```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace Phase2SchooDbWebAPI.Models
{
    [Table("Students")]
    public class Students
    {
        [Key]
        public int StudentID { get; set; }

        [Required]
        [StringLength(50)]
        public string FirstName { get; set; }

        [Required]
        [StringLength(50)]
        public string LastName { get; set; }

        [Required]
        [DataType(DataType.Date)]
        public DateTime DateOfBirth { get; set; }

        [Required]
        [StringLength(10)]
        public string Gender { get; set; }

        [Required]
        [StringLength(100)]
        public string Address { get; set; }
    }
}

using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace Phase2SchoolDbWebAPI.Models
{
    [Table("Subjects")]
    public class Subject
    {
        [Key]
        public int SubID { get; set; }

        [Required]
        [StringLength(50)]
        public string SubName { 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 Phase2SchooDbWebAPI.Data;
using Phase2SchooDbWebAPI.Models;

namespace Phase2SchooDbWebAPI.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
```

```
    public class StudentsController : ControllerBase
    {
        private readonly APISchooDbContext _context;

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

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

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

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

            return students;
        }

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

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

            try
            {
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!StudentsExists(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<Students>> PostStudents(Students students)
        {
          if (_context.Students == null)
          {
              return Problem("Entity set 'APISchooDbContext.Students'  is
null.");
          }
            _context.Students.Add(students);
            await _context.SaveChangesAsync();

            return CreatedAtAction("GetStudents", new { id = students.StudentID
}, students);
        }

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

            _context.Students.Remove(students);
            await _context.SaveChangesAsync();

            return NoContent();
        }

        private bool StudentsExists(int id)
        {
            return (_context.Students?.Any(e => e.StudentID ==
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 Phase2SchooDbWebAPI.Data;
using Phase2SchoolDbWebAPI.Models;

namespace Phase2SchooDbWebAPI.Controllers
```

```
{
    [Route("api/[controller]")]
    [ApiController]
    public class SubjectsController : ControllerBase
    {
        private readonly APISchooDbContext _context;

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

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

        // GET: api/Subjects/5
        [HttpGet("{id}")]
        public async Task<ActionResult<Subject>> GetSubject(int id)
        {
          if (_context.Subject == null)
          {
              return NotFound();
          }
            var subject = await _context.Subject.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.Subject == null)
          {
              return Problem("Entity set 'APISchooDbContext.Subject'  is null.");
          }
            _context.Subject.Add(subject);
            await _context.SaveChangesAsync();

            return CreatedAtAction("GetSubject", new { id = subject.SubID },
subject);
        }

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

            _context.Subject.Remove(subject);
            await _context.SaveChangesAsync();

            return NoContent();
        }

        private bool SubjectExists(int id)
        {
            return (_context.Subject?.Any(e => e.SubID ==
id)).GetValueOrDefault();
        }
    }
}
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
```

```
    "APISchooDbContext":
"Server=(localdb)\\mssqllocaldb;Database=Phase2SchooDbWebAPI.Data;Trusted_Connect
ion=True;MultipleActiveResultSets=true;TrustServerCertificate=True;"
  }
}

using System;
using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace Phase2SchooDbWebAPI.Migrations
{
    /// <inheritdoc />
    public partial class WebAPISchoolDb : Migration
    {
        /// <inheritdoc />
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "Students",
                columns: table => new
                {
                    StudentID = table.Column<int>(type: "int", nullable: false)
                        .Annotation("SqlServer:Identity", "1, 1"),
                    FirstName = table.Column<string>(type: "nvarchar(50)",
maxLength: 50, nullable: false),
                    LastName = table.Column<string>(type: "nvarchar(50)",
maxLength: 50, nullable: false),
                    DateOfBirth = table.Column<DateTime>(type: "datetime2",
nullable: false),
                    Gender = table.Column<string>(type: "nvarchar(10)",
maxLength: 10, nullable: false),
                    Address = table.Column<string>(type: "nvarchar(100)",
maxLength: 100, nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Students", x => x.StudentID);
                });

            migrationBuilder.CreateTable(
                name: "Subjects",
                columns: table => new
                {
                    SubID = table.Column<int>(type: "int", nullable: false)
                        .Annotation("SqlServer:Identity", "1, 1"),
                    SubName = table.Column<string>(type: "nvarchar(50)",
maxLength: 50, nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Subjects", x => x.SubID);
                });
        }

        /// <inheritdoc />
        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DropTable(
                name: "Students");

            migrationBuilder.DropTable(
```

```
                name: "Subjects");
        }
    }
}
```