

Roll number – 19BCE123

Name – Jash Mavani

Batch - EL2

Course Code & Name – 2CSDE93 & Blockchain Technology

Practical – 9

Aim : To write a Solidity contract that implements a distributed ticket sales system. Anybody can create an event (specifying the initial price and number of tickets). Anybody can then purchase one of the initial tickets or sell those tickets peer-to-peer. At the event, gate agents will check that each attendee is listed in the final attendees list on the blockchain. (Ethereum programming)

Smart Contract :

```
pragma solidity >=0.8.0;

contract Ticket { uint256 ticketPrice = 100 wei; address
owner;

mapping (address => uint256) public ticketHolders;

constructor() payable { owner = msg.sender; }

function buyTickets(address _user, uint256 _amount) payable public
{
    require(msg.value >= ticketPrice * _amount); addTickets(_user, _amount); }

function useTickets(address _user, uint256 _amount) public
{
    subTickets(_user, _amount); } function addTickets(address _user, uint256 _amount) internal
{
    ticketHolders[_user] = ticketHolders[_user] + _amount; } function subTickets(address _user,
uint256 _amount) internal
{
```

```

        require(ticketHolders[_user] >= _amount, "You do not have enough tickets!");
        ticketHolders[_user] = ticketHolders[_user] - _amount; }

```

function withdraw() public

```
{
```

```

    require(msg.sender == owner, "You are not the owner!"); (bool success, ) = owner.call{ value:
    address(this).balance}(""); require(success);

```

```
}
```

```
}
```

Output:

The screenshot displays a web application for managing tickets. On the left, a sidebar contains a 'Deploy' button and a section for 'At Address' with a 'Load contract from Address' button. Below this, it shows 'Transactions recorded' and 'Deployed Contracts'. The main area features a code editor with the following Solidity code:

```

13 function buyTickets(address _user, uint256 _amount) payable public
14 {
15     require(msg.value >= ticketPrice * _amount);
16     addTickets(_user, _amount);
17 }
18
19 function useTickets(address _user, uint256 _amount) public
20 {
21     subTickets(_user, _amount);
22 }
23
24 function addTickets(address _user, uint256 _amount) internal
25 {
26     ticketHolders[_user] = ticketHolders[_user] + _amount;
27 }
28
29 function subTickets(address _user, uint256 _amount) internal
30 {
31     require(ticketHolders[_user] >= _amount, "You do not have enough tickets!");
32 }

```

Below the code editor, there is a search bar and a 'listen on network' checkbox. A transaction is being executed, with a confirmation message 'creation of Ticket pending...' and a green checkmark. A tooltip indicates 'buyTickets - transact (payable)'.

DEPLOY & RUN TRANSACTIONS

buyTickets

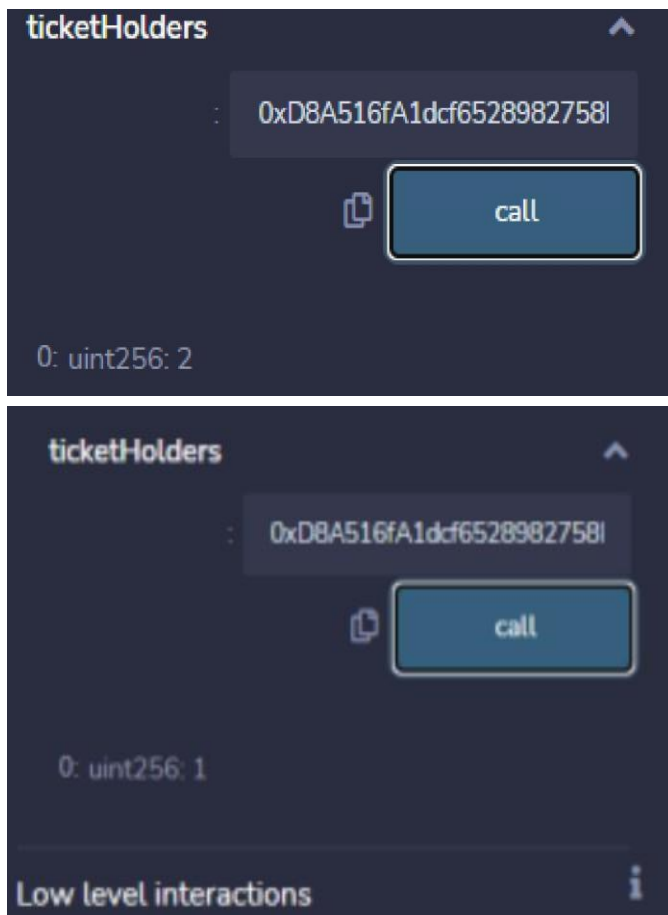
_user:

_amount:

useTickets

_user:

_amount:



Conclusion :

After completion of this practical, I learnt how to write smart contract for sales system using RemixIDE.