Github Link : https://github.com/sivakumar999/PracticeProject4.git

Main files i am writing here

Model file

```
namespace RainbowScholApi.Models
{
    public class RainbowSchool
    {
        public int Id { get; set; }
        public string StudentName { get; set; }
        public string Subject { get; set; }
        public double SubjectMarks { get; set; }
    }
}
```

Controller

```
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 RainbowScholApi.Data;
using RainbowScholApi.Models;

namespace RainbowScholApi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class RainbowSchoolsController : ControllerBase
    {
        private readonly RainbowScholApiContext _context;

        public RainbowSchoolsController(RainbowScholApiContext context)
        {
            _context = context;
        }

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

        // GET: api/RainbowSchools/5
        [HttpGet("{id}")]
        public async Task<ActionResult<RainbowSchool>> GetRainbowSchool(int id)
        {
          if (_context.RainbowSchool == null)
          {
              return NotFound();
          }
            var rainbowSchool = await _context.RainbowSchool.FindAsync(id);

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

            return rainbowSchool;
        }

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

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

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

            return NoContent();
        }

        // POST: api/RainbowSchools
        // To protect from overposting attacks, see
https://go.microsoft.com/fwlink/?linkid=2123754
        [HttpPost]
        public async Task<ActionResult<RainbowSchool>>
PostRainbowSchool(RainbowSchool rainbowSchool)
        {
          if (_context.RainbowSchool == null)
          {
              return Problem("Entity set 'RainbowScholApiContext.RainbowSchool'
is null.");
          }
            _context.RainbowSchool.Add(rainbowSchool);
            await _context.SaveChangesAsync();

            return CreatedAtAction("GetRainbowSchool", new { id =
rainbowSchool.Id }, rainbowSchool);
        }

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

            _context.RainbowSchool.Remove(rainbowSchool);
            await _context.SaveChangesAsync();

            return NoContent();
        }

        private bool RainbowSchoolExists(int id)
        {
            return (_context.RainbowSchool?.Any(e => e.Id == 
id)).GetValueOrDefault();
        }
    }
}
```

Program.cs

```
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.DependencyInjection;
using RainbowScholApi.Data;
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddDbContext<RainbowScholApiContext>(options =>

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

// Add services to the container.

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}


// Configure the HTTP request pipeline.

app.UseAuthorization();

app.MapControllers();

app.Run();
```