Experiment No.5
To design a smart contract using Solidity and Remix IDE
Date of Performance: 07/09/2023
Date of Submission: 07/09/2023

AIM: To design a smart contract using Solidity and Remix IDE

Objective: To develop a program in solidity to demonstrate smart contract in ethereum blockchain

Theory:

Smart contracts are simply programs stored on a blockchain that run when predetermined conditions are met. They typically are used to automate the execution of an agreement so that all participants can be immediately certain of the outcome, without any intermediary's involvement or time loss. They can also automate a workflow, triggering the next action when conditions are met.

Smart contracts work by following simple "if/when…then…" statements that are written into code on a blockchain. A network of computers executes the actions when predetermined conditions have been met and verified. These actions could include releasing funds to the appropriate parties, registering a vehicle, sending notifications, or issuing a ticket. The blockchain is then updated when the transaction is completed. That means the transaction cannot be changed, and only parties who have been granted permission can see the results.

Within a smart contract, there can be as many stipulations as needed to satisfy the participants that the task will be completed satisfactorily. To establish the terms, participants must determine how transactions and their data are represented on the blockchain, agree on the "if/when...then…" rules that govern those transactions, explore all possible exceptions, and define a framework for resolving disputes.

Then the smart contract can be programmed by a developer – although increasingly, organizations that use blockchain for business provide templates, web interfaces, and other online tools to simplify structuring smart contracts.

Solidity for smart contracts

Solidity is an object-oriented programming language created specifically by the Ethereum Network team for constructing and designing smart contracts on Blockchain platforms.

It's used to create smart contracts that implement business logic and generate a chain of transaction records in the blockchain system.

It acts as a tool for creating machine-level code and compiling it on the Ethereum Virtual Machine (EVM).

It has a lot of similarities with C and C++ and is pretty simple to learn and understand. For example, a "main" in C is equivalent to a "contract" in Solidity.

Like other programming languages, Solidity programming also has variables, functions, classes, arithmetic operations, string manipulation, and many other concepts.

- The Ethereum Virtual Machine (EVM) provides a runtime environment for Ethereum smart contracts.

- It is primarily concerned with ensuring the security and execution of untrusted programs through the use of an international network of public nodes.

- EVM is specialized in preventing Denial-of-Service attacks and certifies that the programs do not have access to each other's state, as well as establishing communication, with no possible interference.

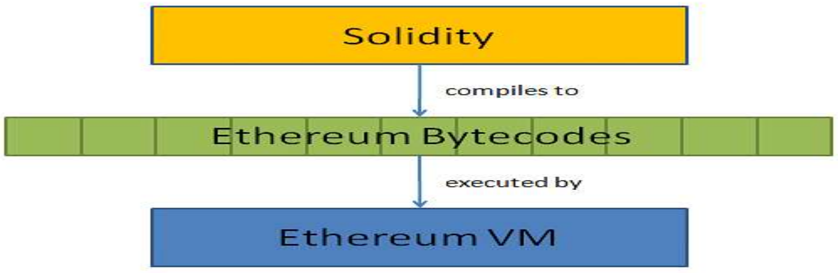


Fig.5.1 EVM in Ethereum Blockchain

- Smart contracts refer to high-level program codes compiled into EVM before being posted to the Ethereum blockchain for execution.

- It enables you to conduct trustworthy transactions without the involvement of a third party; these transactions are traceable and irreversible.

Process:

Step 1. Open Remix **IDE** by typing URL https://remix.ethereum.org/.

Step 2. In Remix IDE select 'Solidity' plugins

Step 3. Click on File Explorer

Step 4. In the default workspace click on 'create new file'

Step 5. Give suitable name to the *file* with extension .sol

Step 6. Type the (smart contract) code in the editor section of the Remix IDE for newly created file (.sol)

Step 7. After typing the code for the smart contract in the newly creates .sol file,click on the compiler option available and then compile the file

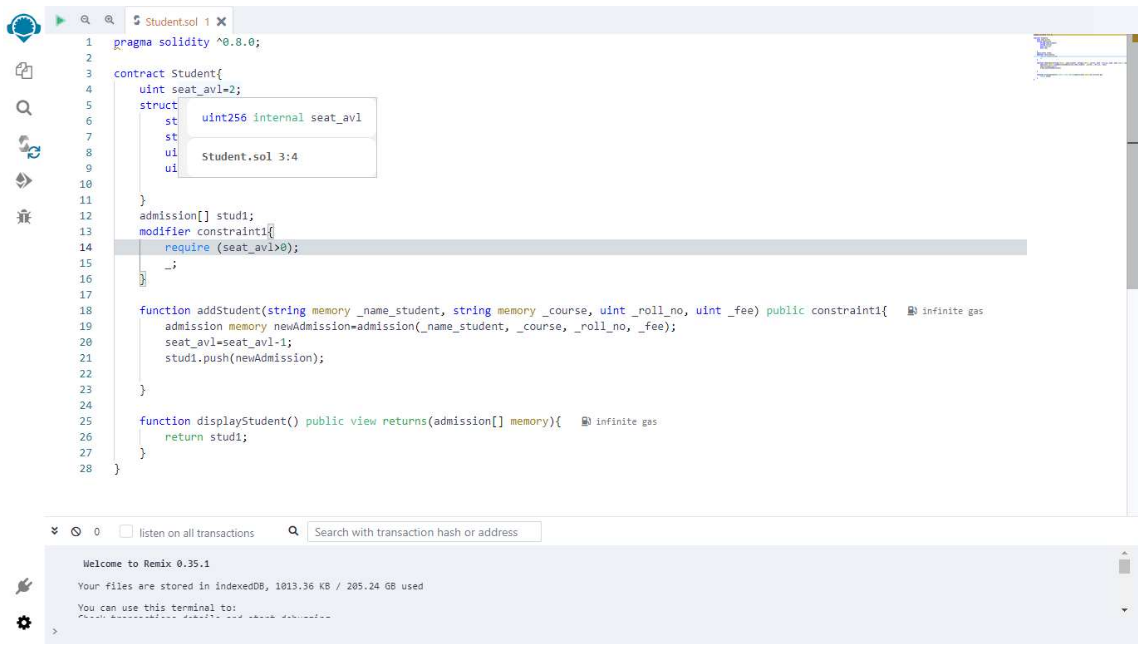
Step 8. If no error, then click on the 'Deploy and Run' to execute the smart contract

Code:

```
pragma solidity ^0.8.0;



contract Student{

   uint seat_avl=2;

   struct admission{

      string name_student;

      string course;

      uint roll_no;

      uint fee;



   }

   admission[] stud1;

   modifier constraint1{

      require (seat_avl>0);
```

```
        _;
    }

     function addStudent(string memory _name_student, string memory _course, uint _roll_no,
uint _fee) public constraint1{
        admission memory newAdmission=admission(_name_student, _course, _roll_no, _fee);
        seat_avl=seat_avl-1;
        stud1.push(newAdmission);

    }

    function displayStudent() public view returns(admission[] memory){
        return stud1;
    }
}
```

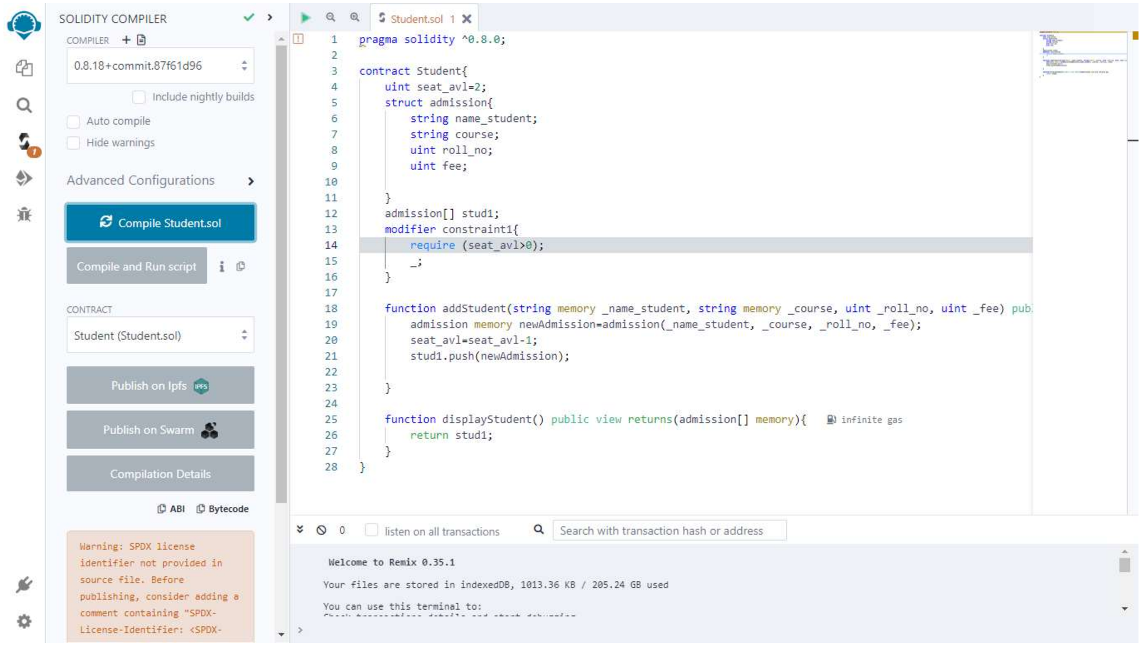
Output:



Student.sol 1

```
pragma solidity ^0.8.0;

contract Student{
    uint seat_avl=2;
    struct
        st      uint256 internal seat_avl
        st
        ui      Student.sol 3:4
        ui

    }
    admission[] stud1;
    modifier constraint1{
        require (seat_avl>0);
        _;
    }

    function addStudent(string memory _name_student, string memory _course, uint _roll_no, uint _fee) public constraint1{    infinite gas
        admission memory newAdmission=admission(_name_student, _course, _roll_no, _fee);
        seat_avl=seat_avl-1;
        stud1.push(newAdmission);

    }

    function displayStudent() public view returns(admission[] memory){    infinite gas
        return stud1;
    }
}
```

```
0    listen on all transactions    Search with transaction hash or address

Welcome to Remix 0.35.1

Your files are stored in indexedDB, 1013.36 KB / 205.24 GB used

You can use this terminal to:
```



SOLIDITY COMPILER

COMPILER

0.8.18+commit.87f61d96

Include nightly builds

Auto compile

Hide warnings

Advanced Configurations

Compile Student.sol

Compile and Run script

CONTRACT

Student (Student.sol)

Publish on Ipfs

Publish on Swarm

Compilation Details

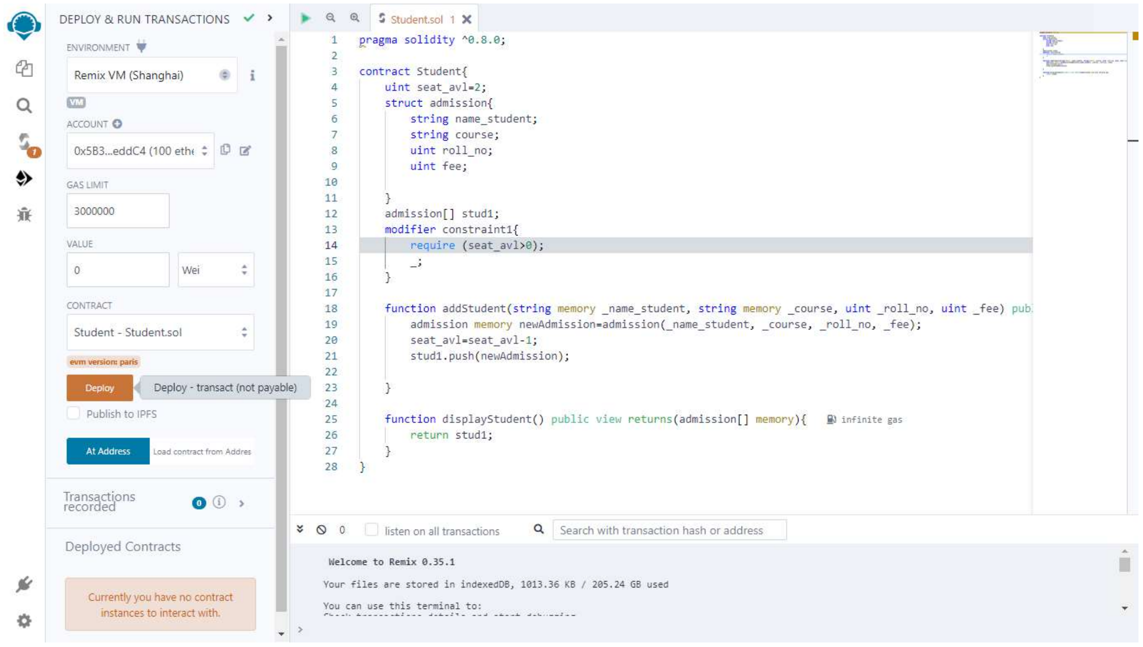
ABI Bytecode

```
Warning: SPDX license
identifier not provided in
source file. Before
publishing, consider adding a
comment containing "SPDX-
License-Identifier: <SPDX-
```

Student.sol 1

```
pragma solidity ^0.8.0;

contract Student{
    uint seat_avl=2;
    struct admission{
        string name_student;
        string course;
        uint roll_no;
        uint fee;

    }
    admission[] stud1;
    modifier constraint1{
        require (seat_avl>0);
        _;
    }

    function addStudent(string memory _name_student, string memory _course, uint _roll_no, uint _fee) pub
        admission memory newAdmission=admission(_name_student, _course, _roll_no, _fee);
        seat_avl=seat_avl-1;
        stud1.push(newAdmission);

    }

    function displayStudent() public view returns(admission[] memory){    infinite gas
        return stud1;
    }
}
```

```
0    listen on all transactions    Search with transaction hash or address

Welcome to Remix 0.35.1

Your files are stored in indexedDB, 1013.36 KB / 205.24 GB used

You can use this terminal to:
```



DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Remix VM (Shanghai)
ACCOUNT: 0x5B3...eddC4 (100 ethe
GAS LIMIT: 3000000
VALUE: 0 Wei
CONTRACT: Student - Student.sol
evm version: paris

Deploy — Deploy - transact (not payable)

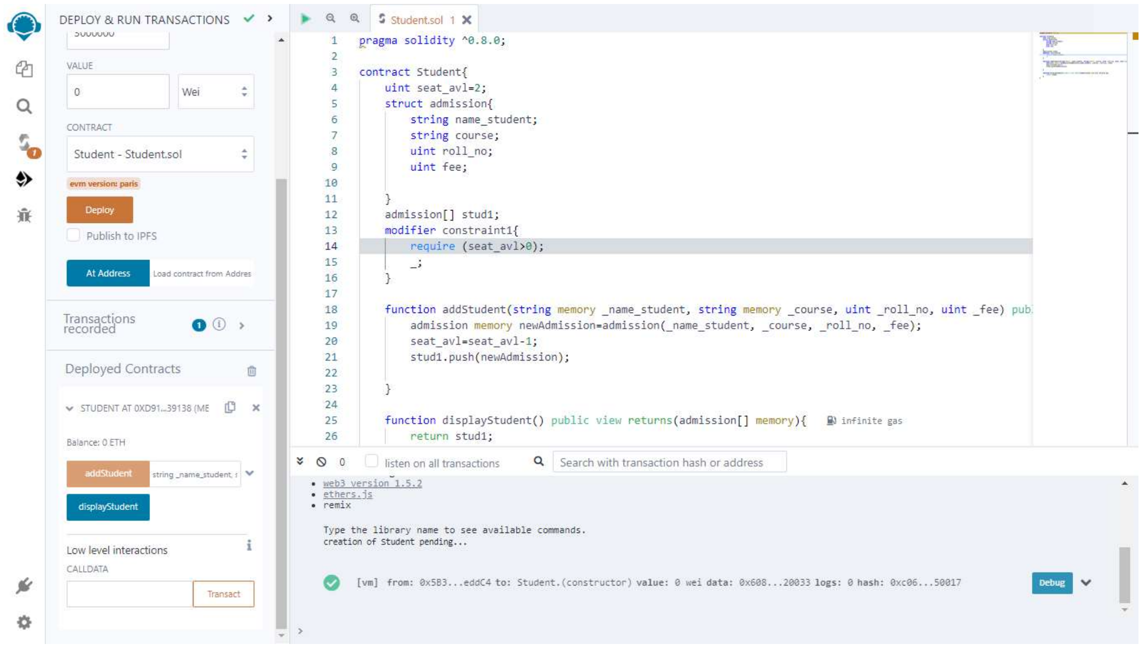
Transactions recorded: 0

Deployed Contracts: Currently you have no contract instances to interact with.

Student.sol

```
pragma solidity ^0.8.0;

contract Student{
    uint seat_avl=2;
    struct admission{
        string name_student;
        string course;
        uint roll_no;
        uint fee;

    }
    admission[] stud1;
    modifier constraint1{
        require (seat_avl>0);
        _;
    }

    function addStudent(string memory _name_student, string memory _course, uint _roll_no, uint _fee) pub
        admission memory newAdmission=admission(_name_student, _course, _roll_no, _fee);
        seat_avl=seat_avl-1;
        stud1.push(newAdmission);

    }

    function displayStudent() public view returns(admission[] memory){
        return stud1;
    }
}
```

```
Welcome to Remix 0.35.1

Your files are stored in indexedDB, 1013.36 KB / 205.24 GB used

You can use this terminal to:
```



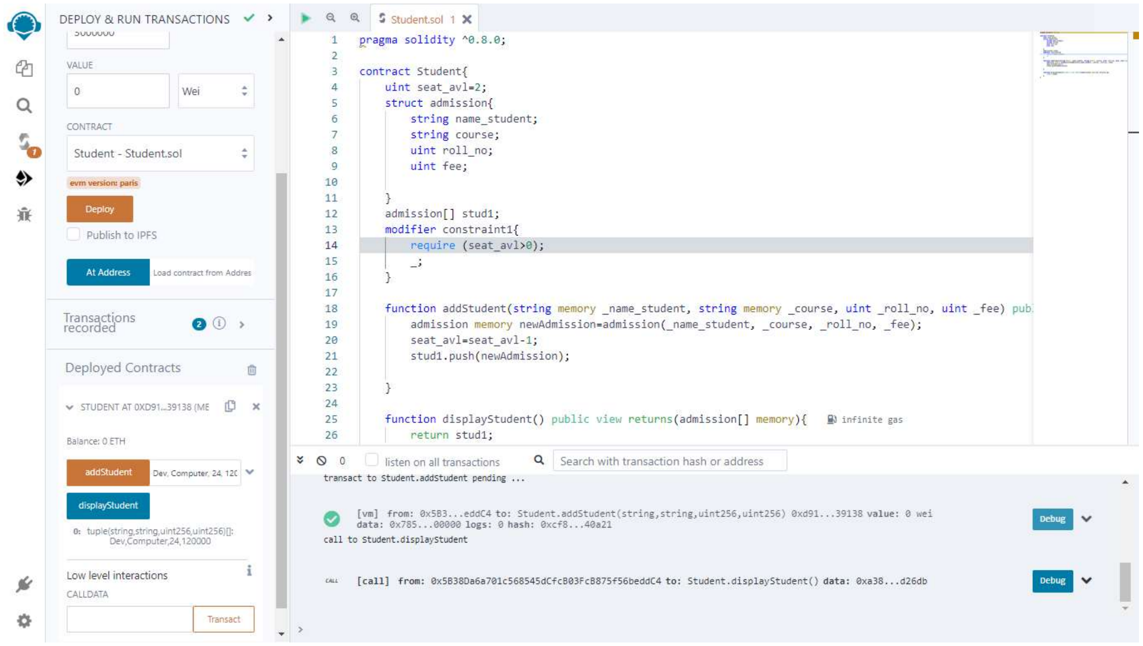
VALUE: 0 Wei
CONTRACT: Student - Student.sol
evm version: paris

Transactions recorded: 1

Deployed Contracts: STUDENT AT 0XD91...39138 (ME
Balance: 0 ETH
addStudent — string _name_student, :
displayStudent

Low level interactions
CALLDATA — Transact

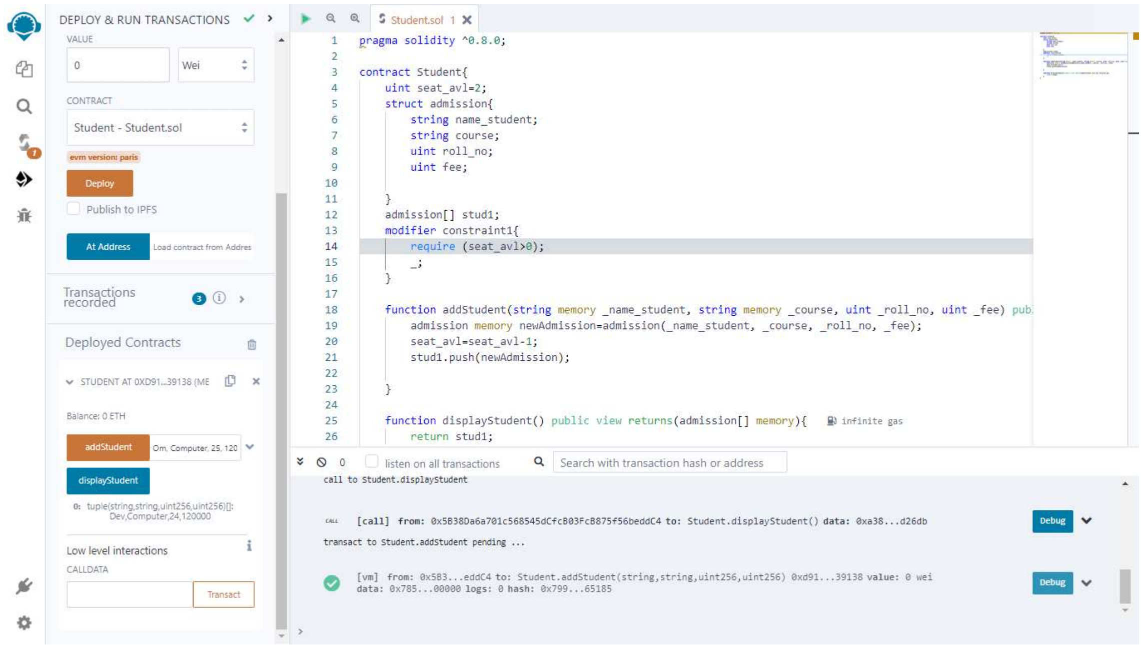
```
• web3 version 1.5.2
• ethers.js
• remix

Type the library name to see available commands.
creation of Student pending...

[vm] from: 0x5B3...eddC4 to: Student.(constructor) value: 0 wei data: 0x608...20033 logs: 0 hash: 0xc06...50017
```



```
pragma solidity ^0.8.0;

contract Student{
    uint seat_avl=2;
    struct admission{
        string name_student;
        string course;
        uint roll_no;
        uint fee;

    }
    admission[] stud1;
    modifier constraint1{
        require (seat_avl>0);
        _;
    }

    function addStudent(string memory _name_student, string memory _course, uint _roll_no, uint _fee) pub
        admission memory newAdmission=admission(_name_student, _course, _roll_no, _fee);
        seat_avl=seat_avl-1;
        stud1.push(newAdmission);

    }

    function displayStudent() public view returns(admission[] memory){
        return stud1;
```

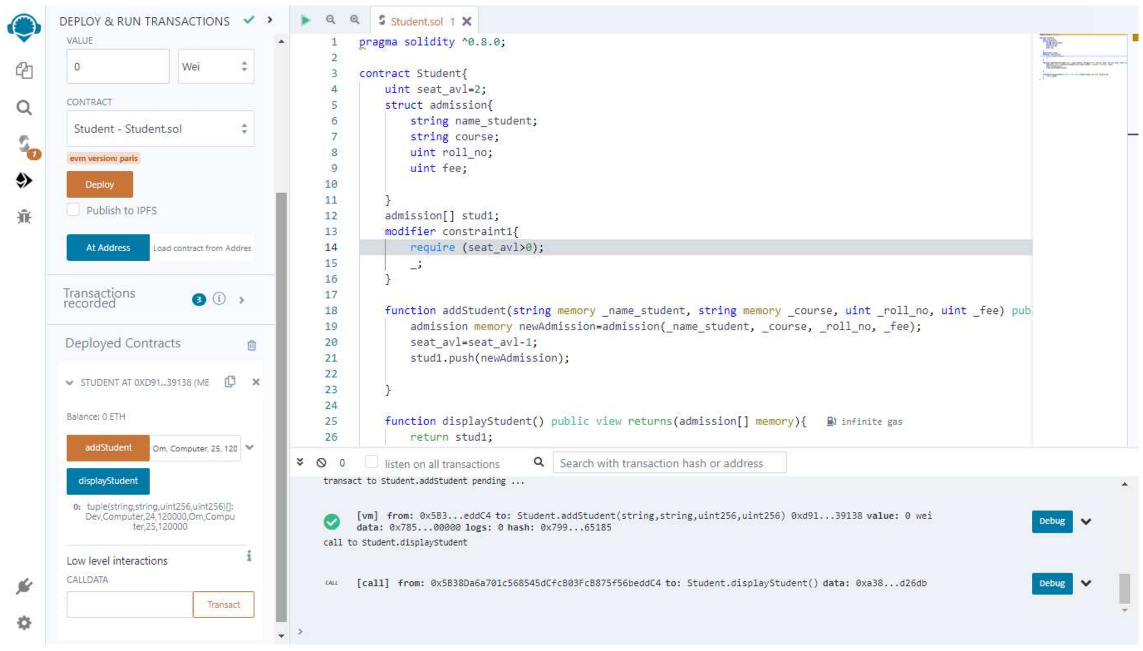
```
transact to Student.addStudent pending ...

[vm] from: 0x5B3...eddC4 to: Student.addStudent(string,string,uint256,uint256) 0xd91...39138 value: 0 wei
data: 0x785...00000 logs: 0 hash: 0xcf8...40a21
call to Student.displayStudent

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Student.displayStudent() data: 0xa38...d26db
```

```
0: tuple(string,string,uint256,uint256)[]:
   Dev,Computer,24,120000
```



```
call to Student.displayStudent

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Student.displayStudent() data: 0xa38...d26db
transact to Student.addStudent pending ...

[vm] from: 0x5B3...eddC4 to: Student.addStudent(string,string,uint256,uint256) 0xd91...39138 value: 0 wei
data: 0x785...00000 logs: 0 hash: 0x799...65185
```

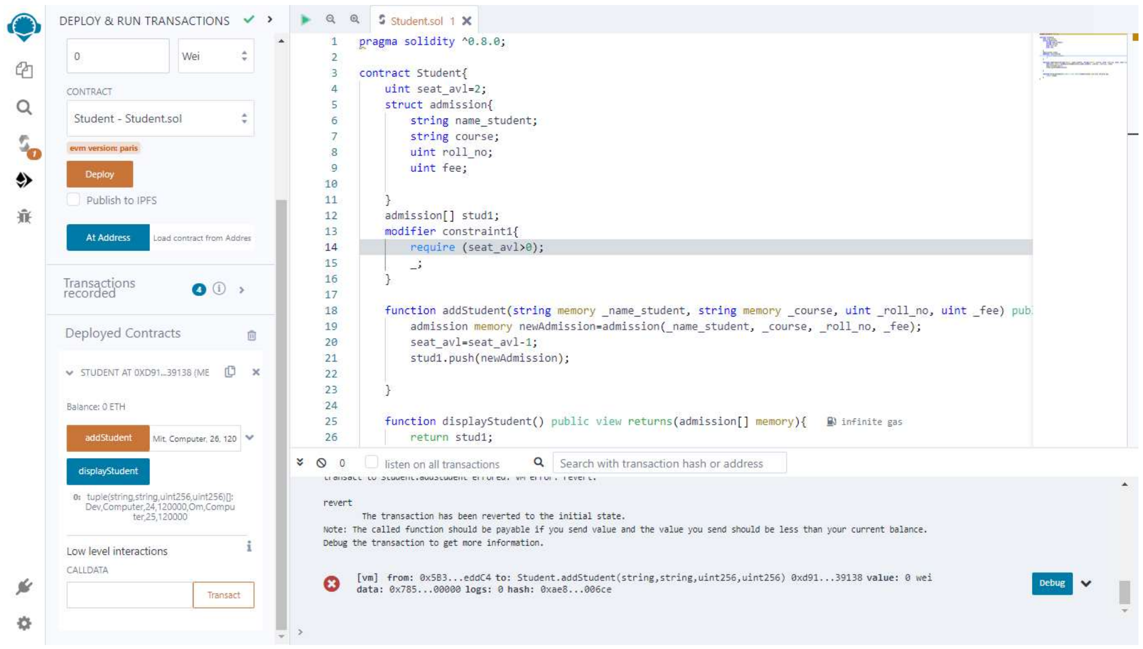
```
0: tuple(string,string,uint256,uint256)[]:
   Dev,Computer,24,120000
```



```
0: tuple(string,string,uint256,uint256)[]:
Dev,Computer,24,120000,Om,Compu
ter,25,120000
```

```
pragma solidity ^0.8.0;

contract Student{
    uint seat_avl=2;
    struct admission{
        string name_student;
        string course;
        uint roll_no;
        uint fee;

    }
    admission[] stud1;
    modifier constraint1{
        require (seat_avl>0);
        _;
    }

    function addStudent(string memory _name_student, string memory _course, uint _roll_no, uint _fee) pub
        admission memory newAdmission=admission(_name_student, _course, _roll_no, _fee);
        seat_avl=seat_avl-1;
        stud1.push(newAdmission);

    }

    function displayStudent() public view returns(admission[] memory){    infinite gas
        return stud1;
```

```
transact to Student.addStudent pending ...

[vm] from: 0x5B3...eddC4 to: Student.addStudent(string,string,uint256,uint256) 0xd91...39138 value: 0 wei
data: 0x785...00000 logs: 0 hash: 0x799...65185
call to Student.displayStudent

CALL  [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Student.displayStudent() data: 0xa38...d26db
```



```
0: tuple(string,string,uint256,uint256)[]:
Dev,Computer,24,120000,Om,Compu
ter,25,120000
```

```
pragma solidity ^0.8.0;

contract Student{
    uint seat_avl=2;
    struct admission{
        string name_student;
        string course;
        uint roll_no;
        uint fee;

    }
    admission[] stud1;
    modifier constraint1{
        require (seat_avl>0);
        _;
    }

    function addStudent(string memory _name_student, string memory _course, uint _roll_no, uint _fee) pub
        admission memory newAdmission=admission(_name_student, _course, _roll_no, _fee);
        seat_avl=seat_avl-1;
        stud1.push(newAdmission);

    }

    function displayStudent() public view returns(admission[] memory){    infinite gas
        return stud1;
```

```
transact to Student.addStudent errored: VM error: revert.

revert
        The transaction has been reverted to the initial state.
Note: The called function should be payable if you send value and the value you send should be less than your current balance.
Debug the transaction to get more information.

[vm] from: 0x5B3...eddC4 to: Student.addStudent(string,string,uint256,uint256) 0xd91...39138 value: 0 wei
data: 0x785...00000 logs: 0 hash: 0xae8...006ce
```

Conclusion:

1. Designed for Ethereum: Solidity is specifically designed for Ethereum, making it a natural choice for building smart contracts on this blockchain. It is Ethereum's native language, and its compatibility ensures smooth integration with the Ethereum Virtual Machine (EVM).

2. Safety and Security: Solidity includes features like static analysis tools and automated testing frameworks that help developers write secure smart contracts. It has been used extensively, and its security patterns are well-understood, reducing the risk of vulnerabilities.

3. Community and Documentation: Solidity has a large and active developer community, which means there are abundant resources, tutorials, and documentation available for learning and troubleshooting. This support network makes it easier for developers to create reliable smart contracts.

4. EVM Compatibility: Solidity is designed to compile into EVM bytecode, which is the execution code that runs on the Ethereum blockchain. This compatibility ensures that smart contracts written in Solidity can be executed on the Ethereum network without compatibility issues.

5. Widely Adopted: Solidity is one of the most widely adopted programming languages for Ethereum smart contracts. This widespread usage means that many existing projects and decentralized applications (DApps) are written in Solidity, creating interoperability and integration opportunities.