

SKILL 4

[CS601] – Cryptography and Blockchain

Date – 14/02/2023 | *By* Aishwarya Suryakant Waghmare, PRN – 2001106059

Title :

To implement escrow example using Solidity Programming..

Skills/Competencies to be acquired :

- To implement escrow example using Solidity Programming.
- To understand the escrow with the related.

Time taken to complete the activity :

One Hour

Purpose of the activity :

Escrow is the third party which holds the asset(asset can be money, bond, stocks) on the presence of two parties. Escrow will release the fund when certain conditions are met.

For Example, "A" is a seller and wants to sell his car, "B" is a buyer who wants to buy "A"'s car so they will contact Escrow "C"(an arbiter) which hold the asset until "B" receives the car. When this condition will be met, Escrow will release the fund to "A". This solves the issue of trust and prevents any discrepancy.

Let's the write a smart contract for the Escrow using solidity language.

Here Smart contract will hold the asset, which will be released on when conditions are fulfilled..

Steps performed in this activity :

Code written in Remix IDE :

```
pragma solidity 0.6.0;
```

```
// Defining a Contract
```

```
contract escrow{
```

```
    // Declaring the state variables
```

```
    address payable public buyer;
```

```
    address payable public seller;
```

```
    address payable public arbiter;
```

```
    mapping(address => uint) TotalAmount;
```

```
    // Defining an enumerator 'State'
```

```

enum State{

    // Following are the data members
    awate_payment, awate_delivery, complete
}

// Declaring the object of the enumerator
State public state;

// Defining function modifier 'instate'
modifier instantiate(State expected_state){

    require(state == expected_state);

    _;
}

// Defining function modifier 'onlyBuyer'
modifier onlyBuyer(){
    require(msg.sender == buyer ||
            msg.sender == arbiter);

    _;
}

// Defining function modifier 'onlySeller'
modifier onlySeller(){
    require(msg.sender == seller);

    _;
}

// Defining a constructor
constructor(address payable _buyer,
            address payable _sender) public{
    // Assigning the values of the
    // state variables
    arbiter = msg.sender;
    buyer = _buyer;
    seller = _sender;
    state = State.awate_payment;
}

// Defining function to confirm payment

```

```

function confirm_payment() onlyBuyer instate(
State.awate_payment) public payable{
    state = State.awate_delivery;
}

// Defining function to confirm delivery
function confirm_Delivery() onlyBuyer instate(
State.awate_delivery) public{

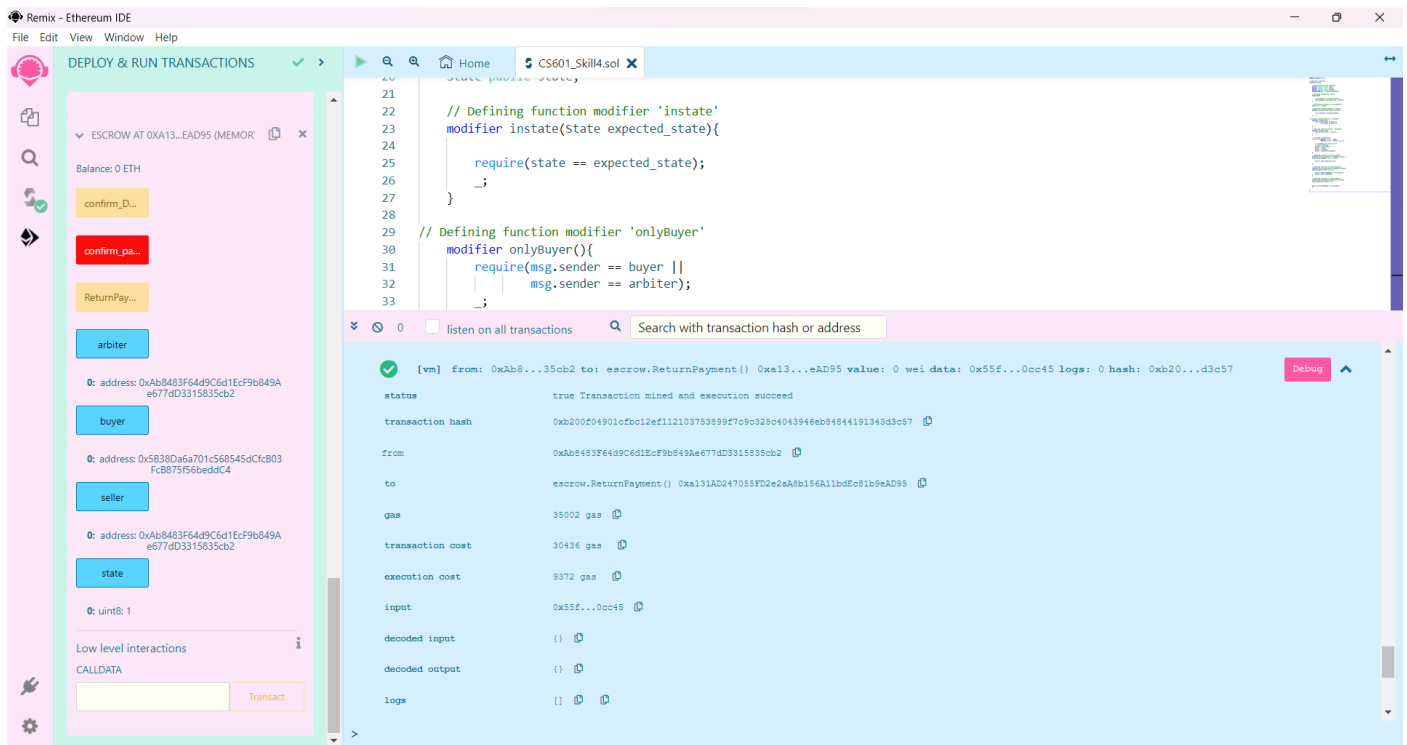
    seller.transfer(address(this).balance);
    state = State.complete;
}

// Defining function to return payment
function ReturnPayment() onlySeller instate(
State.awate_delivery)public{
    buyer.transfer(address(this).balance);
}
}






```

Output/ Presentation prepared :

The screenshot displays the Remix - Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel shows the deployment of a contract named 'ESCROW AT 0xA13...EAD95 (MEMOR)'. The contract's state is set to 'state - call'. The main editor shows the smart contract code, including the 'instate' and 'onlyBuyer' modifiers. The bottom panel displays the transaction logs, showing the execution of the 'confirm_payment' function, the 'ReturnPayment' function, and the 'confirm_Delivery' function. The logs include details such as the transaction hash, the sender and receiver addresses, and the data passed to the functions.



Resources/ tools used for the skill activity :

-  Laptop
-  Remix IDE
-  Lecture Notes
-  E-References
-  E-books.

Skills/ Competencies acquired and Result/ Conclusion :

Therefore, understood and implemented the escrow smart contract by making use of the solidity programming in the Remix IDE.