using System; using System.Collections.Generic;

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

using System; using System.Collections.Generic;

namespace WebAppManageSchoolDatabase.Models { public partial class Class { public int Class1 { 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 WebAppManageSchoolDatabase.Models;

namespace WebAppManageSchoolDatabase.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 WebAppManageSchoolDatabase.Models;

namespace WebAppManageSchoolDatabase.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(); } } }

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 WebAppManageSchoolDatabase.Models;

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

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

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

// GET: api/Classes/5 [HttpGet("{id}")] public async Task<ActionResult<Class» GetClass(int id) { if (_context.Classes == null) { return NotFound(); } var @class = await _context.Classes.FindAsync(id);

if (@class == null) { return NotFound(); }

return @class; }

// PUT: api/Classes/5 // To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754 [HttpPut("{id}")] pub-

```
lic async Task<IActionResult> PutClass(int id, Class @class) { if (id !=
@class.Class1) { return BadRequest(); }

_context.Entry(@class).State = EntityState.Modified;

try { await _context.SaveChangesAsync(); } catch (DbUpdateConcurrencyEx-
ception) { if (!ClassExists(id)) { return NotFound(); } else { throw; } }

return NoContent(); }

// POST: api/Classes // To protect from overposting attacks, see
https://go.microsoft.com/fwlink/?linkid=2123754 [HttpPost] public async
Task<ActionResult<Class» PostClass(Class @class) { if (_context.Classes ==
null) { return Problem(”Entity set 'School_ProjectContext.Classes' is null.”);
} _context.Classes.Add(@class); try { await _context.SaveChangesAsync();
} catch (DbUpdateException) { if (ClassExists(@class.Class1)) { return
Conflict(); } else { throw; } }

return CreatedAtAction(”GetClass”, new { id = @class.Class1 }, @class); }

// DELETE: api/Classes/5 [HttpDelete(”{id}”)] public async Task<IActionResult>
DeleteClass(int id) { if (_context.Classes == null) { return NotFound(); } var
@class = await _context.Classes.FindAsync(id); if (@class == null) { return
NotFound(); }

_context.Classes.Remove(@class); await _context.SaveChangesAsync();

return NoContent(); }

private bool ClassExists(int id) { return (_context.Classes?.Any(e => e.Class1
== id)).GetValueOrDefault(); } } }

using System; using System.Collections.Generic; using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata;

namespace WebAppManageSchoolDatabase.Models { public partial class
School_ProjectContext : DbContext { public School_ProjectContext() { }

public School_ProjectContext(DbContextOptions<School_ProjectContext>
options) : base(options) { }

public virtual DbSet<Class> Classes { get; set; } = null!; public virtual Db-
Set<Student> Students { get; set; } = null!; public virtual DbSet<Subject>
Subjects { get; set; } = null!;

protected override void OnConfiguring(DbContextOptionsBuilder options-
Builder) { if (!optionsBuilder.IsConfigured) { #warning To protect po-
tentially sensitive information in your connection string, you should
move it out of source code. You can avoid scaffolding the connection
string by using the Name= syntax to read it from configuration - see
https://go.microsoft.com/fwlink/?linkid=2131148. For more guidance on stor-
ing connection strings, see http://go.microsoft.com/fwlink/?LinkId=723263.
optionsBuilder.UseSqlServer(”server=DESKTOP-U064AL2;database=School_Project;trusted_connection=tr
} }

protected override void OnModelCreating(ModelBuilder modelBuilder) { modelBuilder.Entity<Class>(entity => { entity.HasKey(e => e.Class1) .HasName(”PK___classes___71DF78ECC4C206B5”);

entity.ToTable(”classes”);

entity.Property(e => e.Class1) .ValueGeneratedNever() .HasColumnName(”class”); });

modelBuilder.Entity<Student>(entity => { entity.HasKey(e => e.StuId) .HasName(”PK___student___E53CAB210370AD06”);

entity.ToTable(”student”);

entity.Property(e => e.StuId) .ValueGeneratedNever() .HasColumnName(”stu_id”);

entity.Property(e => e.StuClass).HasColumnName(”stu_class”);

entity.Property(e => e.StuName) .HasMaxLength(20) .IsUnicode(false) .HasColumnName(”stu_name”); });

modelBuilder.Entity<Subject>(entity => { entity.HasKey(e => e.SubId) .HasName(”PK___subjects___694106B0F1FB024F”);

entity.ToTable(”subjects”);

entity.Property(e => e.SubId) .ValueGeneratedNever() .HasColumnName(”sub_id”);

entity.Property(e => e.SubName) .HasMaxLength(20) .IsUnicode(false) .HasColumnName(”sub_name”); });

OnModelCreatingPartial(modelBuilder); }

partial void OnModelCreatingPartial(ModelBuilder modelBuilder); } }
```