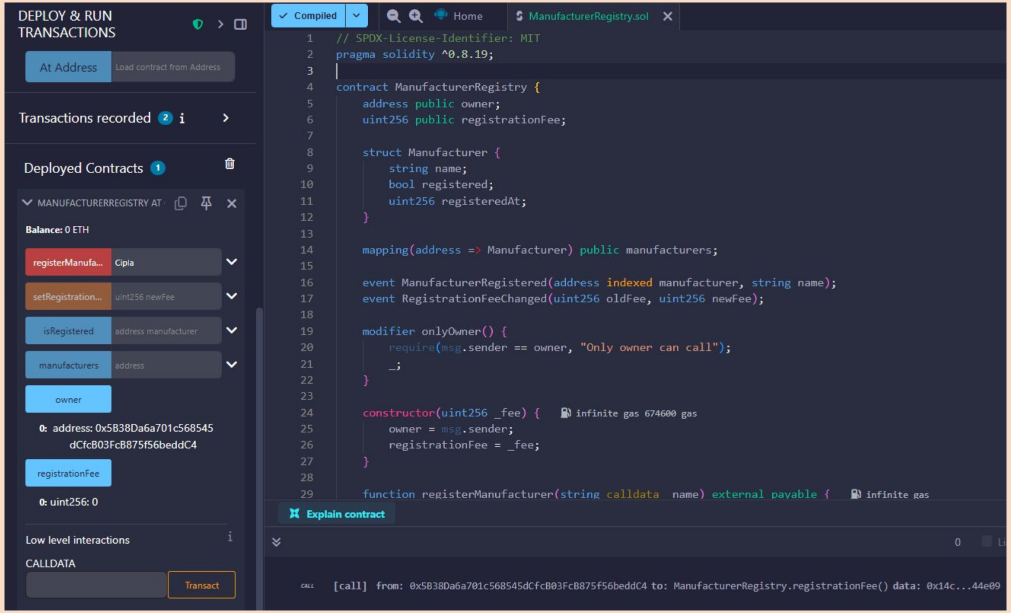
Verification

Consumers scan and verify authenticity and provenance.

Chapter 2 / Stakeholders and Contracts


Manufacturer
[Contracts - registerMan(), registerBatch(),transferOwnerShip()]
Records Batch Details
Wholesaler
[Contracts - transferOwnership(), confirmReceipt()]
Declares Change in ownership
Blockchain Ledger
Verify details about batch
Consumer
[Contracts - verifyBatch()]
Pharmacy
[Contracts - confirmReceipt(), transferOwnership()]
Changes the ownership and generates final receipt

Chapter 3 / Smart Contracts [Exp 6]



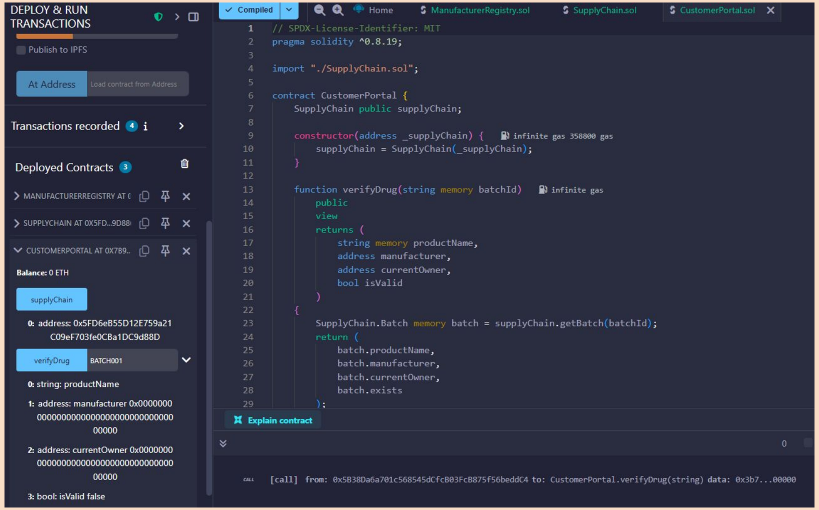
```
Balance: 0 ETH
registerManufa...  Cipla
setRegistration... uint256 newFee
isRegistered       address manufacturer
manufacturers      address
owner
0: address: 0x5B38Da6a701c568545
   dCfcB03FcB875f56beddC4
registrationFee
0: uint256: 0
```

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

contract ManufacturerRegistry {
    address public owner;
    uint256 public registrationFee;

    struct Manufacturer {
        string name;
        bool registered;
        uint256 registeredAt;
    }

    mapping(address => Manufacturer) public manufacturers;

    event ManufacturerRegistered(address indexed manufacturer, string name);
    event RegistrationFeeChanged(uint256 oldFee, uint256 newFee);

    modifier onlyOwner() {
        require(msg.sender == owner, "Only owner can call");
        _;
    }

    constructor(uint256 _fee) {     infinite gas 674600 gas
        owner = msg.sender;
        registrationFee = _fee;
    }

    function registerManufacturer(string calldata _name) external payable {     infinite gas
```

```
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: ManufacturerRegistry.registrationFee() data: 0x14c...44e09
```

ManufacturerRegistery.sol

CustomerPortal.sol

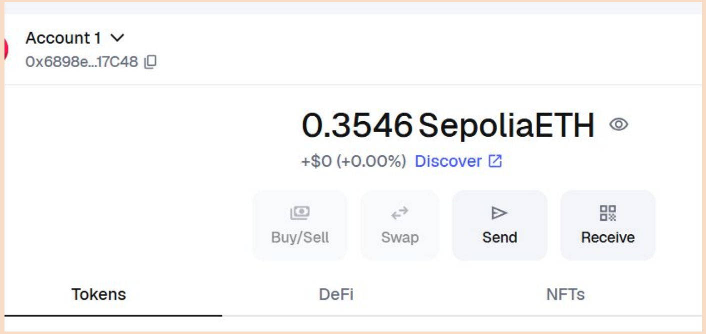


```
Balance: 0 ETH
supplyChain
0: address: 0x5FD6eB55D12E759a21
   C09eF703fe0CBa1DC9d88D
verifyDrug  BATCH001
0: string: productName
1: address: manufacturer 0x0000000
   000000000000000000000000000
   00000
2: address: currentOwner 0x0000000
   000000000000000000000000000
   00000
3: bool: isValid false
```

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

import "./SupplyChain.sol";

contract CustomerPortal {
    SupplyChain public supplyChain;

    constructor(address _supplyChain) {     infinite gas 358800 gas
        supplyChain = SupplyChain(_supplyChain);
    }

    function verifyDrug(string memory batchId)     infinite gas
        public
        view
        returns (
            string memory productName,
            address manufacturer,
            address currentOwner,
            bool isValid
        )
    {
        SupplyChain.Batch memory batch = supplyChain.getBatch(batchId);
        return (
            batch.productName,
            batch.manufacturer,
            batch.currentOwner,
            batch.exists
        );
```

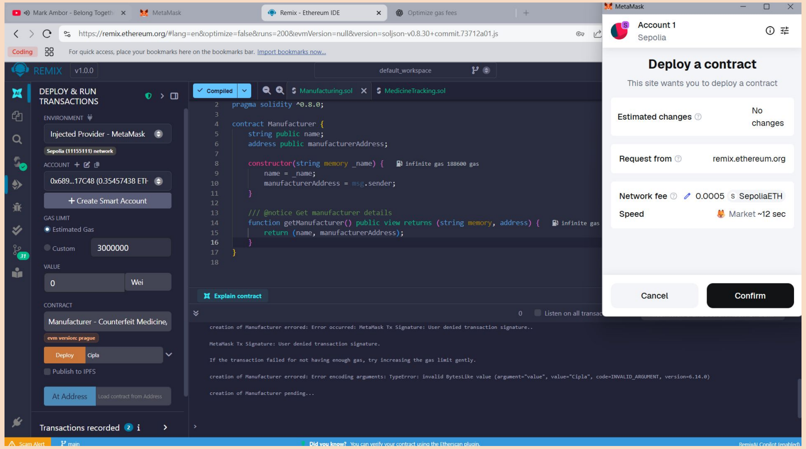
```
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: CustomerPortal.verifyDrug(string) data: 0x3b7...00000
```

Chapter 4 / Using Sepolia TestNet [Exp 7]



Account 1
0x6898e...17C48

0.3546 SepoliaETH
+$0 (+0.00%) Discover



https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.30+commit.73712a01.js

For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Injected Provider - MetaMask
Sepolia (11155111) network
ACCOUNT: 0x689...17C48 (0.35457438 ETH
GAS LIMIT: Estimated Gas
Custom 3000000
VALUE: 0 Wei
CONTRACT: Manufacturer - Counterfeit Medicine,
evm version: prague
Deploy Cipla
Publish to IPFS

```
pragma solidity ^0.8.0;

contract Manufacturer {
    string public name;
    address public manufacturerAddress;

    constructor(string memory _name) {    infinite gas 188600 gas
        name = _name;
        manufacturerAddress = msg.sender;
    }

    /// @notice Get manufacturer details
    function getManufacturer() public view returns (string memory, address) {    infinite gas
        return (name, manufacturerAddress);
    }
}
```

```
creation of Manufacturer errored: Error occurred: MetaMask Tx Signature: User denied transaction signature..

MetaMask Tx Signature: User denied transaction signature.

If the transaction failed for not having enough gas, try increasing the gas limit gently.

creation of Manufacturer errored: Error encoding arguments: TypeError: invalid BytesLike value (argument="value", value="Cipla", code=INVALID_ARGUMENT, version=6.14.0)

creation of Manufacturer pending...
```

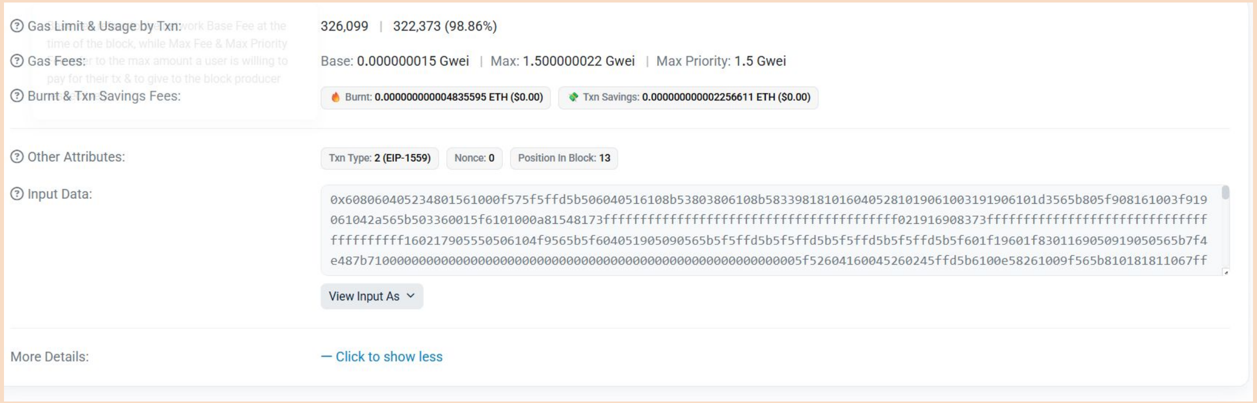
Did you know? You can verify your contract using the Etherscan plugin.

MetaMask
Account 1
Sepolia

Deploy a contract

This site wants you to deploy a contract

Estimated changes	No changes
Request from	remix.ethereum.org
Network fee	0.0005 SepoliaETH
Speed	Market ~12 sec



Gas Limit & Usage by Txn:	326,099 | 322,373 (98.86%)
Gas Fees:	Base: 0.000000015 Gwei | Max: 1.500000022 Gwei | Max Priority: 1.5 Gwei
Burnt & Txn Savings Fees:	Burnt: 0.000000000004835595 ETH ($0.00) Txn Savings: 0.000000000002256611 ETH ($0.00)
Other Attributes:	Txn Type: 2 (EIP-1559) Nonce: 0 Position In Block: 13
Input Data:	see below

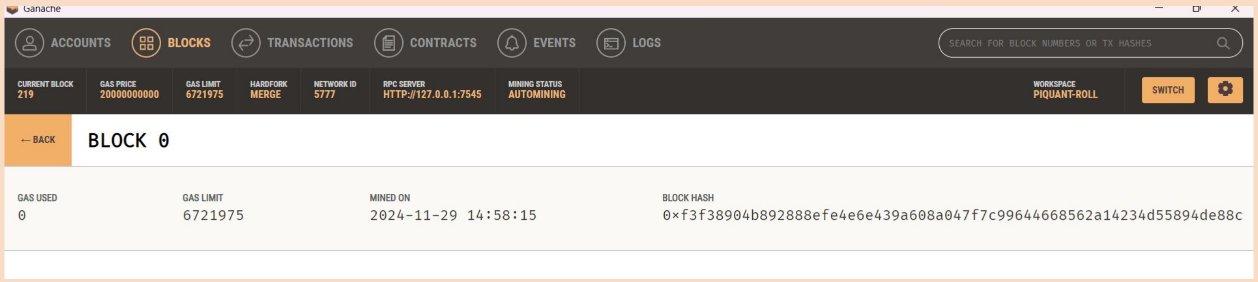
work Base Fee at the time of the block, while Max Fee & Max Priority r to the max amount a user is willing to pay for their tx & to give to the block producer

```
0x608060405234801561000f575f5ffd5b506040516108b53803806108b5833981810160405281019061003191906101d3565b805f908161003f919
061042a565b503360015f6101000a81548173ffffffffffffffffffffffffffffffffffffffff021916908373fffffffffffffffffffffffffffffff
fffffffff160217905550506104f9565b5f604051905090565b5f5ffd5b5f5ffd5b5f5ffd5b5f5ffd5b5f601f19601f8301169050919050565b7f4
e487b71000000000000000000000000000000000000000000000000000000005f52604160045260245ffd5b6100e58261009f565b810181811067ff
```

View Input As

More Details: — Click to show less

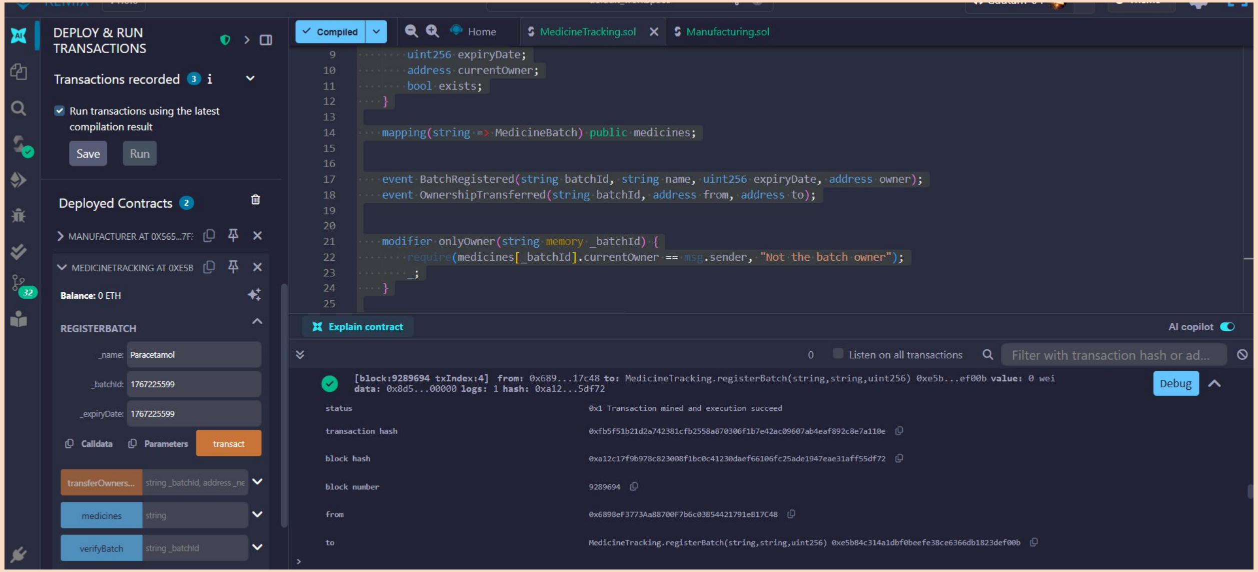
Chapter 5 / Using Ganache [Exp 8]



CURRENT BLOCK 219 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MERGE | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE PIQUANT-ROLL

BLOCK 0

GAS USED	GAS LIMIT	MINED ON	BLOCK HASH
0	6721975	2024-11-29 14:58:15	0×f3f38904b892888efe4e6e439a608a047f7c99644668562a14234d55894de88c



Balance: 0 ETH

_name:	Paracetamol
_batchId:	1767225599
_expiryDate:	1767225599
transferOwners...	string _batchId, address _ne
medicines	string
verifyBatch	string _batchId

```
        uint256 expiryDate;
        address currentOwner;
        bool exists;
    }

    mapping(string => MedicineBatch) public medicines;


    event BatchRegistered(string batchId, string name, uint256 expiryDate, address owner);
    event OwnershipTransferred(string batchId, address from, address to);


    modifier onlyOwner(string memory _batchId) {
        require(medicines[_batchId].currentOwner == msg.sender, "Not the batch owner");
        _;
    }

```

```
[block:9289694 txIndex:4] from: 0x689...17c48 to: MedicineTracking.registerBatch(string,string,uint256) 0xe5b...ef00b value: 0 wei
data: 0x8d5...00000 logs: 1 hash: 0xa12...5df72
status                  0x1 Transaction mined and execution succeed
transaction hash        0xfb5f51b21d2a742381cfb2558a870306f1b7e42ac09607ab4eaf892c8e7a110e
block hash              0xa12c17f9b978c823008f1bc0c41230daef66106fc25ade1947eae31aff55df72
block number            9289694
from                    0x6898eF3773Aa88700F7b6c03B54421791eB17C48
to                      MedicineTracking.registerBatch(string,string,uint256) 0xe5b84c314a1dbf0beefe38ce6366db1823def00b
```