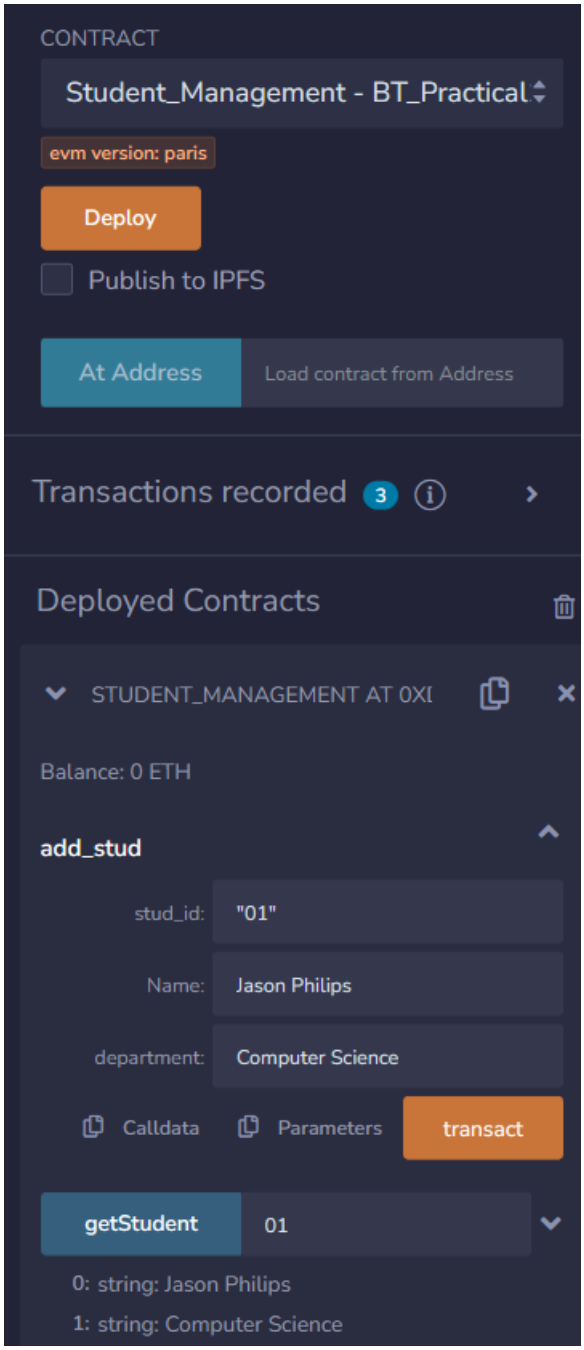
Blockchain Technology **Practical 4 : Program in solidity to create student data**

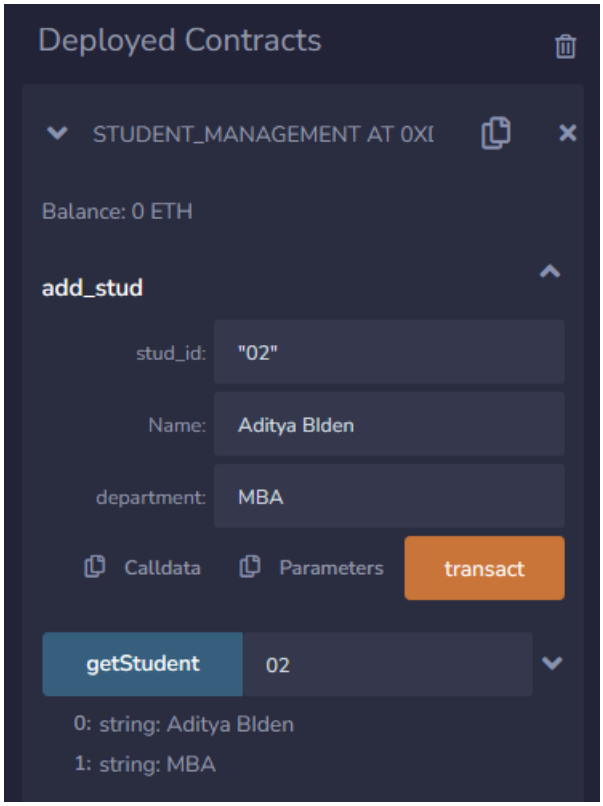
Program Code :

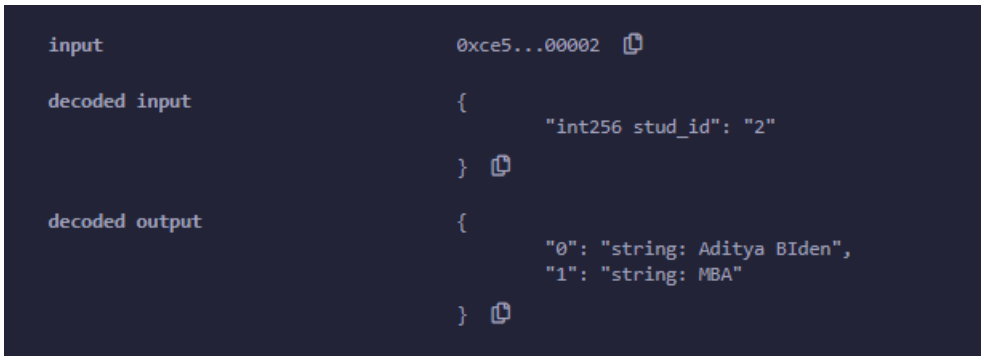
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.18;
contract Student_Management{
    struct Student{
        int stud_id;
        string name;
        string department;
    }
    Student[] Students;
    function add_stud(int stud_id,string memory Name, string memory department) public{
        Student memory stud = Student(stud_id,Name,department);
        Students.push(stud);
    }

    function getStudent(int stud_id) public view returns(string memory, string memory) {

        for (uint i= 0;i<Students.length;i++){
            Student memory stud=Students[i];
            if(stud.stud_id == stud_id){
                return (stud.name,stud.department);
            }
        }
        return ("Not Found","Not Found");
    }
}
```

Console Output:


CONTRACT
Student_Management - BT_Practical
evm version: paris
Deploy
Publish to IPFS
At Address
Load contract from Address
Transactions recorded 3
Deployed Contracts
STUDENT_MANAGEMENT AT 0XI
Balance: 0 ETH
add_stud
stud_id: "01"
Name: Jason Philips
department: Computer Science
Calldata
Parameters
transact
getStudent
01
0: string: Jason Philips
1: string: Computer Science


input
0xce5...00001
decoded input
{
"int256 stud_id": "1"
}
decoded output
{
"0": "string: Jason Philips",
"1": "string: Computer Science"
}


Deployed Contracts
STUDENT_MANAGEMENT AT 0XI
Balance: 0 ETH
add_stud
stud_id:
"02"
Name:
Aditya Blden
department:
MBA
Calldata
Parameters
transact
getStudent
02
0: string: Aditya Blden
1: string: MBA

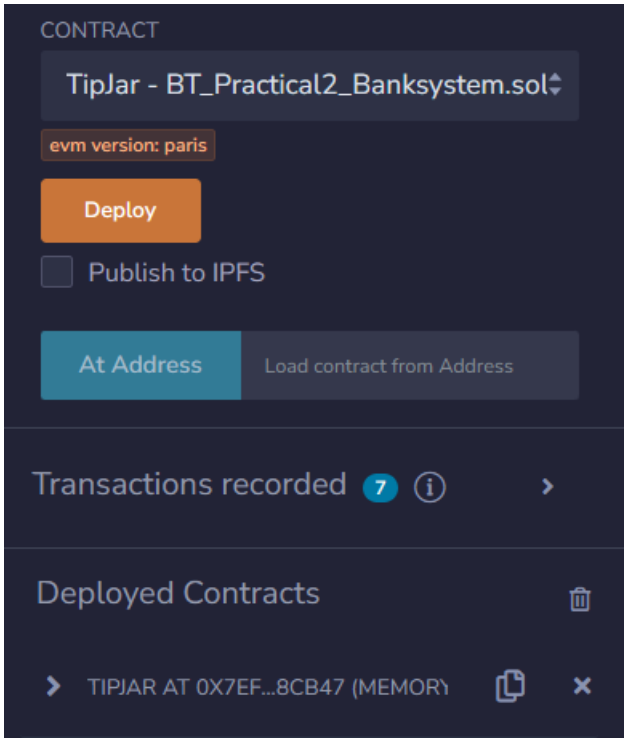

input
0xce5...00002
decoded input
{
"int256 stud_id": "2"
}
decoded output
{
"0": "string: Aditya BIden",
"1": "string: MBA"
}

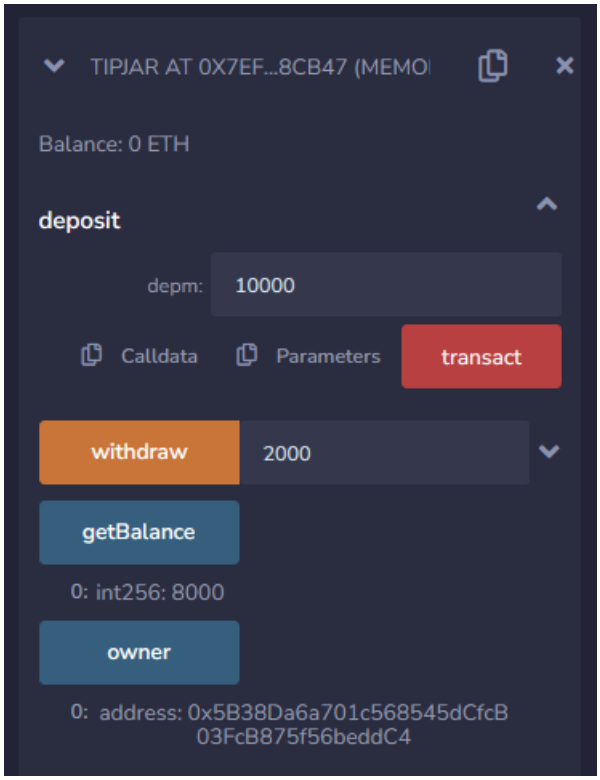
Practical 3 Smart Contact for Bank system :

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

contract TipJar {
    int depmoney;
    int withdraw_trans;
    int balance;
    address public owner; // Current owner of the contract

    constructor() {
        owner = msg.sender;
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "Only the owner can call this function");
        _;
    }

    function withdraw(int witm) public onlyOwner {
        withdraw_trans= witm;
        depmoney=depmoney-witm;
        payable(owner).transfer(address(this).balance);
    }

    function deposit(int depm) public payable {
       depmoney = depm;
```

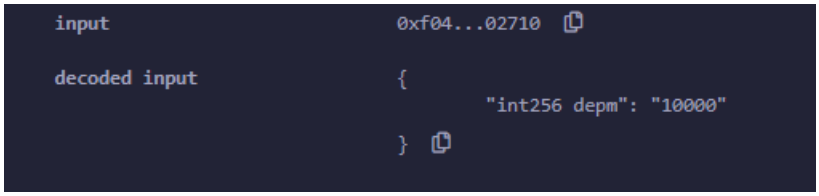
```
        // No need to specify an amount; the function should be called with the desired value.
    }

    function getBalance() public view returns (int256) {
        //balance = depmoney;
        return depmoney;
       //address(this).balance;
    }
}
```

Output Screenshots :

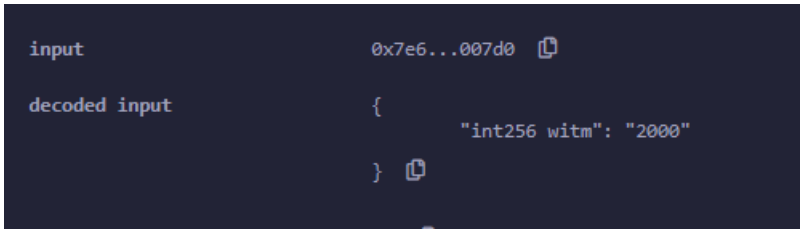




Deposit :



WithDraw :



Balance :