```
using System.ComponentModel.DataAnnotations;

namespace Simplona.Models
{
    public class EmpProfile
    {
        [Key]
        public int EmpCode { get; set; }
        public DateTime DateOfBirth { get; set; }
        public string EmpName { get; set; }
        public string Email { get; set; }
        public int DeptCode { get; set; }
        public virtual DeptMaster DeptMaster { 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 Simplona.Data;
using Simplona.Models;

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

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

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

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

```
          }
            var empProfile = await _context.EmpProfile.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.EmpProfile == null)
          {
              return Problem("Entity set 'SimplonaContext.EmpProfile'  is null.");
          }
            _context.EmpProfile.Add(empProfile);
            await _context.SaveChangesAsync();
```

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

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

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

            return NoContent();
        }

        private bool EmpProfileExists(int id)
        {
            return (_context.EmpProfile?.Any(e => e.EmpCode ==
id)).GetValueOrDefault();
        }
    }
}
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.DependencyInjection;
using Simplona.Data;
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddDbContext<SimplonaContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("SimplonaContext") ??
throw new InvalidOperationException("Connection string 'SimplonaContext' not
found.")));

// 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.UseAuthorization();

app.MapControllers();

app.Run();
```

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "SimplonaContext": "Server=DESKTOP-
FDHH6M8;Database=SimplonaDb;Trusted_Connection=True;MultipleActiveResultSets=true;Tr
ustServerCertificate=True;"
  }
}
```