STUDENT RESULT MANAGEMENT SYSTEM USING BLOCK CHAIN TECHNOLOGY

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

ADITYA INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(Approved by AICTE, New Delhi, Affiliated to JNTUK, Accredited by NBA & NAAC)

(AUTONOMOUS) K. KOTTURU, TEKKALI-532201

2023-2024

Submitted by

ARJI PAVAN KUMAR 21A51A1270

Organised By Department Of IT

aitam
ADITYA INSTITUTE OF TECHNOLOGY AND MANAGEMENT

AITAM ... DEVELOPING ... Leaders | Entrepreneurs | Innovators

An Autonomous Institution
NAAC A+,NBA
(CSE,MECH,CIVIL,ECE,EEE,IT)

CERTIFICATE

This is to certify that the Skill Advanced Course entitled "**STUDENT RESULT MANAGEMENT SYSTEM USING BLOCK CHAIN TECHNOLOGY**" is being submitted by **ARJI PAVAN KUMAR(21A51A1270);** Conducted by APSSDC-CM's Center of excellence in partial fulfilment of requirements for the award of Bachelor of Technology in **INFORMATION TECHNOLOGY** during the year 2023 Aditya Institute of Technology and Management, Tekkali is a record of bonfide work carried out by them under my guidance and supervision during the academic year 2023-24.

Dr. Y. RAMESH	**SRI DASARI REVENTH RAJ**
MTech, PhD	**MTech**
Professor & Head of the Dept.	**Professor & Coordinator**
Department of IT	**Department of IT**

TABLE OF CONTENTS

INTRODUCTION TO BLOCKCHAIN

Blockchain technology is a decentralized and distributed ledger system that securely records transactions across multiple computers in a way that ensures the security, transparency, and immutability of the data. Each set of transactions is stored in a "block," and these blocks are linked together in a chronological chain, hence the name "blockchain."

What makes blockchain unique is its decentralized nature. Traditional databases are typically centralized, meaning they are stored in one location, making them vulnerable to hacks and data manipulation. In contrast, blockchain operates on a network of computers (nodes), where each node has a copy of the entire blockchain. This decentralization makes it extremely difficult for any single entity to control the entire blockchain, enhancing security.

Moreover, blockchain uses cryptographic techniques to secure the data, ensuring that once a transaction is recorded, it cannot be altered without changing all subsequent blocks. This immutability ensures the integrity of the data, making blockchain ideal for applications where trust and transparency are crucial, such as in finance, supply chain management, healthcare, and more.

Blockchain technology also enables the use of smart contracts, which are self-executing contracts with the terms of the agreement directly written into lines of code. These smart contracts automatically execute and enforce the terms of the agreement when predefined conditions are met, eliminating the need for intermediaries and reducing the risk of disputes.

Overall, blockchain technology has the potential to revolutionize various industries by providing a secure, transparent, and efficient way to record and verify transactions and digital assets.

<u>**FEATURES OF BLOCKCHAIN TECHNOLOGY**</u>

Decentralization: Decentralization in blockchain refers to transferring control and decision making from a centralized entity (individual, organization, or group) to a distributed network. Decentralized blockchain networks use transparency to reduce the need for trust among participants. These networks also deter participants from exerting authority or control over one another in ways that degrade the functionality of the network.

Immutability: Immutability means something cannot be changed or altered. No participant can tamper with a transaction once someone has recorded it to the shared ledger. If a transaction record includes an error, you must add a new transaction to reverse the mistake, and both transactions are visible to the network.

SOLIDITY

Solidity is a high-level, statically-typed programming language designed for developing smart contracts on blockchain platforms, primarily Ethereum. Smart contracts are self-executing contracts with the terms of the agreement between buyer and seller directly written into code. Solidity allows developers to create these smart contracts, which run on the Ethereum Virtual Machine (EVM), enabling decentralized applications (DApps) to be built on the Ethereum blockchain.

Key features of Solidity include:

1. **Smart Contracts:** Solidity is primarily used for writing smart contracts, enabling developers to create automated agreements that execute predefined actions when specific conditions are met.

2.**Ethereum Compatibility:** Solidity is specifically designed for Ethereum, making it one of the most popular languages for Ethereum-based DApps and smart contracts.

3. **Statically Typed:** Solidity is statically typed, meaning variable types are explicitly declared at compile-time. This helps catch type-related errors before the code is executed.

4. **Object-Oriented:** Solidity supports object-oriented programming concepts such as inheritance, encapsulation, and polymorphism, making it more versatile and powerful for complex smart contract development.

5. **Security Focus:** Solidity includes various security features and best practices to help developers write secure contracts. However, it's important for developers to be aware of potential vulnerabilities and follow security best practices.

ABSTRACT

Blockchain technology has the potential to revolutionize various industries, including education. When applied to student result management systems (SRMS), blockchain can offer enhanced security, transparency, and efficiency in managing and verifying student records and results. Here's some information about the use of blockchain in SRMS:

1. Blockchain-Based SRMS: Several research papers and articles propose the use of blockchain in creating fully functional Student Result Management Systems. These systems leverage blockchain to maintain students' records, course registrations, and academic results, ensuring data integrity and security

2. Enhanced Security: Blockchain technology employs encryption algorithms, making student information highly secure. It has been reported that the security of SRMS based on blockchain technology can increase significantly, up to 87% in some cases, compared to traditional systems

3. Transparency and Trust: Blockchain's decentralized nature ensures transparency and trust in the management of student records and results. It eliminates the need for intermediaries and allows stakeholders to verify data independently.

4. Efficiency: Blockchain-based SRMS can streamline administrative processes, reducing paperwork and manual verification efforts. This can lead to more efficient and faster result processing

.

PROBLEM STATEMENT

Blockchain technology can address these problems by providing a decentralized, secure, and transparent platform for managing student results. It enables tamper-proof record-keeping, reduces the reliance on centralized authorities, and improves data accessibility and security

1. **Lack of Data Security**: Traditional student result management systems often face security issues, including data breaches and unauthorized access to sensitive student information. This jeopardizes the confidentiality and integrity of student records.
2. **Trust Issues**: Current systems rely on centralized databases managed by institutions, creating a single point of failure and the need to trust the institution completely. This can lead to concerns about data manipulation and tampering.
3. **Inefficient Verification**: Verification of academic credentials is a time-consuming and complex process, often requiring manual checks with educational institutions. This inefficiency can delay employment verification and academic transfers.
4. **Data Redundancy**: Existing systems may result in redundant data entry and storage, leading to inefficiencies in data management and potential discrepancies in student records.
5. **Transparency**: There is a need for greater transparency in the management of student results, allowing students, employers, and educational institutions to access and verify academic records easily.
6. **Data Accessibility**: Ensuring that authorized parties can access and verify student records securely and conveniently is a challenge in traditional systems.

EXISTING SYSTEM

The existing system of a Student Result Management System typically involves a digital platform or software designed to efficiently manage and process students academic results. Here is an overview of the existing system :

1. **Modules**: Student Result Management Systems are often divided into two main modules: one for students and one for administrators. The admin module typically includes features such as an admin dashboard and the ability to add/update classes and student information.
2. **Automation**: These systems aim to automate the result management process for all categories of students, providing a seamless and interactive platform for administrators and students to access and manage academic results.
3. **Data Mining Techniques**: Some systems utilize data mining techniques to predict student performance based on generated rules. This can help in identifying students who may require additional support or intervention.
4. **Web-Based**: Many Student Result Management Systems are web-based, allowing users to access and manage results from anywhere with an internet connection. These systems are often programmed using server-side languages.
5. **Administrator Panel**: Administrators can edit content and manage the system through an administrator panel, which is then reflected on the public website where students can access their results.

PROPOSED SYSTEM

The proposed Student Result Management System (SRMS) using Blockchain is designed to enhance the management and security of student academic records. Here is an overview of the proposed system:

1. Blockchain Integration: The proposed system will integrate blockchain technology to create a decentralized and secure ledger for storing student result data. This blockchain ledger will ensure the immutability and tamper-proof nature of academic records.

2. User-Friendly Interface: The system will provide a user-friendly web interface for faculties, administrators, and students. Faculties can input marks, while students can securely access their results.

3. Data Security: Blockchain's cryptographic techniques will be utilized to enhance data security and prevent unauthorized access or tampering of student result data.

4. Transparency: The proposed system will offer transparency by allowing easy verification of academic records. This will reduce fraud and mistrust in the authenticity of results.

5. Efficiency: The system will streamline result recording and verification processes, reducing administrative overhead and manual verification efforts.

6. Decentralization: Student result data will be stored in a decentralized manner across multiple nodes, eliminating the need for a central authority and reducing the risk of data loss.

7. Smart Contracts: The system may leverage smart contracts on the blockchain to automate tasks such as grade calculations, result notifications, and record updates.

8. Security Testing: Rigorous security testing and experiments will be conducted to ensure the robustness and security of the blockchain-based SRMS.

Implementation of Blockchain in Student Result Management System

1. Decentralized Ledger: Blockchain operates on a decentralized ledger where each block contains a set of transactions, including student results. These blocks are linked together chronologically, forming a secure and tamper-resistant chain of data.

2. Data Encryption: Student result data is encrypted using cryptographic techniques before being added to the blockchain. This ensures that sensitive academic information is protected from unauthorized access.

3. User Authentication: Blockchain-based SRMS often incorporate robust user authentication mechanisms. Students, faculty, and administrators have secure access to the system using cryptographic keys.

4. Immutable Records: Once academic results are added to the blockchain, they become immutable. This means that once recorded, the data cannot be altered or deleted, ensuring the authenticity of student records.

5. Transparency: Blockchain's transparency allows stakeholders to verify academic records independently. Students can easily access and verify their results without relying on intermediaries.

6. Smart Contracts: Smart contracts can be used to automate processes related to result management. For example, when exam scores are submitted, a smart contract can automatically calculate grades and update records.

7. Verification and Credentialing: Blockchain can be used for issuing digital credentials and certificates. Graduates can have their degrees stored on the blockchain, making it easy for employers and institutions to verify qualifications.

8. Eliminating Fraud: The tamper-resistant nature of blockchain eliminates the risk of result manipulation and fraud. It ensures that academic records remain accurate and trustworthy.

9. Reduced Administrative Overheads: Automation of result management processes on the blockchain can reduce administrative overheads and streamline operations.

10. Integration: Blockchain-based SRMS can be integrated with other educational systems and databases, creating a seamless ecosystem for managing student data.

SOURCE CODE:

```
pragma solidity ^0.8.0;
// Build the Contract
contract MarksManagmtSys
{
    // Create a structure for
    // employee details
    struct Student
    {
        string Student_ID;
        string Name;
        uint Dccn_marks;
        uint Oose_marks;
        uint Dm_marks;
        uint Toc_marks;
        string Sac_grade;
        string Internship_grade;
    }
    struct StudentGrades
    {
        string student_ID;
        string Name;
        string Dccn_grade;
        string Oose_grade;
        string Dm_grade;
        string Toc_grade;
        string Sac_grade;
        string Internship_grade;
    }
    struct StudentGradePoints
    {
        string student_ID;
        string name;
        uint total_marks;
        string GradePoints;
    }
    struct showresult
    {
     string Student_ID;
        string Name;
        uint Dccn_marks;
        string Dccn_grade;
        uint Oose_marks;
        string Oose_grade;
        uint Dm_marks;
        string Dm_grade;
        uint Toc_marks;
        string Toc_grade;
        string Sac_grade;
        string Internship_grade;
        uint total_marks;
        string GradePoints;
    }
    address owner;
    uint public StudentCount = 0;
    mapping(uint => Student) private StudentRecords;
    mapping(uint=>StudentGrades)private Student_grade_records;
    mapping(uint=>StudentGradePoints)public Student_grade_Points;
    mapping(uint=>uint)private Student_analysis;
    mapping(uint=>showresult)public Show_result;
    modifier onlyOwner
    {
        require(owner == msg.sender);
        _;
    }
    constructor()
    {
        owner=msg.sender;
    }
```

```
// Create a function to add
// the new records
function addNewRecords(string memory student_ID,string memory Name,uint Dccn_marks,
   uint Oose_marks,uint Dm_marks,uint Toc_marks,string memory Sac_grade,
   string memory Internship_grade) public onlyOwner
{
   // Increase the count by 1
   StudentCount = StudentCount + 1;

   // Fetch the student details
   // with the help of stdCount
   StudentRecords[StudentCount] = Student(student_ID,Name,Dccn_marks,
   Oose_marks,Dm_marks,Toc_marks,Sac_grade,Internship_grade);
}
function Student_grades()public onlyOwner
{
   string memory Student_id;
   string memory Dccn_grade;
   string memory Oose_grade;
   string memory Toc_grade;
   string memory Dm_grade;
   for(uint i=1;i<=StudentCount;i++)
   {
      if(StudentRecords[i].Dccn_marks<=40)
          Dccn_grade="F";
      else if(StudentRecords[i].Dccn_marks<=50)
          Dccn_grade="E";
      else if(StudentRecords[i].Dccn_marks<=60)
          Dccn_grade="D";
      else if(StudentRecords[i].Dccn_marks<=70)
          Dccn_grade="C";
      else if(StudentRecords[i].Dccn_marks<=80)
          Dccn_grade="B";
      else if(StudentRecords[i].Dccn_marks<=90)
          Dccn_grade="A";
```

```
else
    Dccn_grade="O";
if(StudentRecords[i].Oose_marks<=40)
    Oose_grade="F";
else if(StudentRecords[i].Oose_marks<=50)
    Oose_grade="E";
else if(StudentRecords[i].Oose_marks<=60)
    Oose_grade="D";
else if(StudentRecords[i].Oose_marks<=70)
    Oose_grade="C";
else if(StudentRecords[i].Oose_marks<=80)
    Oose_grade="B";
else if(StudentRecords[i].Oose_marks<=90)
    Oose_grade="A";
else
    Oose_grade="O";
if(StudentRecords[i].Dm_marks<=40)
    Dm_grade="F";
else if(StudentRecords[i].Dm_marks<=50)
    Dm_grade="E";
else if(StudentRecords[i].Dm_marks<=60)
    Dm_grade="D";
else if(StudentRecords[i].Dm_marks<=70)
    Dm_grade="C";
else if(StudentRecords[i].Dm_marks<=80)
    Dm_grade="B";
else if(StudentRecords[i].Dm_marks<=90)
    Dm_grade="A";
else
    Dm_grade="O";
if(StudentRecords[i].Toc_marks<=40)
    Toc_grade="F";
else if(StudentRecords[i].Toc_marks<=50)
    Toc_grade="E";
else if(StudentRecords[i].Toc_marks<=60)
```

```
            Toc_grade="D";
        else if(StudentRecords[i].Toc_marks<=70)
            Toc_grade="C";
        else if(StudentRecords[i].Toc_marks<=80)
            Toc_grade="B";
        else if(StudentRecords[i].Toc_marks<=90)
            Toc_grade="A";
        else
            Toc_grade="O";
        Student_id=StudentRecords[i].Student_ID;
        Student_grade_records[i]=StudentGrades(Student_id,StudentRecords[i].Name,
                                        Dccn_grade,Oose_grade,Dm_grade,Toc_grade,StudentRecords[i].Sac_grade,
                                         StudentRecords[i].Internship_grade);
    }
}
function GradePoints()public onlyOwner
{
    uint sum;
    uint total;

    for(uint i=1;i<=StudentCount;i++)
    {
        sum=0;
        sum=sum+grade_point_analysis(Student_grade_records[i].Dccn_grade);
        sum=sum+grade_point_analysis(Student_grade_records[i].Oose_grade);
        sum=sum+grade_point_analysis(Student_grade_records[i].Dm_grade);
        sum=sum+grade_point_analysis(Student_grade_records[i].Toc_grade);
        sum=sum+grade_point_analysis(Student_grade_records[i].Sac_grade);
        sum=sum+grade_point_analysis(Student_grade_records[i].Internship_grade);
        string memory grade_points;
        uint score;
        if(sum>60)
          {
            grade_points="FAIL";
            score=0;
            }
          else
            {
              uint256 quotieent=sum/6;
              uint256 remainder=(sum*100/6)%100;
              score=quotieent*100+remainder;
              string memory str1=convert(quotieent);
              string memory str2=convert(remainder);
              grade_points = string(abi.encodePacked(str1, ".", str2));

            }
            Student_analysis[i]=score;

total=StudentRecords[i].Dccn_marks+StudentRecords[i].Dm_marks+StudentRecords[i].Oose_marks+

StudentRecords[i].Toc_marks+sum1(StudentRecords[i].Sac_grade)+sum1(StudentRecords[i].Internship_gra
de);

Student_grade_Points[i]=StudentGradePoints(StudentRecords[i].Student_ID,StudentRecords[i].Name,total,g
rade_points);
        }
    }
    function convert(uint256 n) private pure returns (string memory) {
        // Convert the uint256 to a string
        string memory a = uint2str(n);
        return a;
    }
    function uint2str(uint256 _i) internal pure returns (string memory) {
        if (_i == 0) {
            return "0";
        }
        uint256 j = _i;
        uint256 len;
        while (j != 0) {
            len++;
            j /= 10;
        }
        bytes memory bstr = new bytes(len);
    uint256 k = len;
        while (_i != 0) {
            k = k - 1;
            uint8 temp = (48 + uint8(_i % 10));
            bytes1 b1 = bytes1(temp);
            bstr[k] = b1;
            _i /= 10;
        }
        return string(bstr);
    }
    function sum1(string memory grade)private returns(uint)
    {
         if(keccak256(abi.encodePacked(grade))==keccak256(abi.encodePacked("O")))
          return 100;
        else if(keccak256(abi.encodePacked(grade))==keccak256(abi.encodePacked("A")))
          return 90;
        else if(keccak256(abi.encodePacked(grade))==keccak256(abi.encodePacked("B")))
          return 80;
        else if(keccak256(abi.encodePacked(grade))==keccak256(abi.encodePacked("C")))
          return 70;
        else if(keccak256(abi.encodePacked(grade))==keccak256(abi.encodePacked("D")))
          return 60;
        else if(keccak256(abi.encodePacked(grade))==keccak256(abi.encodePacked("E")))
          return 50;
        else
          return 0;
    }
    function grade_point_analysis(string memory grade)private returns (uint)
    {
        if(keccak256(abi.encodePacked(grade))==keccak256(abi.encodePacked("O")))
          return 10;
        else if(keccak256(abi.encodePacked(grade))==keccak256(abi.encodePacked("A")))
          return 9;
        else if(keccak256(abi.encodePacked(grade))==keccak256(abi.encodePacked("B")))
```

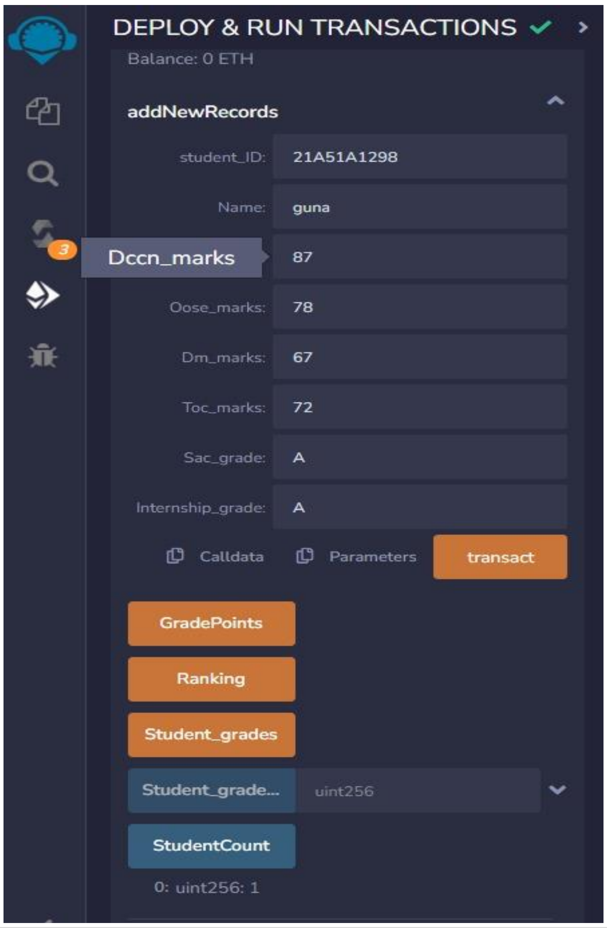
```
          return 8;
else if(keccak256(abi.encodePacked(grade))==keccak256(abi.encodePacked("C")))
          return 7;
        else if(keccak256(abi.encodePacked(grade))==keccak256(abi.encodePacked("D")))
          return 6;
        else if(keccak256(abi.encodePacked(grade))==keccak256(abi.encodePacked("E")))
          return 5;
        else
          return 60;
    }
    function Ranking()public onlyOwner
    {
      for(uint i=1;i<StudentCount;i++)
      {
           for(uint j=i+1;j<=StudentCount;j++)
           {
                 if(Student_analysis[i]<Student_analysis[j])
                 {
                       StudentRecords[0]=StudentRecords[i];
                       StudentRecords[i]=StudentRecords[j];
                       StudentRecords[j]=StudentRecords[0];

                       Student_grade_records[0]=Student_grade_records[i];
                       Student_grade_records[i]=Student_grade_records[j];
                       Student_grade_records[j]=Student_grade_records[0];

                       Student_grade_Points[0]=Student_grade_Points[i];
                       Student_grade_Points[i]=Student_grade_Points[j];
                       Student_grade_Points[j]=Student_grade_Points[0];
                 }


           }
      }
    }
    function searchmyresult(string memory _id)public onlyOwner
    {
        for(uint i=1;i<=StudentCount;i++)
        {
if(keccak256(abi.encodePacked(Student_grade_Points[i].student_ID))==keccak256(abi.encodePacked(_id)))
          {
              Show_result[0]=showresult(Student_grade_Points[i].student_ID,Student_grade_Points[i].name,
                                        StudentRecords[i].Dccn_marks,Student_grade_records[i].Dccn_grade,
                                        StudentRecords[i].Oose_marks,Student_grade_records[i].Oose_grade,
                                        StudentRecords[i].Dm_marks,Student_grade_records[i].Dm_grade,
                                        StudentRecords[i].Toc_marks,Student_grade_records[i].Toc_grade,
                                        StudentRecords[i].Sac_grade,StudentRecords[i].Internship_grade,
                                        Student_grade_Points[i].total_marks,Student_grade_Points[i].GradePoints);
          }
        }
    }
}
```

OUTPUT:



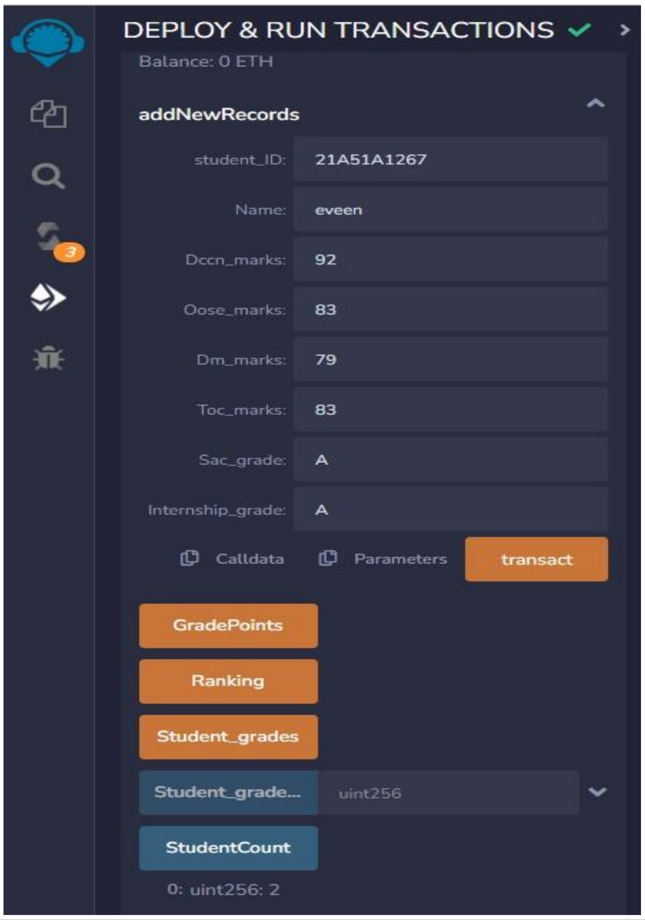
DEPLOY & RUN TRANSACTIONS

Balance: 0 ETH

addNewRecords

student_ID: 21A51A1298
Name: guna
Dccn_marks 87
Oose_marks: 78
Dm_marks: 67
Toc_marks: 72
Sac_grade: A
Internship_grade: A

0: uint256: 1



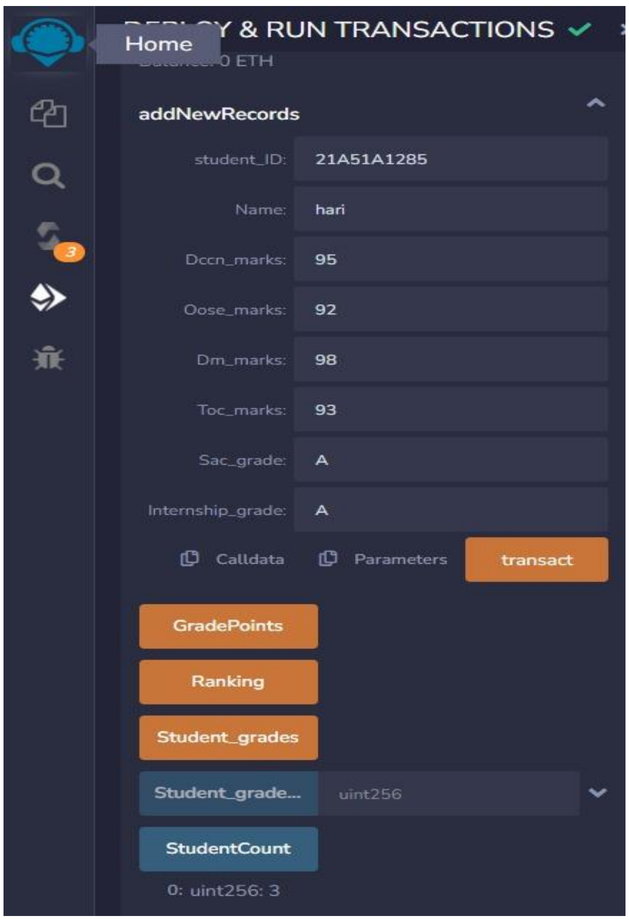
DEPLOY & RUN TRANSACTIONS

Balance: 0 ETH

addNewRecords

student_ID: 21A51A1267
Name: eveen
Dccn_marks: 92
Oose_marks: 83
Dm_marks: 79
Toc_marks: 83
Sac_grade: A
Internship_grade: A

0: uint256: 2



Home Y & RUN TRANSACTIONS

0 ETH

addNewRecords

student_ID: 21A51A1285

Name: hari

Dccn_marks: 95

Oose_marks: 92

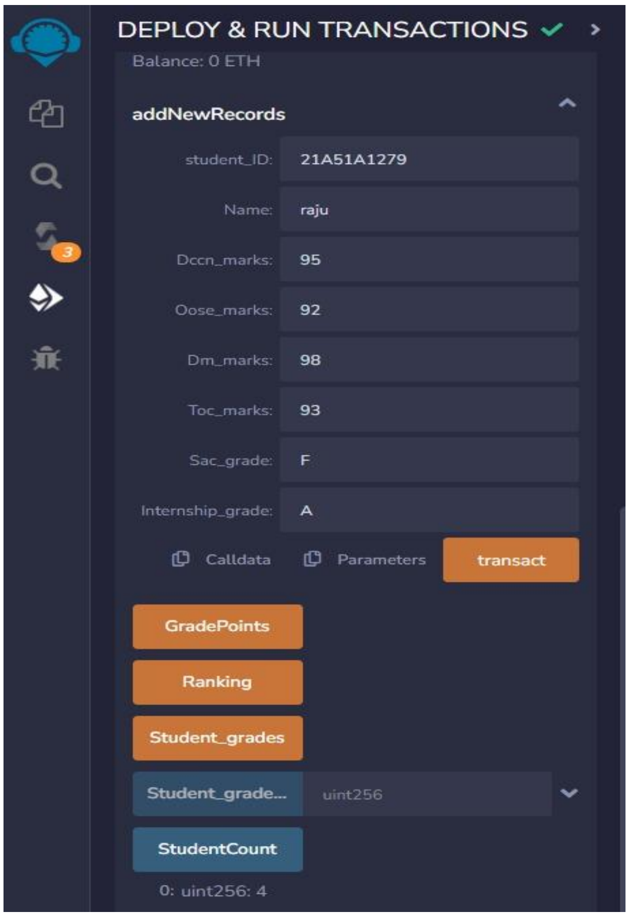
Dm_marks: 98

Toc_marks: 93

Sac_grade: A

Internship_grade: A

```
0: uint256: 3
```



DEPLOY & RUN TRANSACTIONS

Balance: 0 ETH

addNewRecords

student_ID: 21A51A1279

Name: raju

Dccn_marks: 95

Oose_marks: 92

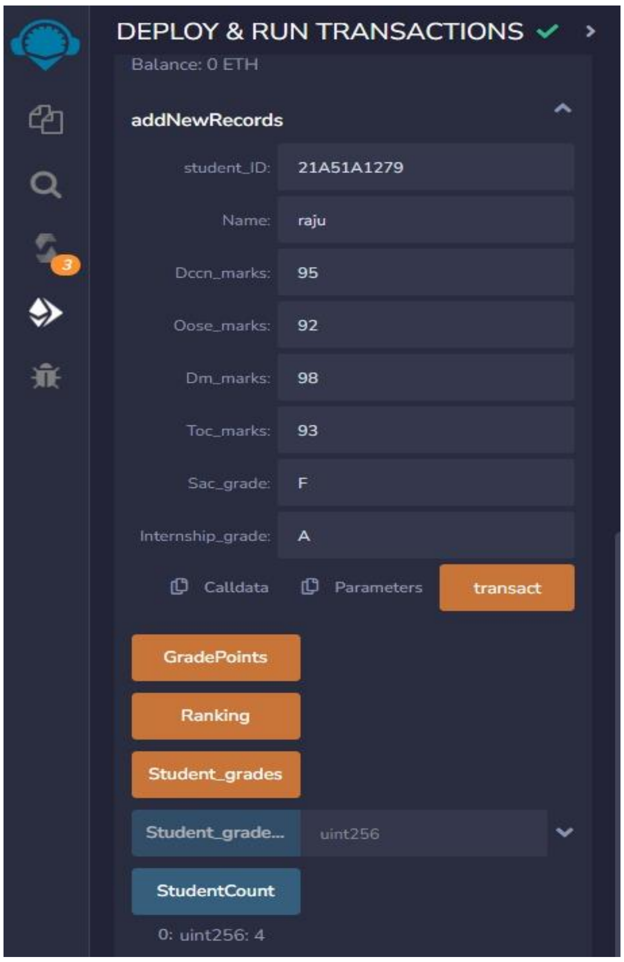
Dm_marks: 98

Toc_marks: 93

Sac_grade: F

Internship_grade: A

```
0: uint256: 4
```



DEPLOY & RUN TRANSACTIONS

Balance: 0 ETH

addNewRecords

student_ID: 21A51A1279

Name: raju

Dccn_marks: 95

Oose_marks: 92

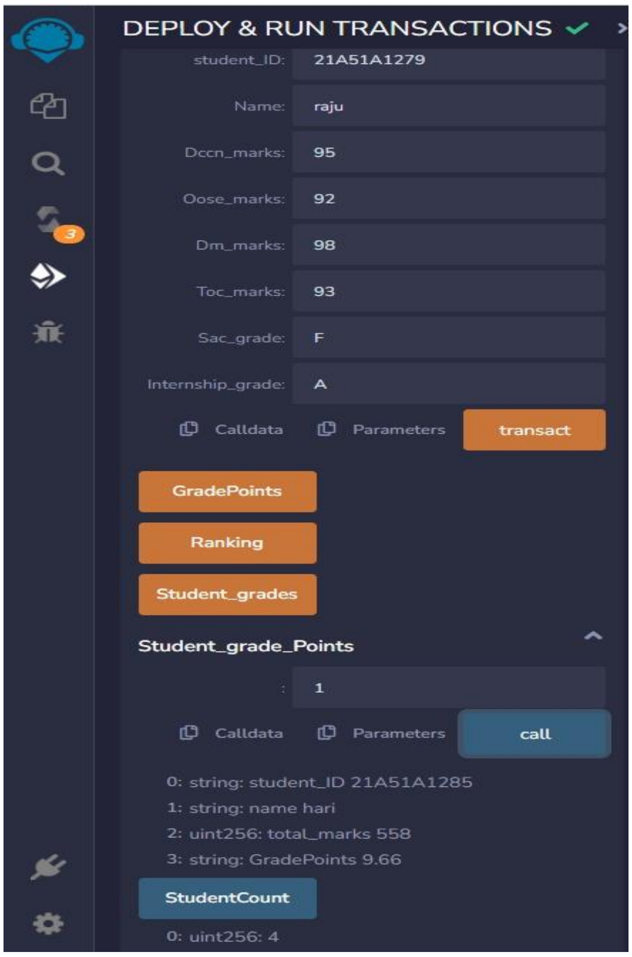
Dm_marks: 98

Toc_marks: 93

Sac_grade: F

Internship_grade: A

```
0: uint256: 4
```



DEPLOY & RUN TRANSACTIONS

student_ID: 21A51A1279

Name: raju

Dccn_marks: 95

Oose_marks: 92

Dm_marks: 98

Toc_marks: 93

Sac_grade: F

Internship_grade: A

Student_grade_Points

: 1

```
0: string: student_ID 21A51A1285
1: string: name hari
2: uint256: total_marks 558
3: string: GradePoints 9.66
```

```
0: uint256: 4
```

CONCLUSION:

Blockchain-based Student Result Management Systems are becoming increasingly popular due to their enhanced security and transparency, ensuring the reliability of student records and results. They are also efficient and user-friendly, making it easier for students and faculties to manage and access academic information.

REFERENCES

Using Blockchain in University Management Systems (researchgate.net)

A blockchain-based models for student information systems - ScienceDirect

Blockchain Based Student Information Management System | IEEE Conference Publication | IEEE Xplore