// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract StudentData {

    // Structure to store student details

    struct Student {

        uint id;

        string name;

        uint age;

        string course;

    }

    // Array to store multiple students

    Student[] public students;

    uint public studentCount = 0;

    // Mapping to check if a student ID already exists

    mapping(uint => bool) private studentExists;

    // Event to log the addition of a new student

    event StudentAdded(uint id, string name, uint age, string course);

    // Function to add a new student

    function addStudent(string memory \_name, uint \_age, string memory \_course) public {

        require(bytes(\_name).length > 0, "Name is required");

        require(\_age > 0, "Age must be greater than zero");

        // Increment student count for unique ID

        studentCount++;

        uint studentId = studentCount;

        // Ensure the student ID is unique

        require(!studentExists[studentId], "Student ID already exists");

        // Add new student to the array

        students.push(Student(studentId, \_name, \_age, \_course));

        studentExists[studentId] = true;

        // Emit the StudentAdded event

        emit StudentAdded(studentId, \_name, \_age, \_course);

    }

    // Function to get details of a student by index

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

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

        Student memory student = students[index];

        return (student.id, student.name, student.age, student.course);

    }

    // Fallback function to handle any Ether sent to the contract

    fallback() external payable {

        // This function is called if no other function matches the call or when Ether is sent without data

    }

    // Receive function to handle direct Ether transfer

    receive() external payable {}

    // Function to check the contract's balance

    function getContractBalance() public view returns (uint) {

        return address(this).balance;

    }

}