

Steps for the demo:

- Create the Entities/Models
 - Configure the DbContext
 - Setup & Configure Repository
 - Create CRUD endpoints (Get, Post, Push & Delete)
 - Entity Framework Relationship
 - Unit tests for controllers/Integration testing
-

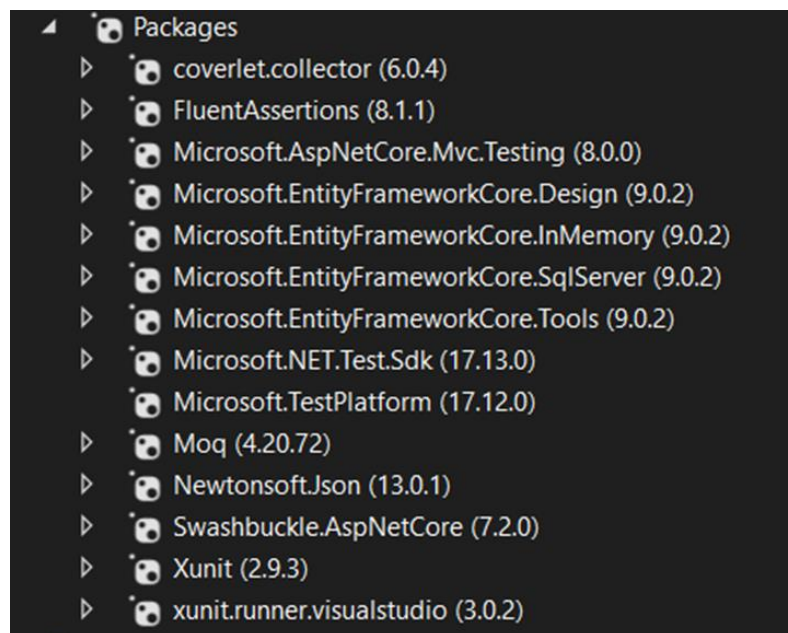
Getting started.....

1. Open Visual Studio 2022.
 2. Click Create a new project → Select ASP.NET Core Web API.
 3. Name it StudentManagementAPI and choose .NET 8.0.
 4. Click Create.
-

Install Packages:

Go to Tools>>NuGet Package Manager>>Package Manager Console/ or browse for each package to install...

Install the following [all may not be required, check the versions]:



All may not be required at this stage.... Once successfully installed, follow next step.....

Create Entities/Models

1. Right-click on the StudentManagementAPI project>>Add>>New Folder

2. Name this folder 'Models'
3. Right-click on 'Models'>>Add>>Class
4. Name the class: Student.cs

The code should be the same as below:

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

namespace StudentManagementAPI.Models
{
    20 references
    public class Student
    {
        [Key]
        4 references
        public int Id { get; set; }

        [Required]
        2 references
        public string Name { get; set; } = string.Empty;

        [Required]
        2 references
        public string Email { get; set; } = string.Empty;

        3 references
        public List<Course>? Courses { get; set; }
    }
}
```

Once this is done....Create the Course class

1. Right-click on 'Models'>>Add>>Class
2. Name this class: Course.cs

Refer to image for the code:

```
using System.ComponentModel.DataAnnotations;

namespace StudentManagementAPI.Models
{
    3 references
    public class Course
    {
        [Key]
        0 references
        public int Id { get; set; }

        [Required]
        0 references
        public string CourseName { get; set; } = string.Empty;

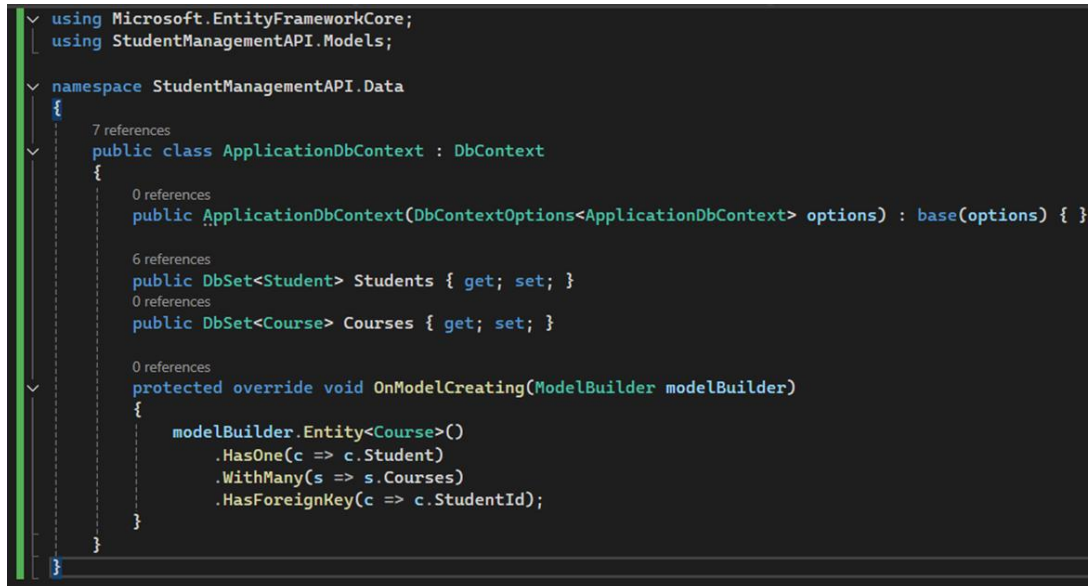
        1 reference
        public int StudentId { get; set; }

        1 reference
        public Student? Student { get; set; }
    }
}
```

Configure the DbContext

1. Right-click on the StudentManagementAPI project>>Add>>New Folder
2. Name this folder: 'Data'
3. Right-click on the 'Data' folder>>Add>>Class
4. Name this class: ApplicationDbContext.cs

Refer to code in the image:



```
using Microsoft.EntityFrameworkCore;
using StudentManagementAPI.Models;

namespace StudentManagementAPI.Data
{
    7 references
    public class ApplicationDbContext : DbContext
    {
        0 references
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options) { }

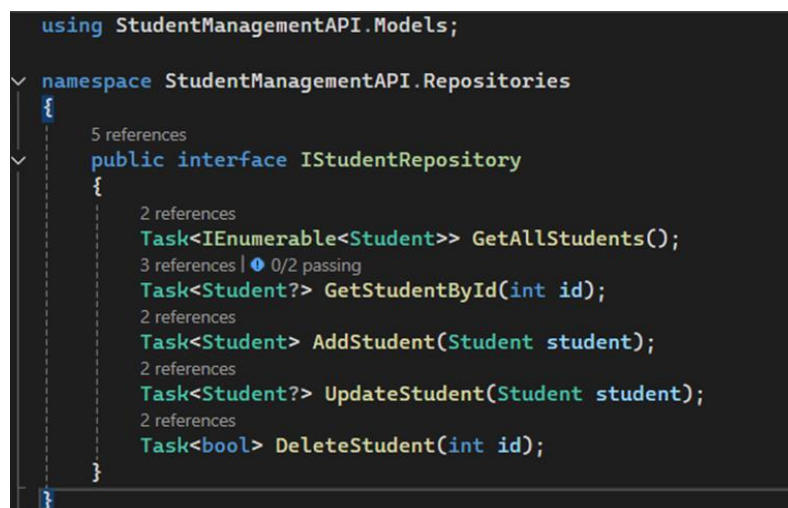
        6 references
        public DbSet<Student> Students { get; set; }
        0 references
        public DbSet<Course> Courses { get; set; }

        0 references
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Course>()
                .HasOne(c => c.Student)
                .WithMany(s => s.Courses)
                .HasForeignKey(c => c.StudentId);
        }
    }
}
```

Configure Repository Pattern

1. Right-click on the StudentManagementAPI project>>Add>>New Folder
2. Name this folder: 'Repositories'
3. Right-click on the 'Repositories' folder>>Add>>Class
4. Name this class: IStudentRepository.cs [this is the interface class]

Refer to the code below:



```
using StudentManagementAPI.Models;

namespace StudentManagementAPI.Repositories
{
    5 references
    public interface IStudentRepository
    {
        2 references
        Task<IEnumerable<Student>> GetAllStudents();
        3 references | 0/2 passing
        Task<Student?> GetStudentById(int id);
        2 references
        Task<Student> AddStudent(Student student);
        2 references
        Task<Student?> UpdateStudent(Student student);
        2 references
        Task<bool> DeleteStudent(int id);
    }
}
```

Create the implementation class:

1. Right-click on the 'Repositories' folder>>Add>>Class
2. Name this class: StudentRepository.cs [this is the implementation class]

Refer to the code below:

```
using Microsoft.EntityFrameworkCore;
using StudentManagementAPI.Data;
using StudentManagementAPI.Models;

namespace StudentManagementAPI.Repositories
{
    2 references
    public class StudentRepository : IStudentRepository
    {
        private readonly ApplicationDbContext _context;

        0 references
        public StudentRepository(ApplicationDbContext context)
        {
            _context = context;
        }

        2 references
        public async Task<IEnumerable<Student>> GetAllStudents()
        {
            return await _context.Students.Include(s => s.Courses).ToListAsync();
        }

        3 references | 0/2 passing
        public async Task<Student?> GetStudentById(int id)
        {
            return await _context.Students.Include(s => s.Courses).FirstOrDefaultAsync(s => s.Id == id);
        }
    }
}
```

```
2 references
public async Task<Student> AddStudent(Student student)
{
    _context.Students.Add(student);
    await _context.SaveChangesAsync();
    return student;
}

2 references
public async Task<Student?> UpdateStudent(Student student)
{
    var existingStudent = await _context.Students.FindAsync(student.Id);
    if (existingStudent == null) return null;

    existingStudent.Name = student.Name;
    existingStudent.Email = student.Email;
    await _context.SaveChangesAsync();
    return existingStudent;
}
}
```

```
2 references
public async Task<bool> DeleteStudent(int id)
{
    var student = await _context.Students.FindAsync(id);
    if (student == null) return false;

    _context.Students.Remove(student);
    await _context.SaveChangesAsync();
    return true;
}
}
```

Register Services in Program.cs

1. Click on Program.cs

[services are defined and includes repository and database configuration]

Refer to the code:

```
using Microsoft.EntityFrameworkCore;
using StudentManagementAPI.Data;
using StudentManagementAPI.Repositories;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();
builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));
builder.Services.AddScoped<IStudentRepository, StudentRepository>();

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

var app = builder.Build();

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

app.UseHttpsRedirection();
app.UseAuthorization();
app.MapControllers();

app.Run();
```

Add Database Connection & Run Migrations

Modify appsettings.json:

```
"ConnectionStrings": {
  "DefaultConnection":
    "Server=(localdb)\\mssqllocaldb;Database=StudentDB;Trusted_Connection=True;"
}
```

Run the following commands in Package Manager Console:

```
Add-Migration InitialCreate
Update-Database
```

Create Student Controller with CRUD Endpoints

1. Right-click on the 'Controllers' folder >> Add >> Class
2. Name the class: StudentController.cs

The following code should be included:

```
using Microsoft.AspNetCore.Mvc;
using StudentManagementAPI.Models;
using StudentManagementAPI.Repositories;

namespace StudentManagementAPI.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    2 references
    public class StudentController : ControllerBase
    {
        private readonly IStudentRepository _repository;

        1 reference | 0/2 passing
        public StudentController(IStudentRepository repository)
        {
            _repository = repository;
        }

        [HttpGet]
        0 references
        public async Task<ActionResult<IEnumerable<Student>>> GetAllStudents()
        {
            return Ok(await _repository.GetAllStudents());
        }
    }
}
```

```
[HttpGet("{id}")]
2 references | 0/2 passing
public async Task<ActionResult<Student>> GetStudent(int id)
{
    var student = await _repository.GetStudentById(id);
    return student == null ? NotFound() : Ok(student);
}

[HttpPost]
0 references
public async Task<ActionResult<Student>> AddStudent(Student student)
{
    var newStudent = await _repository.AddStudent(student);
    return CreatedAtAction(nameof(GetStudent), new { id = newStudent.Id }, newStudent);
}

[HttpPut("{id}")]
0 references
public async Task<IActionResult> UpdateStudent(int id, Student student)
{
    if (id != student.Id) return BadRequest();
    var updatedStudent = await _repository.UpdateStudent(student);
    return updatedStudent == null ? NotFound() : NoContent();
}
}
```

```
[HttpDelete("{id}")]
0 references
public async Task<IActionResult> DeleteStudent(int id)
{
    return await _repository.DeleteStudent(id) ? NoContent() : NotFound();
}
}
```

Write Unit Tests

1. Right-click on: Solution 'StudentManagementAPI' >>Add >> New Project
2. Search and select xUnit.
3. Name the project: 'StudentManagementAPI.Tests'
4. Right-click on: StudentManagementAPI.Tests >> Add >> New Folder
5. Name the folder: 'Controller'
6. Right-click on: Controller >>Add >> Class
7. Name the class: StudentControllerTests.cs

The following code must be entered:

```
using Microsoft.AspNetCore.Mvc;
using Moq;
using StudentManagementAPI.Controllers;
using StudentManagementAPI.Models;
using StudentManagementAPI.Repositories;

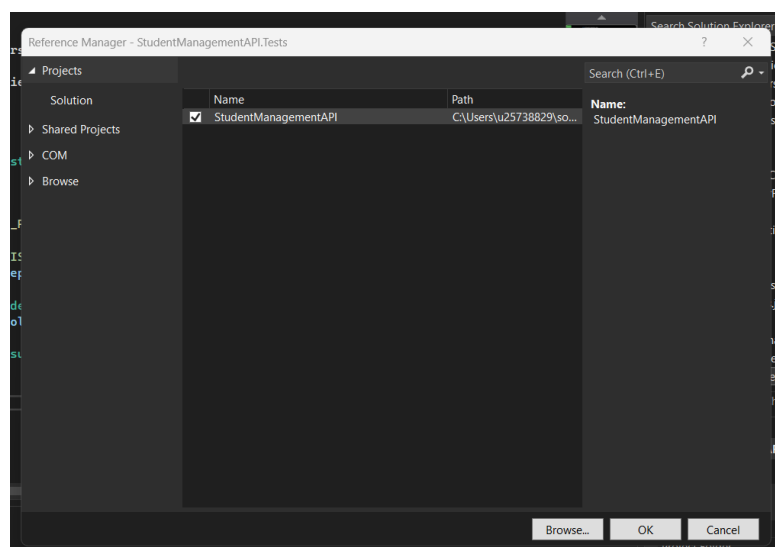
namespace StudentManagementAPI.Tests
{
    0 references
    public class StudentControllerTests
    {
        [Fact]
        0 references
        public async Task GetStudent_ReturnsNotFound_WhenStudentDoesNotExist()
        {
            var mockRepo = new Mock<IStudentRepository>();
            mockRepo.Setup(repo => repo.GetStudentById(1)).ReturnsAsync((Student)null);

            var controller = new StudentController(mockRepo.Object);
            var result = await controller.GetStudent(1);

            Assert.IsType<NotFoundResult>(result.Result);
        }
    }
}
```

Make sure the project reference is selected:

1. Right-click on: 'StudentManagementAPI.Tests >> Add >> Project Reference
2. Tick the 'StudentManagementAPI' project



1. Click on: View >> Terminal
2. Type the following for the test:

dotnet test

```
PS C:\Users\u25738829\source\repos\StudentManagementAPI> dotnet test
```

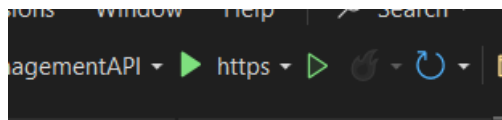
The following should be displayed [no errors or failed tests]:

```
PS C:\Users\u25738829\source\repos\StudentManagementAPI> dotnet test
Restore complete (0,8s)
StudentManagementAPI succeeded (0,4s) -> StudentManagementAPI\bin\Debug\net8.0\StudentManagementAPI.dll
[xUnit.net 00:00:00.00] xUnit.net VSTest Adapter v3.0.2+dd36e86129 (64-bit .NET 8.0.12)
[xUnit.net 00:00:00.08] Discovering: StudentManagementAPI
[xUnit.net 00:00:00.10] Discovered: StudentManagementAPI
StudentManagementAPI test succeeded with 1 warning(s) (0,8s)
C:\Program Files\dotnet\sdk\9.0.102\Microsoft.TestPlatform.targets(48,5): warning : No test is available in C:\Users\u25738829\source\repos\StudentManagementAPI\StudentManagementAPI\bin\Debug\net8.0\StudentManagementAPI.dll. Make sure that test discoverer & executors are registered and platform & framework version settings are appropriate and try again.
StudentManagementAPI.Tests succeeded (0,3s) -> StudentManagementAPI.Tests\bin\Debug\net8.0\StudentManagementAPI.Tests.dll
[xUnit.net 00:00:00.00] xUnit.net VSTest Adapter v2.5.3.1+6b60a9e56a (64-bit .NET 8.0.12)
[xUnit.net 00:00:00.06] Discovering: StudentManagementAPI.Tests
[xUnit.net 00:00:00.09] Discovered: StudentManagementAPI.Tests
[xUnit.net 00:00:00.09] Starting: StudentManagementAPI.Tests
[xUnit.net 00:00:00.16] Finished: StudentManagementAPI.Tests
[xUnit.net 00:00:00.00] xUnit.net VSTest Adapter v3.0.2+dd36e86129 (64-bit .NET 8.0.12)
[xUnit.net 00:00:00.03] Discovering: StudentManagementAPI.Tests
[xUnit.net 00:00:00.04] Discovered: StudentManagementAPI.Tests
[xUnit.net 00:00:00.05] Starting: StudentManagementAPI.Tests
[xUnit.net 00:00:00.06] Finished: StudentManagementAPI.Tests
StudentManagementAPI.Tests test succeeded (0,9s)

Test summary: total: 4, failed: 0, succeeded: 4, skipped: 0, duration: 2,0s
Build succeeded with 1 warning(s) in 3,5s

Workload updates are available. Run `dotnet workload list` for more information.
PS C:\Users\u25738829\source\repos\StudentManagementAPI>
```

Run the application by selecting:



A snippet of the following should appear:

