Source Code

Subject.cs

```csharp
namespace PracticeProject3.Models
{
    public class Subject
    {
        public int Id { get; set; }
        public int StudentId { get; set; }
        public string StudentName { get; set; }

        public int SubjectId { get; set; }
        public string SubjectName { get; set; }

        public int SubjectMark { get; set; }
    }
}
```

StudentsController.cs

```csharp
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 PracticeProject3.Data;
using PracticeProject3.Models;

namespace PracticeProject3.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class SubjectsController : ControllerBase
    {
        private readonly PP3DbContext _context;

        public SubjectsController(PP3DbContext 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)
```

```csharp
        {
            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.Id)
            {
                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 'PP3DbContext.Subject'  is null.");
            }
            _context.Subject.Add(subject);
            await _context.SaveChangesAsync();

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

        // DELETE: api/Subjects/5
        [HttpDelete("{id}")]
```

```csharp
        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.Id == id)).GetValueOrDefault();
        }
    }
}
```