CA-3

Name:- Surendra Khyalia

Section:- K22CS

Course Code:-CSC307

Reg No.:- 12222968

Submitted To:- Piyush Gururani

Problem Statement

Write a Solidity smart contract function for a basic decentralized marketplace where users can list items for sale. The function should allow sellers to set the price and buyers to purchase the item by paying the specified amount. Ensure that only the buyer receives ownership of the item after purchase. Include checks to prevent underpayments or overpayments.

Contract code

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

contract Marketplace {
    struct Item {
        uint id;
        string name;
        uint priceInEth;
        address payable seller;
        address buyer;
        bool sold;
    }

    uint public itemCount;
    mapping(uint => Item) public items;

    event ItemListed(uint id, string name, uint priceInEth, address seller);
    event ItemPurchased(uint id, address buyer, uint priceInEth);
    function listItem(string memory _name, uint _priceInEth) public {
        require(_priceInEth > 0, "Price must be greater than 0 Ether");
        itemCount++;
        items[itemCount] = Item(itemCount, _name, _priceInEth, payable(msg.sender), address(0), false);
        emit ItemListed(itemCount, _name, _priceInEth, msg.sender);
    }
    function buyItem(uint _id) public payable {
        Item storage item = items[_id];
        require(item.id > 0 && item.id <= itemCount, "Item does not exist");
        require(!item.sold, "Item already sold");
        require(msg.value == item.priceInEth * 1 ether, "Incorrect payment amount");
        require(msg.sender != item.seller, "Seller cannot buy their own item");

        item.buyer = msg.sender;
        item.sold = true;
        item.seller.transfer(item.priceInEth * 1 ether);

        emit ItemPurchased(item.id, msg.sender, item.priceInEth);
    }
}
```

Contract Overview

The Marketplace.sol smart contract provides a decentralized platform where users can list items for sale and purchase them securely. Sellers can list items for sale by specifying a name and price and Buyers can purchase items by sending the exact amount of Ether. Ownership is transferred to the buyer upon successful purchase.

Basic Implementaon

struct Item: Defines the structure of an item in the marketplace.

uint id: A unique identifier for each item.

string name: The name of the item listed for sale.

uint priceInEth: Price of the item, stored in Ether.

address payable seller: Address of the seller listing the item.

address buyer: Address of the buyer who purchases the item.

bool sold: Status indicating whether the item has been sold or not.

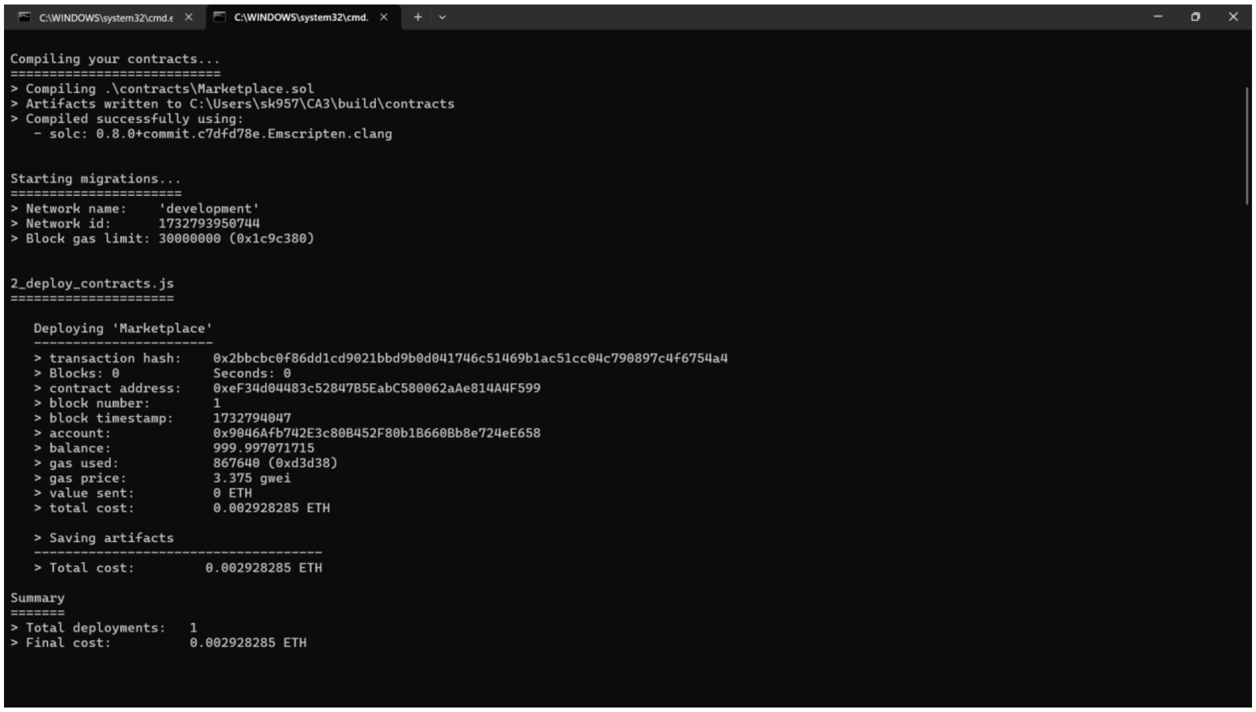
uint public itemCount: Counts the total number of items listed in the marketplace.

mapping(uint => Item) public items: A mapping to store all listed items, where the key is the item's ID, and the value is the Item struct.

ItemListed(uint id, string name, uint priceInEth, address seller): Emitted when a new item is listed.

ItemPurchased(uint id, address buyer, uint priceInEth): Emitted when an item is purchased.

Screenshot Of the Deployed Contract

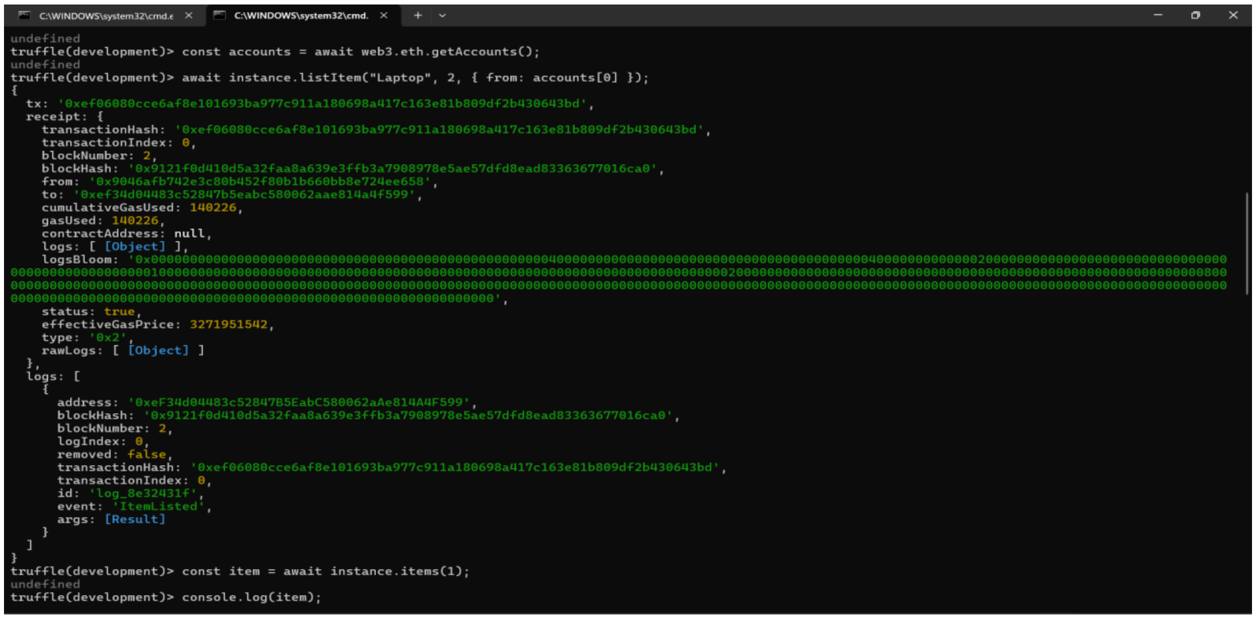


```
Compiling your contracts...
===========================
> Compiling .\contracts\Marketplace.sol
> Artifacts written to C:\Users\sk957\CA3\build\contracts
> Compiled successfully using:
   - solc: 0.8.0+commit.c7dfd78e.Emscripten.clang


Starting migrations...
======================
> Network name:    'development'
> Network id:      1732793950744
> Block gas limit: 30000000 (0x1c9c380)


2_deploy_contracts.js
=====================

   Deploying 'Marketplace'
   -----------------------
   > transaction hash:    0x2bbcbc0f86dd1cd9021bbd9b0d041746c51469b1ac51cc04c790897c4f6754a4
   > Blocks: 0            Seconds: 0
   > contract address:    0xeF34d04483c52847B5EabC580062aAe814A4F599
   > block number:        1
   > block timestamp:     1732794047
   > account:             0x9046Afb742E3c80B452F80b1B660Bb8e724eE658
   > balance:             999.997071715
   > gas used:            867640 (0xd3d38)
   > gas price:           3.375 gwei
   > value sent:          0 ETH
   > total cost:          0.002928285 ETH

   > Saving artifacts
   -------------------------------------
   > Total cost:         0.002928285 ETH

Summary
=======
> Total deployments:   1
> Final cost:          0.002928285 ETH
```

Testing the Contract

1. To list an item

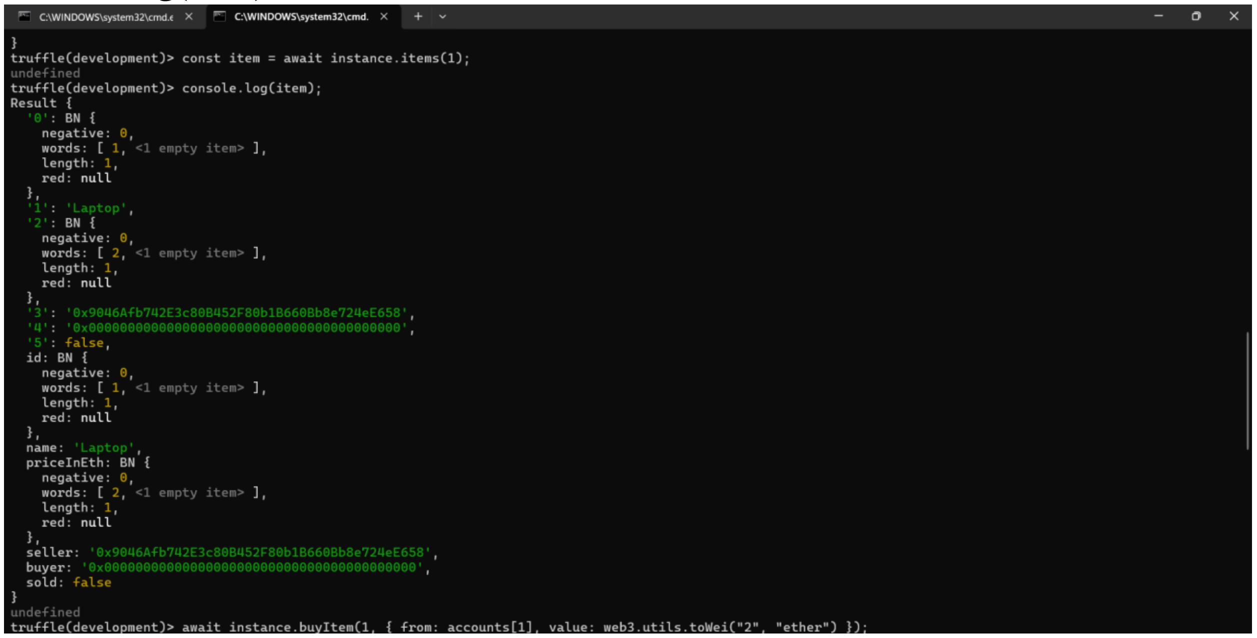
await instance.listItem("Laptop", 2, { from: accounts[0] });



```
undefined
truffle(development)> const accounts = await web3.eth.getAccounts();
undefined
truffle(development)> await instance.listItem("Laptop", 2, { from: accounts[0] });
{
  tx: '0xef06080cce6af8e101693ba977c911a180698a417c163e81b809df2b430643bd',
  receipt: {
    transactionHash: '0xef06080cce6af8e101693ba977c911a180698a417c163e81b809df2b430643bd',
    transactionIndex: 0,
    blockNumber: 2,
    blockHash: '0x9121f0d410d5a32faa8a639e3ffb3a7908978e5ae57dfd8ead83363677016ca0',
    from: '0x9046afb742e3c80b452f80b1b660bb8e724ee658',
    to: '0xef34d04483c52847b5eabc580062aae814a4f599',
    cumulativeGasUsed: 140226,
    gasUsed: 140226,
    contractAddress: null,
    logs: [ [Object] ],
    logsBloom: '0x000000000000000000000000000000000000000000000000000040000000000000000000000000000000000000000400000000000002000000000000000000000000000000
000000000000000000010000000000000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000800
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000',
    status: true,
    effectiveGasPrice: 3271951542,
    type: '0x2',
    rawLogs: [ [Object] ]
  },
  logs: [
    {
      address: '0xeF34d04483c52847B5EabC580062aAe814A4F599',
      blockHash: '0x9121f0d410d5a32faa8a639e3ffb3a7908978e5ae57dfd8ead83363677016ca0',
      blockNumber: 2,
      logIndex: 0,
      removed: false,
      transactionHash: '0xef06080cce6af8e101693ba977c911a180698a417c163e81b809df2b430643bd',
      transactionIndex: 0,
      id: 'log_8e32431f',
      event: 'ItemListed',
      args: [Result]
    }
  ]
}
truffle(development)> const item = await instance.items(1);
undefined
truffle(development)> console.log(item);
```

2. To view an item

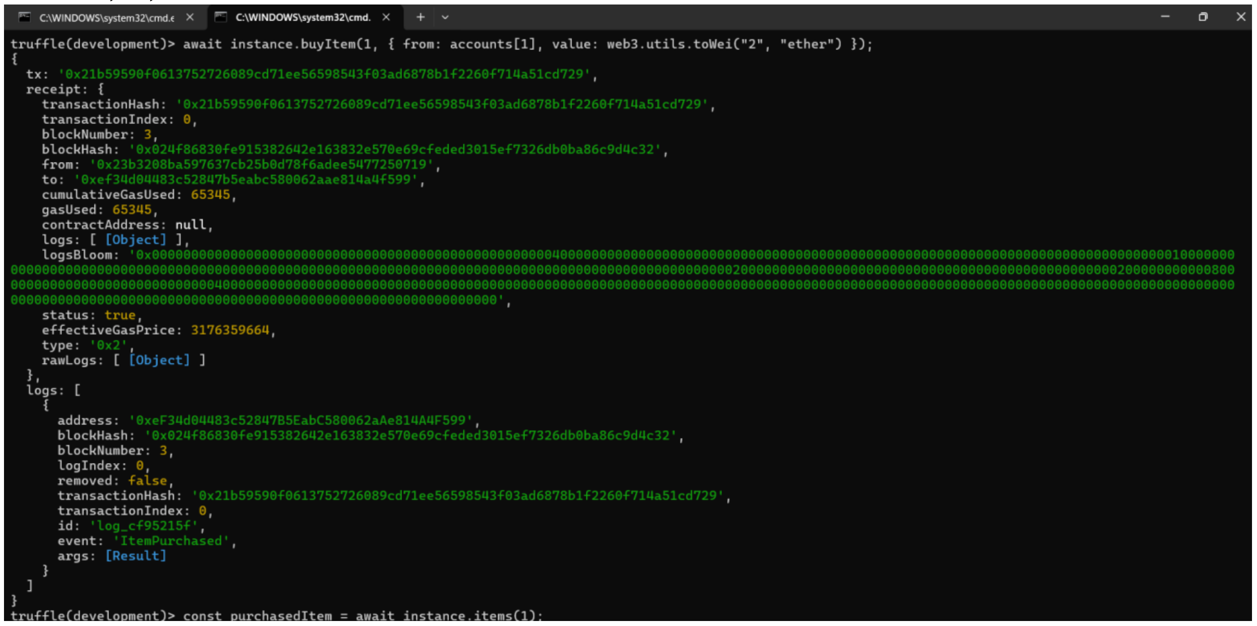
let item = await instance.items(1);
console.log(item);



```
}
truffle(development)> const item = await instance.items(1);
undefined
truffle(development)> console.log(item);
Result {
  '0': BN {
    negative: 0,
    words: [ 1, <1 empty item> ],
    length: 1,
    red: null
  },
  '1': 'Laptop',
  '2': BN {
    negative: 0,
    words: [ 2, <1 empty item> ],
    length: 1,
    red: null
  },
  '3': '0x9046Afb742E3c80B452F80b1B660Bb8e724eE658',
  '4': '0x0000000000000000000000000000000000000000',
  '5': false,
  id: BN {
    negative: 0,
    words: [ 1, <1 empty item> ],
    length: 1,
    red: null
  },
  name: 'Laptop',
  priceInEth: BN {
    negative: 0,
    words: [ 2, <1 empty item> ],
    length: 1,
    red: null
  },
  seller: '0x9046Afb742E3c80B452F80b1B660Bb8e724eE658',
  buyer: '0x0000000000000000000000000000000000000000',
  sold: false
}
undefined
truffle(development)> await instance.buyItem(1, { from: accounts[1], value: web3.utils.toWei("2", "ether") });
```

3. To buy an item

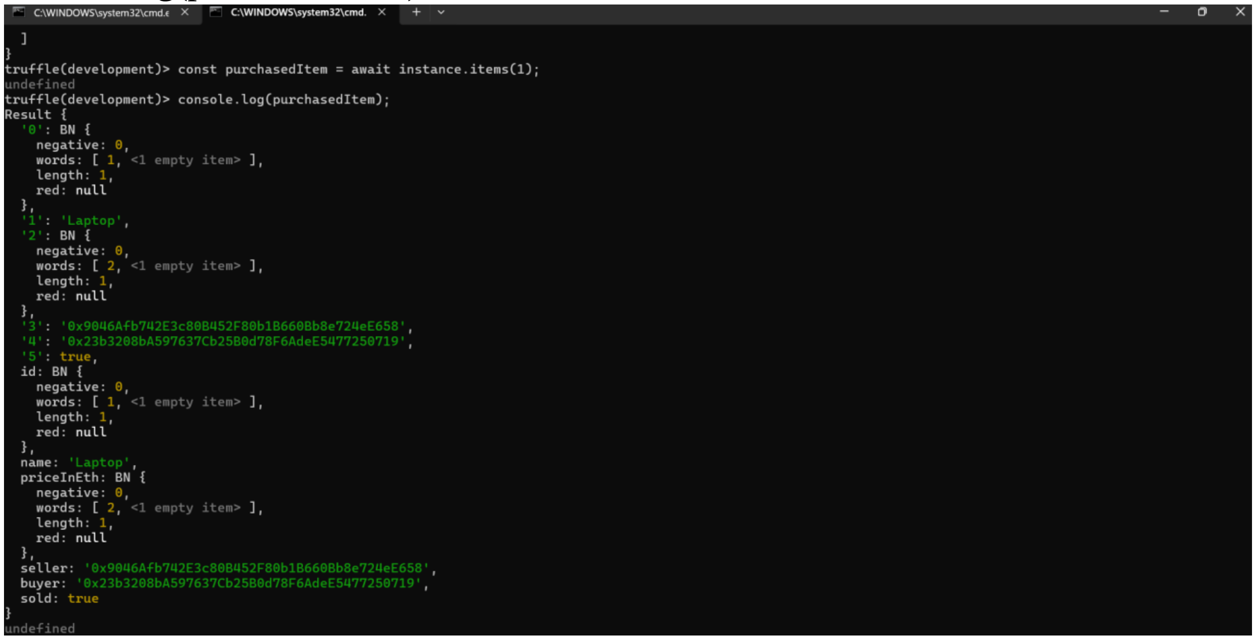
await instance.buyItem(1, { from: accounts[1], value: web3.utils.toWei("2", "ether") });



```
truffle(development)> await instance.buyItem(1, { from: accounts[1], value: web3.utils.toWei("2", "ether") });
{
  tx: '0x21b59590f0613752726089cd71ee56598543f03ad6878b1f2260f714a51cd729',
  receipt: {
    transactionHash: '0x21b59590f0613752726089cd71ee56598543f03ad6878b1f2260f714a51cd729',
    transactionIndex: 0,
    blockNumber: 3,
    blockHash: '0x024f86830fe915382642e163832e570e69cfeded3015ef7326db0ba86c9d4c32',
    from: '0x23b3208ba597637cb25b0d78f6adee5477250719',
    to: '0xef34d04483c52847b5eabc580062aae814a4f599',
    cumulativeGasUsed: 65345,
    gasUsed: 65345,
    contractAddress: null,
    logs: [ [Object] ],
    logsBloom: '0x000000000000000000000000000000000000000000000000000400000000000000000000000000000000000000000000000000000000000000000000000000000000000010000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000002000000000000000000000000000000000000000000000000200000000000800
000000000000000000000000004000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000',
    status: true,
    effectiveGasPrice: 3176359664,
    type: '0x2',
    rawLogs: [ [Object] ]
  },
  logs: [
    {
      address: '0xeF34d04483c52847B5EabC580062aAe814A4F599',
      blockHash: '0x024f86830fe915382642e163832e570e69cfeded3015ef7326db0ba86c9d4c32',
      blockNumber: 3,
      logIndex: 0,
      removed: false,
      transactionHash: '0x21b59590f0613752726089cd71ee56598543f03ad6878b1f2260f714a51cd729',
      transactionIndex: 0,
      id: 'log_cf95215f',
      event: 'ItemPurchased',
      args: [Result]
    }
  ]
}
truffle(development)> const purchasedItem = await instance.items(1);
```

4. To verify ownership

```
let purchasedItem = await instance.items(1);
console.log(purchasedItem);
```



```
  ]
}
truffle(development)> const purchasedItem = await instance.items(1);
undefined
truffle(development)> console.log(purchasedItem);
Result {
  '0': BN {
    negative: 0,
    words: [ 1, <1 empty item> ],
    length: 1,
    red: null
  },
  '1': 'Laptop',
  '2': BN {
    negative: 0,
    words: [ 2, <1 empty item> ],
    length: 1,
    red: null
  },
  '3': '0x9046Afb742E3c80B452F80b1B660Bb8e724eE658',
  '4': '0x23b3208bA597637Cb25B0d78F6AdeE5477250719',
  '5': true,
  id: BN {
    negative: 0,
    words: [ 1, <1 empty item> ],
    length: 1,
    red: null
  },
  name: 'Laptop',
  priceInEth: BN {
    negative: 0,
    words: [ 2, <1 empty item> ],
    length: 1,
    red: null
  },
  seller: '0x9046Afb742E3c80B452F80b1B660Bb8e724eE658',
  buyer: '0x23b3208bA597637Cb25B0d78F6AdeE5477250719',
  sold: true
}
undefined
```

References

Solidity Documentaon: https://soliditylang.org/

OpenZeppelin Contracts: https://www.openzeppelin.com/solidity-contracts/