StudentData.sol

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

// Defining a contract named StudentRecords

contract StudentRecords {

    // Struct to store student information

    struct Student {

        string name;

        uint age;

        string course;

    }

    // Array to hold the list of students

    Student[] public students;

    // Mapping to track student index by name for easy removal

    mapping(string => uint) private studentIndex;

    // Function to add a new student

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

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

        studentIndex[\_name] = students.length - 1; // Store index for easy removal

    }

    // Function to remove a student by name

    function removeStudent(string memory \_name) public {

        require(students.length > 0, "No students to remove."); // Ensure the array is not empty

        uint index = studentIndex[\_name];

        require(index < students.length, "Student does not exist.");

        // Move the last student into the place of the removed student to maintain array integrity

        students[index] = students[students.length - 1];

        // Update index in the mapping

        studentIndex[students[index].name] = index;

        students.pop(); // Remove the last element

    }

    // Fallback function to receive Ether and log it

    fallback() external payable {

        emit LogFallback(msg.sender, msg.value); // Simple logging mechanism for the fallback function

    }

    // Receive function to receive Ether directly

    receive() external payable {

        emit LogReceived(msg.sender, msg.value);

    }

    // Function to check the balance of the contract

    function getBalance() public view returns (uint) {

        return address(this).balance; // Measured in Wei (the smallest unit of Ether)

    }

    // Payable function to allow users to pay for adding a student record

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

        require(msg.value > 0, "You need to send some Ether to add a student record.");

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

        studentIndex[\_name] = students.length - 1;

    }

    // Event to log fallback transactions

    event LogFallback(address indexed sender, uint value);

    // Event to log received Ether

    event LogReceived(address indexed sender, uint value);

}