```
using Microsoft.EntityFrameworkCore; using WebAppSchoolDb.Models;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container. builder.Services.AddDbContext<School_ProjectContext>(options
=> options.UseSqlServer(builder.Configuration.GetConnectionString(”EFConStr”)));
builder.Services.AddControllersWithViews();

var app = builder.Build();

// Configure the HTTP request pipeline. if (!app.Environment.IsDevelopment())
{ app.UseExceptionHandler(”/Home/Error”); } app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute( name: ”default”, pattern: ”{controller=Home}/{action=Index}/{id?}”);

app.Run();

<!DOCTYPE html> <html lang=”en”> <head> <meta charset=”utf-
8” /> <meta name=”viewport” content=”width=device-width, initial-
scale=1.0” /> <title>@ViewData[”Title”] - WebAppSchoolDb</title>
<link rel=”stylesheet” href=”~/lib/bootstrap/dist/css/bootstrap.min.css”
/> <link rel=”stylesheet” href=”~/css/site.css” asp-append-version=”true”
/> <link rel=”stylesheet” href=”~/WebAppSchoolDb.styles.css” asp-append-
version=”true” /> </head> <body> <header> <nav class=”navbar navbar-
expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-
shadow mb-3”> <div class=”container-fluid”> <a class=”navbar-brand” asp-
area=”” asp-controller=”Home” asp-action=”Index”>WebAppSchoolDb</a>
<button class=”navbar-toggler” type=”button” data-bs-toggle=”collapse”
data-bs-target=”.navbar-collapse” aria-controls=”navbarSupportedContent”
aria-expanded=”false” aria-label=”Toggle navigation”> <span class=”navbar-
toggler-icon”></span> </button> <div class=”navbar-collapse collapse
d-sm-inline-flex justify-content-between”> <ul class=”navbar-nav flex-
grow-1”> <li class=”nav-item”> <a class=”nav-link text-dark” asp-area=””
asp-controller=”Home” asp-action=”Index”>Home</a> </li> <li class=”nav-
item”> <a class=”nav-link text-dark” asp-area=”” asp-controller=”Home”
asp-action=”Privacy”>Privacy</a> </li>

<li class=”nav-item”> <a class=”nav-link text-dark” asp-area=”” asp-
controller=”Students” asp-action=”Index”>Student</a> </li> <li class=”nav-
item”> <a class=”nav-link text-dark” asp-area=”” asp-controller=”Subjects”
asp-action=”Index”>Subject</a> </li> <li class=”nav-item”> <a
class=”nav-link text-dark” asp-area=”” asp-controller=”Classes” asp-
action=”Index”>Class</a> </li>

</ul> </div> </div> </nav> </header> <div class=”container”> <main
role=”main” class=”pb-3”> @RenderBody() </main> </div>
```

<footer class=”border-top footer text-muted”> <div class=”container”> &copy; 2024 - WebAppSchoolDb - <a asp-area=”” asp-controller=”Home” asp-action=”Privacy”>Privacy</a> </div> </footer> <script src=”~/lib/jquery/dist/jquery.min.js”></script> <script src=”~/lib/bootstrap/dist/js/bootstrap.bundle.min.js”></script> <script src=”~/js/site.js” asp-append-version=”true”></script> @await RenderSectionAsync(”Scripts”, required: false) </body> </html>

using System; using System.Collections.Generic; using System.Linq; using System.Threading.Tasks; using Microsoft.AspNetCore.Mvc; using Microsoft.AspNetCore.Mvc.Rendering; using Microsoft.EntityFrameworkCore; using WebAppSchoolDb.Models;

namespace WebAppSchoolDb.Controllers { public class StudentsController : Controller { private readonly School_ProjectContext _context;

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

// GET: Students public async Task<IActionResult> Index() { return _context.Students != null ? View(await _context.Students.ToListAsync()) : Problem(”Entity set ’School_ProjectContext.Students’ is null.”); }

// GET: Students/Details/5 public async Task<IActionResult> Details(int? id) { if (id == null || _context.Students == null) { return NotFound(); }

var student = await _context.Students .FirstOrDefaultAsync(m => m.StuId == id); if (student == null) { return NotFound(); }

return View(student); }

// GET: Students/Create public IActionResult Create() { return View(); }

// POST: Students/Create // To protect from overposting attacks, enable the specific properties you want to bind to. // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598. [HttpPost] [ValidateAntiForgeryToken] public async Task<IActionResult> Create([Bind(”StuId,StuName,StuClass”)] Student student) { if (ModelState.IsValid) { _context.Add(student); await _context.SaveChangesAsync(); return RedirectToAction(nameof(Index)); } return View(student); }

// GET: Students/Edit/5 public async Task<IActionResult> Edit(int? id) { if (id == null || _context.Students == null) { return NotFound(); }

var student = await _context.Students.FindAsync(id); if (student == null) { return NotFound(); } return View(student); }

// POST: Students/Edit/5 // To protect from overposting attacks, enable the specific properties you want to bind to. // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598. [HttpPost] [ValidateAntiForgeryToken] public async Task<IActionResult> Edit(int id, [Bind(”StuId,StuName,StuClass”)] Student student) { if (id != student.StuId) { return NotFound(); }

if (ModelState.IsValid) { try { _context.Update(student); await _context.SaveChangesAsync(); } catch (DbUpdateConcurrencyException) { if (!StudentExists(student.StuId)) { return NotFound(); } else { throw; } } return RedirectToAction(nameof(Index)); } return View(student); }

// GET: Students/Delete/5 public async Task<IActionResult> Delete(int? id) { if (id == null || _context.Students == null) { return NotFound(); }

var student = await _context.Students .FirstOrDefaultAsync(m => m.StuId == id); if (student == null) { return NotFound(); }

return View(student); }

// POST: Students/Delete/5 [HttpPost, ActionName(”Delete”)] [ValidateAntiForgeryToken] public async Task<IActionResult> DeleteConfirmed(int id) { if (_context.Students == null) { return Problem(”Entity set ’School_ProjectContext.Students’ is null.”); } var student = await _context.Students.FindAsync(id); if (student != null) { _context.Students.Remove(student); }

await _context.SaveChangesAsync(); return RedirectToAction(nameof(Index)); }

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.Mvc; using Microsoft.AspNetCore.Mvc.Rendering; using Microsoft.EntityFrameworkCore; using WebAppSchoolDb.Models;

namespace WebAppSchoolDb.Controllers { public class SubjectsController : Controller { private readonly School_ProjectContext _context;

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

// GET: Subjects public async Task<IActionResult> Index() { return _context.Subjects != null ? View(await _context.Subjects.ToListAsync()) : Problem(”Entity set ’School_ProjectContext.Subjects’ is null.”); }

// GET: Subjects/Details/5 public async Task<IActionResult> Details(int? id) { if (id == null || _context.Subjects == null) { return NotFound(); }

var subject = await _context.Subjects .FirstOrDefaultAsync(m => m.SubId == id); if (subject == null) { return NotFound(); }

return View(subject); }

// GET: Subjects/Create public IActionResult Create() { return View(); }

// POST: Subjects/Create // To protect from overposting attacks, enable the specific properties you want to bind to. // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598. [HttpPost] [ValidateAntiForgeryToken] public async Task<IActionResult> Create([Bind("SubId,SubName")] Subject subject) { if (ModelState.IsValid) { _context.Add(subject); await _context.SaveChangesAsync(); return RedirectToAction(nameof(Index)); } return View(subject); }

// GET: Subjects/Edit/5 public async Task<IActionResult> Edit(int? id) { if (id == null || _context.Subjects == null) { return NotFound(); }

var subject = await _context.Subjects.FindAsync(id); if (subject == null) { return NotFound(); } return View(subject); }

// POST: Subjects/Edit/5 // To protect from overposting attacks, enable the specific properties you want to bind to. // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598. [HttpPost] [ValidateAntiForgeryToken] public async Task<IActionResult> Edit(int id, [Bind("SubId,SubName")] Subject subject) { if (id != subject.SubId) { return NotFound(); }

if (ModelState.IsValid) { try { _context.Update(subject); await _context.SaveChangesAsync(); } catch (DbUpdateConcurrencyException) { if (!SubjectExists(subject.SubId)) { return NotFound(); } else { throw; } } return RedirectToAction(nameof(Index)); } return View(subject); }

// GET: Subjects/Delete/5 public async Task<IActionResult> Delete(int? id) { if (id == null || _context.Subjects == null) { return NotFound(); }

var subject = await _context.Subjects .FirstOrDefaultAsync(m => m.SubId == id); if (subject == null) { return NotFound(); }

return View(subject); }

// POST: Subjects/Delete/5 [HttpPost, ActionName("Delete")] [ValidateAntiForgeryToken] public async Task<IActionResult> DeleteConfirmed(int id) { if (_context.Subjects == null) { return Problem("Entity set 'School_ProjectContext.Subjects' is null."); } var subject = await _context.Subjects.FindAsync(id); if (subject != null) { _context.Subjects.Remove(subject); }

await _context.SaveChangesAsync(); return RedirectToAction(nameof(Index)); }

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.Mvc; using Microsoft.AspNetCore.Mvc.Rendering; using Microsoft.EntityFrameworkCore; using WebAppSchoolDb.Models;

namespace WebAppSchoolDb.Controllers { public class ClassesController : Controller { private readonly School_ProjectContext _context;

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

// GET: Classes public async Task<IActionResult> Index() { return _context.Classes != null ? View(await _context.Classes.ToListAsync()) : Problem(”Entity set ’School_ProjectContext.Classes’ is null.”); }

// GET: Classes/Details/5 public async Task<IActionResult> Details(int? id) { if (id == null || _context.Classes == null) { return NotFound(); }

var @class = await _context.Classes .FirstOrDefaultAsync(m => m.Class1 == id); if (@class == null) { return NotFound(); }

return View(@class); }

// GET: Classes/Create public IActionResult Create() { return View(); }

// POST: Classes/Create // To protect from overposting attacks, enable the specific properties you want to bind to. // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598. [HttpPost] [ValidateAntiForgeryToken] public async Task<IActionResult> Create([Bind(”Class1”)] Class @class) { if (ModelState.IsValid) { _context.Add(@class); await _context.SaveChangesAsync(); return RedirectToAction(nameof(Index)); } return View(@class); }

// GET: Classes/Edit/5 public async Task<IActionResult> Edit(int? id) { if (id == null || _context.Classes == null) { return NotFound(); }

var @class = await _context.Classes.FindAsync(id); if (@class == null) { return NotFound(); } return View(@class); }

// POST: Classes/Edit/5 // To protect from overposting attacks, enable the specific properties you want to bind to. // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598. [HttpPost] [ValidateAntiForgeryToken] public async Task<IActionResult> Edit(int id, [Bind(”Class1”)] Class @class) { if (id != @class.Class1) { return NotFound(); }

if (ModelState.IsValid) { try { _context.Update(@class); await _context.SaveChangesAsync(); } catch (DbUpdateConcurrencyException) { if (!ClassExists(@class.Class1)) { return NotFound(); } else { throw; } } return RedirectToAction(nameof(Index)); } return View(@class); }

// GET: Classes/Delete/5 public async Task<IActionResult> Delete(int? id) { if (id == null || _context.Classes == null) { return NotFound(); }

var @class = await _context.Classes .FirstOrDefaultAsync(m => m.Class1 == id); if (@class == null) { return NotFound(); }

return View(@class); }

// POST: Classes/Delete/5 [HttpPost, ActionName(”Delete”)] [ValidateAntiForgeryToken] public async Task<IActionResult> DeleteConfirmed(int id) { if (_context.Classes == null) { return Problem("Entity set 'School_ProjectContext.Classes' is null."); } var @class = await _context.Classes.FindAsync(id); if (@class != null) { _context.Classes.Remove(@class); }

await _context.SaveChangesAsync(); return RedirectToAction(nameof(Index)); }

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