// SPDX-License-Identifier: MIT

 pragma solidity ^0.8.0;

 contract StudentData {

    // Structure to store student details

    struct Student {

        uint id;

        string name;

        uint8 age;

        string department;

    }

    // Array to store the list of students

    Student[] public students;

    // Mapping to track if a student with an id already exists

    mapping(uint => bool) public studentExists;

    // Event to emit student added

    event StudentAdded(uint id, string name, uint8 age, string department);

    // Function to add a new student

    function addStudent(uint \_id, string memory \_name, uint8 \_age, string memory \_department) public {

 // Ensure that the student does not already exist

        require(!studentExists[\_id], "Student with this ID already exists.");

 // Adding student to the array

        students.push(Student(\_id, \_name, \_age, \_department));

        studentExists[\_id] = true;

 // Emit the event

        emit StudentAdded(\_id, \_name, \_age, \_department);

    }

    // Function to get student details by index

 //Ensure the index is within bounds

    function getStudent(uint index) public view returns (uint, string memory, uint8, string memory) {

        require(index < students.length, "Invalid student index.");

        Student memory s = students[index];

        return (s.id, s.name, s.age, s.department);

    }

    // Function to get the total number of students

    function getTotalStudents() public view returns (uint) {

        return students.length;

    }

    // Receive function to handle direct Ether transfers (without data)

    receive() external payable {

 //The contract can receive Ether through this function

    }

    // Fallback function to handle other types of Ether transfers or undefined calls

    fallback() external payable {

 // The contract can also receive Ether and handle calls to undefined functions

 }

 }

Output-

Add student=1, "Alice", 20, "Computer Science"