Experiment No. 07

Semester	B.E. Semester VII – Computer Engineering
Subject	Blockchain Lab (CSDL7022)
Subject Professor In-charge	Prof. Swapnil S. Sonawane
Academic Year	2024-25

Student Name	Deep Salunkhe
Roll Number	21102A0014

Title: Tracking Supply Chain Transactions

Code:

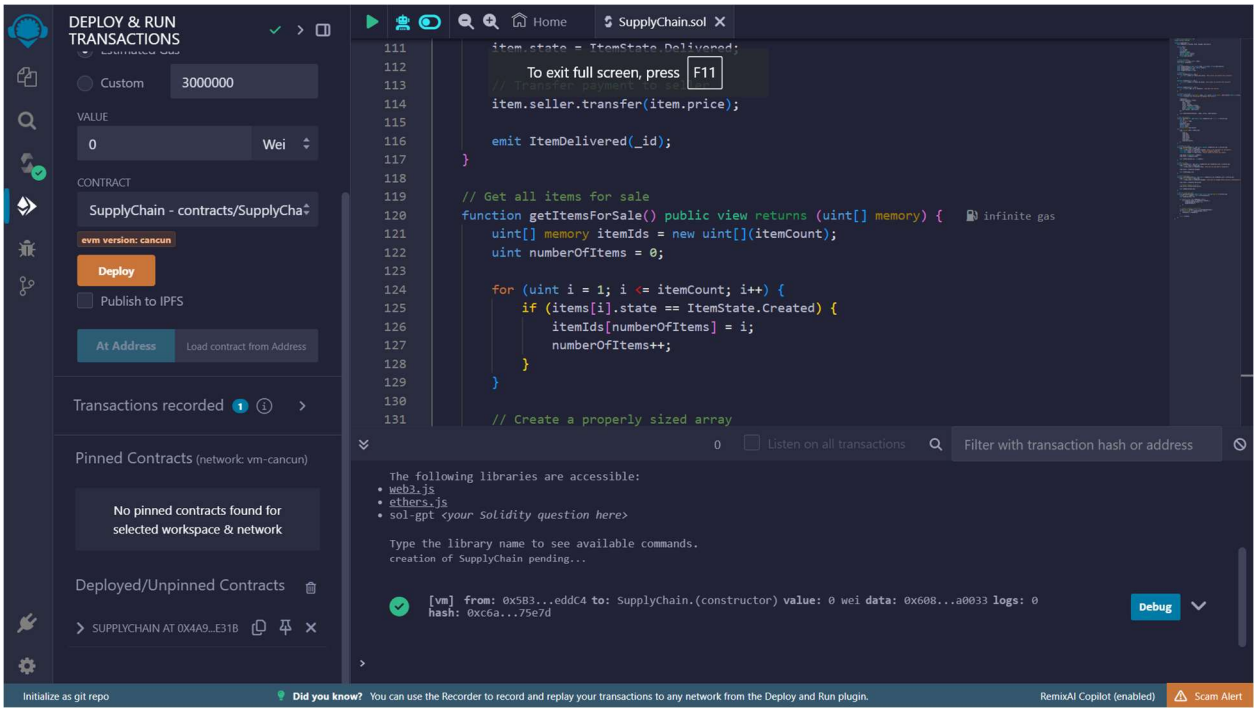
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract SupplyChain {
    enum ItemState { Created, Paid, Shipped, Delivered }

    struct Item {
        uint id;
        string name;
        uint price;
        ItemState state;
        address payable seller;
        address payable buyer;
        string description;
    }

    // State variables
    mapping(uint => Item) public items;
    uint public itemCount;

    // Events
    event ItemCreated(uint id, string name, uint price, string description);
```

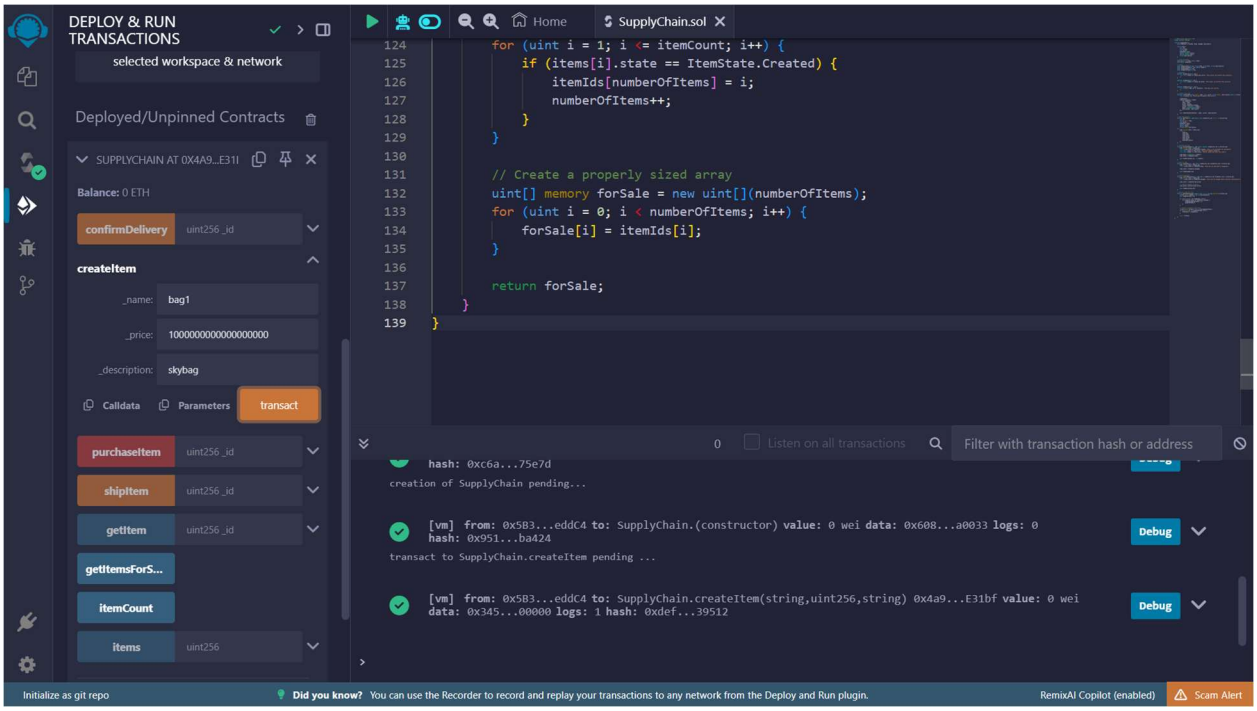
```
    event ItemPurchased(uint id, address buyer,uint price);
    event ItemShipped(uint id);
    event ItemDelivered(uint id);

    // Modifiers
    modifier onlySeller(uint _id) {
        require(msg.sender == items[_id].seller, "Only seller can perform this
action");
        _;
    }

    modifier onlyBuyer(uint _id) {
        require(msg.sender == items[_id].buyer, "Only buyer can perform this
action");
        _;
    }

    modifier itemExists(uint _id) {
        require(_id > 0 && _id <= itemCount, "Item does not exist");
        _;
    }

    // Create a new item
    function createItem(string memory _name, uint _price, string memory
_description) public {
        require(_price > 0, "Price must be greater than zero");

        itemCount++;
        items[itemCount] = Item({
            id: itemCount,
            name: _name,
            price: _price,
            state: ItemState.Created,
            seller: payable(msg.sender),
            buyer: payable(address(0)),
            description: _description
        });

        emit ItemCreated(itemCount, _name, _price, _description);
    }

    // Get item details
    function getItem(uint _id) public view itemExists(_id) returns (
        uint id,
        string memory name,
        uint price,
        ItemState state,
```

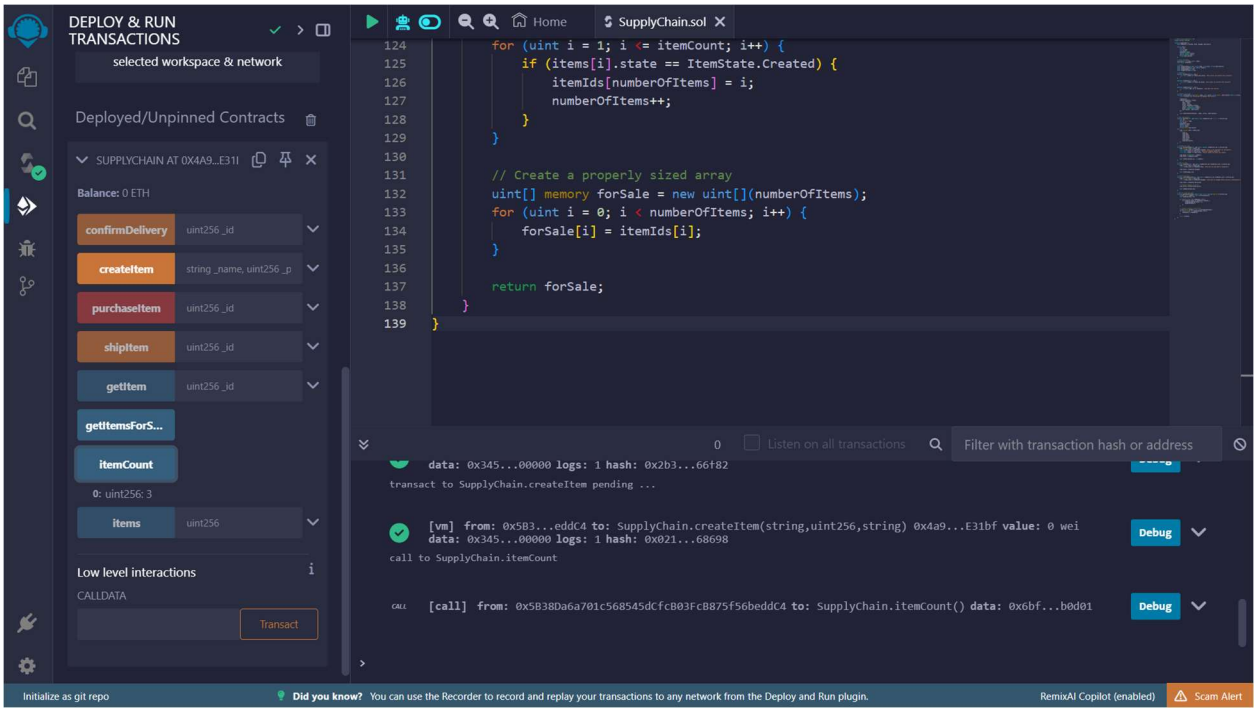
```
        address seller,
        address buyer,
        string memory description
    ) {
        Item storage item = items[_id];
        return (
            item.id,
            item.name,
            item.price,
            item.state,
            item.seller,
            item.buyer,
            item.description
        );
    }

    // Purchase an item
    function purchaseItem(uint _id,uint _price) public payable itemExists(_id) {
        Item storage item = items[_id];
        require(item.state == ItemState.Created, "Item is not available for
purchase");
        require(_price == item.price, "Incorrect payment amount");
        require(msg.sender != item.seller, "Seller cannot buy their own item");

        item.buyer = payable(msg.sender);
        item.state = ItemState.Paid;

        emit ItemPurchased(_id, msg.sender,_price);
    }

    // Ship an item
    function shipItem(uint _id) public itemExists(_id) onlySeller(_id) {
        Item storage item = items[_id];
        require(item.state == ItemState.Paid, "Item must be paid before
shipping");

        item.state = ItemState.Shipped;

        emit ItemShipped(_id);
    }

    // Confirm delivery
    function confirmDelivery(uint _id) public itemExists(_id) onlyBuyer(_id) {
        Item storage item = items[_id];
        require(item.state == ItemState.Shipped, "Item must be shipped before
delivery confirmation");
```

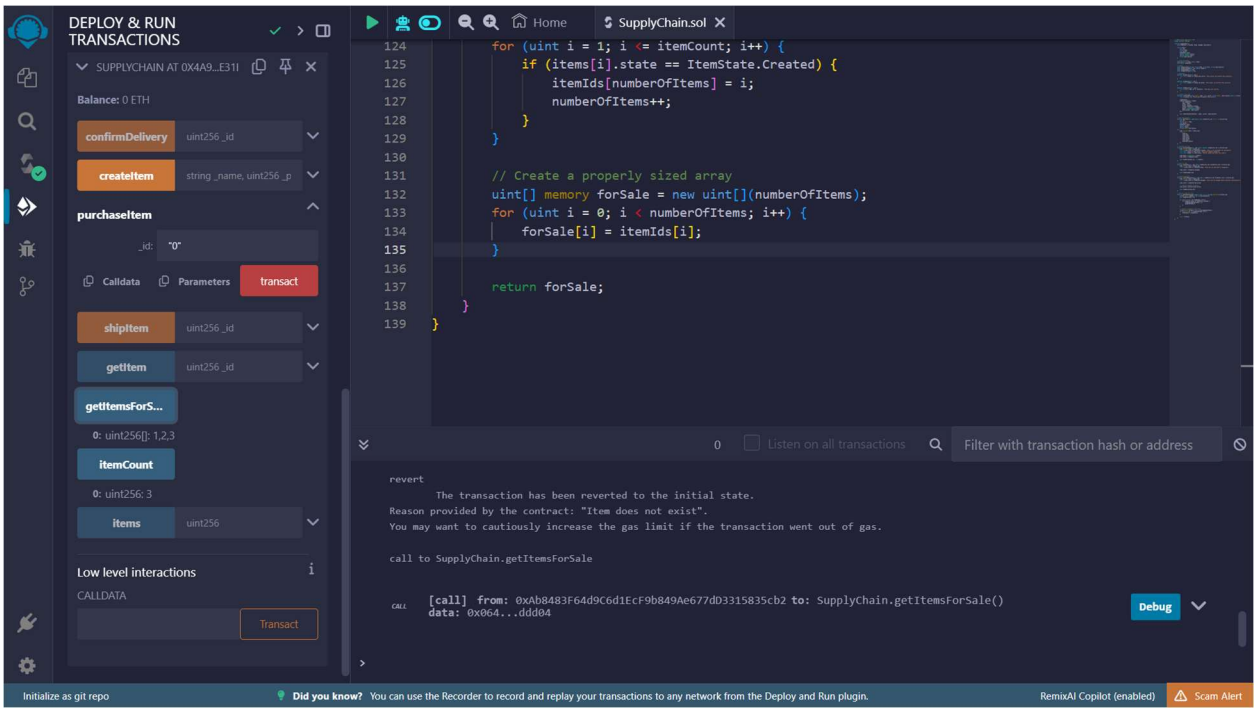
```
        item.state = ItemState.Delivered;

        // Transfer payment to seller
        item.seller.transfer(item.price);

        emit ItemDelivered(_id);
    }

    // Get all items for sale
    function getItemsForSale() public view returns (uint[] memory) {
        uint[] memory itemIds = new uint[](itemCount);
        uint numberOfItems = 0;

        for (uint i = 1; i <= itemCount; i++) {
            if (items[i].state == ItemState.Created) {
                itemIds[numberOfItems] = i;
                numberOfItems++;
            }
        }

        // Create a properly sized array
        uint[] memory forSale = new uint[](numberOfItems);
        for (uint i = 0; i < numberOfItems; i++) {
            forSale[i] = itemIds[i];
        }

        return forSale;
    }
}
```

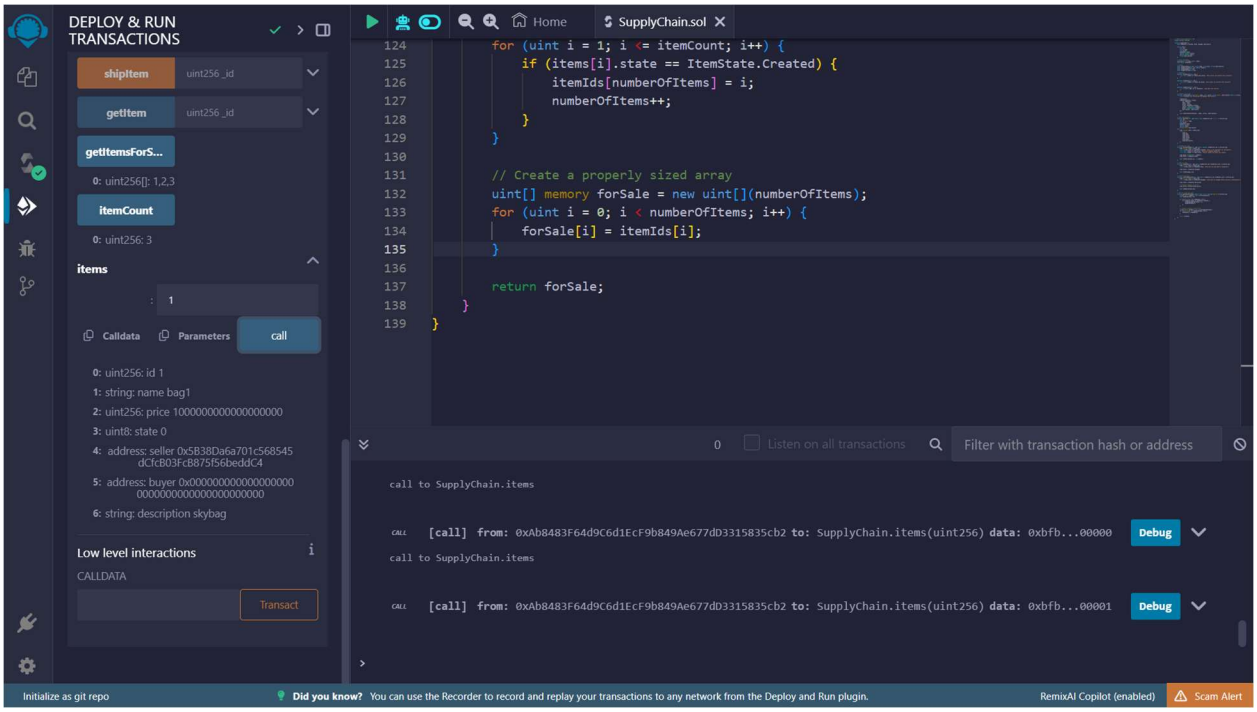
Output:

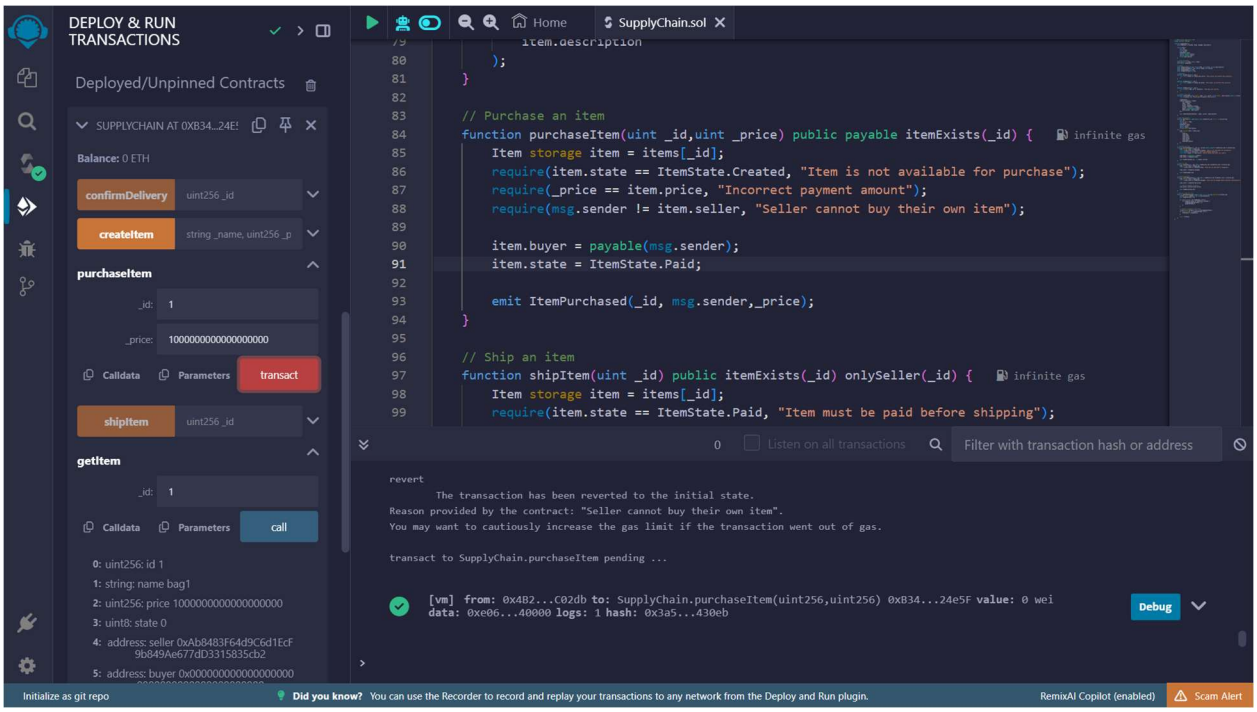

DEPLOY & RUN TRANSACTIONS
Custom
3000000
VALUE
0
Wei
CONTRACT
SupplyChain - contracts/SupplyCha
evm version: cancun
Deploy
Publish to IPFS
At Address
Load contract from Address
Transactions recorded
Pinned Contracts (network: vm-cancun)
No pinned contracts found for selected workspace & network
Deployed/Unpinned Contracts
SUPPLYCHAIN AT 0X4A9...E31B
Home
SupplyChain.sol
To exit full screen, press F11
item.seller.transfer(item.price);
emit ItemDelivered(_id);
// Get all items for sale
function getItemsForSale() public view returns (uint[] memory) {
infinite gas
uint[] memory itemIds = new uint[](itemCount);
uint numberOfItems = 0;
for (uint i = 1; i <= itemCount; i++) {
if (items[i].state == ItemState.Created) {
itemIds[numberOfItems] = i;
numberOfItems++;
}
}
// Create a properly sized array
Listen on all transactions
Filter with transaction hash or address
The following libraries are accessible:
web3.js
ethers.js
sol-gpt <your Solidity question here>
Type the library name to see available commands.
creation of SupplyChain pending...
[vm] from: 0x5B3...eddC4 to: SupplyChain.(constructor) value: 0 wei data: 0x608...a0033 logs: 0
hash: 0xc6a...75e7d
Debug
Initialize as git repo
Did you know? You can use the Recorder to record and replay your transactions to any network from the Deploy and Run plugin.
RemixAI Copilot (enabled)
Scam Alert

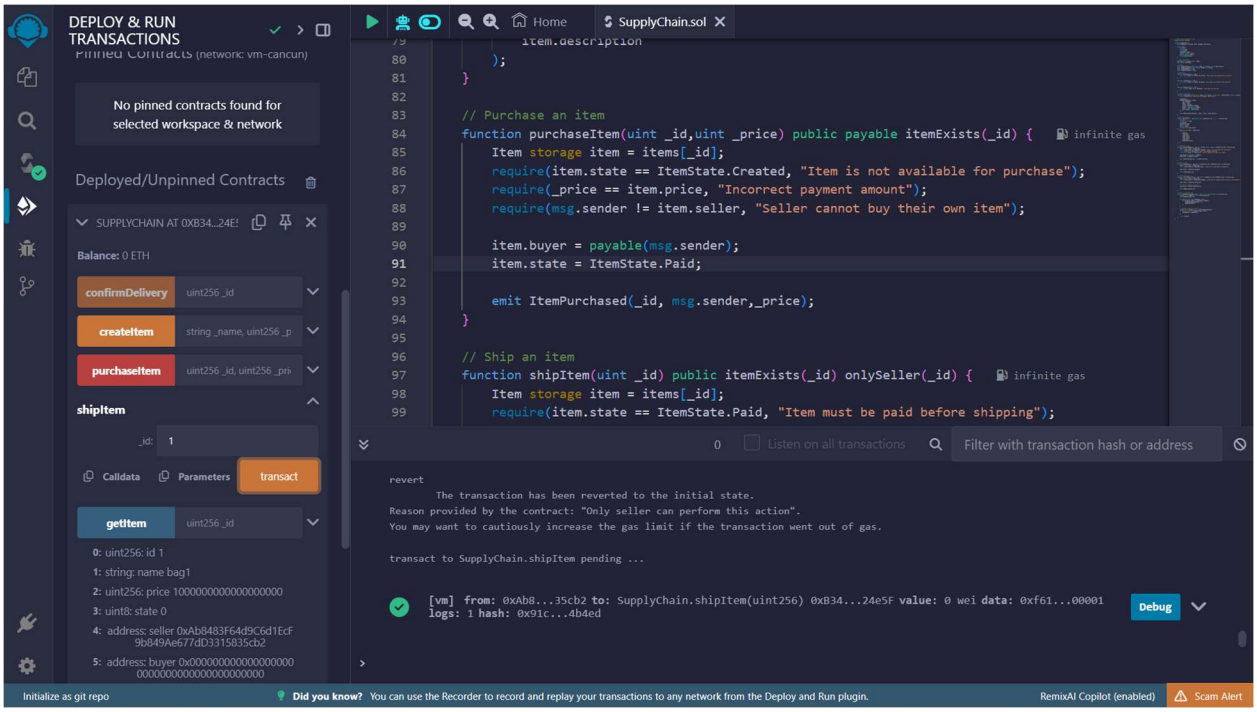

DEPLOY & RUN TRANSACTIONS
selected workspace & network
Deployed/Unpinned Contracts
SUPPLYCHAIN AT 0X4A9...E31I
Balance: 0 ETH
confirmDelivery
uint256 _id
createItem
_name: bag1
_price: 100000000000000000
_description: skybag
Calldata
Parameters
transact
purchaseItem
shipItem
getItem
getItemsForS...
itemCount
items
uint256
Home
SupplyChain.sol
for (uint i = 1; i <= itemCount; i++) {
if (items[i].state == ItemState.Created) {
itemIds[numberOfItems] = i;
numberOfItems++;
}
}
// Create a properly sized array
uint[] memory forSale = new uint[](numberOfItems);
for (uint i = 0; i < numberOfItems; i++) {
forSale[i] = itemIds[i];
}
return forSale;
}
}
Listen on all transactions
Filter with transaction hash or address
hash: 0xc6a...75e7d
creation of SupplyChain pending...
[vm] from: 0x5B3...eddC4 to: SupplyChain.(constructor) value: 0 wei data: 0x608...a0033 logs: 0
hash: 0x951...ba424
Debug
transact to SupplyChain.createItem pending ...
[vm] from: 0x5B3...eddC4 to: SupplyChain.createItem(string,uint256,string) 0x4a9...E31bf value: 0 wei
data: 0x345...00000 logs: 1 hash: 0xdef...39512
Debug
Initialize as git repo
Did you know? You can use the Recorder to record and replay your transactions to any network from the Deploy and Run plugin.
RemixAI Copilot (enabled)
Scam Alert

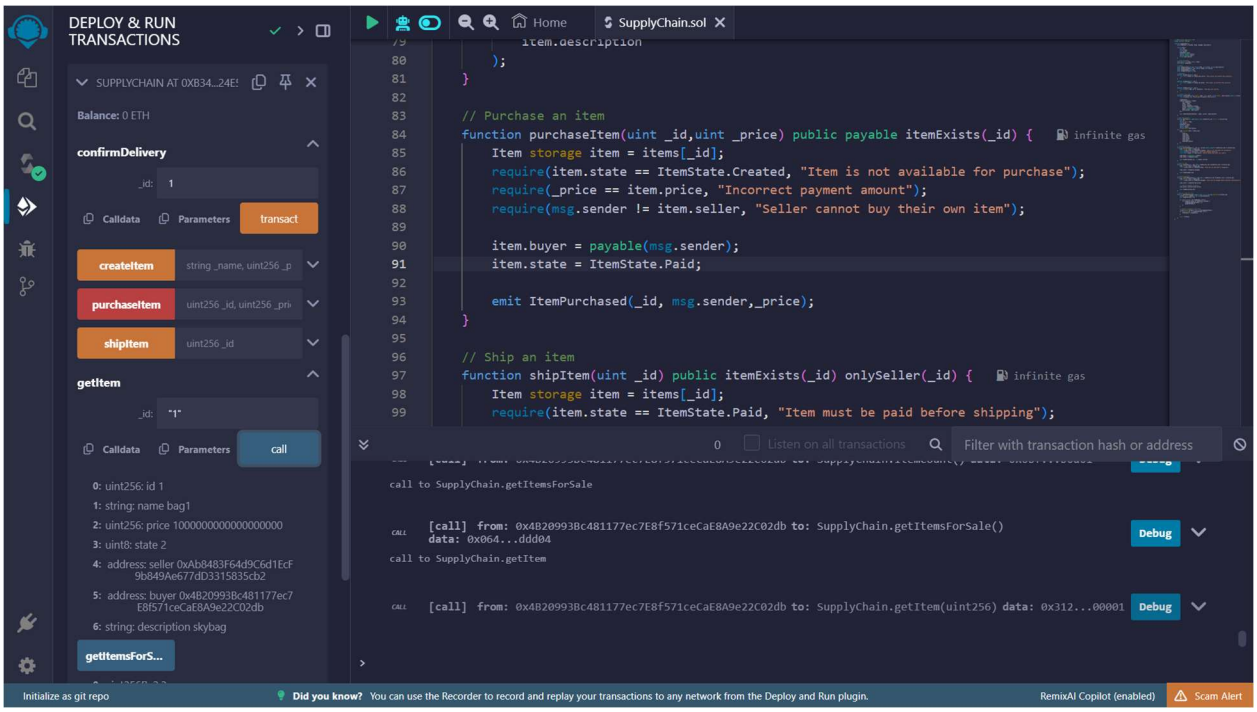

DEPLOY & RUN TRANSACTIONS
selected workspace & network
Deployed/Unpinned Contracts
SUPPLYCHAIN AT 0X4A9...E31I
Balance: 0 ETH
confirmDelivery uint256 _id
createItem string _name, uint256 _p
purchaseItem uint256 _id
shipItem uint256 _id
getItem uint256 _id
getItemsForS...
itemCount
0: uint256: 3
items uint256
Low level interactions
CALLDATA
Transact
Home
SupplyChain.sol
for (uint i = 1; i <= itemCount; i++) {
if (items[i].state == ItemState.Created) {
itemIds[numberOfItems] = i;
numberOfItems++;
}
}
// Create a properly sized array
uint[] memory forSale = new uint[](numberOfItems);
for (uint i = 0; i < numberOfItems; i++) {
forSale[i] = itemIds[i];
}
return forSale;
}
}
0
Listen on all transactions
Filter with transaction hash or address
data: 0x345...00000 logs: 1 hash: 0x2b3...66f82
transact to SupplyChain.createItem pending ...
[vm] from: 0x5B3...eddC4 to: SupplyChain.createItem(string,uint256,string) 0x4a9...E31bf value: 0 wei
data: 0x345...00000 logs: 1 hash: 0x021...68698
Debug
call to SupplyChain.itemCount
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: SupplyChain.itemCount() data: 0x6bf...b0d01
Debug
Initialize as git repo
Did you know? You can use the Recorder to record and replay your transactions to any network from the Deploy and Run plugin.
RemixAI Copilot (enabled)
Scam Alert


DEPLOY & RUN TRANSACTIONS
SUPPLYCHAIN AT 0X4A9...E31I
Balance: 0 ETH
confirmDelivery uint256 _id
createItem string _name, uint256 _p
purchaseItem
_id: "0"
Calldata
Parameters
transact
shipItem uint256 _id
getItem uint256 _id
getItemsForS...
0: uint256[]: 1,2,3
itemCount
0: uint256: 3
items uint256
Low level interactions
CALLDATA
Transact
Home
SupplyChain.sol
for (uint i = 1; i <= itemCount; i++) {
if (items[i].state == ItemState.Created) {
itemIds[numberOfItems] = i;
numberOfItems++;
}
}
// Create a properly sized array
uint[] memory forSale = new uint[](numberOfItems);
for (uint i = 0; i < numberOfItems; i++) {
forSale[i] = itemIds[i];
}
return forSale;
}
}
0
Listen on all transactions
Filter with transaction hash or address
revert
The transaction has been reverted to the initial state.
Reason provided by the contract: "Item does not exist".
You may want to cautiously increase the gas limit if the transaction went out of gas.
call to SupplyChain.getItemsForSale
[call] from: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 to: SupplyChain.getItemsForSale()
data: 0x064...ddd04
Debug
Initialize as git repo
Did you know? You can use the Recorder to record and replay your transactions to any network from the Deploy and Run plugin.
RemixAI Copilot (enabled)
Scam Alert


DEPLOY & RUN TRANSACTIONS
shipItem
uint256 _id
getItem
uint256 _id
getItemsForS...
0: uint256[]: 1,2,3
itemCount
0: uint256: 3
items
1
Calldata
Parameters
call
0: uint256: id 1
1: string: name bag1
2: uint256: price 1000000000000000000
3: uint8: state 0
4: address: seller 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
5: address: buyer 0x0000000000000000000000000000000000000000
6: string: description skybag
Low level interactions
CALLDATA
Transact
Home
SupplyChain.sol
for (uint i = 1; i <= itemCount; i++) {
if (items[i].state == ItemState.Created) {
itemIds[numberOfItems] = i;
numberOfItems++;
}
}
// Create a properly sized array
uint[] memory forSale = new uint[](numberOfItems);
for (uint i = 0; i < numberOfItems; i++) {
forSale[i] = itemIds[i];
}
return forSale;
}
}
Listen on all transactions
Filter with transaction hash or address
call to SupplyChain.items
[call] from: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 to: SupplyChain.items(uint256) data: 0xbfb...00000
Debug
call to SupplyChain.items
[call] from: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 to: SupplyChain.items(uint256) data: 0xbfb...00001
Debug
Initialize as git repo
Did you know? You can use the Recorder to record and replay your transactions to any network from the Deploy and Run plugin.
RemixAI Copilot (enabled)
Scam Alert


DEPLOY & RUN TRANSACTIONS
Deployed/Unpinned Contracts
SUPPLYCHAIN AT 0XB34...24E5
Balance: 0 ETH
confirmDelivery
uint256 _id
createItem
string _name, uint256 _p
purchaseItem
_id: 1
_price: 1000000000000000000
Calldata
Parameters
transact
shipItem
uint256 _id
getItem
_id: 1
Calldata
Parameters
call
0: uint256: id 1
1: string: name bag1
2: uint256: price 1000000000000000000
3: uint8: state 0
4: address: seller 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
5: address: buyer 0x0000000000000000000000000000000000000000
Home
SupplyChain.sol
item.description
);
}
// Purchase an item
function purchaseItem(uint _id,uint _price) public payable itemExists(_id) {
infinite gas
Item storage item = items[_id];
require(item.state == ItemState.Created, "Item is not available for purchase");
require(_price == item.price, "Incorrect payment amount");
require(msg.sender != item.seller, "Seller cannot buy their own item");
item.buyer = payable(msg.sender);
item.state = ItemState.Paid;
emit ItemPurchased(_id, msg.sender,_price);
}
// Ship an item
function shipItem(uint _id) public itemExists(_id) onlySeller(_id) {
infinite gas
Item storage item = items[_id];
require(item.state == ItemState.Paid, "Item must be paid before shipping");
Listen on all transactions
Filter with transaction hash or address
revert
The transaction has been reverted to the initial state.
Reason provided by the contract: "Seller cannot buy their own item".
You may want to cautiously increase the gas limit if the transaction went out of gas.
transact to SupplyChain.purchaseItem pending ...
[vm] from: 0x4B2...C02db to: SupplyChain.purchaseItem(uint256,uint256) 0xB34...24e5F value: 0 wei
data: 0xe06...40000 logs: 1 hash: 0x3a5...430eb
Debug
Initialize as git repo
Did you know? You can use the Recorder to record and replay your transactions to any network from the Deploy and Run plugin.
RemixAI Copilot (enabled)
Scam Alert


DEPLOY & RUN TRANSACTIONS
No pinned contracts found for selected workspace & network
Deployed/Unpinned Contracts
SUPPLYCHAIN AT 0XB34...24E5
Balance: 0 ETH
confirmDelivery
createItem
purchaseItem
shipItem
_id: 1
Calldata
Parameters
transact
getItem
0: uint256: id 1
1: string: name bag1
2: uint256: price 100000000000000000
3: uint8: state 0
4: address: seller 0xAb8483F64d9C6d1EcF 9b849Ae677dD3315835cb2
5: address: buyer 0x0000000000000000000 0000000000000000000
Home
SupplyChain.sol
// Purchase an item
function purchaseItem(uint _id,uint _price) public payable itemExists(_id) {
Item storage item = items[_id];
require(item.state == ItemState.Created, "Item is not available for purchase");
require(_price == item.price, "Incorrect payment amount");
require(msg.sender != item.seller, "Seller cannot buy their own item");
item.buyer = payable(msg.sender);
item.state = ItemState.Paid;
emit ItemPurchased(_id, msg.sender,_price);
// Ship an item
function shipItem(uint _id) public itemExists(_id) onlySeller(_id) {
Item storage item = items[_id];
require(item.state == ItemState.Paid, "Item must be paid before shipping");
Listen on all transactions
Filter with transaction hash or address
revert
The transaction has been reverted to the initial state.
Reason provided by the contract: "Only seller can perform this action".
You may want to cautiously increase the gas limit if the transaction went out of gas.
transact to SupplyChain.shipItem pending ...
[vm] from: 0xAb8...35cb2 to: SupplyChain.shipItem(uint256) 0xB34...24e5F value: 0 wei data: 0xf61...00001 logs: 1 hash: 0x91c...4b4ed
Debug
Initialize as git repo
Did you know? You can use the Recorder to record and replay your transactions to any network from the Deploy and Run plugin.
RemixAI Copilot (enabled)
Scam Alert


DEPLOY & RUN TRANSACTIONS
SUPPLYCHAIN AT 0XB34...24E5
Balance: 0 ETH
confirmDelivery
createItem
purchaseItem
shipItem
getItem
_id: 1
_id: "1"
Calldata
Parameters
transact
call
0: uint256: id 1
1: string: name bag1
2: uint256: price 100000000000000000
3: uint8: state 2
4: address: seller 0xAb8483F64d9C6d1EcF 9b849Ae677dD3315835cb2
5: address: buyer 0x4B20993Bc481177ec7 E8f571ceCaE8A9e22C02db
6: string: description skybag
getItemsForS...
Home
SupplyChain.sol
// Purchase an item
function purchaseItem(uint _id,uint _price) public payable itemExists(_id) {
Item storage item = items[_id];
require(item.state == ItemState.Created, "Item is not available for purchase");
require(_price == item.price, "Incorrect payment amount");
require(msg.sender != item.seller, "Seller cannot buy their own item");
item.buyer = payable(msg.sender);
item.state = ItemState.Paid;
emit ItemPurchased(_id, msg.sender,_price);
// Ship an item
function shipItem(uint _id) public itemExists(_id) onlySeller(_id) {
Item storage item = items[_id];
require(item.state == ItemState.Paid, "Item must be paid before shipping");
Listen on all transactions
Filter with transaction hash or address
call to SupplyChain.getItemsForSale
[call] from: 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db to: SupplyChain.getItemsForSale()
data: 0x064...ddd04
Debug
call to SupplyChain.getItem
[call] from: 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db to: SupplyChain.getItem(uint256) data: 0x312...00001
Debug
Initialize as git repo
Did you know? You can use the Recorder to record and replay your transactions to any network from the Deploy and Run plugin.
RemixAI Copilot (enabled)
Scam Alert