PROJECT TITLE: Smart Contract Project - Event Organization

CODE:

```
//SPDX-License-Identifier: Unlicense
pragma solidity >=0.5.0 <0.9.0;

contract EventContract {
 struct Event{
   address organizer;
   string name;
   uint date; //0 1 2
   uint price;
   uint ticketCount;  //1 sec  0.5 sec
   uint ticketRemain;
 }

 mapping(uint=>Event) public events;
 mapping(address=>mapping(uint=>uint)) public
tickets;
 uint public nextId;


 function createEvent(string memory name,uint
date,uint price,uint ticketCount) external{
   require(date>block.timestamp,"You can organize
event for future date");
   require(ticketCount>0,"You can organize event only
if you create more than 0 tickets");

   events[nextId] =
Event(msg.sender,name,date,price,ticketCount,ticketCo
unt);
   nextId++;
 }

 function buyTicket(uint id,uint quantity) external
payable{
   require(events[id].date!=0,"Event does not
exist");
   require(events[id].date>block.timestamp,"Event has
already occured");
```

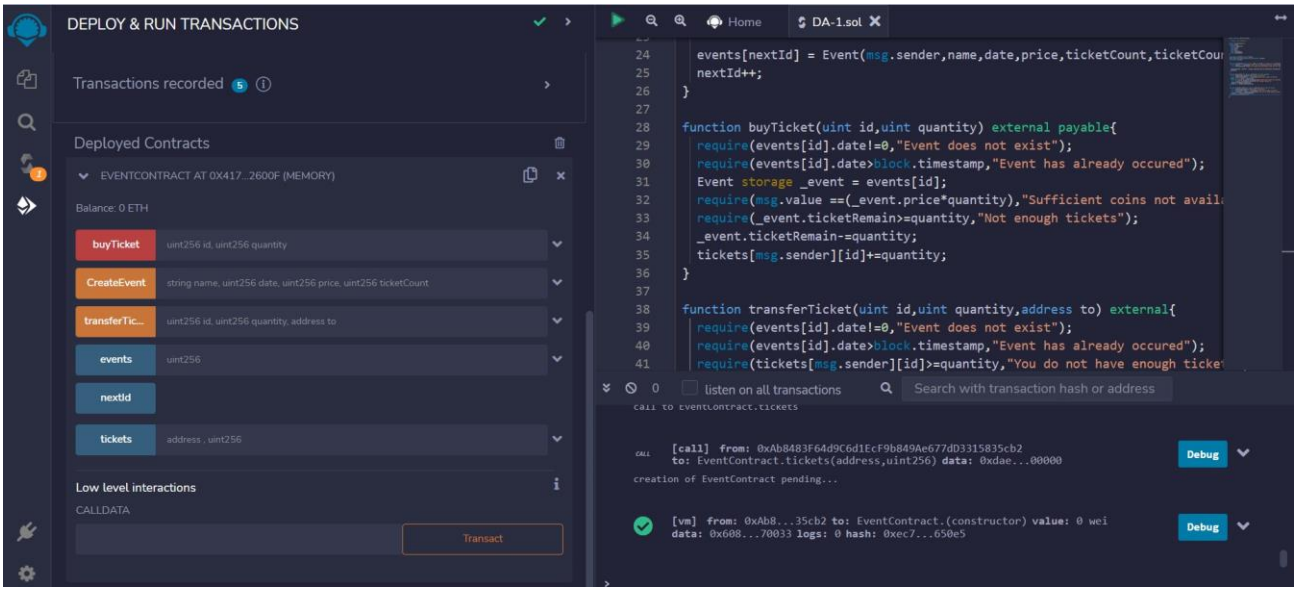
```
    Event storage _event = events[id];
    require(msg.value==(_event.price*quantity),"Ethere
is not enough");
    require(_event.ticketRemain>=quantity,"Not enough
tickets");
    _event.ticketRemain-=quantity;
    tickets[msg.sender][id]+=quantity;

  }


  function transferTicket(uint id,uint
quantity,address to) external{
    require(events[id].date!=0,"Event does not
exist");
    require(events[id].date>block.timestamp,"Event has
already occured");
    require(tickets[msg.sender][id]>=quantity,"You do
not have enough tickets");
    tickets[msg.sender][id]-=quantity;
    tickets[to][id]+=quantity;
  }
}
```

P.T.O.

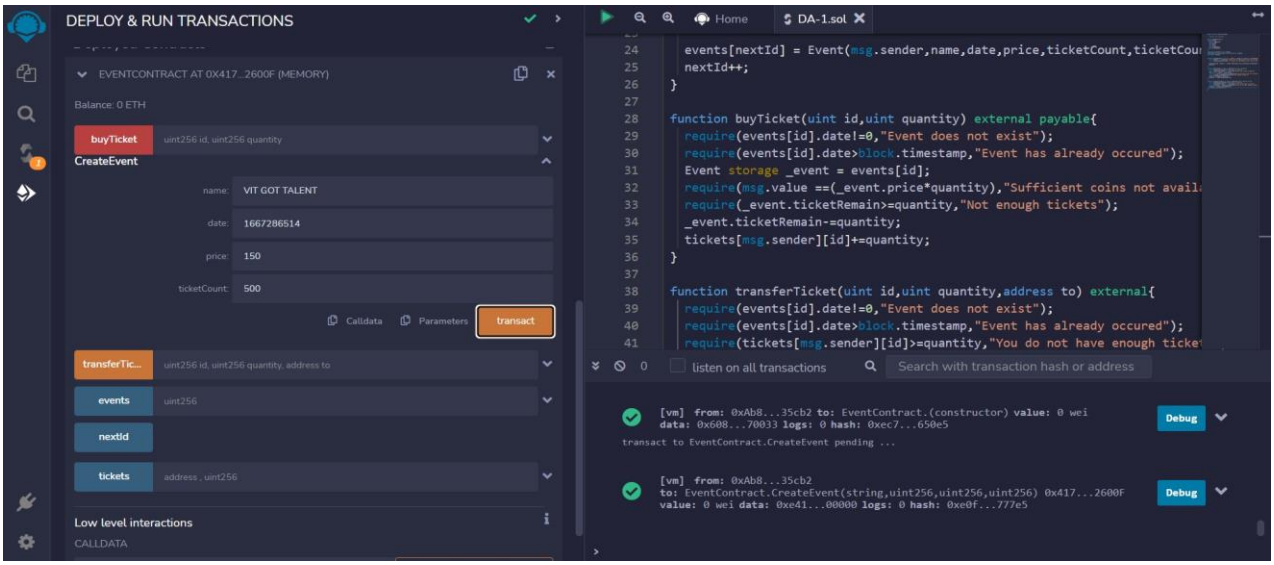
SCREENSHOTS:

We Events Application:

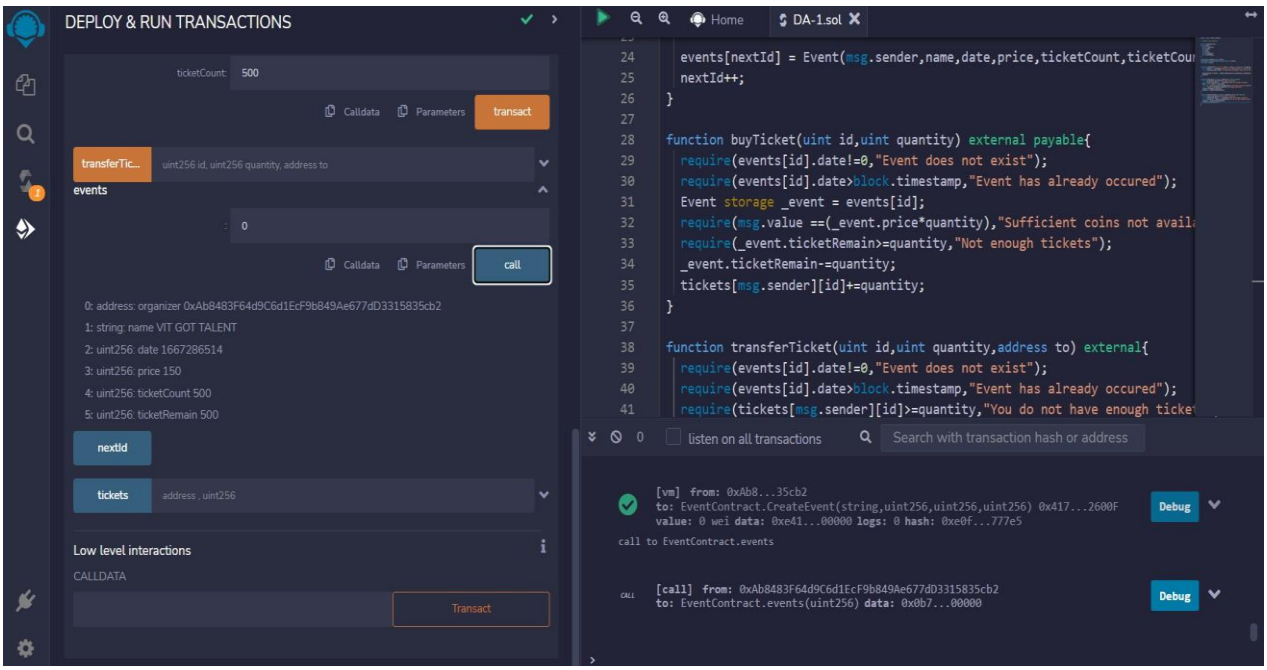
I have used solidity as the blockchain framework and built smart contracts using the proof of concept to organise the events.



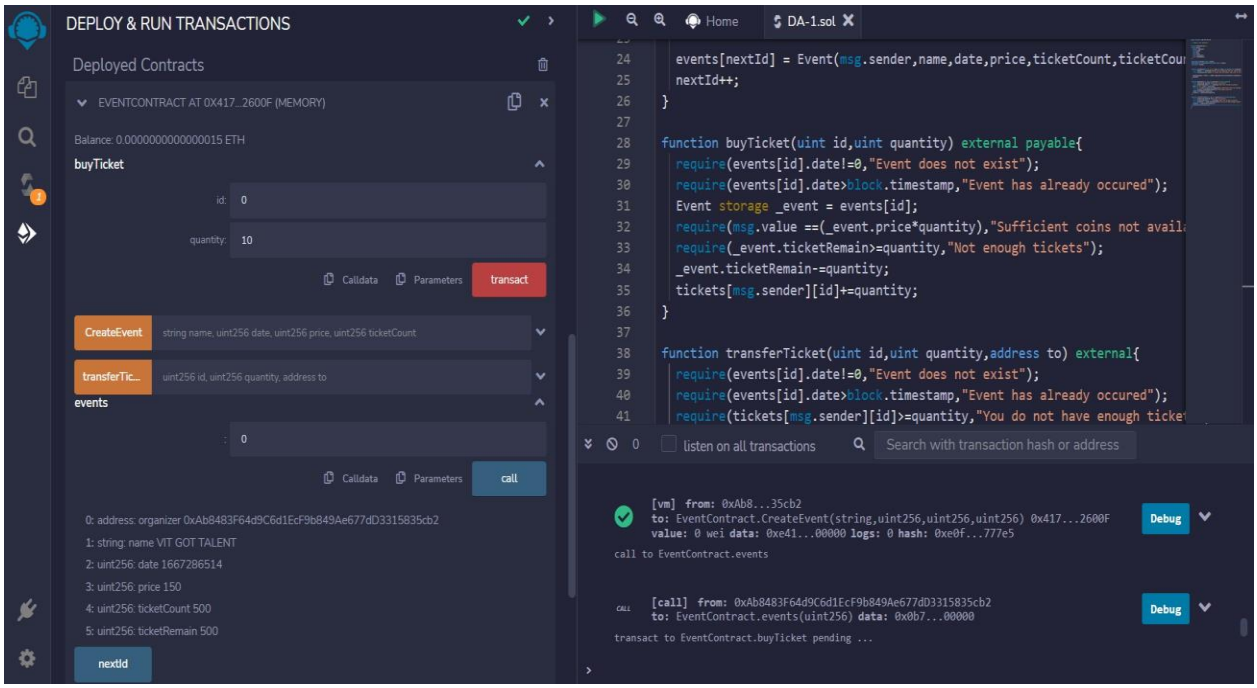
Buy Tickets:



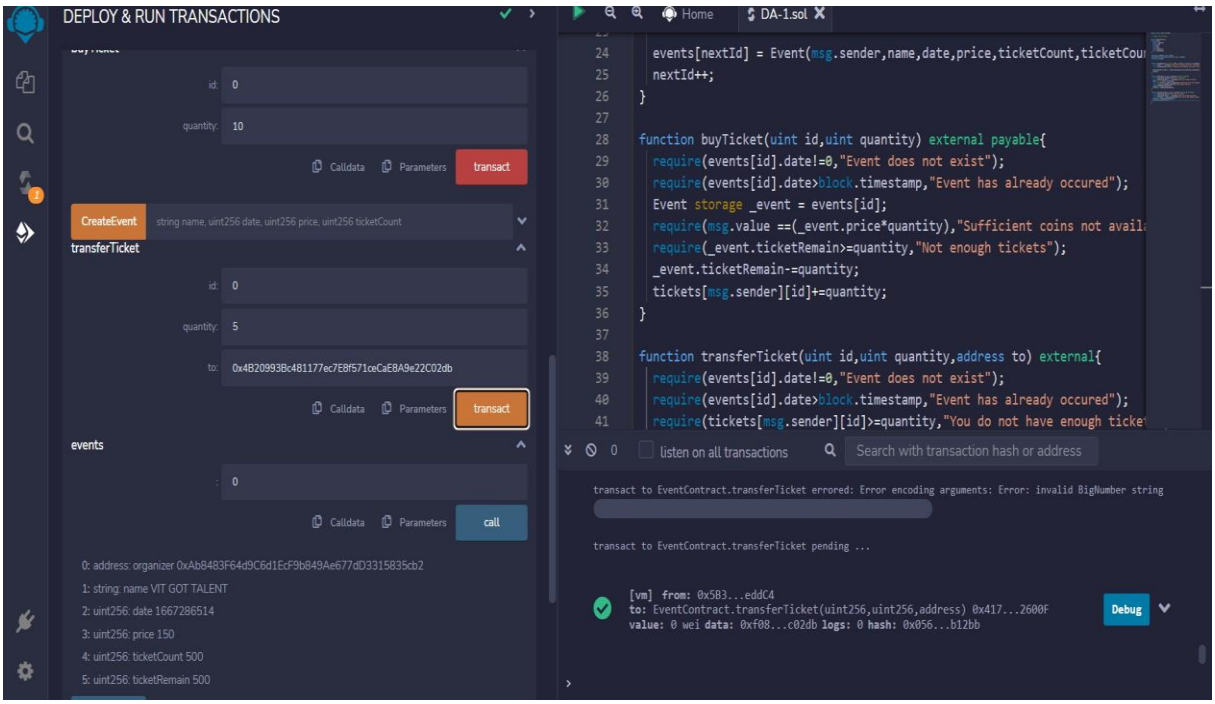
Events:



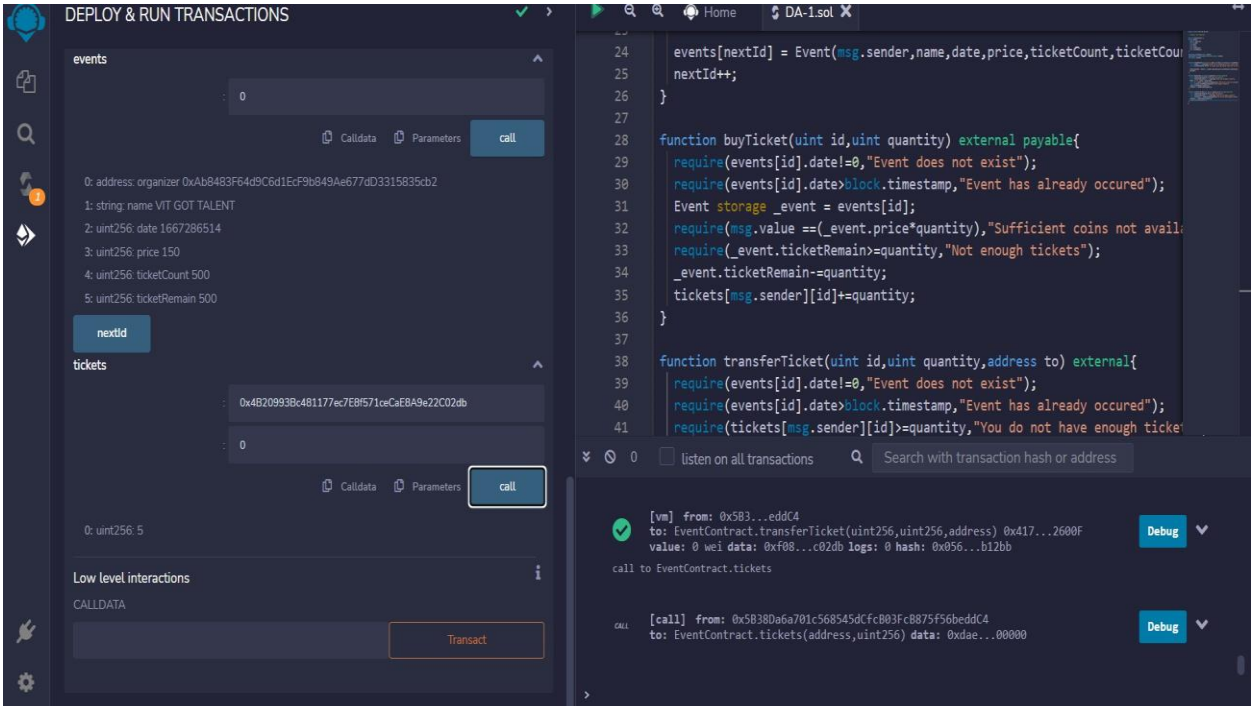
Buy Ticket:



Transfer Ticket:



Tickets:



PROJECT DESCRIPTION:

This is an event organizing project. In this project there are two entities: first the organizer and the second the attendee. The organizer will organize the event and attendee will be the person who will be attending the event by buying the ticket for the event. An organizer will be able to create multiple events and the attendee can also buy multiple tickets for the multiple events.

Functionalities:

- **Events** - Date, tickets, ticket price.
- **Organizer** - Creates an event.
- **Attendee** - Attendee buys tickets for the event.

EVENTS IN SOLIDITY:

Event is an inheritable member of a contract. An event is emitted, it stores the arguments passed in transaction logs. These logs are stored on blockchain and are accessible using address of the contract till the contract is present on the blockchain. An event generated is not accessible from within contracts, not even the one which have created and emitted them.

An event can be declared using event keyword.

```
//Declare an Event
event Deposit(address indexed _from, bytes32 indexed _id, uint
_value);

//Emit an event
emit Deposit(msg.sender, _id, msg.value);
```

Solidity events are crucial for smart contract developers because they allow smart contracts to index variables in order to rebuild the

storage stage, help to automatically update the user interface, and allow for testing of specific variables.

In Solidity, events are dispatched signals that smart contracts can fire. When you call events, they cause the arguments to be stored in the transaction's log, which is a special data structure in the blockchain. Events notify external users, such as a listening frontend website or client application, that something has happened on the blockchain.

DIFFERENCE BETWEEN EVENTS AND FUNCTIONS IN SOLIDITY:

While both functions and events accept arguments and can be called, functions modify smart contracts directly while events have the role of informing services outside of the blockchain to let users know that something has happened.

Functions in Solidity allow developers to read, write, change, and store data in the smart contract. You can pass arguments or parameters into a function. You can also call a function whenever it is needed in the code.

Events also accept arguments, but these are stored in the transaction's log, which is inaccessible to smart contracts. Contact data lives in the States trie and event data is stored in the Transaction Receipts trie, meaning contracts cannot read event data. Events are simply fired and forgotten.

Like functions, events can be called. However, the *emit* keyword is used to call/dispatch events. This allows developers to know when an event or a function is being called.

RELATION BETWEEN EVENTS AND LOGS IN SOLIDITY:

The Ethereum Virtual Machine (EVM) has a logging function that is used to write data, including Solidity events, to a structure outside smart contracts.

Logs and events are often referred to synonymously. Events allow you to 'print' information to the logging structure in a way that is more gas-efficient since the information is not stored in a storage variable, which takes up more gas. Events, or logs, live in the Transaction Receipts trie, which is inaccessible to smart contracts.

SOLIDITY EVENT TYPES:

There are two types of Solidity events: those which are indexed and those which are not.

When parameters do not have the indexed attribute, they are ABI-encoded into the data portion of the log. These parameters form the byte array of the event. Data is encoded according to its type and can be decoded according to a schema.

Indexed parameters are also known as "topics", are the searchable parameters in events. The indexed parameters will allow you to search for these events using the indexed parameters as filters. You can add an attribute indexed up to 4 parameters or 3 parameters based on whether the events are anonymous or not, respectively.

WORKING OF EVENTS IN SOLIDITY:

Solidity events are declared, emitted, and registered.

1. First, the event type has to be declared with the *event* keyword in Solidity.
2. Next, the event has to be emitted with the keyword emit.
3. Anytime something in the blockchain changes, your program will automatically register this change and trigger the event

EMITTING EVENTS IN SOLIDITY:

After the event is defined, you can trigger the event using the keyword *emit*. Once an event is emitted, the arguments passed are stored in transaction logs.

The following syntax shows you how to use emit in Solidity:

```
emit transfer(_from, _to, amount);
```

NEED OF EVENTS:

Events are used to inform external users that something happened on the blockchain. Smart contracts themselves cannot listen to any events.

All information in the blockchain is public and any actions can be found by looking into the transactions close enough but events are a shortcut to ease the development of outside systems in cooperation with smart contracts.