Name: Raj K Patel
Roll No: 20BCE218
Course: 2CSDE93 - Blockchain Technology
Practical No: 8
Aim: To design and develop end-to-end decentralized applications (Dapps).

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"
        );
```

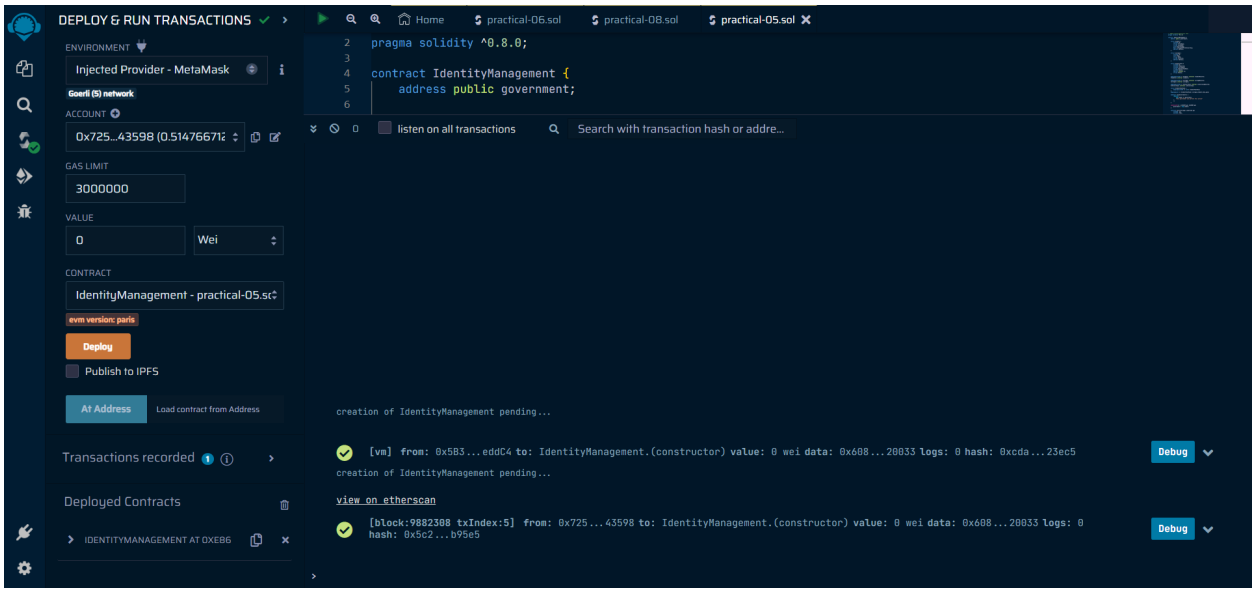
```
        College memory c = College({
            Id: _id,
            Fees: _fees,
            Name: _name,
            Location: _location,
            Address: _address
        });
        collegeRecords[c.Id] = c;
        colleges.push(c);
    }

    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;
    }
```

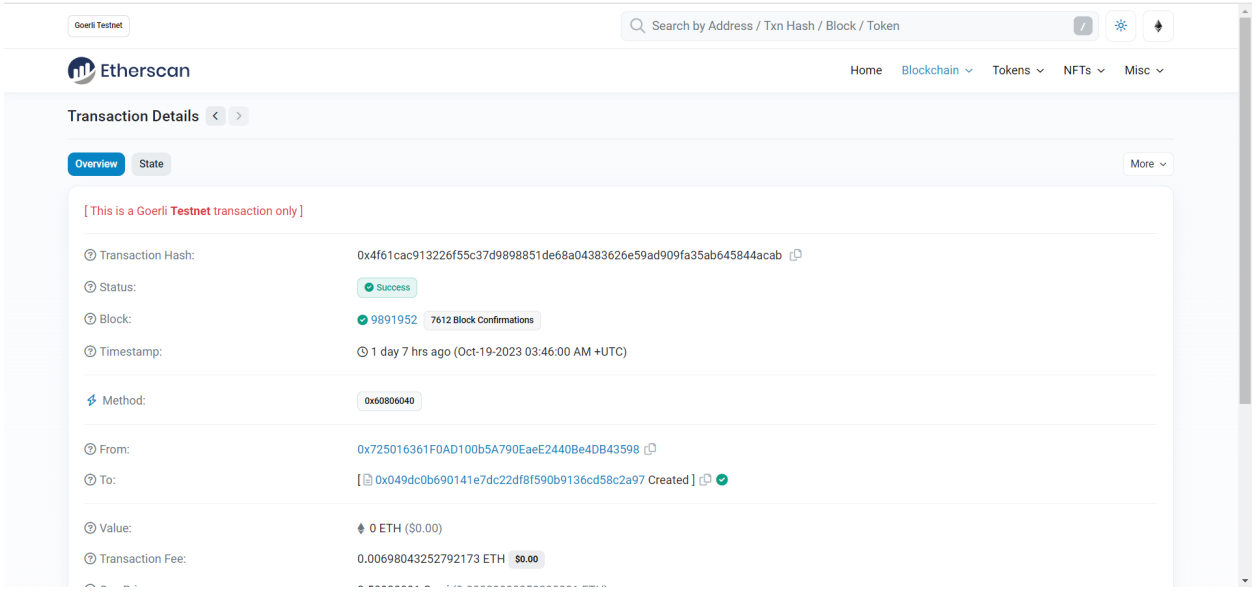
```
function getStudentDetails(
    uint256 _Id
) public view returns (Student memory) {
    require(
        studentRecords[_Id].Id != 0,
        "No student with the given ID found"
    );
    return studentRecords[_Id];
}

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(
```

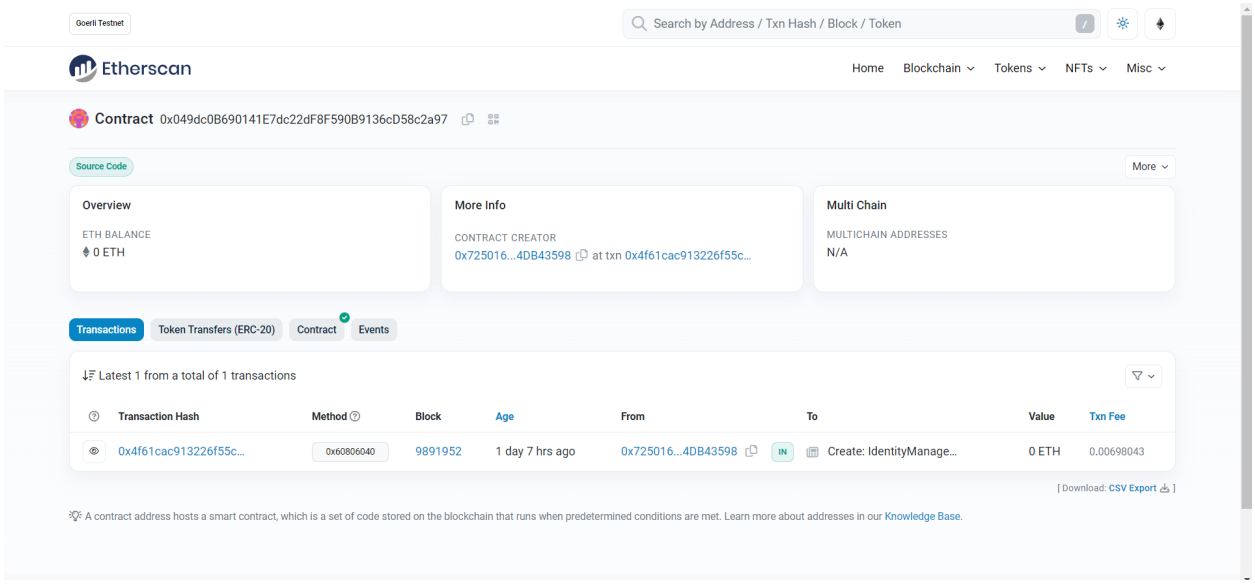
```
            scholarshipRecords[_StudentId].StudentId == 0,
            "Scholarship with the given ID already exists"
        );
        require(
            isStudentEligibleForScholarship(_StudentId),
            "Student is not eligible for a scholarship"
        );

        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"
        );
```

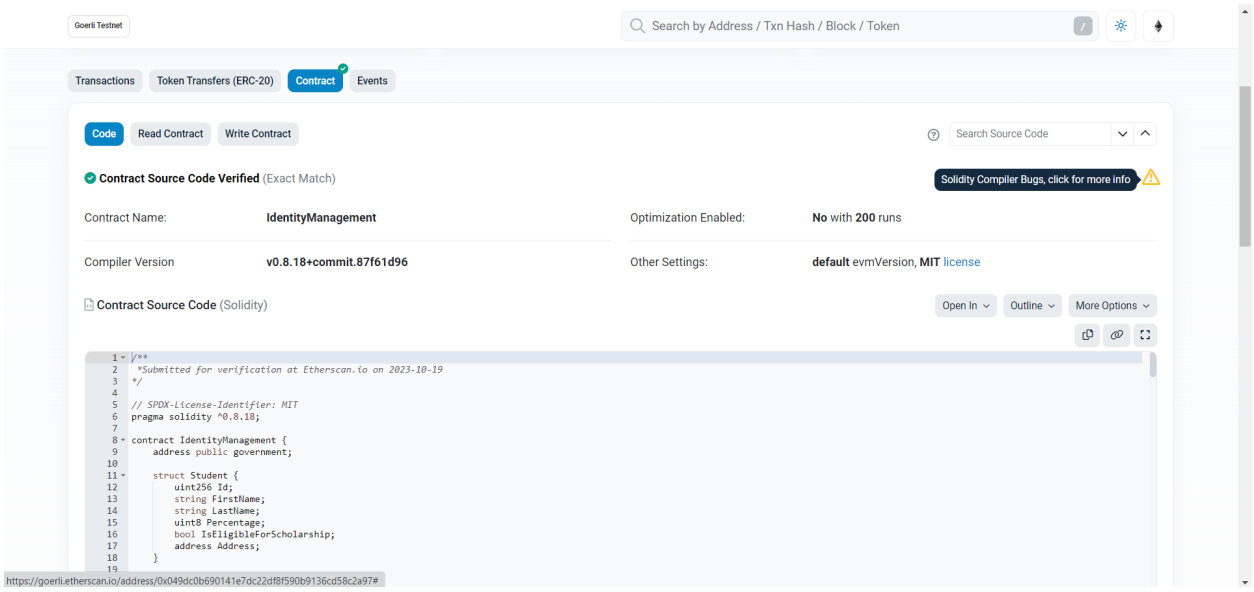
```
        require(
            keccak256(
                abi.encodePacked(scholarship_of_registered_stduent.Status)
            ) == keccak256(abi.encodePacked("Pending")),
            "Scholarship is not pending"
        );

        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";
    }
```

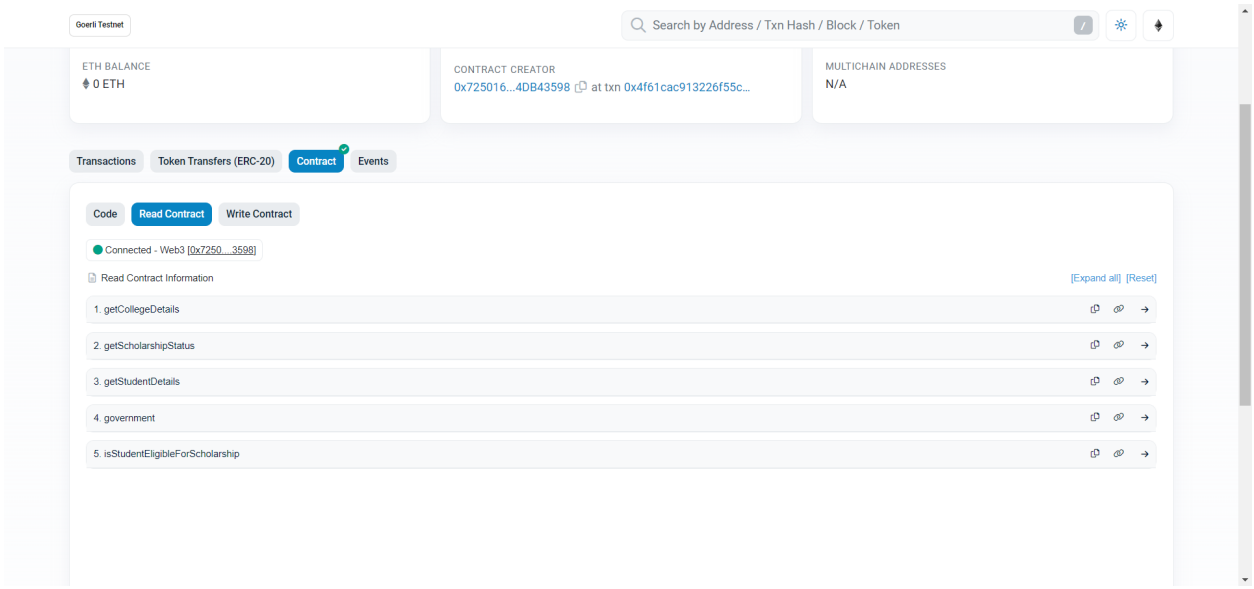
```
    function updateScholarshipStatusToPaid(
        uint256 _StudentId
    ) public onlyGovernment {
        require(
            scholarshipRecords[_StudentId].Id != 0,
            "Scholarship with the given ID does not exists"
        );
        Scholarship storage _scholarship = scholarshipRecords[_StudentId];
        _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";
    }
}
```

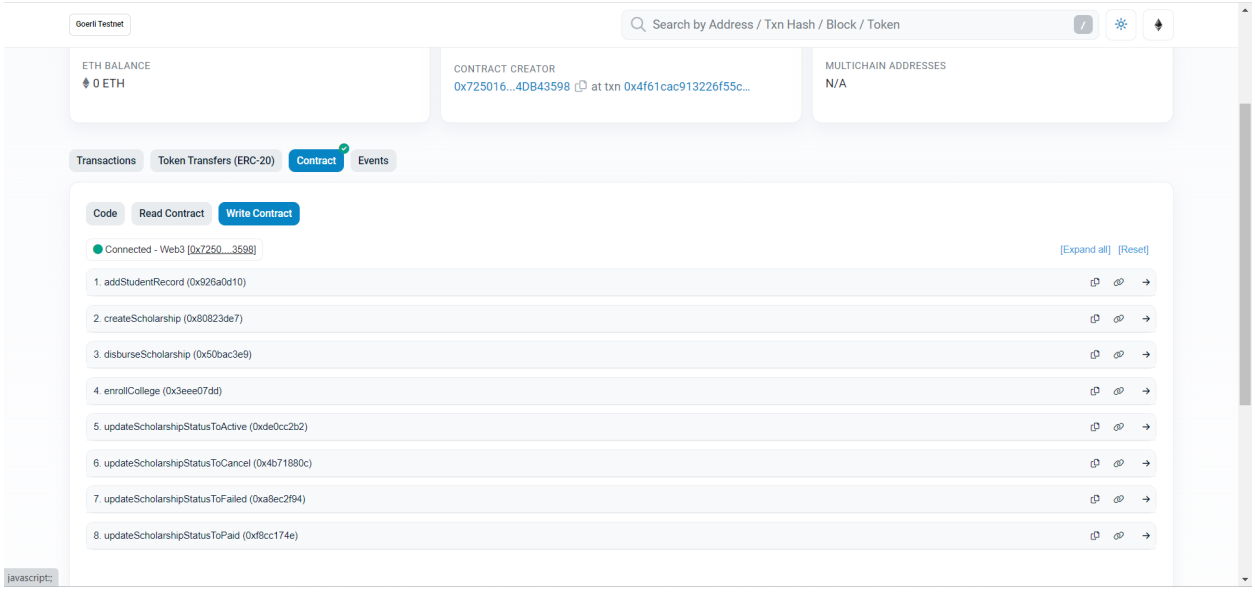

DEPLOY & RUN TRANSACTIONS
ENVIRONMENT
Injected Provider - MetaMask
Goerli (5) network
ACCOUNT
0x725...43598 (0.514766712
GAS LIMIT
3000000
VALUE
0
Wei
CONTRACT
IdentityManagement - practical-05.sc
evm version: paris
Deploy
Publish to IPFS
At Address
Load contract from Address
Transactions recorded 1
Deployed Contracts
IDENTITYMANAGEMENT AT 0XEB6
Home
practical-06.sol
practical-08.sol
practical-05.sol
pragma solidity ^0.8.0;
contract IdentityManagement {
address public government;
listen on all transactions
Search with transaction hash or addre...
creation of IdentityManagement pending...
[vm] from: 0x5B3...eddC4 to: IdentityManagement.(constructor) value: 0 wei data: 0x608...20033 logs: 0 hash: 0xcda...23ec5
Debug
creation of IdentityManagement pending...
view on etherscan
[block:9882308 txIndex:5] from: 0x725...43598 to: IdentityManagement.(constructor) value: 0 wei data: 0x608...20033 logs: 0 hash: 0x5c2...b95e5
Debug


Goerli Testnet
Search by Address / Txn Hash / Block / Token
Etherscan
Home
Blockchain
Tokens
NFTs
Misc
Transaction Details
Overview
State
More
[This is a Goerli Testnet transaction only]
Transaction Hash:
0x4f61cac913226f55c37d9898851de68a04383626e59ad909fa35ab645844acab
Status:
Success
Block:
9891952
7612 Block Confirmations
Timestamp:
1 day 7 hrs ago (Oct-19-2023 03:46:00 AM +UTC)
Method:
0x60806040
From:
0x725016361F0AD100b5A790EaeE2440Be4DB43598
To:
[0x049dc0b690141e7dc22df8f590b9136cd58c2a97 Created]
Value:
0 ETH ($0.00)
Transaction Fee:
0.00698043252792173 ETH
$0.00


Goerli Testnet
Search by Address / Txn Hash / Block / Token
Etherscan
Home
Blockchain
Tokens
NFTs
Misc
Contract 0x049dc0B690141E7dc22dF8F590B9136cD58c2a97
Source Code
More
Overview
ETH BALANCE
0 ETH
More Info
CONTRACT CREATOR
0x725016...4DB43598 at txn 0x4f61cac913226f55c...
Multi Chain
MULTICHAIN ADDRESSES
N/A
Transactions
Token Transfers (ERC-20)
Contract
Events
Latest 1 from a total of 1 transactions
Transaction Hash
Method
Block
Age
From
To
Value
Txn Fee
0x4f61cac913226f55c...
0x60806040
9891952
1 day 7 hrs ago
0x725016...4DB43598
IN
Create: IdentityManage...
0 ETH
0.00698043
[Download: CSV Export]
A contract address hosts a smart contract, which is a set of code stored on the blockchain that runs when predetermined conditions are met. Learn more about addresses in our Knowledge Base.


Goerli Testnet
Search by Address / Txn Hash / Block / Token
Transactions
Token Transfers (ERC-20)
Contract
Events
Code
Read Contract
Write Contract
Search Source Code
Contract Source Code Verified (Exact Match)
Solidity Compiler Bugs, click for more info
Contract Name:
IdentityManagement
Optimization Enabled:
No with 200 runs
Compiler Version
v0.8.18+commit.87f61d96
Other Settings:
default evmVersion, MIT license
Contract Source Code (Solidity)
Open In
Outline
More Options
/**
*Submitted for verification at Etherscan.io on 2023-10-19
*/
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.18;
contract IdentityManagement {
address public government;
struct Student {
uint256 Id;
string FirstName;
string LastName;
uint8 Percentage;
bool IsEligibleForScholarship;
address Address;
}
https://goerli.etherscan.io/address/0x049dc0b690141e7dc22df8f590b9136cd58c2a97#


Goerli Testnet
Search by Address / Txn Hash / Block / Token
ETH BALANCE
0 ETH
CONTRACT CREATOR
0x725016...4DB43598 at txn 0x4f61cac913226f55c...
MULTICHAIN ADDRESSES
N/A
Transactions
Token Transfers (ERC-20)
Contract
Events
Code
Read Contract
Write Contract
Connected - Web3 [0x7250...3598]
Read Contract Information
[Expand all] [Reset]
1. getCollegeDetails
2. getScholarshipStatus
3. getStudentDetails
4. government
5. isStudentEligibleForScholarship


Goerli Testnet
Search by Address / Txn Hash / Block / Token
ETH BALANCE
0 ETH
CONTRACT CREATOR
0x725016...4DB43598 at txn 0x4f61cac913226f55c...
MULTICHAIN ADDRESSES
N/A
Transactions
Token Transfers (ERC-20)
Contract
Events
Code
Read Contract
Write Contract
Connected - Web3 [0x7250...3598]
[Expand all] [Reset]
1. addStudentRecord (0x926a0d10)
2. createScholarship (0x80823de7)
3. disburseScholarship (0x50bac3e9)
4. enrollCollege (0x3eee07dd)
5. updateScholarshipStatusToActive (0xde0cc2b2)
6. updateScholarshipStatusToCancel (0x4b71880c)
7. updateScholarshipStatusToFailed (0xa8ec2f94)
8. updateScholarshipStatusToPaid (0xf8cc174e)
javascript:;