**#BT16**

**// SPDX-License-Identifier: MIT**

**pragma solidity ^0.8.0;**

**contract StudentData {**

**// Structure to store student data**

**struct Student {**

**uint id;**

**string name;**

**uint age;**

**uint grade;**

**}**

**// Array to store multiple Student structures**

**Student[] public students;**

**// Mapping to keep track of student IDs to avoid duplicates**

**mapping(uint => bool) private studentExists;**

**// Event for adding a new student**

**event StudentAdded(uint id, string name, uint age, uint grade);**

**// Function to add a new student**

**function addStudent(uint \_id, string memory \_name, uint \_age, uint \_grade) public {**

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

**Student memory newStudent = Student({**

**id: \_id,**

**name: \_name,**

**age: \_age,**

**grade: \_grade**

**});**

**students.push(newStudent);**

**studentExists[\_id] = true;**

**emit StudentAdded(\_id, \_name, \_age, \_grade);**

**}**

**// Function to retrieve a student's data by array index**

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

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

**Student memory student = students[index];**

**return (student.id, student.name, student.age, student.grade);**

**}**

**// Function to get the total number of students**

**function getStudentCount() public view returns (uint) {**

**return students.length;**

**}**

**// Receive function to handle simple Ether transfers**

**receive() external payable {**

**// Logic for receiving Ether (if any) can go here**

**}**

**// Fallback function to handle unexpected calls or Ether with data**

**fallback() external payable {**

**// Additional logic for fallback, if needed**

**}**

**}**