Name: Hanuman Bavane
Roll No: 14108
Class: BE – A – A1

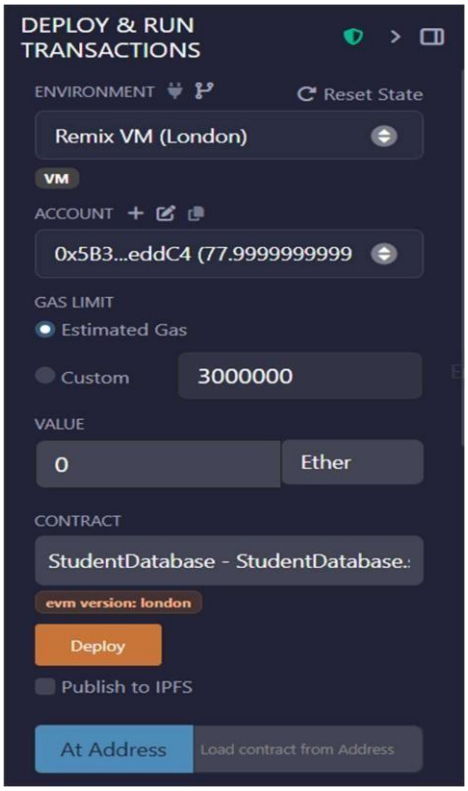
Practical 4

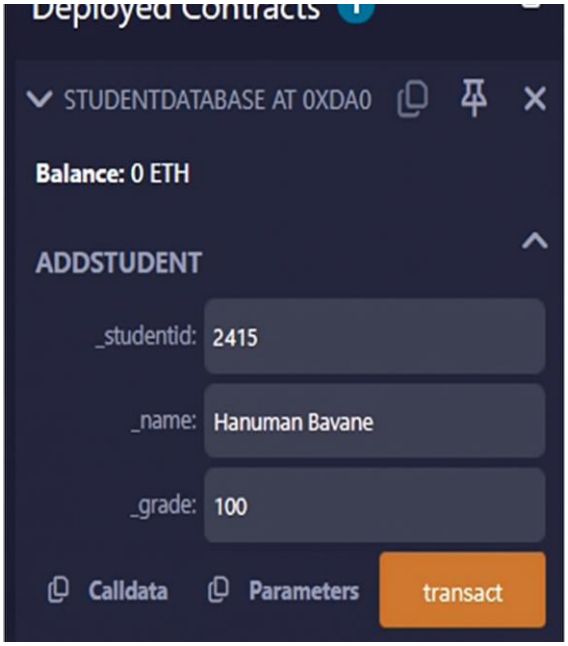
Program:

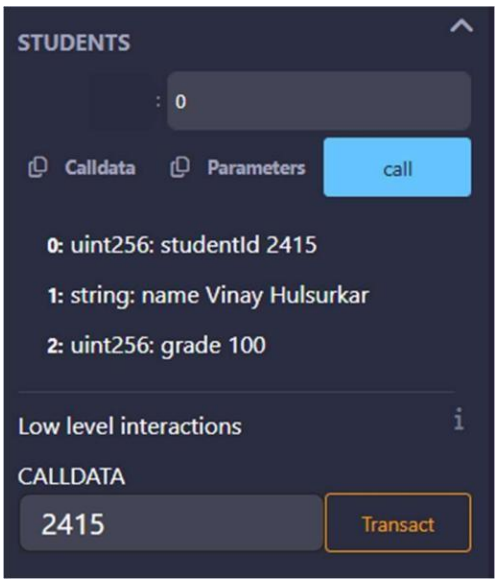
```
// SPDX-License-Identifier: MIT pragma
solidity ^0.8.20;

/**
* @title StudentDatabase
* @dev This contract stores and manages student records.
* It demonstrates the use of structs, arrays, and fallback functions.
 */
contract StudentDatabase {

    /**
* @dev Defines the structure for a Student.
* Each student has a unique ID, a name, and a grade.
     */
    struct  Student  {
uint         studentId;
string name;
        uint grade;
    }

    /**
* @dev An array to store all the student records.
* The 'public' keyword automatically creates a getter function for it.
     */
    Student[] public students;

    /**
* @dev Adds a new student to the `students` array.
* @param _studentId The unique identifier for the student.
* @param _name The name of the student.
* @param _grade The grade of the student.
     */
    function addStudent(uint _studentId, string memory _name, uint _grade) public {     //
Creates a new Student struct in memory and adds it to the storage array.
        students.push(Student(_studentId, _name, _grade));
    }

    /**
* @dev A receive function is the modern way to handle plain Ether transfers.
* This function is executed when the contract receives Ether without any data.
     */
    receive() external payable {
        // This function is intentionally left simple.
        // In a real-world scenario, you might emit an event or log the deposit.
```

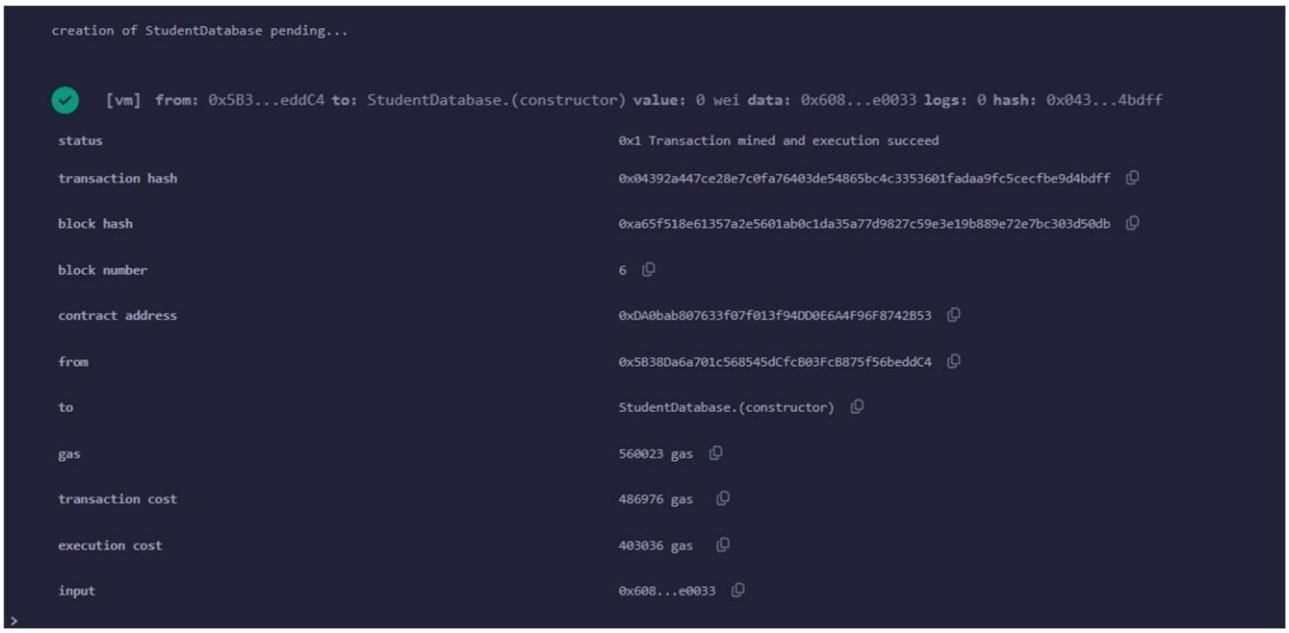
```
    }

    /**
* @dev A fallback function is executed when a function that doesn't exist
* is called on the contract, or if Ether is sent with some data but        * no other function
  matches.
     */
    fallback() external payable {
        // This function is also kept simple for demonstration.
    }
}
```

Output:

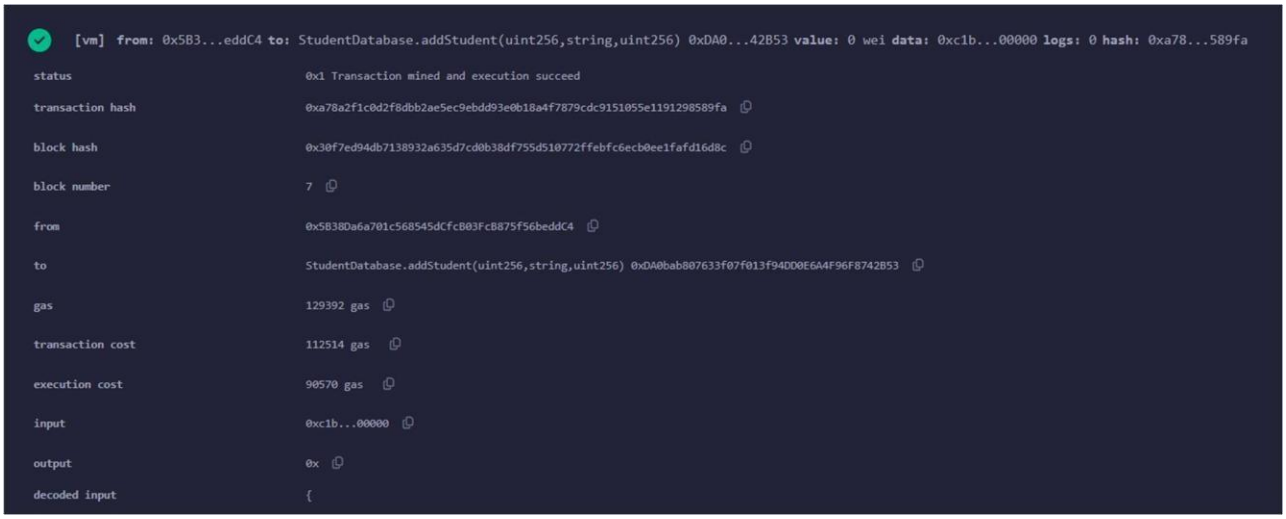




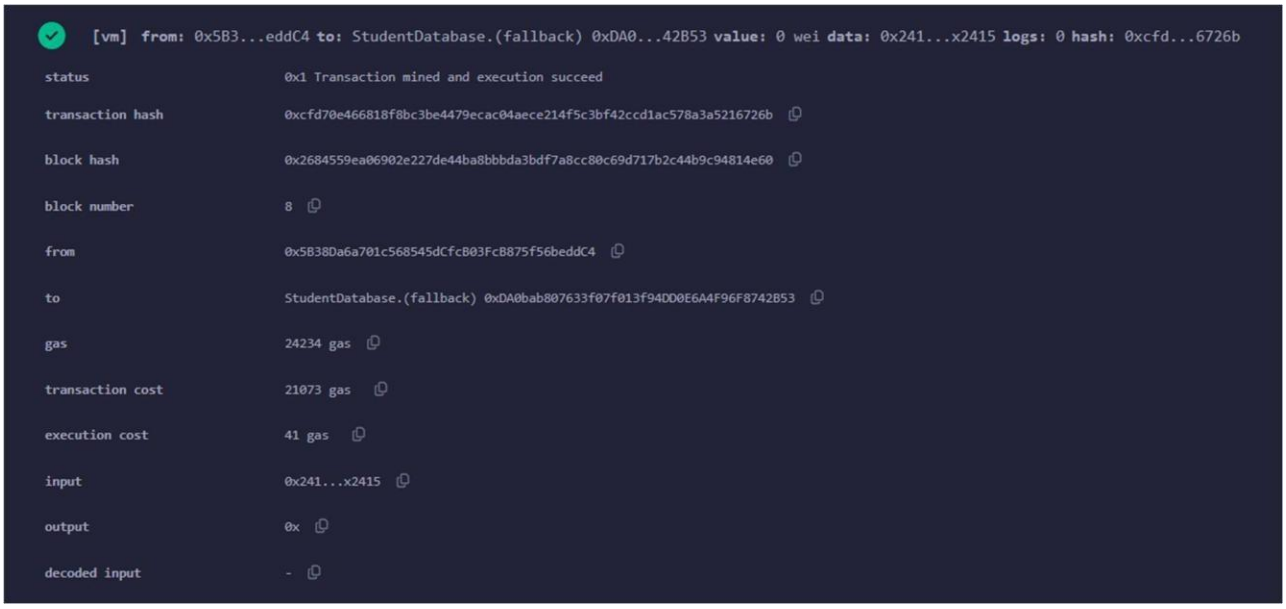




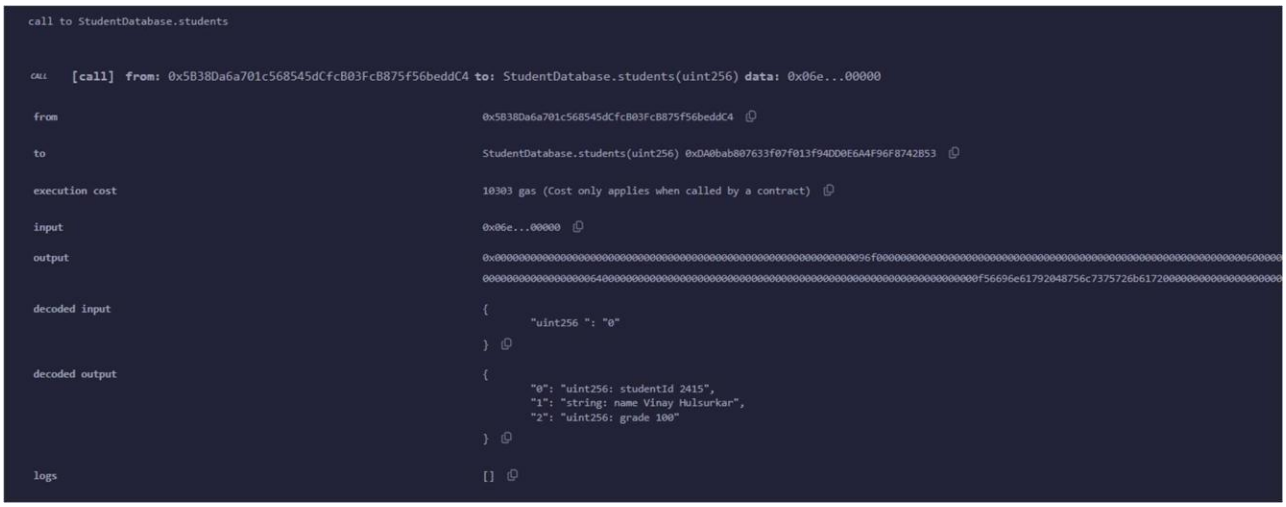
Creation Of DB



Add Student



Fallback



View Database