```
using System.ComponentModel.DataAnnotations;
namespace RainbowSchoool.Models
{
    public class Student
    {
        [Key]
        public int StudentId { get; set; }
        [Required]
        public string StudentName { get; set; } = null!;
        [Required]
        public int StudentClass { get; set; }
        [Required]
        public string Subject { get; set; } = null!;
        public int TeacherId { get; set; }
        public int? SubjectMark { get; set; }

    }
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using RainbowSchoool.Data;
using RainbowSchoool.Models;

namespace RainbowSchoool.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class StudentsController : ControllerBase
    {
        private readonly RainbowSchooolContext _context;

        public StudentsController(RainbowSchooolContext context)
        {
            _context = context;
        }

        // GET: api/Students
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Student>>> GetStudent()
        {
          if (_context.Student == null)
          {
              return NotFound();
          }
            return await _context.Student.ToListAsync();
        }

        // GET: api/Students/5
        [HttpGet("{id}")]
        public async Task<ActionResult<Student>> GetStudent(int id)
        {
          if (_context.Student == null)
          {
              return NotFound();
          }
            var student = await _context.Student.FindAsync(id);

            if (student == null)
            {
                return NotFound();
            }

            return student;
        }

        // PUT: api/Students/5
        // To protect from overposting attacks, see 
https://go.microsoft.com/fwlink/?linkid=2123754
        [HttpPut("{id}")]
        public async Task<IActionResult> PutStudent(int id, Student student)
        {
            if (id != student.StudentId)
            {
                return BadRequest();
            }

            _context.Entry(student).State = EntityState.Modified;

            try
            {
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!StudentExists(id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }

            return NoContent();
        }

        // POST: api/Students
        // To protect from overposting attacks, see 
https://go.microsoft.com/fwlink/?linkid=2123754
        [HttpPost]
        public async Task<ActionResult<Student>> PostStudent(Student student)
        {
          if (_context.Student == null)
          {
              return Problem("Entity set 'RainbowSchooolContext.Student'  is 
null.");
          }
            _context.Student.Add(student);
            await _context.SaveChangesAsync();

            return CreatedAtAction("GetStudent", new { id = student.StudentId },
student);
        }

        // DELETE: api/Students/5
        [HttpDelete("{id}")]
        public async Task<IActionResult> DeleteStudent(int id)
        {
            if (_context.Student == null)
            {
                return NotFound();
            }
            var student = await _context.Student.FindAsync(id);
            if (student == null)
            {
                return NotFound();
            }

            _context.Student.Remove(student);
            await _context.SaveChangesAsync();

            return NoContent();
        }
        [HttpPost("AddOrUpdateMarks/{id}")]
        public async Task<IActionResult> AddOrUpdateMarks(int id, int? SubjectMark)
        {
            var student = await _context.Student.FindAsync(id);

            if (student == null)
            {
                return NotFound();
            }

            // Update the SubjectMark property if a value is provided
            if (SubjectMark.HasValue)
            {
                student.SubjectMark = SubjectMark.Value;
            }

            try
            {
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!StudentExists(id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }

            return NoContent();
        }


        private bool StudentExists(int id)
        {
            return (_context.Student?.Any(e => e.StudentId ==
id)).GetValueOrDefault();
        }
    }
}
```