using Microsoft.EntityFrameworkCore; using Project_1.Models;

var builder = WebApplication.CreateBuilder(args);

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

{ "Logging": { "LogLevel": { "Default": "Information", "Microsoft.AspNetCore": "Warning" } }, "ConnectionStrings": { "EFConStr": "server=DESKTOP-U064AL2;database=EMSdatabase;trusted_connection=true;" }, "AllowedHosts": "*" }

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

namespace Project_1.Models { public partial class EMSdatabaseContext : DbContext { public EMSdatabaseContext() { }

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

public virtual DbSet<DeptMaster> DeptMasters { get; set; } = null!; public virtual DbSet<EmpProfile> EmpProfiles { get; set; } = null!;

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) { if (!optionsBuilder.IsConfigured) { #warning To protect potentially 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 storing connection strings, see http://go.microsoft.com/fwlink/?LinkId=723263. optionsBuilder.UseSqlServer("server=DESKTOP-U064AL2;database=EMSdatabase;trusted_connection=true } }

protected override void OnModelCreating(ModelBuilder modelBuilder) { modelBuilder.Entity<DeptMaster>(entity => { entity.HasKey(e => e.DeptCode) .HasName("PK___DeptMast___BB9B955122785A77");

entity.ToTable("DeptMaster");

entity.Property(e => e.DeptCode).ValueGeneratedNever();

entity.Property(e => e.DeptName) .HasMaxLength(20) .IsUnicode(false); });

modelBuilder.Entity<EmpProfile>(entity => { entity.HasKey(e => e.EmpCode) .HasName("PK___EmpProfi___7DA847CB85A9D4D3");

entity.ToTable("EmpProfile");

entity.Property(e => e.EmpCode).ValueGeneratedNever();

entity.Property(e => e.DateOfBirth).HasColumnType("datetime");

entity.Property(e => e.Email) .HasMaxLength(20) .IsUnicode(false);

entity.Property(e => e.EmpName) .HasMaxLength(20) .IsUnicode(false);

entity.HasOne(d => d.DeptCodeNavigation) .WithMany(p => p.EmpProfiles) .HasForeignKey(d => d.DeptCode) .HasConstraintName("FK___EmpProfil___DeptC___4BAC3F29"); });

OnModelCreatingPartial(modelBuilder); }

partial void OnModelCreatingPartial(ModelBuilder modelBuilder); } }

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 Project_1.Models;

namespace Project_1.Controllers { [Route("api/[controller]")] [ApiController] public class DeptMastersController : ControllerBase { private readonly EMSdatabaseContext _context;

public DeptMastersController(EMSdatabaseContext context) { _context = context; }

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

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

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

return deptMaster; }

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

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

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

return NoContent(); }

// POST: api/DeptMasters // To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754 [HttpPost] public async Task<ActionResult<DeptMaster» PostDeptMaster(DeptMaster deptMaster) { if (_context.DeptMasters == null) { return Problem("Entity set 'EMSdatabaseContext.DeptMasters' is null."); } _context.DeptMasters.Add(deptMaster); try { await _context.SaveChangesAsync(); } catch (DbUpdateException) { if (DeptMasterExists(deptMaster.DeptCode)) { return Conflict(); } else { throw; } }

return CreatedAtAction("GetDeptMaster", new { id = deptMaster.DeptCode }, deptMaster); }

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

_context.DeptMasters.Remove(deptMaster); await _context.SaveChangesAsync();

return NoContent(); }

private bool DeptMasterExists(int id) { return (_context.DeptMasters?.Any(e => e.DeptCode == 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 Project_1.Models;

namespace Project_1.Controllers { [Route("api/[controller]")] [ApiController] public class EmpProfilesController : ControllerBase { private readonly EMSdatabaseContext _context;

public EmpProfilesController(EMSdatabaseContext context) { _context = context; }

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

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

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

return empProfile; }

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

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

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

return NoContent(); }

// POST: api/EmpProfiles // To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754 [HttpPost] public async Task<ActionResult<EmpProfile» PostEmpProfile(EmpProfile empProfile) { if (_context.EmpProfiles == null) { return Problem("Entity set 'EMSdatabaseContext.EmpProfiles' is null."); } _context.EmpProfiles.Add(empProfile); try { await _context.SaveChangesAsync(); } catch (DbUpdateException) { if (EmpProfileExists(empProfile.EmpCode)) { return Conflict(); } else { throw; } }

return CreatedAtAction("GetEmpProfile", new { id = empProfile.EmpCode }, empProfile); }

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

_context.EmpProfiles.Remove(empProfile); await _context.SaveChangesAsync();

return NoContent(); }

private bool EmpProfileExists(int id) { return (_context.EmpProfiles?.Any(e => e.EmpCode == id)).GetValueOrDefault(); } } }