

| **Title:**  Development of smart contract for any simple application - II |
| --- |

**Objective:**

To create smart contracts for Ecommerce sites & deploy it using Remix, Truffle and Ganache.

**Expected Outcome of Experiment:**

| **CO** | **Outcome** |
| --- | --- |
| CO2 | Learn Solidity language & Multiple Technology-based developments. |
| CO3 | Apply the algorithm and techniques used in Blockchain |

**Books/ Journals/ Websites referred:**

* https://trufflesuite.com/ganache/
* https://trufflesuite.com/truffle/
* https://remix-ide.readthedocs.io/en/latest/run.html
* https://remix.ethereum.org/
* https://www.web3.university/tracks/create-a-smart-contract/

**Abstract**: **Smart Contract**

A "smart contract" is simply a program that runs on the Ethereum blockchain. It's a collection of code (its functions) and data (its state) that resides at a specific address on the Ethereum blockchain. Smart contracts are a type of Ethereum account. This means they have a balance and can be the target of transactions. However they're not controlled by a user, instead they are deployed to the network and run as programmed. User accounts can then interact with a smart contract by submitting transactions that execute a function defined on the smart contract. Smart contracts can define rules, like a regular contract, and automatically enforce them via the code. Smart contracts cannot be deleted by default, and interactions with them are irreversible.

**Related Theory: -**

Smart contracts allow developers to build a wide variety of decentralized apps and tokens. They’re used in everything from new financial tools to logistics and game experiences, and they’re stored on a blockchain like any other crypto transaction. Once a smart-contract app has been added to the blockchain, it generally can’t be reversed or changed (although there are some exceptions).

Smart-contract-powered apps are often referred to as “decentralized applications” or “dapps” – and they include decentralized finance (or DeFi) tech that aims to transform the banking industry. DeFi apps allow cryptocurrency holders to engage in complex financial transactions — saving, loans, insurance — without a bank or other financial institution taking a cut and from anywhere in the world.

Features of Smart Contracts

* Distributed: Everyone on the network is guaranteed to have a copy of all the conditions of the smart contract and they cannot be changed by one of the parties. A smart contract is replicated and distributed by all the nodes connected to the network.
* Deterministic: Smart contracts can only perform functions for which they are designed only when the required conditions are met. The final outcome will not vary, no matter who executes the smart contract.
* Immutable: Once deployed smart contract cannot be changed, it can only be removed as long as the functionality is implemented previously.
* Autonomy: There is no third party involved. The contract is made by you and shared between the parties. No intermediaries are involved which minimizes bullying and grants full authority to the dealing parties. Also, the smart contract is maintained and executed by all the nodes on the network, thus removing all the controlling power from any one party’s hand.
* Customizable: Smart contracts have the ability for modification or we can say customization before being launched to do what the user wants it to do.
* Transparent: Smart contracts are always stored on a public distributed ledger called blockchain due to which the code is visible to everyone, whether or not they are participants in the smart contract.
* Trustless: These are not required by third parties to verify the integrity of the process or to check whether the required conditions are met.
* Self-verifying: These are self-verifying due to automated possibilities.
* Self-enforcing: These are self-enforcing when the conditions and rules are met at all stages.

Applications of Smart Contracts

* Real Estate: Reduce money paid to the middleman and distribute between the parties actually involved. For example, a smart contract to transfer ownership of an apartment once a certain amount of resources have been transferred to the seller’s account(or wallet).
* Vehicle ownership: A smart contract can be deployed in a blockchain that keeps track of vehicle maintenance and ownership. The smart contract can, for example, enforce vehicle maintenance service every six months; failure of which will lead to suspension of driving license.
* Music Industry: The music industry could record the ownership of music in a blockchain. A smart contract can be embedded in the blockchain and royalties can be credited to the owner’s account when the song is used for commercial purposes. It can also work in resolving ownership disputes.
* Government elections: Once the votes are logged in the blockchain, it would be very hard to decrypt the voter address and modify the vote leading to more confidence against the ill practices.

Ganache:

Ganache is a personal blockchain for rapid Ethereum and Corda distributed application development. You can use Ganache across the entire development cycle; enabling you to develop, deploy, and test your dApps in a safe and deterministic environment. Ganache comes in two flavors: a UI and CLI. Ganache UI is a desktop application supporting both Ethereum and Corda technology. Our more robust command-line tool, ganache, is available for Ethereum development. It offers:

* Console.log in Solidity ,Zero-config Mainnet and testnet forking
* Fork any Ethereum network without waiting to sync
* Ethereum JSON-RPC support, fast-forward time
* Snapshot/revert state with mine blocks instantly, on demand, or at an interval

**Implementation Details:**

**1. Enlist all the Steps followed and various options explored**

1. Explored truffle and ganache environment for Vscode
2. Created smart contract for e-commerce site with

An array for the products registered. It includes product id, title, description, seller, price, buyer and delivered (true of false).

1. registerProduct(): To register the product. The price of the product while registering should be greater than 0. All the details of the product will be mentioned such as product id, title, description, price and seller address.
2. buy(): If the buyer doesn’t enter the exact amount of the product while buying then the message “Please pay the exact price” will be shown. The seller of the product cannot be the buyer.
3. delivery(): Only the buyer can confirm the delivery. When the delivery is confirmed by the buyer then the amount is transferred from the buyer account to the seller.

C. Deploy the smart contract using truffle and ganache

* truffle compile – To compile smart contract using truffle on Vscode
* truffle migrate – To deploy smart contract
* Check ganache for results of transactions

**2. Explain your program logic, classes and methods used.**

Ecommerce smart contracts has function :

registerProduct(): To register the product. The price of the product while registering should be greater than 0. All the details of the product will be mentioned such as product id, title, description, price and seller address.

buy(): If the buyer doesn’t enter the exact amount of the product while buying then the message “Please pay the exact price” will be shown. The seller of the product cannot be the buyer.

delivery(): Only the buyer can confirm the delivery. When the delivery is confirmed by the buyer then the amount is transferred from the buyer account to the seller.

**3. Explain the Importance of the approach followed by you**

Code:

// SPDX-License-Identifier: GPL-3.0

pragma solidity 0.8.17;

contract Ecommerce {

struct Product {

string title;

string desc;

address payable seller;

uint productId;

uint price;

address buyer;

bool delivered;

}

uint counter = 1;

Product[] public products;

event registered(string title, uint productId, address seller);

event bought(uint productId, address buyer);

event delivered(uint productId);

function registerProduct(string memory \_title, string memory \_desc,

uint \_price) public {

require(\_price>0, "Price should be greater than zero");

Product memory tempProduct;

tempProduct.title = \_title;

tempProduct.desc = \_desc;

tempProduct.price = \_price\* 10\*\*18;

tempProduct.seller = payable(msg.sender);

tempProduct.productId = counter;

products.push(tempProduct);

counter++;

emit registered(\_title, tempProduct.productId, msg.sender);

}

function buy (uint \_productId) payable public {

require(products [\_productId-1].price==msg.value, "Please pay the exact price");

require(products [\_productId-1].seller!=msg.sender, "Seller cannot be the buyer");

products [\_productId-1]. buyer=msg.sender;

emit bought (\_productId, msg.sender);

}

function delivery (uint \_productId) public {

require(products[\_productId-1].buyer== msg.sender, "Only buyer can confirm it");

products [\_productId-1]. delivered = true;

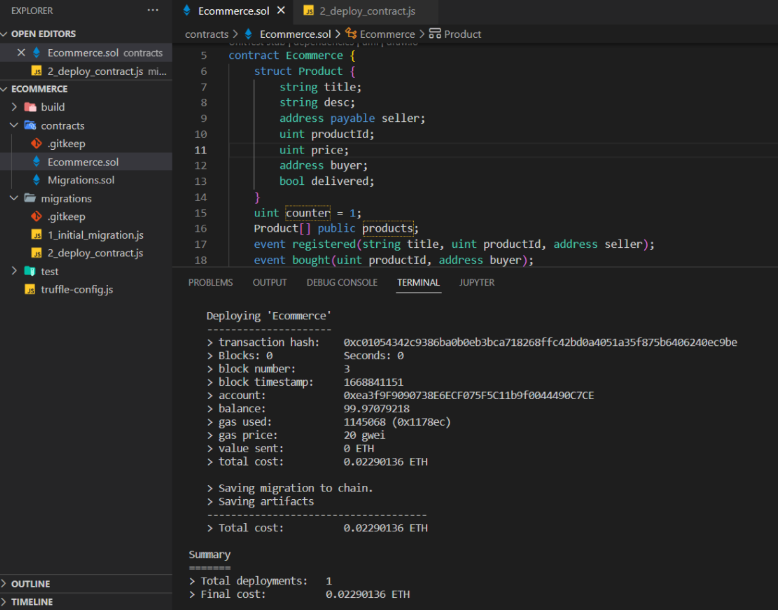
products [\_productId-1].seller.transfer (products[\_productId-1].price);

emit delivered (\_productId);

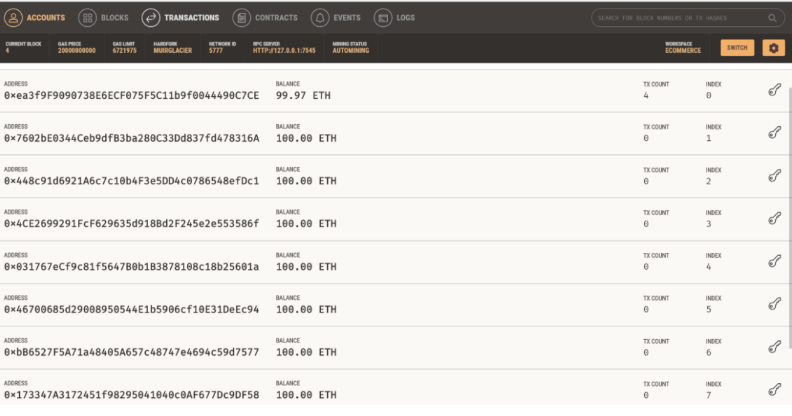
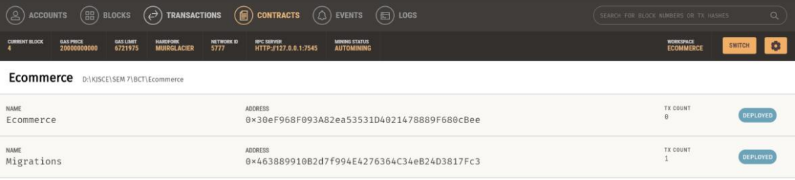
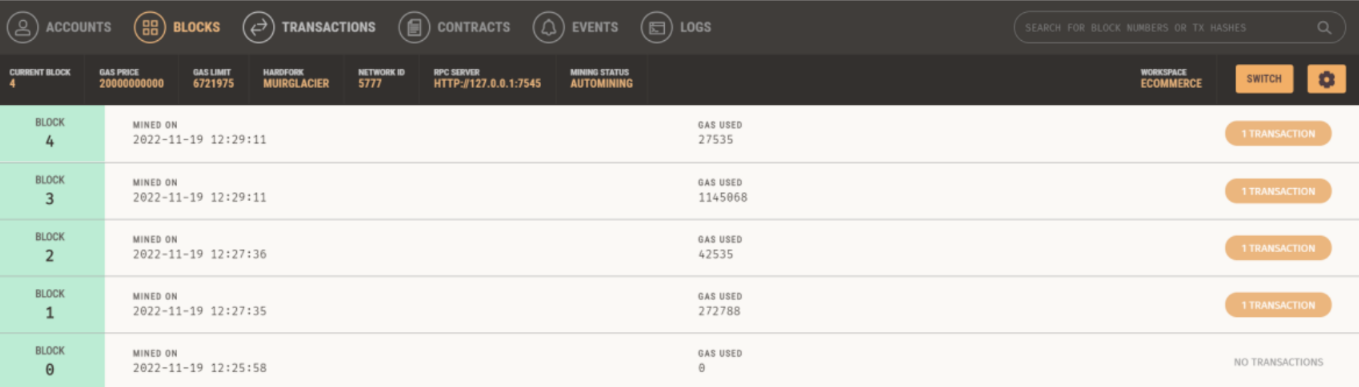
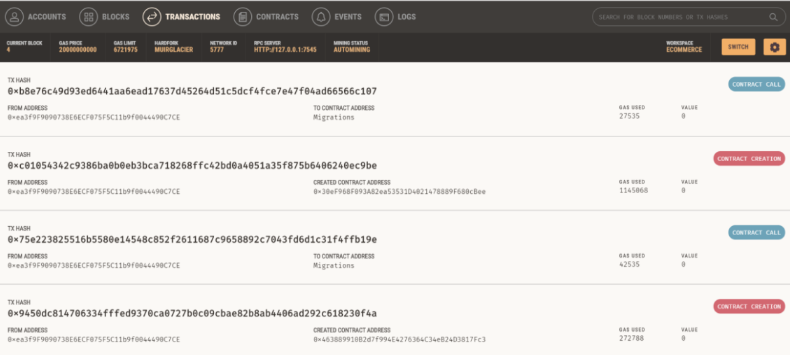
}

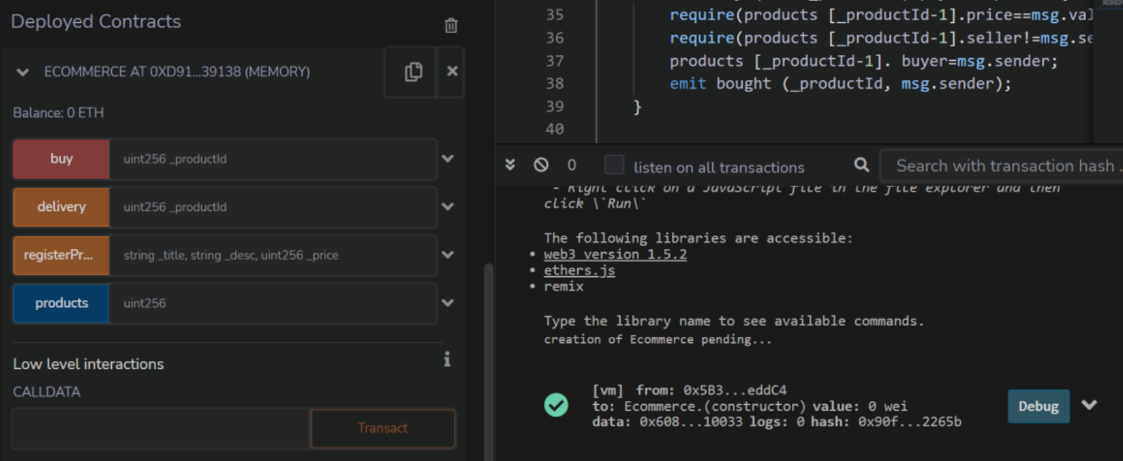
}

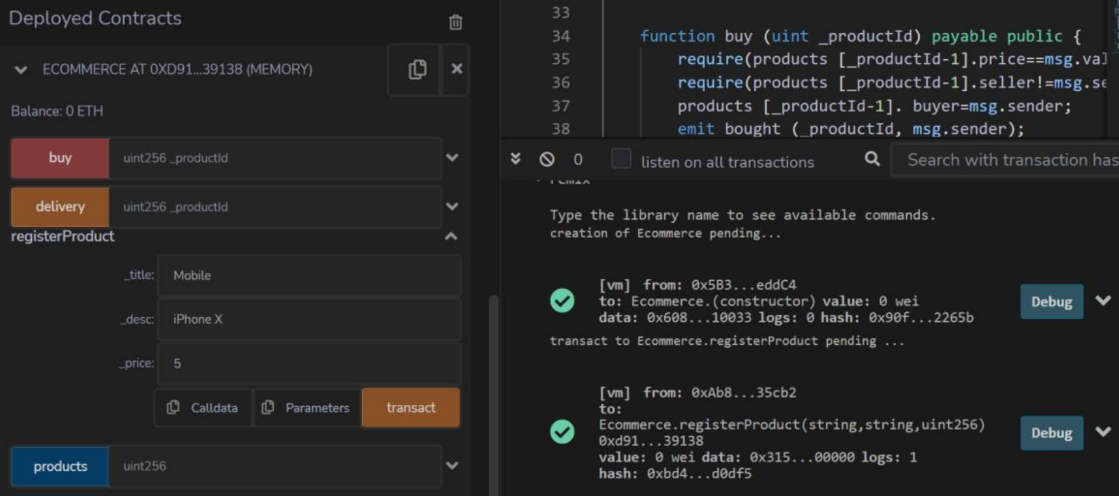
Output:

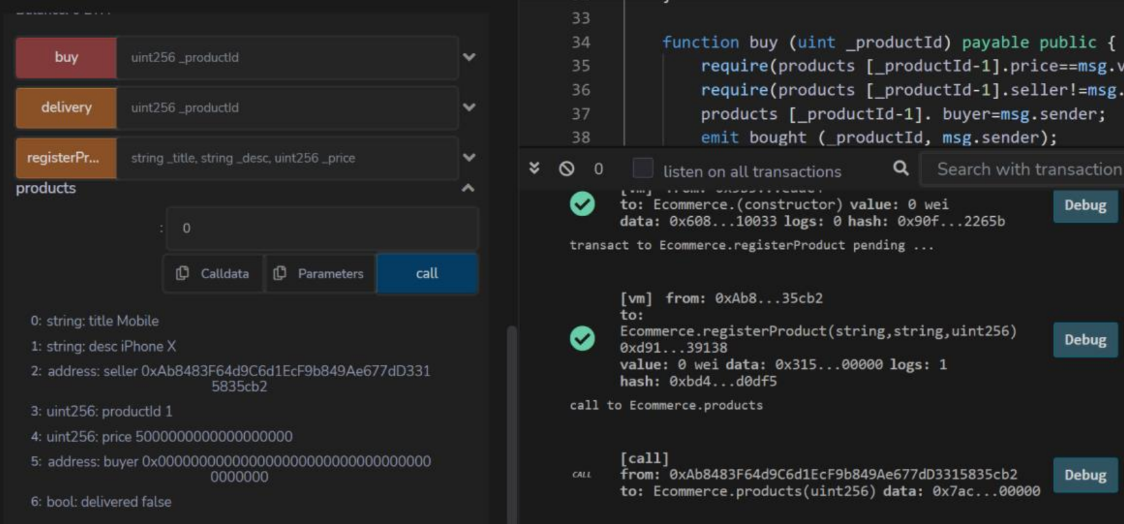


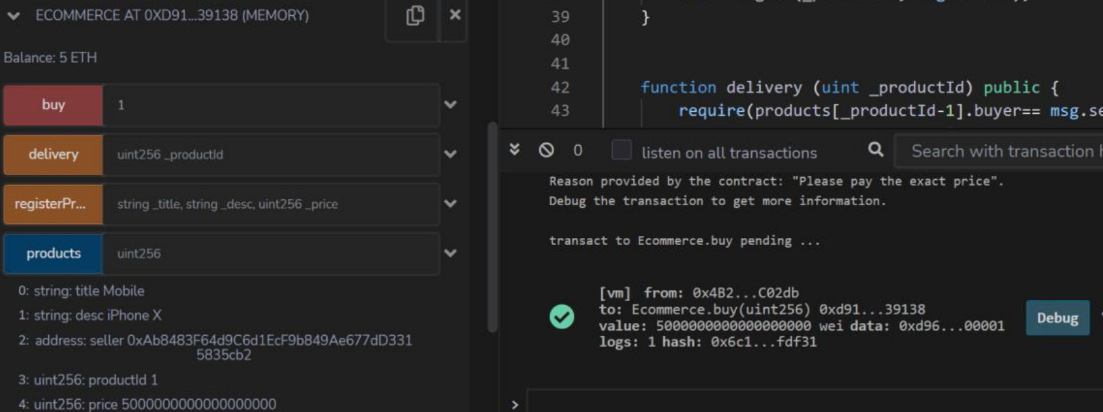
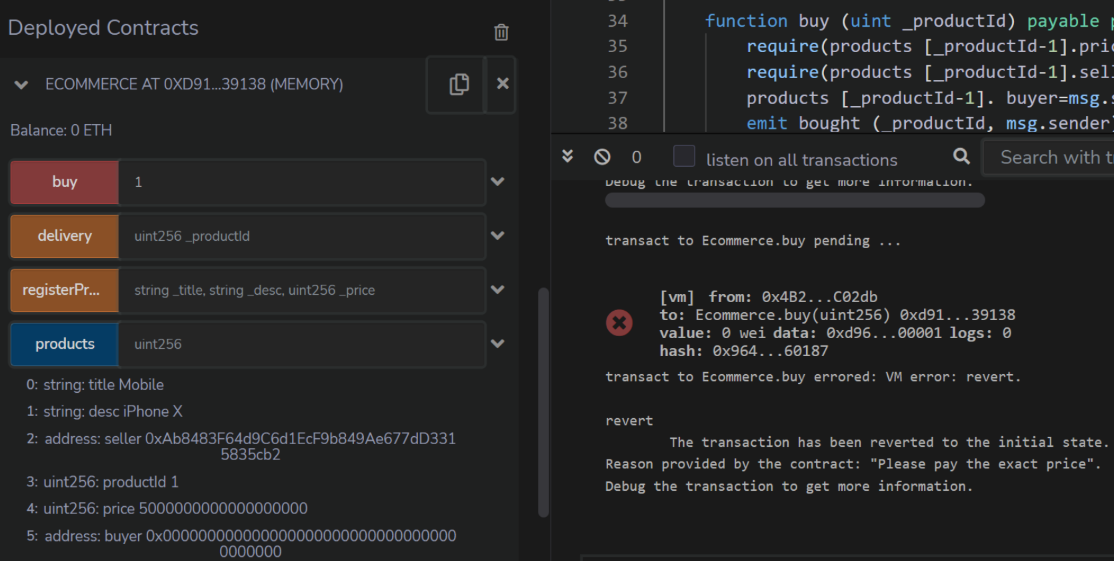
Deployed using truffle & ganache

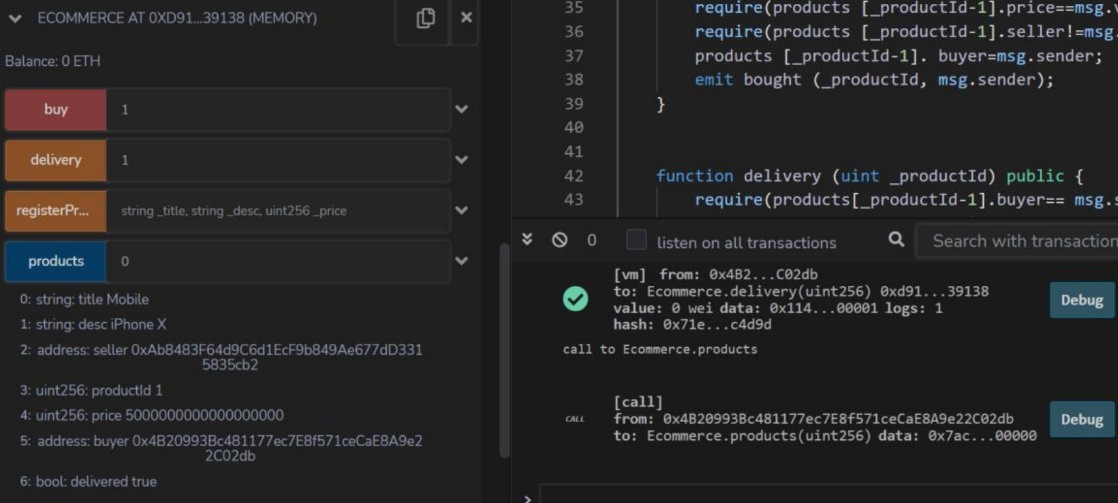
Contracts:Transactions:Functions of buy, register product and delivery:

Product Register

Viewing product details:

Buy Products

Delivery function:



**Conclusion:-**

We have successfully understood & implemented the concept of smart contracts by creating and deploying an e-commerce smart contract with the help of Remix, Truffle and Ganache.