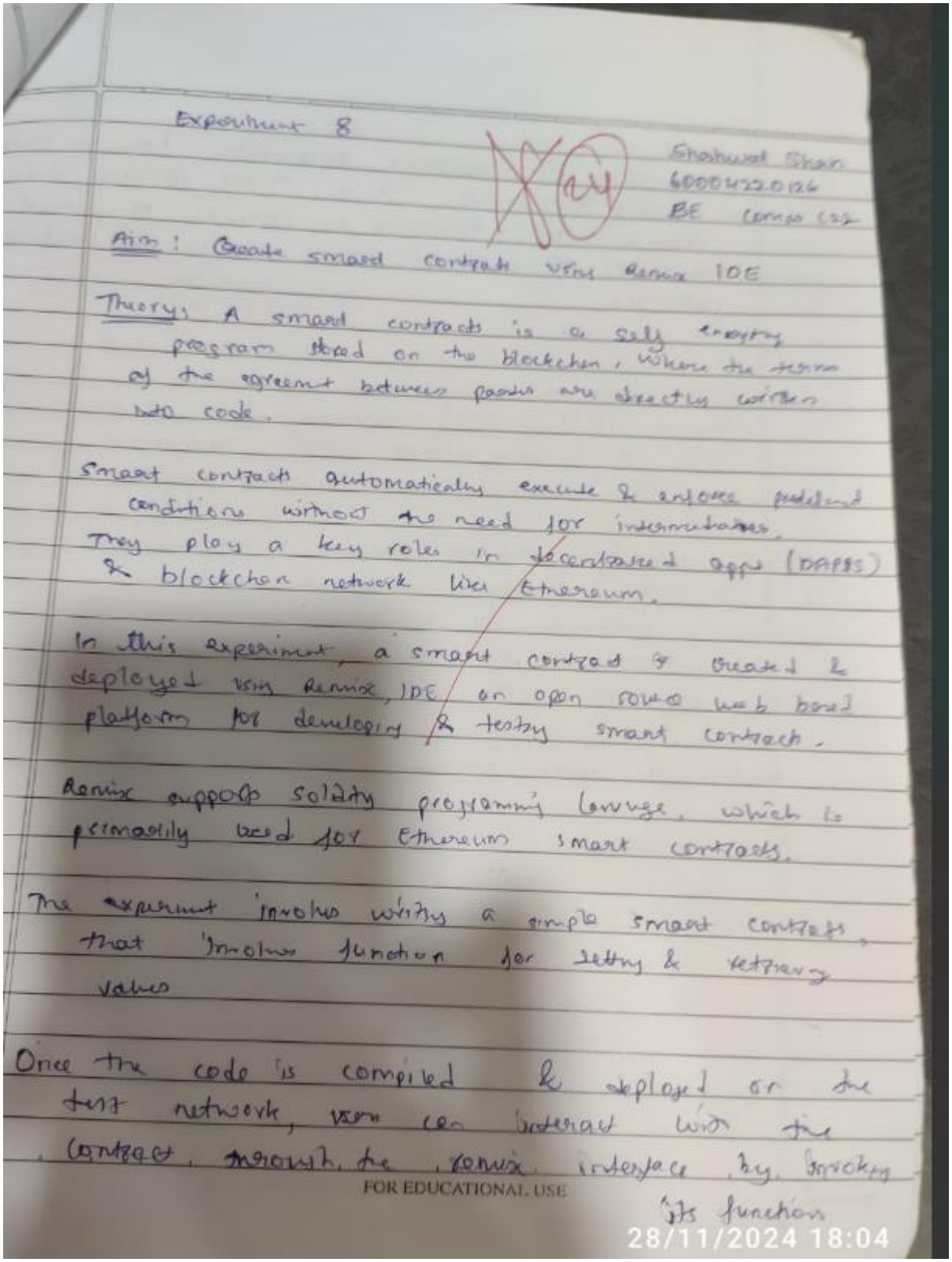
BLOCKCHAIN TECHNOLOGY

EXPERIMENT NO.08



Experiment 8

Shashwat Shah
60004220126
BE Comps (3)

Aim : Create smart contract using Remix IDE

Theory: A smart contracts is a self executing program stored on the blockchain, where the terms of the agreement between parties are directly written into code.

Smart contracts automatically execute & enforce predefined conditions without the need for intermediaries. They play a key roles in decentralized apps (DAPPS) & blockchain network like Ethereum.

In this experiment, a smart contract is created & deployed using Remix IDE an open source web based platform for developing & testing smart contracts.

Remix supports solidity programming language, which is primarily used for Ethereum smart contracts.

The experiment involves writing a simple smart contracts, that involves function for setting & retrieving values

Once the code is compiled & deployed on the test network, user can interact with the contract through the remix interface by invoking its functions

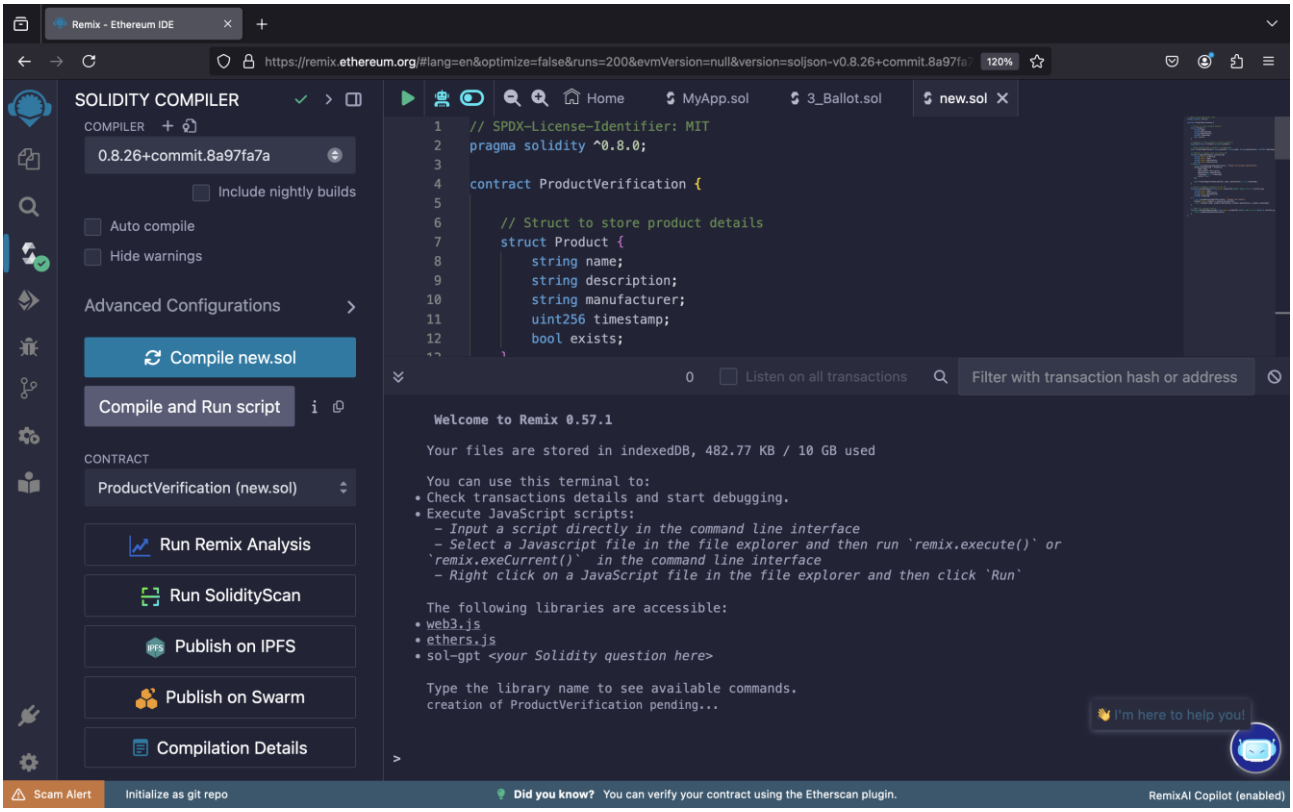
FOR EDUCATIONAL USE

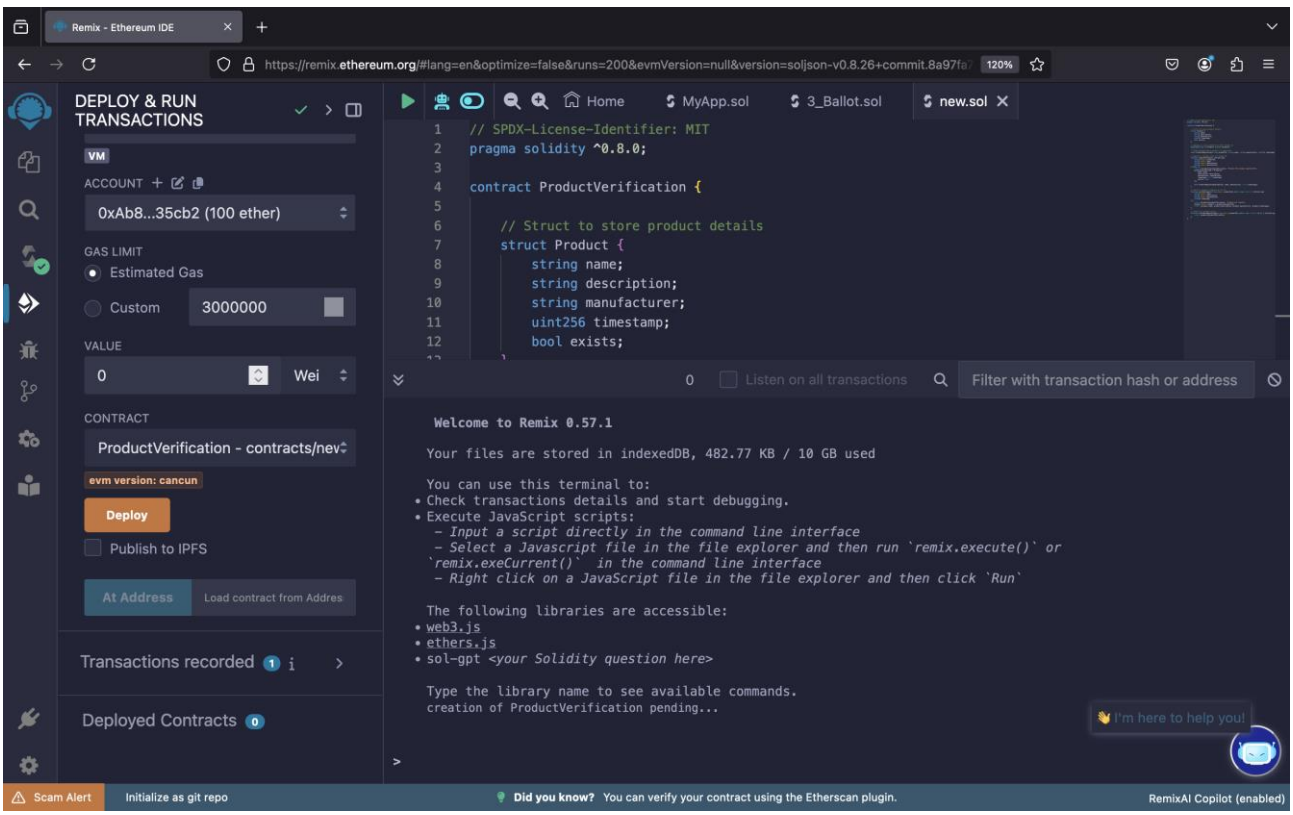
28/11/2024 18:04

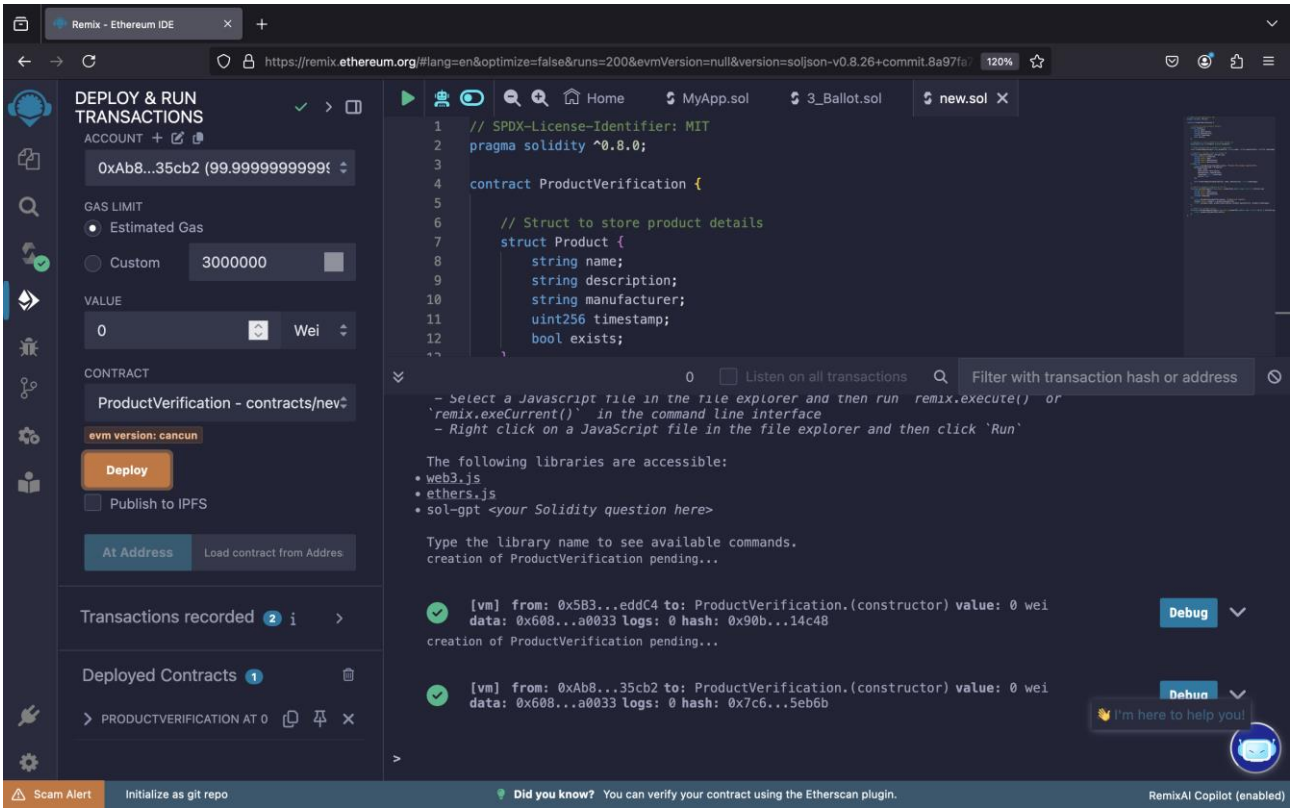
CODE & OUTPUT:-

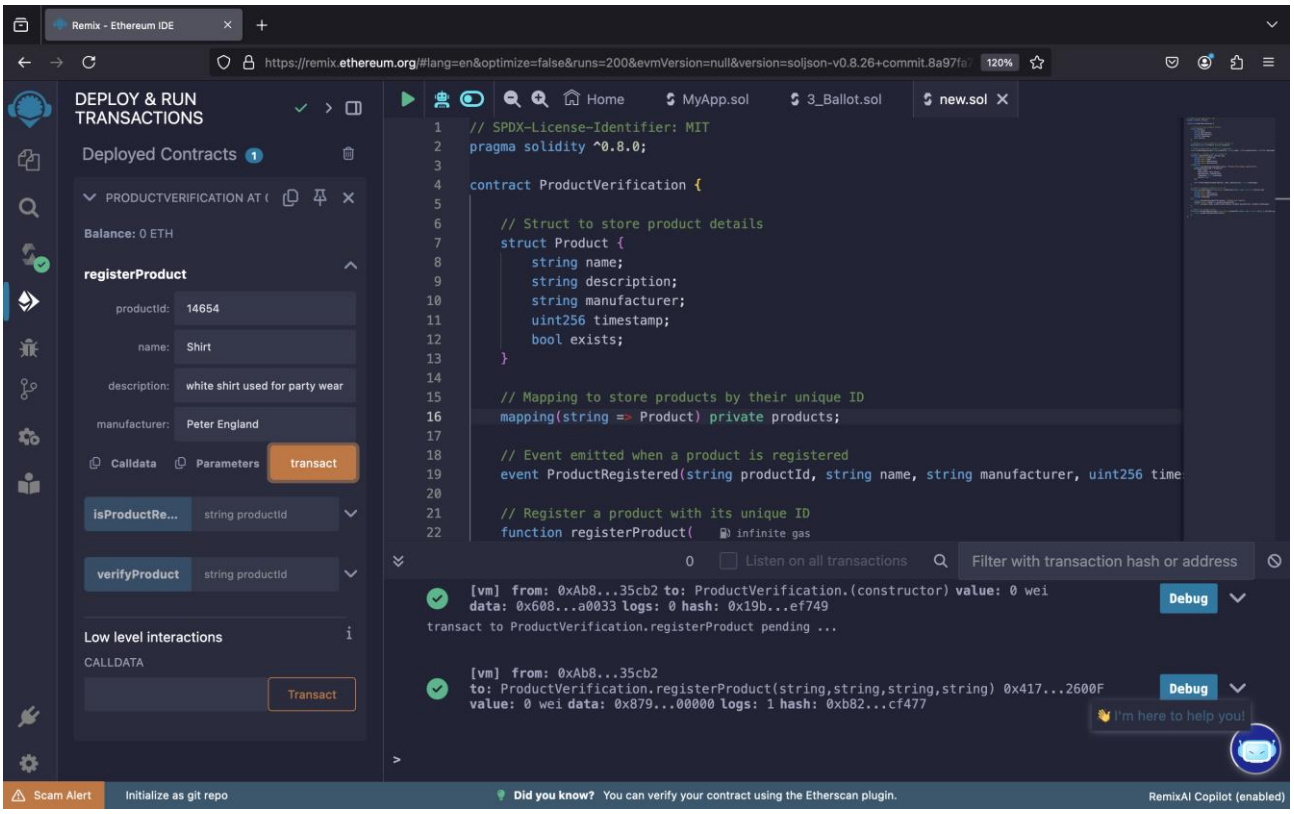
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
```

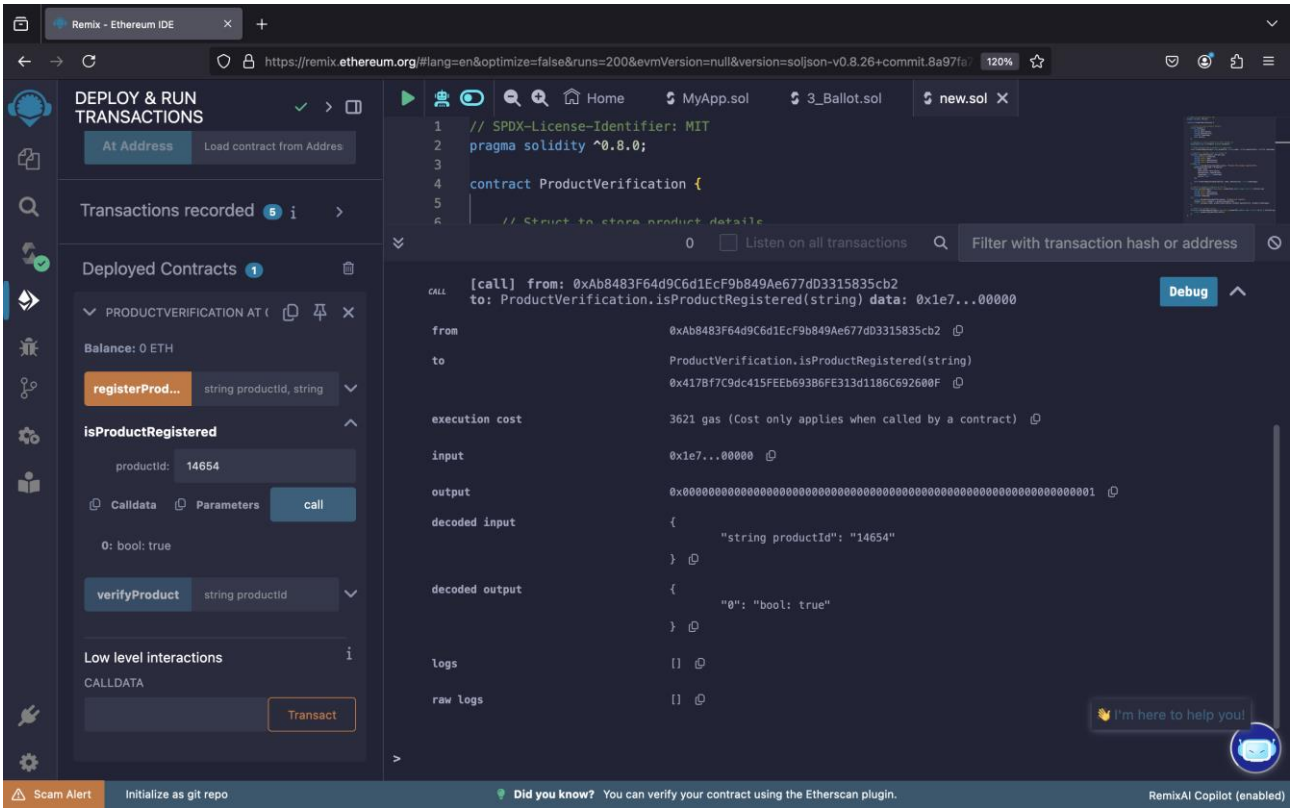
```
contract ProductVerification {

    // Struct to store product details
    struct Product {
        string name;
        string description;
        string manufacturer;
        uint256 timestamp;
        bool exists;
    }

    // Mapping to store products by their unique ID
    mapping(string => Product) private products;

    // Event emitted when a product is registered
    event ProductRegistered(string productId, string name, string manufacturer, uint256 timestamp);

    // Register a product with its unique ID
    function registerProduct(
        string memory productId,
        string memory name,
        string memory description,
        string memory manufacturer
    ) public {
        require(!products[productId].exists, "Product ID already registered");
        products[productId] = Product({
            name: name,
            description: description,
            manufacturer: manufacturer,
            timestamp: block.timestamp,
            exists: true
        });

        emit ProductRegistered(productId, name, manufacturer, block.timestamp);
    }

    // Verify a product's details by its ID
    function verifyProduct(string memory productId) public view returns (
        string memory name,
        string memory description,
        string memory manufacturer,
        uint256 timestamp
    ) {
        require(products[productId].exists, "Product not found");
        Product memory product = products[productId];
        return (product.name, product.description, product.manufacturer, product.timestamp);
    }

    // Check if a product exists
    function isProductRegistered(string memory productId) public view returns (bool) {
        return products[productId].exists;
    }
}
```

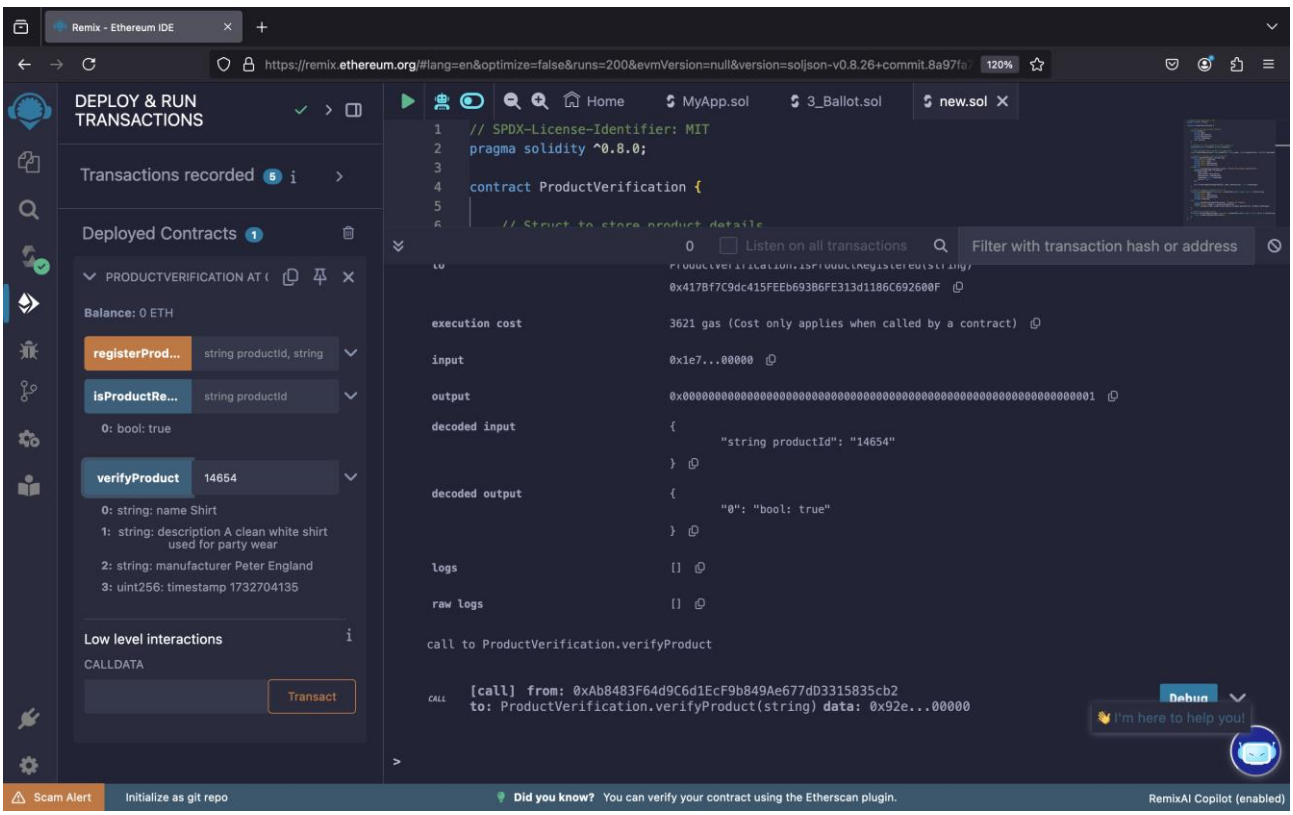

Remix - Ethereum IDE
https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.26+commit.8a97fa
SOLIDITY COMPILER
COMPILER
0.8.26+commit.8a97fa7a
Include nightly builds
Auto compile
Hide warnings
Advanced Configurations
Compile new.sol
Compile and Run script
CONTRACT
ProductVerification (new.sol)
Run Remix Analysis
Run SolidityScan
Publish on IPFS
Publish on Swarm
Compilation Details
Home
MyApp.sol
3_Ballot.sol
new.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract ProductVerification {
// Struct to store product details
struct Product {
string name;
string description;
string manufacturer;
uint256 timestamp;
bool exists;
Listen on all transactions
Filter with transaction hash or address
Welcome to Remix 0.57.1
Your files are stored in indexedDB, 482.77 KB / 10 GB used
You can use this terminal to:
Check transactions details and start debugging.
Execute JavaScript scripts:
Input a script directly in the command line interface
Select a Javascript file in the file explorer and then run `remix.execute()` or `remix.exeCurrent()` in the command line interface
Right click on a JavaScript file in the file explorer and then click `Run`
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 ProductVerification pending...
I'm here to help you!
Scam Alert
Initialize as git repo
Did you know? You can verify your contract using the Etherscan plugin.
RemixAI Copilot (enabled)


Remix - Ethereum IDE
https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.26+commit.8a97fa
DEPLOY & RUN TRANSACTIONS
VM
ACCOUNT
0xAb8...35cb2 (100 ether)
GAS LIMIT
Estimated Gas
Custom
3000000
VALUE
0
Wei
CONTRACT
ProductVerification - contracts/nev
evm version: cancun
Deploy
Publish to IPFS
At Address
Load contract from Addres
Transactions recorded 1
Deployed Contracts 0
Home
MyApp.sol
3_Ballot.sol
new.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract ProductVerification {
// Struct to store product details
struct Product {
string name;
string description;
string manufacturer;
uint256 timestamp;
bool exists;
Listen on all transactions
Filter with transaction hash or address
Welcome to Remix 0.57.1
Your files are stored in indexedDB, 482.77 KB / 10 GB used
You can use this terminal to:
Check transactions details and start debugging.
Execute JavaScript scripts:
Input a script directly in the command line interface
Select a Javascript file in the file explorer and then run `remix.execute()` or `remix.exeCurrent()` in the command line interface
Right click on a JavaScript file in the file explorer and then click `Run`
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 ProductVerification pending...
I'm here to help you!
Scam Alert
Initialize as git repo
Did you know? You can verify your contract using the Etherscan plugin.
RemixAI Copilot (enabled)


Remix - Ethereum IDE
https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.26+commit.8a97fa
120%
DEPLOY & RUN TRANSACTIONS
ACCOUNT
0xAb8...35cb2 (99.99999999995
GAS LIMIT
Estimated Gas
Custom
3000000
VALUE
0
Wei
CONTRACT
ProductVerification - contracts/nev
evm version: cancun
Deploy
Publish to IPFS
At Address
Load contract from Addres
Transactions recorded 2
Deployed Contracts 1
PRODUCTVERIFICATION AT 0
Home
MyApp.sol
3_Ballot.sol
new.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract ProductVerification {
// Struct to store product details
struct Product {
string name;
string description;
string manufacturer;
uint256 timestamp;
bool exists;
0
Listen on all transactions
Filter with transaction hash or address
- Right click on a JavaScript file in the file explorer and then click `Run`
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 ProductVerification pending...
[vm] from: 0x5B3...eddC4 to: ProductVerification.(constructor) value: 0 wei
data: 0x608...a0033 logs: 0 hash: 0x90b...14c48
Debug
creation of ProductVerification pending...
[vm] from: 0xAb8...35cb2 to: ProductVerification.(constructor) value: 0 wei
data: 0x608...a0033 logs: 0 hash: 0x7c6...5eb6b
I'm here to help you!
Scam Alert
Initialize as git repo
Did you know? You can verify your contract using the Etherscan plugin.
RemixAI Copilot (enabled)


Remix - Ethereum IDE
https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.26+commit.8a97fa
120%
DEPLOY & RUN TRANSACTIONS
Deployed Contracts 1
PRODUCTVERIFICATION AT (
Balance: 0 ETH
registerProduct
productId: 14654
name: Shirt
description: white shirt used for party wear
manufacturer: Peter England
Calldata
Parameters
transact
isProductRe...
string productId
verifyProduct
string productId
Low level interactions
CALLDATA
Transact
Home
MyApp.sol
3_Ballot.sol
new.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract ProductVerification {
// Struct to store product details
struct Product {
string name;
string description;
string manufacturer;
uint256 timestamp;
bool exists;
}
// Mapping to store products by their unique ID
mapping(string => Product) private products;
// Event emitted when a product is registered
event ProductRegistered(string productId, string name, string manufacturer, uint256 time
// Register a product with its unique ID
function registerProduct(infinite gas
0
Listen on all transactions
Filter with transaction hash or address
[vm] from: 0xAb8...35cb2 to: ProductVerification.(constructor) value: 0 wei
data: 0x608...a0033 logs: 0 hash: 0x19b...ef749
Debug
transact to ProductVerification.registerProduct pending ...
[vm] from: 0xAb8...35cb2
to: ProductVerification.registerProduct(string,string,string,string) 0x417...2600F
value: 0 wei data: 0x879...00000 logs: 1 hash: 0xb82...cf477
Debug
I'm here to help you!
Scam Alert
Initialize as git repo
Did you know? You can verify your contract using the Etherscan plugin.
RemixAI Copilot (enabled)


Remix - Ethereum IDE
https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.26+commit.8a97fa
DEPLOY & RUN TRANSACTIONS
At Address
Load contract from Addres
Transactions recorded 5
Deployed Contracts 1
PRODUCTVERIFICATION AT (
Balance: 0 ETH
registerProd...
string productId, string
isProductRegistered
productId: 14654
Calldata
Parameters
call
0: bool: true
verifyProduct
string productId
Low level interactions
CALLDATA
Transact
Home
MyApp.sol
3_Ballot.sol
new.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract ProductVerification {
Listen on all transactions
Filter with transaction hash or address
[call] from: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
to: ProductVerification.isProductRegistered(string) data: 0x1e7...00000
Debug
from 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
to ProductVerification.isProductRegistered(string)
0x417Bf7C9dc415FEEb693B6FE313d1186C692600F
execution cost 3621 gas (Cost only applies when called by a contract)
input 0x1e7...00000
output 0x0000000000000000000000000000000000000000000000000000000000000001
decoded input { "string productId": "14654" }
decoded output { "0": "bool: true" }
logs []
raw logs []
I'm here to help you!
Scam Alert
Initialize as git repo
Did you know? You can verify your contract using the Etherscan plugin.
RemixAI Copilot (enabled)


Remix - Ethereum IDE
https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.26+commit.8a97fa
DEPLOY & RUN TRANSACTIONS
Transactions recorded 5
Deployed Contracts 1
PRODUCTVERIFICATION AT (
Balance: 0 ETH
registerProd...
string productId, string
isProductRe...
string productId
0: bool: true
verifyProduct
14654
0: string: name Shirt
1: string: description A clean white shirt used for party wear
2: string: manufacturer Peter England
3: uint256: timestamp 1732704135
Low level interactions
CALLDATA
Transact
Home
MyApp.sol
3_Ballot.sol
new.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract ProductVerification {
Listen on all transactions
Filter with transaction hash or address
0x417Bf7C9dc415FEEb693B6FE313d1186C692600F
execution cost 3621 gas (Cost only applies when called by a contract)
input 0x1e7...00000
output 0x0000000000000000000000000000000000000000000000000000000000000001
decoded input { "string productId": "14654" }
decoded output { "0": "bool: true" }
logs []
raw logs []
call to ProductVerification.verifyProduct
[call] from: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
to: ProductVerification.verifyProduct(string) data: 0x92e...00000
I'm here to help you!
Scam Alert
Initialize as git repo
Did you know? You can verify your contract using the Etherscan plugin.
RemixAI Copilot (enabled)