**Name**: Raj K Patel

**Roll No**: 20BCE218

**Course**: 2CSDE93 - Blockchain Technology

**Practical No**: 5

**Aim**: To perform thorough study and installation of Remix IDE and Truffle IDE for deploying Smart Contracts and Decentralized Applications (dapps) and create and deploy a Smart Contract for any application such as finance, healthcare etc.

**Code:**

***// SPDX-License-Identifier: MIT***

pragma solidity ^0.8.0;

contract IdentityManagement { address public government;

struct Student { uint256 Id; string FirstName; string LastName; uint8 Percentage;

bool IsEligibleForScholarship; address Address;

**}**

struct College { uint256 Id; uint Fees; string Name; string Location; address Address;

**}**

struct Scholarship { uint256 Id; uint256 StudentId; uint256 CollegeId;

string ScholarshipName; uint256 Amount; address payable To; string Status;

**}**

mapping(uint256 => Student) internal studentRecords; Student[] internal students;

mapping(uint256 => College) internal collegeRecords; College[] internal colleges;

mapping(uint256 => Scholarship) internal scholarshipRecords; Scholarship[] internal scholarships;

struct StudentFeesPaid {

mapping(uint256 => uint) studentFeesPaid;

**}**

mapping(uint => StudentFeesPaid) college\_student\_fees\_paid;

modifier onlyGovernment() { require(

msg.sender == government,

"Only government can perform this action"

**);**

**\_;**

**}**

constructor() {

government = msg.sender;

**}**

function enrollCollege( uint256 \_id, uint256 \_fees, string memory \_name,

string memory \_location, address payable \_address

) public onlyGovernment {

require(\_address != address(0), "Invalid college address"); require(

collegeRecords[\_id].Id == 0,

"College with the given ID already exists"

**);**

require(bytes(\_name).length > 0, "College name cannot be empty"); require(

bytes(\_location).length > 0, "College location cannot be empty"

**);**

College memory c = College({ Id: \_id,

Fees: \_fees, Name: \_name,

Location: \_location, Address: \_address

**});**

collegeRecords[c.Id] = c; colleges.push(c);

**}**

function getCollegeDetails( uint256 \_Id

) public view returns (College memory) { require(

collegeRecords[\_Id].Id != 0,

"No college with the given ID found"

**);**

return collegeRecords[\_Id];

**}**

function addStudentRecord( uint256 \_id,

string memory \_firstName, string memory \_lastName, uint8 \_percentage, address \_address

) public onlyGovernment { require(

studentRecords[\_id].Id == 0,

"Student with the given ID already exists"

**);**

Student memory student = Student({ Id: \_id,

FirstName: \_firstName, LastName: \_lastName, Percentage: \_percentage, IsEligibleForScholarship: false, Address: \_address

**});**

students.push(student);

studentRecords[student.Id] = student;

**}**

function getStudentDetails( uint256 \_Id

) public view returns (Student memory) { require(

studentRecords[\_Id].Id != 0,

"No student with the given ID found"

**);**

return studentRecords[\_Id];

**}**

function isStudentEligibleForScholarship( uint256 \_Id

) public view returns (bool) { require(

studentRecords[\_Id].Id != 0,

"No student with the given ID found"

**);**

Student memory s = getStudentDetails(\_Id); if (s.Percentage >= 80) {

return true;

} else {

return false;

**}**

**}**

***//***

function createScholarship( uint256 \_StudentId, uint256 \_CollegeId

) public onlyGovernment { require(

studentRecords[\_StudentId].Id != 0, "No student with the given ID found"

**);**

require(

collegeRecords[\_CollegeId].Id != 0, "No college with the given ID found"

**);**

require(

scholarshipRecords[\_StudentId].StudentId == 0, "Scholarship with the given ID already exists"

**);**

require(

isStudentEligibleForScholarship(\_StudentId), "Student is not eligible for a scholarship"

**);**

College memory c = collegeRecords[\_CollegeId]; uint min\_amount = 2000000000000000000;

uint scholarship\_amount = 0;

if (c.Fees < min\_amount) { scholarship\_amount = c.Fees;

} else {

scholarship\_amount = min\_amount;

**}**

Scholarship memory new\_scholarship = Scholarship({ Id: scholarships.length + 1,

StudentId: \_StudentId, CollegeId: \_CollegeId,

ScholarshipName: "Merit Scholarship", Amount: scholarship\_amount,

To: payable(c.Address), Status: "Pending"

**});**

scholarshipRecords[\_StudentId] = new\_scholarship; scholarships.push(new\_scholarship);

**}**

function disburseScholarship( uint256 \_StudentId

) public payable onlyGovernment {

Scholarship

storage scholarship\_of\_registered\_stduent = scholarshipRecords[

\_StudentId

**];**

require(

scholarship\_of\_registered\_stduent.StudentId != 0,

"Scholarship for the given student not found"

**);**

require(

keccak256(

abi.encodePacked(scholarship\_of\_registered\_stduent.Status)

) == keccak256(abi.encodePacked("Pending")), "Scholarship is not pending"

**);**

scholarship\_of\_registered\_stduent.Status = "Awarded"; scholarship\_of\_registered\_stduent.To.transfer(

scholarship\_of\_registered\_stduent.Amount

**);**

StudentFeesPaid storage getCollege = college\_student\_fees\_paid[ scholarship\_of\_registered\_stduent.CollegeId

**];**

getCollege.studentFeesPaid[ scholarship\_of\_registered\_stduent.StudentId

] = scholarship\_of\_registered\_stduent.Amount;

**}**

function getScholarshipStatus( uint256 \_StudentId

) public view returns (string memory) { require(

scholarshipRecords[\_StudentId].Id != 0, "Scholarship with the given ID does not exists"

**);**

Scholarship memory scholarship = scholarshipRecords[\_StudentId]; return scholarship.Status;

**}**

function updateScholarshipStatusToCancel( uint256 \_StudentId

) public onlyGovernment { require(

scholarshipRecords[\_StudentId].Id != 0, "Scholarship with the given ID does not exists"

**);**

Scholarship storage \_scholarship = scholarshipRecords[\_StudentId];

\_scholarship.Status = "Cancel";

**}**

function updateScholarshipStatusToPaid( uint256 \_StudentId

) public onlyGovernment { require(

scholarshipRecords[\_StudentId].Id != 0, "Scholarship with the given ID does not exists"

**);**

Scholarship storage \_scholarship = scholarshipRecords[\_StudentId];

\_scholarship.Status = "Paid";

**}**

function updateScholarshipStatusToFailed( uint256 \_StudentId

) public onlyGovernment { require(

scholarshipRecords[\_StudentId].Id != 0, "Scholarship with the given ID does not exists"

**);**

Scholarship storage \_scholarship = scholarshipRecords[\_StudentId];

\_scholarship.Status = "Failed";

**}**

function updateScholarshipStatusToActive( uint256 \_StudentId

) public onlyGovernment { require(

scholarshipRecords[\_StudentId].Id != 0, "Scholarship with the given ID does not exists"

**);**

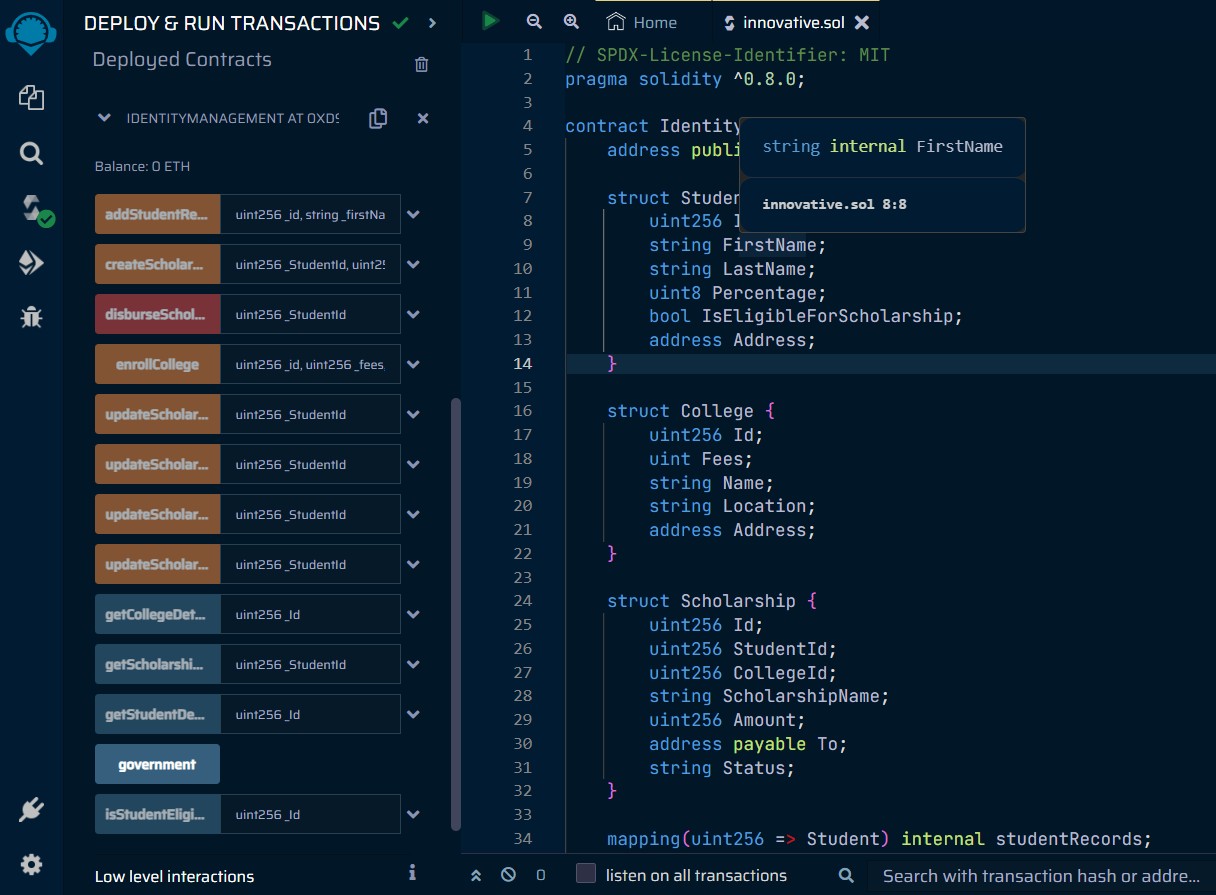
Scholarship storage \_scholarship = scholarshipRecords[\_StudentId];

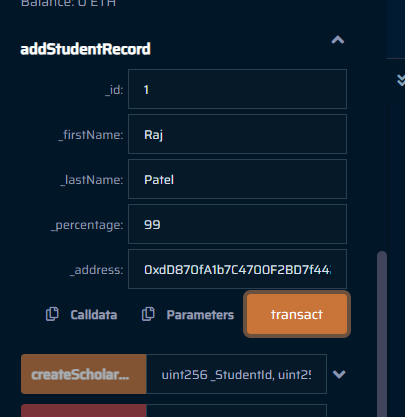
\_scholarship.Status = "Active";

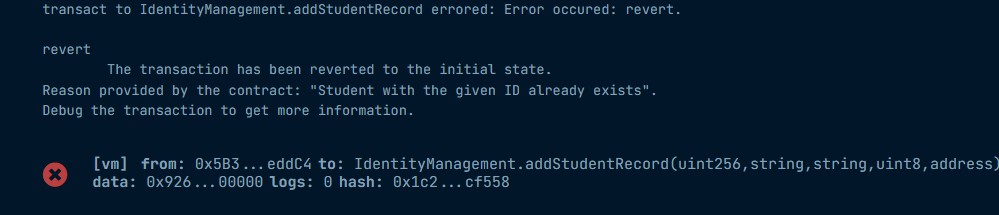
**}**

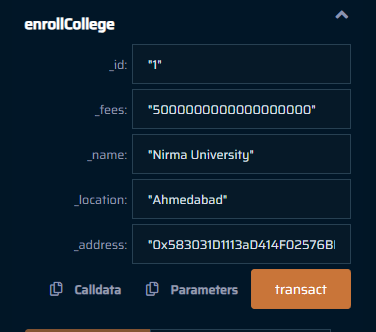
**}**

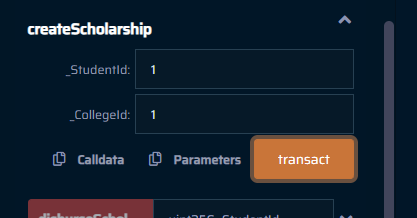
**Output:**

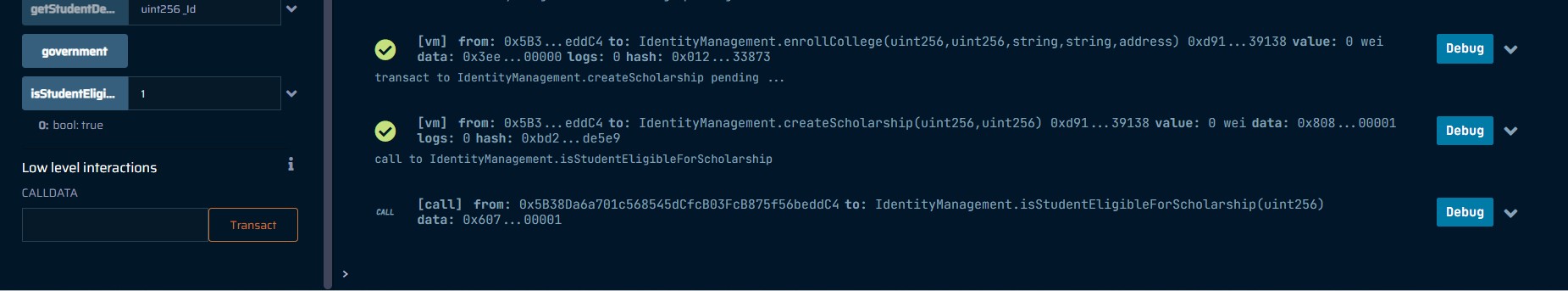












"' 6nqaa 'Jn'\_L':, **rem**

: ,)cp

0

,0,lj :.JI,]

