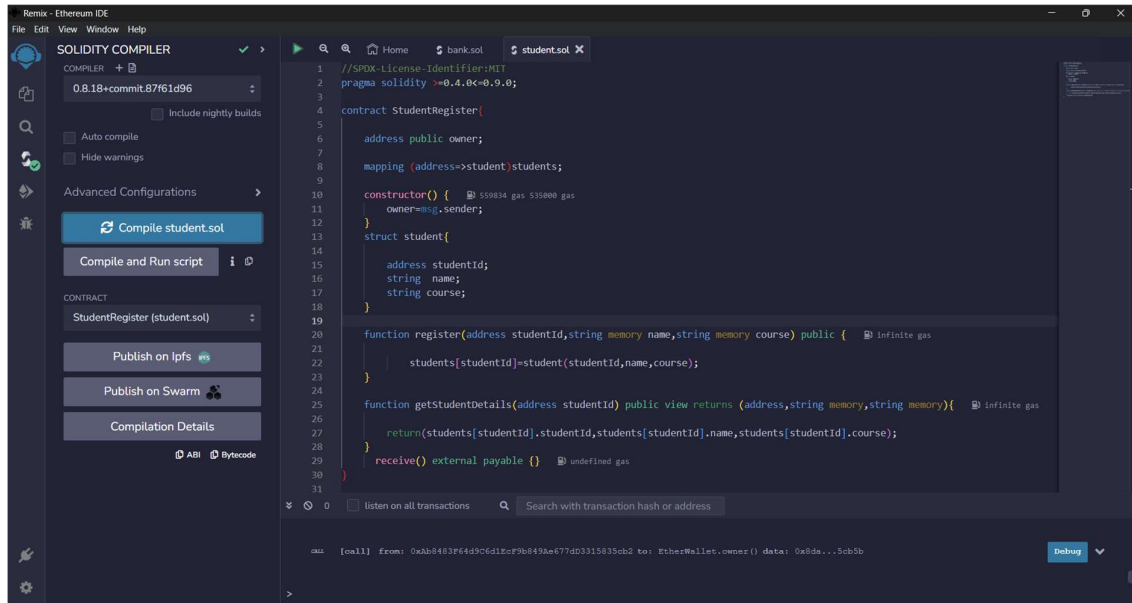


**Code:**

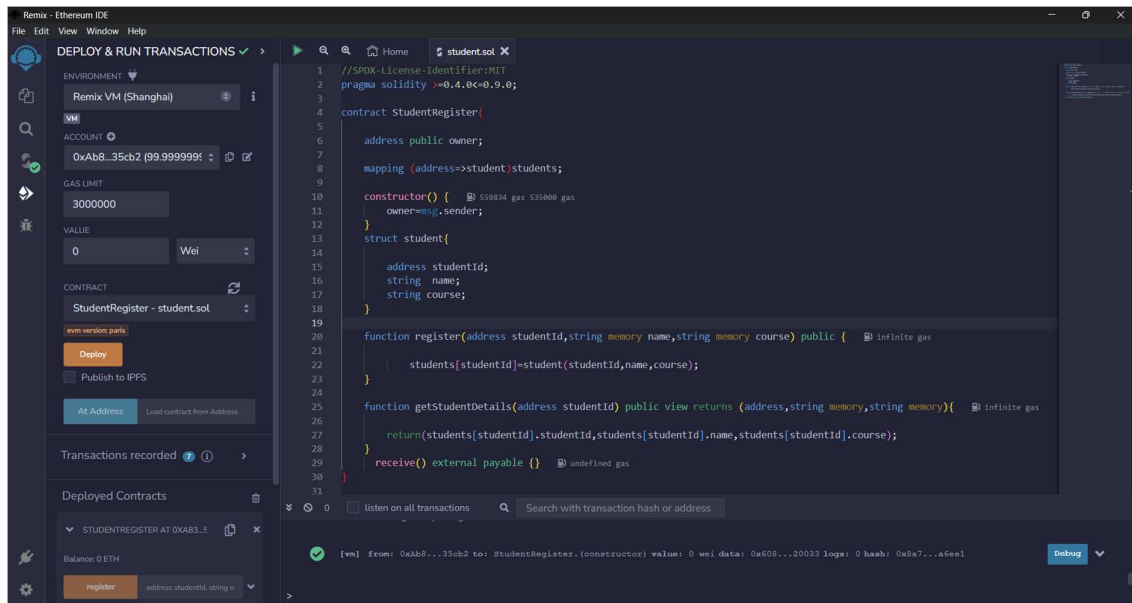
```
//SPDX-License-Identifier:MIT
pragma solidity >=0.4.0<=0.9.0;
contract StudentRegister{
    address public owner;
    mapping (address=>student)students;
    constructor() {
        owner=msg.sender;
    }
    struct student{
        address studentId;
        string name;
        string course;
    }
    function register(address studentId,string memory name,string memory course) public {
        students[studentId]=student(studentId,name,course);
    }
    function getStudentDetails(address studentId) public view returns (address,string
memory,string memory){
    return(students[studentId].studentId,students[studentId].name,students[studentId].course);
    }
    receive() external payable {}
}
```

## Output:

### Step 1: Compile



### Step 2: Deploy



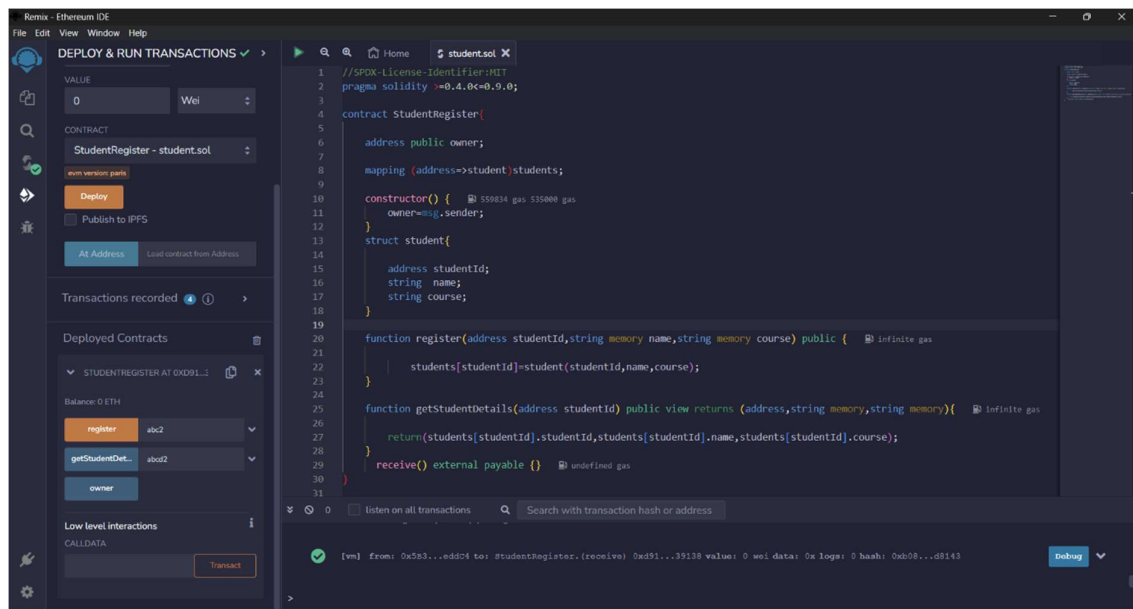
### Step 3: Add the student details and click on the Register (We are adding 3 student details)

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is active. The 'CONTRACT' dropdown shows 'StudentRegister - student.sol'. The 'VALUE' field is set to '0' and the 'ACCOUNT' is 'Wei'. The 'Deploy' button is highlighted. Below it, the 'Transactions recorded' section shows a list of transactions. The 'Deployed Contracts' section shows the 'STUDENTREGISTER AT 0xD91...' contract with a balance of '0 ETH'. The 'register' button is highlighted. The 'Low level interactions' section shows the 'CALLDATA' field. The main editor displays the Solidity code for the 'StudentRegister' contract. The code includes a constructor, a 'register' function, and a 'getStudentDetails' function. The 'register' function is being executed, and the console shows the transaction details: '[vm] from: 0x583...add04 to: StudentRegister, (receive) 0xd91...39138 value: 0 wei data: 0x logs: 0 hashes: 0xfec...1e499'.

```
1 //SPDX-License-Identifier:MIT
2 pragma solidity ^0.4.0<0.9.0;
3
4 contract StudentRegister{
5
6     address public owner;
7
8     mapping (address=>student) students;
9
10    constructor() { @ 559834 gas 535000 gas
11        owner=msg.sender;
12    }
13    struct student{
14
15        address studentId;
16        string name;
17        string course;
18    }
19
20    function register(address studentId,string memory name,string memory course) public { @ Infinite gas
21
22        students[studentId]=student(studentId,name,course);
23    }
24
25    function getStudentDetails(address studentId) public view returns (address,string memory,string memory){ @ Infinite gas
26
27        return (students[studentId].studentId,students[studentId].name,students[studentId].course);
28    }
29    receive() external payable {} @ undefined gas
30 }
31
```

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is active. The 'CONTRACT' dropdown shows 'StudentRegister - student.sol'. The 'VALUE' field is set to '0' and the 'ACCOUNT' is 'Wei'. The 'Deploy' button is highlighted. Below it, the 'Transactions recorded' section shows a list of transactions. The 'Deployed Contracts' section shows the 'STUDENTREGISTER AT 0xD91...' contract with a balance of '0 ETH'. The 'register' button is highlighted. The 'Low level interactions' section shows the 'CALLDATA' field. The main editor displays the Solidity code for the 'StudentRegister' contract. The code includes a constructor, a 'register' function, and a 'getStudentDetails' function. The 'register' function is being executed, and the console shows the transaction details: '[vm] from: 0x583...add04 to: StudentRegister, (receive) 0xd91...39138 value: 0 wei data: 0x logs: 0 hashes: 0xf09...a5ac6'.

```
1 //SPDX-License-Identifier:MIT
2 pragma solidity ^0.4.0<0.9.0;
3
4 contract StudentRegister{
5
6     address public owner;
7
8     mapping (address=>student) students;
9
10    constructor() { @ 559834 gas 535000 gas
11        owner=msg.sender;
12    }
13    struct student{
14
15        address studentId;
16        string name;
17        string course;
18    }
19
20    function register(address studentId,string memory name,string memory course) public { @ Infinite gas
21
22        students[studentId]=student(studentId,name,course);
23    }
24
25    function getStudentDetails(address studentId) public view returns (address,string memory,string memory){ @ Infinite gas
26
27        return (students[studentId].studentId,students[studentId].name,students[studentId].course);
28    }
29    receive() external payable {} @ undefined gas
30 }
31
```



#### Step 4: Click on the owner button to check who is the owner

