

EX.NO: 3

Ecommerce System for Remote Purchasing

Date: 12/07/2023

Aim:

To implement the Ecommerce System for the remote purchasing system by using Solidity Smart control

Procedure:

Step1: Open the remix IDE

Step2: Choose the solidity compiler version

Step3: Allow the owner to add the products specify their name, description, price and initial quantity

Step4: Declare the product information such as product id, name, price, quantity, description and product is available or not

Step5: Buyers can purchase the products by providing the product ID and the desired quantity

Step6: Call the withdrawfunds() function to withdraw the contract's balance

Step7: If the product is not available then return "Product Not Available!!!"

Step8: If any insufficient payment is done then return "Insufficient Payment" message

Step9: If the buyer has overpaid then refund the excess amount to the buyer

Step10: After processing all the steps then stop the transaction

Program:

```
pragma solidity ^0.8.0;
contract Ecommerce {
    address public owner;
    uint256 public productCount;
    struct Product {
        uint256 id;
        string name;
        string description;
        uint256 price;
        uint256 quantity;
```

```

        bool available;
    }
    mapping(uint256 => Product) public products;
    event ProductAdded(uint256 id, string name, uint256 price, uint256 quantity);
    event ProductPurchased(uint256 id, string name, uint256 quantity, uint256 totalPrice);
    constructor() {
        owner = msg.sender;
        productCount = 0;
    }
    modifier onlyOwner() {
        require(msg.sender == owner, "Only the owner can perform this action");
        _;
    }
    function addProduct(
        string memory _name,
        string memory _description,
        uint256 _price,
        uint256 _quantity
    ) public onlyOwner {
        productCount++;
        products[productCount] = Product({
            id: productCount,
            name: _name,
            description: _description,
            price: _price,
            quantity: _quantity,
            available: true
        });
        emit ProductAdded(productCount, _name, _price, _quantity);
    }

```

```

function purchaseProduct(uint256 _productId, uint256 _quantity) public payable {
    require(_quantity > 0, "Quantity must be greater than zero");
    require(products[_productId].available, "Product is not available");
    require(products[_productId].quantity >= _quantity, "Not enough quantity available");
    require(msg.value >= products[_productId].price * _quantity, "Insufficient payment");
    products[_productId].quantity -= _quantity;

    if (msg.value > products[_productId].price * _quantity) {
        payable(msg.sender).transfer(msg.value - (products[_productId].price * _quantity));
    }

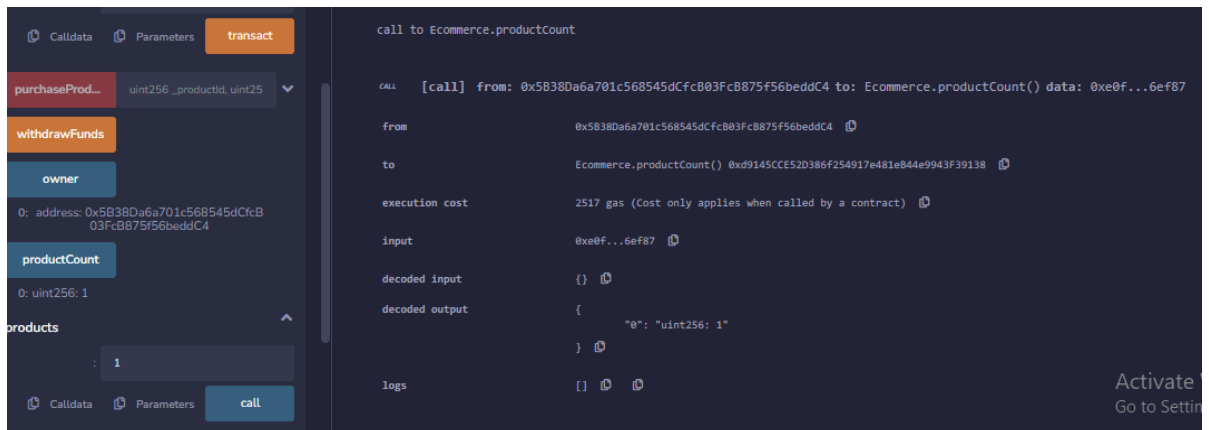
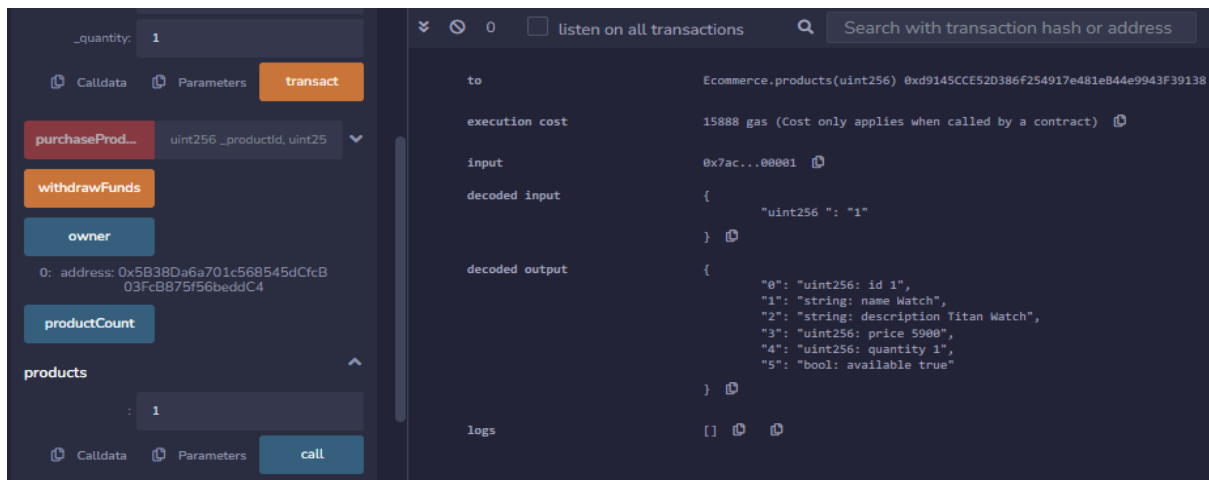
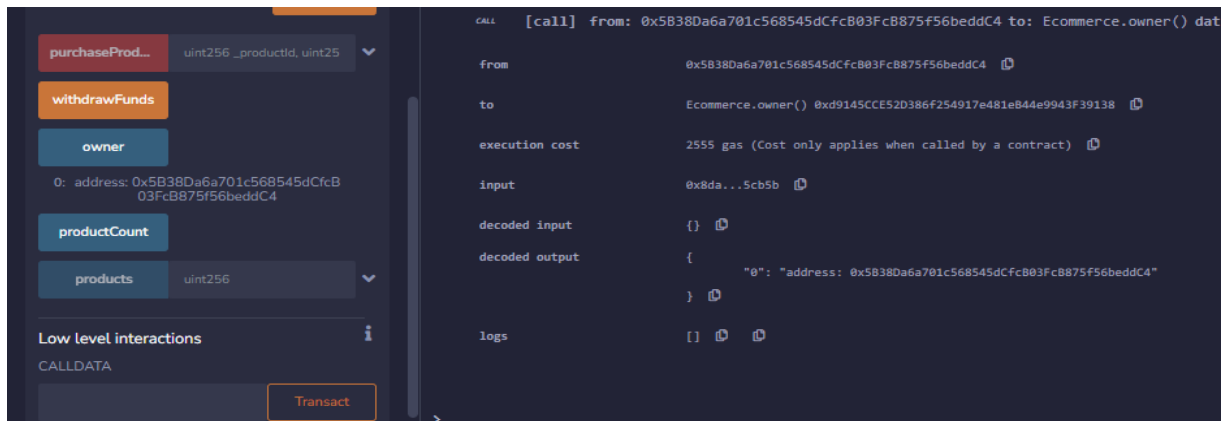
    emit ProductPurchased(_productId, products[_productId].name, _quantity,
        products[_productId].price * _quantity);
}

function withdrawFunds() public onlyOwner {
    payable(owner).transfer(address(this).balance);
}
}

```

Output:

The screenshot shows a web application interface for a smart contract. On the left, there is a sidebar with a form titled 'addProduct' containing input fields for '_name' (Watch), '_description' (Titan Watch), '_price' (5900), and '_quantity' (1). Below the form are buttons for 'transact', 'purchaseProd...' (with a dropdown menu showing 'uint256_productId, uint25'), 'withdrawFunds', 'owner', and 'productCount'. The main area on the right displays transaction details for block number 2. It includes fields for 'from' (0x58380a6a701c568545dcfc883fc8875f56beddc4), 'to' (Ecommerce.addProduct(string,string,uint256,uint256) 0xd9145cc520386f254917e481e844e9943f39138), 'gas' (214382), 'transaction cost' (186419 gas), 'execution cost' (164055 gas), 'input' (0x81c...00000), and 'decoded input' (a JSON object with 'string_name': 'Watch', 'string_description': 'Titan Watch', 'uint256_price': '5000', and 'uint256_quantity': '1'). The 'decoded output' is shown as an empty object {}.



Result:

Thus the Ecommerce system for remote purchasing has been implemented and executed out successfully.