using System; using System.Collections.Generic;

namespace WebAPIAddSubjectMarks.Models { public partial class Student { public int StuId { get; set; } public string? StuName { get; set; } public int? StuClass { get; set; } } }

using System; using System.Collections.Generic;

namespace WebAPIAddSubjectMarks.Models { public partial class Subject { public int SubId { get; set; } public string? SubName { get; set; } } }

{ "Logging": { "LogLevel": { "Default": "Information", "Microsoft.AspNetCore": "Warning" } }, "AllowedHosts": "*" }

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers(); // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle builder.Services.AddEndpointsApiExplorer(); builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline. if (app.Environment.IsDevelopment()) { app.UseSwagger(); app.UseSwaggerUI(); }

app.UseStaticFiles(); app.UseAuthorization();

app.MapControllers();

app.Run();

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 WebAPIAddSubjectMarks.Models;

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

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

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

// GET: api/Students/5 [HttpGet("{id}")] public async Task<ActionResult<Student» GetStudent(int id) { if (_context.Students == null) { return NotFound(); } var student = await _context.Students.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.StuId) { 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.Students == null) { return Problem("Entity set 'School_ProjectContext.Students' is null."); } _context.Students.Add(student); try { await _context.SaveChangesAsync(); } catch (DbUpdateException) { if (StudentExists(student.StuId)) { return Conflict(); } else { throw; } }

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

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

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

return NoContent(); }

private bool StudentExists(int id) { return (_context.Students?.Any(e => e.StuId == id)).GetValueOrDefault(); } } }

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 WebAPIAddSubjectMarks.Models;

namespace WebAPIAddSubjectMarks.Controllers { [Route("api/[controller]")] [ApiController] public class SubjectsController : ControllerBase { private readonly School_ProjectContext _context;

public SubjectsController(School_ProjectContext context) { _context = context; }

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

// GET: api/Subjects/5 [HttpGet("{id}")] public async Task<ActionResult<Subject» GetSubject(int id) { if (_context.Subjects == null) { return NotFound(); } var subject = await _context.Subjects.FindAsync(id);

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

return subject; }

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

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

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

return NoContent(); }

// POST: api/Subjects // To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754 [HttpPost] public async Task<ActionResult<Subject» PostSubject(Subject subject) { if (_context.Subjects == null) { return Problem("Entity set 'School_ProjectContext.Subjects' is null."); } _context.Subjects.Add(subject); try { await _context.SaveChangesAsync(); } catch (DbUpdateException) { if (SubjectExists(subject.SubId)) { return Conflict(); } else { throw; } }

return CreatedAtAction("GetSubject", new { id = subject.SubId }, subject); }

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

_context.Subjects.Remove(subject); await _context.SaveChangesAsync();

return NoContent(); }

private bool SubjectExists(int id) { return (_context.Subjects?.Any(e => e.SubId == id)).GetValueOrDefault(); } } }