Day 2 – Entity Framework Core and SQL Server Integration **Date:** 17-10-2025

Objective: To understand the EF Core Web API project and SQL Server Integration

1. Introduction to Entity Framework Core (EF Core)

Entity Framework Core (EF Core) is an **Object-Relational Mapper (ORM)** developed by Microsoft that allows developers to work with databases using **.NET objects** instead of writing SQL queries manually.
EF Core simplifies data access by mapping **C# classes to database tables**.

2. Why Use EF Core

- **Abstraction**: Developers can focus on business logic instead of SQL code.
- **Maintainability**: Changes to models can automatically update the database.
- **Portability**: Works with multiple databases like SQL Server, MySQL, PostgreSQL, and SQLite.
- **Productivity**: Reduces boilerplate data-access code.

3. EF Core Architecture

The EF Core architecture is built around the following key components:

Component	**Description**
DbContext	The main class that manages database connections and transactions.
DbSet	Represents a table in the database; used for CRUD operations.
Model	Defines the structure of tables, relationships, and constraints.
Migration	Mechanism to keep database schema in sync with C# models.
LINQ (Language Integrated Query)	Allows writing database queries using C# syntax.

4. Database Approaches

EF Core supports **two main approaches** for connecting applications with databases:

1. **Code First Approach**
    - The developer defines **C# classes**, and EF Core generates the database schema automatically.
    - Best suited for new projects where the database doesn’t exist yet.
2. **Database First Approach**
    - The database is already created.
    - EF Core automatically generates C# model classes and DbContext based on the existing tables.
    - Ideal for projects integrating with existing databases.

5. Setting up EF Core in ASP.NET Core

To use EF Core, the following steps are required:

Step 1: Install EF Core Packages

Run these commands in the terminal:

```
dotnet add package Microsoft.EntityFrameworkCore

dotnet add package Microsoft.EntityFrameworkCore.SqlServer

dotnet add package Microsoft.EntityFrameworkCore.Tools
```

Step 2: Configure Database Connection

In the appsettings.json file, define a connection string:

```
"ConnectionStrings": {

  "DefaultConnection":
"Server=localhost;Database=StudentDB;Trusted_Connection=True;TrustServerCertificate=True;"

}
```

Step 3: Register DbContext in Program.cs

```
builder.Services.AddDbContext<StudentDBContext>(options =>

   options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));
```

Step 4: Create DbContext and Model

- **Model (Entity)** → represents a database table.
- **DbContext** → manages database access and queries.
-

6. Understanding DbContext

DbContext is the heart of EF Core. It:

- Opens and closes database connections.
- Tracks entity changes.
- Executes SQL commands.
- Maps database tables to C# objects.

Example:

```
public class StudentDBContext : DbContext
{
   public DbSet<Student> Students { get; set; }
}
```

7. How EF Core Translates C# to SQL

EF Core automatically converts your C# code into SQL statements.
For example:

```
var data = context.Students.ToList();
```

Internally executes:

```
SELECT * FROM Students;
```

8. LINQ in EF Core

LINQ (Language Integrated Query) allows writing queries directly in C#.
Examples:

```
var student = context.Students.FirstOrDefault(s => s.Id == 1);
```

```
var topStudents = context.Students.Where(s => s.Grade == "A").ToList();
```

EF Core translates these into optimized SQL queries.

9. EF Core and SQL Server Integration Steps

1. Create a **SQL Server database** (e.g., StudentDB).
2. Create tables (e.g., Students table).
3. Scaffold database into C# using EF Core Database First.

4. Connect Web API with the database context.
5. Test data retrieval through Swagger or Postman.

Testing the Connection

Run the application and open Swagger UI.
Check endpoints like:

GET /api/students

GET /api/students/{id}

If the database connection is correct, data will appear as JSON output.

Benefits of EF Core in Real Projects

- Simplifies CRUD operations.
- Automatically handles parameterization (prevents SQL injection).
- Integrates with ASP.NET Core dependency injection.
- Supports transactions and concurrency control.
- Works well with REST APIs and microservices.

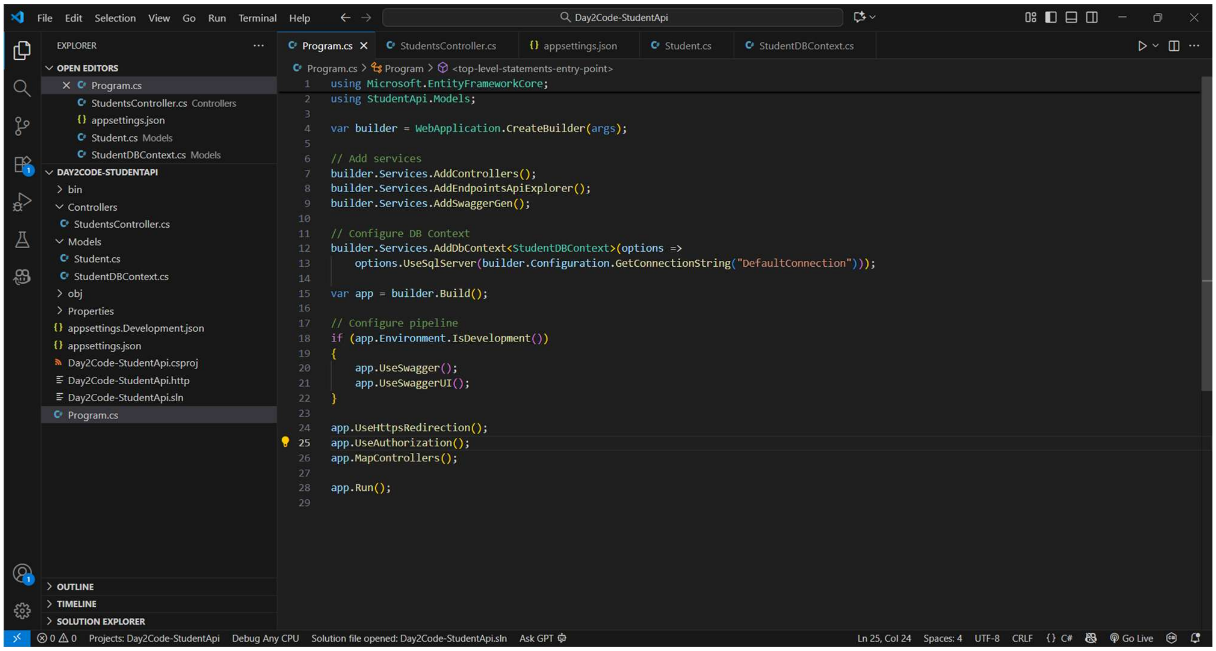
Common Issues and Fixes

Issue	Cause	Solution
Connection string error	Wrong SQL Server name or missing database	Verify connection string in appsettings.json
Migration error	Missing EF Core tools	Install Microsoft.EntityFrameworkCore.Tools
Table not found	Wrong schema or table name	Check OnModelCreating in DbContext
Null reference on DbContext	Forgot to register DbContext	Register in Program.cs

Day-End Mini Task

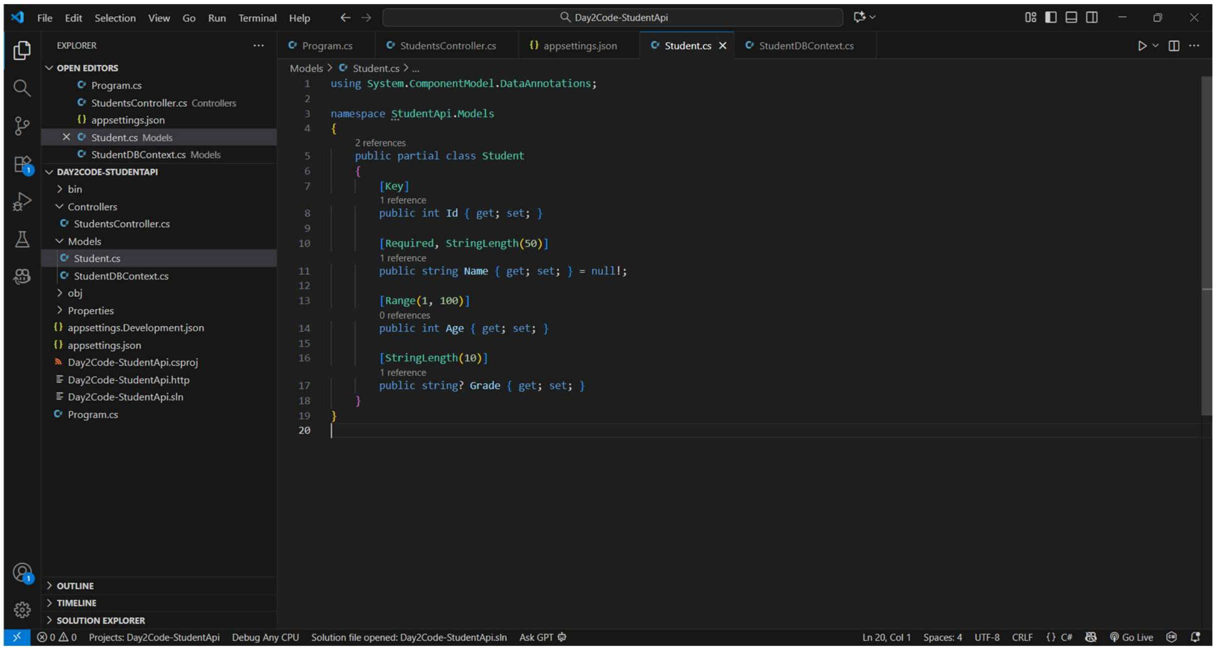
- Create a StudentsController with endpoints:
  - GET /api/students → Retrieve all students
  - GET /api/students/{id} → Retrieve a single student

Snapshots :



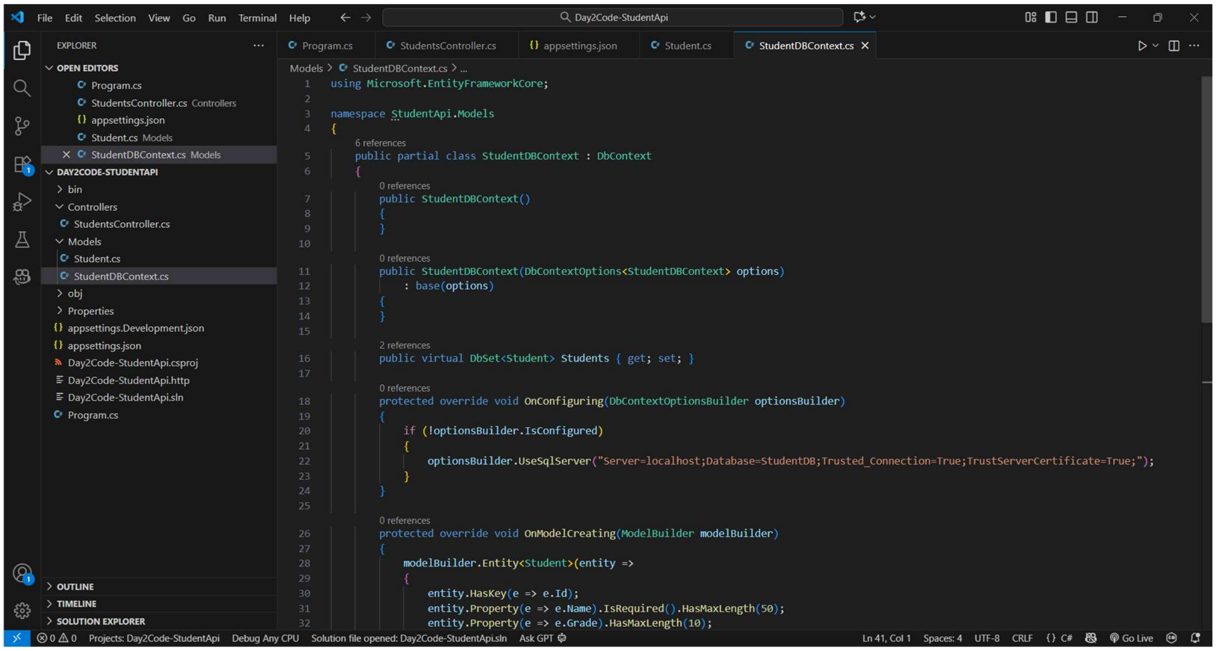
```csharp
using Microsoft.EntityFrameworkCore;
using StudentApi.Models;

var builder = WebApplication.CreateBuilder(args);

// Add services
builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

// Configure DB Context
builder.Services.AddDbContext<StudentDBContext>(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

var app = builder.Build();

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

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

app.Run();
```

Code : Program.cs



```csharp
using System.ComponentModel.DataAnnotations;

namespace StudentApi.Models
{
    public partial class Student
    {
        [Key]
        public int Id { get; set; }

        [Required, StringLength(50)]
        public string Name { get; set; } = null!;

        [Range(1, 100)]
        public int Age { get; set; }

        [StringLength(10)]
        public string? Grade { get; set; }
    }
}
```

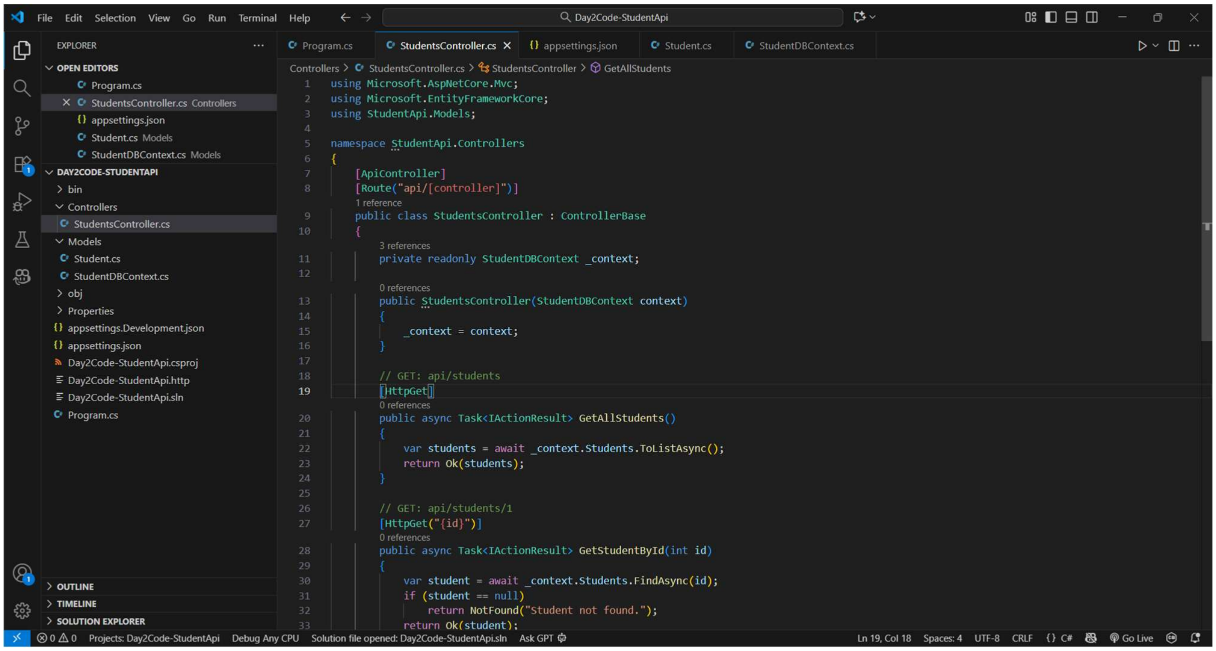
Code : Student.cs



File Edit Selection View Go Run Terminal Help
Day2Code-StudentApi
EXPLORER
OPEN EDITORS
Program.cs
StudentsController.cs Controllers
appsettings.json
Student.cs Models
StudentDBContext.cs Models
DAY2CODE-STUDENTAPI
bin
Controllers
StudentsController.cs
Models
Student.cs
StudentDBContext.cs
obj
Properties
appsettings.Development.json
appsettings.json
Day2Code-StudentApi.csproj
Day2Code-StudentApi.http
Day2Code-StudentApi.sln
Program.cs
OUTLINE
TIMELINE
SOLUTION EXPLORER
Models > StudentDBContext.cs > ...
Projects: Day2Code-StudentApi Debug Any CPU Solution file opened: Day2Code-StudentApi.sln Ask GPT
Ln 41, Col 1 Spaces: 4 UTF-8 CRLF {} C# Go Live

```csharp
using Microsoft.EntityFrameworkCore;

namespace StudentApi.Models
{
    public partial class StudentDBContext : DbContext
    {
        public StudentDBContext()
        {
        }

        public StudentDBContext(DbContextOptions<StudentDBContext> options)
            : base(options)
        {
        }

        public virtual DbSet<Student> Students { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            if (!optionsBuilder.IsConfigured)
            {
                optionsBuilder.UseSqlServer("Server=localhost;Database=StudentDB;Trusted_Connection=True;TrustServerCertificate=True;");
            }
        }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Student>(entity =>
            {
                entity.HasKey(e => e.Id);
                entity.Property(e => e.Name).IsRequired().HasMaxLength(50);
                entity.Property(e => e.Grade).HasMaxLength(10);
```

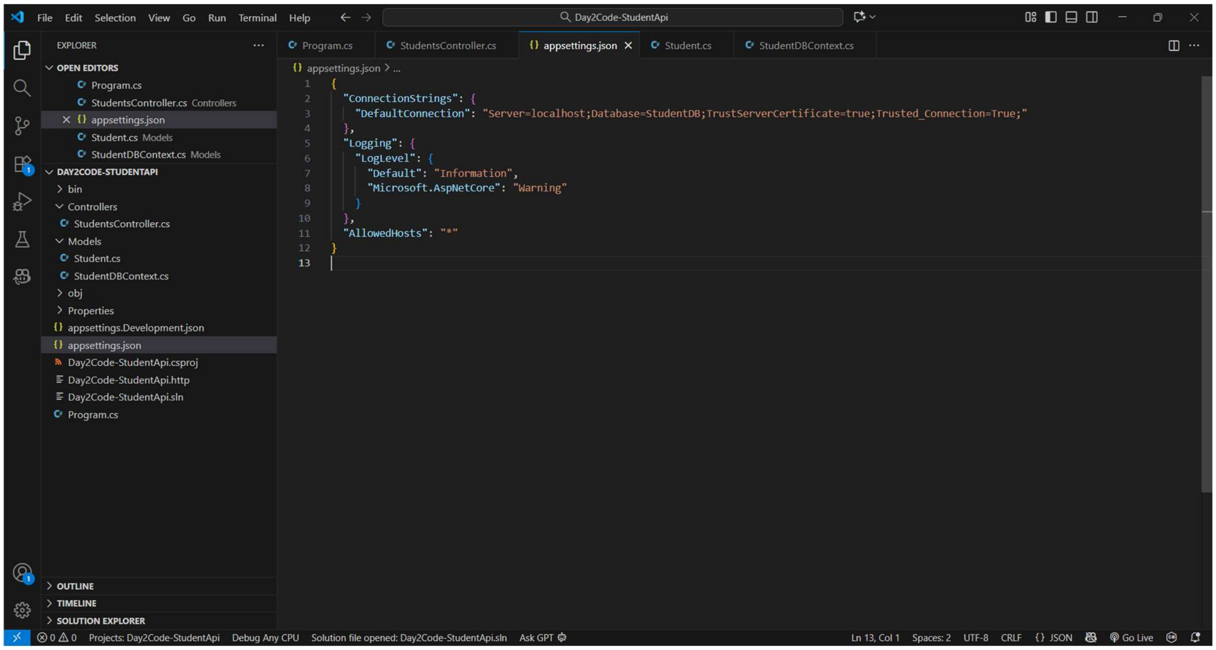
Code : StudentDBContext.cs



File Edit Selection View Go Run Terminal Help
Day2Code-StudentApi
EXPLORER
OPEN EDITORS
Program.cs
StudentsController.cs Controllers
appsettings.json
Student.cs Models
StudentDBContext.cs Models
DAY2CODE-STUDENTAPI
bin
Controllers
StudentsController.cs
Models
Student.cs
StudentDBContext.cs
obj
Properties
appsettings.Development.json
appsettings.json
Day2Code-StudentApi.csproj
Day2Code-StudentApi.http
Day2Code-StudentApi.sln
Program.cs
OUTLINE
TIMELINE
SOLUTION EXPLORER
Controllers > StudentsController.cs > StudentsController > GetAllStudents
Projects: Day2Code-StudentApi Debug Any CPU Solution file opened: Day2Code-StudentApi.sln Ask GPT
Ln 19, Col 18 Spaces: 4 UTF-8 CRLF {} C# Go Live

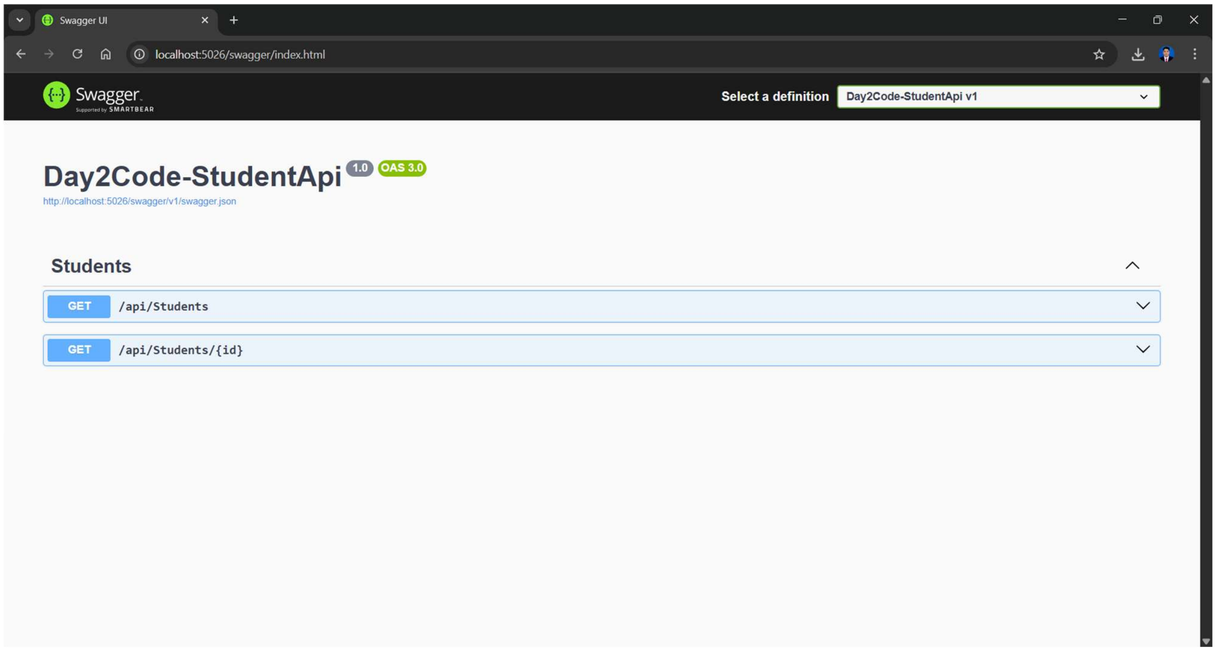
```csharp
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using StudentApi.Models;

namespace StudentApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class StudentsController : ControllerBase
    {
        private readonly StudentDBContext _context;

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

        // GET: api/students
        [HttpGet]
        public async Task<IActionResult> GetAllStudents()
        {
            var students = await _context.Students.ToListAsync();
            return Ok(students);
        }

        // GET: api/students/1
        [HttpGet("{id}")]
        public async Task<IActionResult> GetStudentById(int id)
        {
            var student = await _context.Students.FindAsync(id);
            if (student == null)
                return NotFound("Student not found.");
            return Ok(student);
```

Code : StudentController.cs



```json
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=localhost;Database=StudentDB;TrustServerCertificate=true;Trusted_Connection=True;"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

Code : appsettings.json



Day2Code-StudentApi 1.0 OAS 3.0

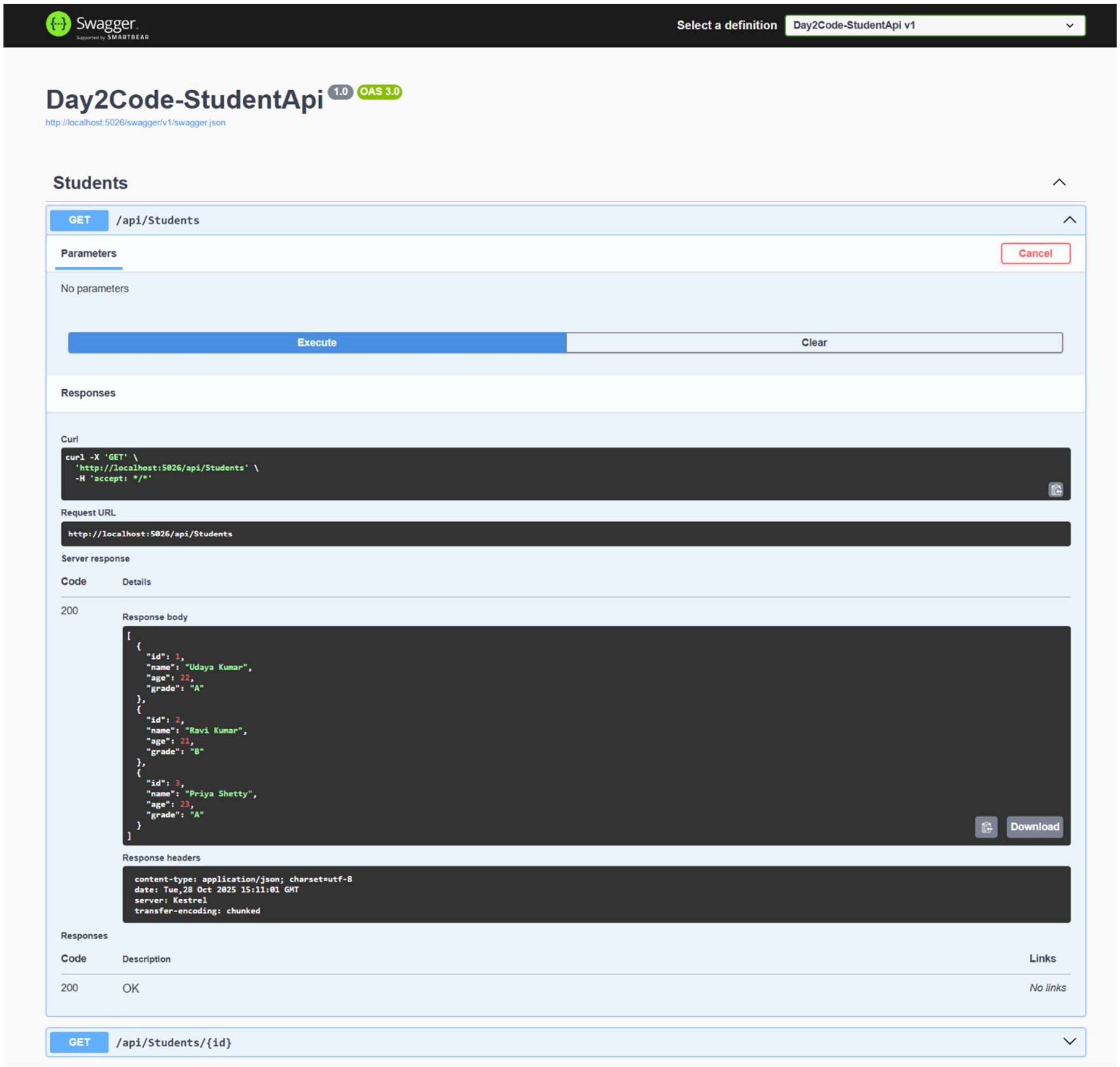
http://localhost:5026/swagger/v1/swagger.json

Students

GET /api/Students

GET /api/Students/{id}

Output : GET method



Day2Code-StudentApi

http://localhost:5026/swagger/v1/swagger.json

/api/Students

No parameters

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5026/api/Students' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5026/api/Students
```

Server response

Code	Details
200	Response body

```
[
  {
    "id": 1,
    "name": "Udaya Kumar",
    "age": 22,
    "grade": "A"
  },
  {
    "id": 2,
    "name": "Ravi Kumar",
    "age": 21,
    "grade": "B"
  },
  {
    "id": 3,
    "name": "Priya Shetty",
    "age": 23,
    "grade": "A"
  }
]
```

Response headers

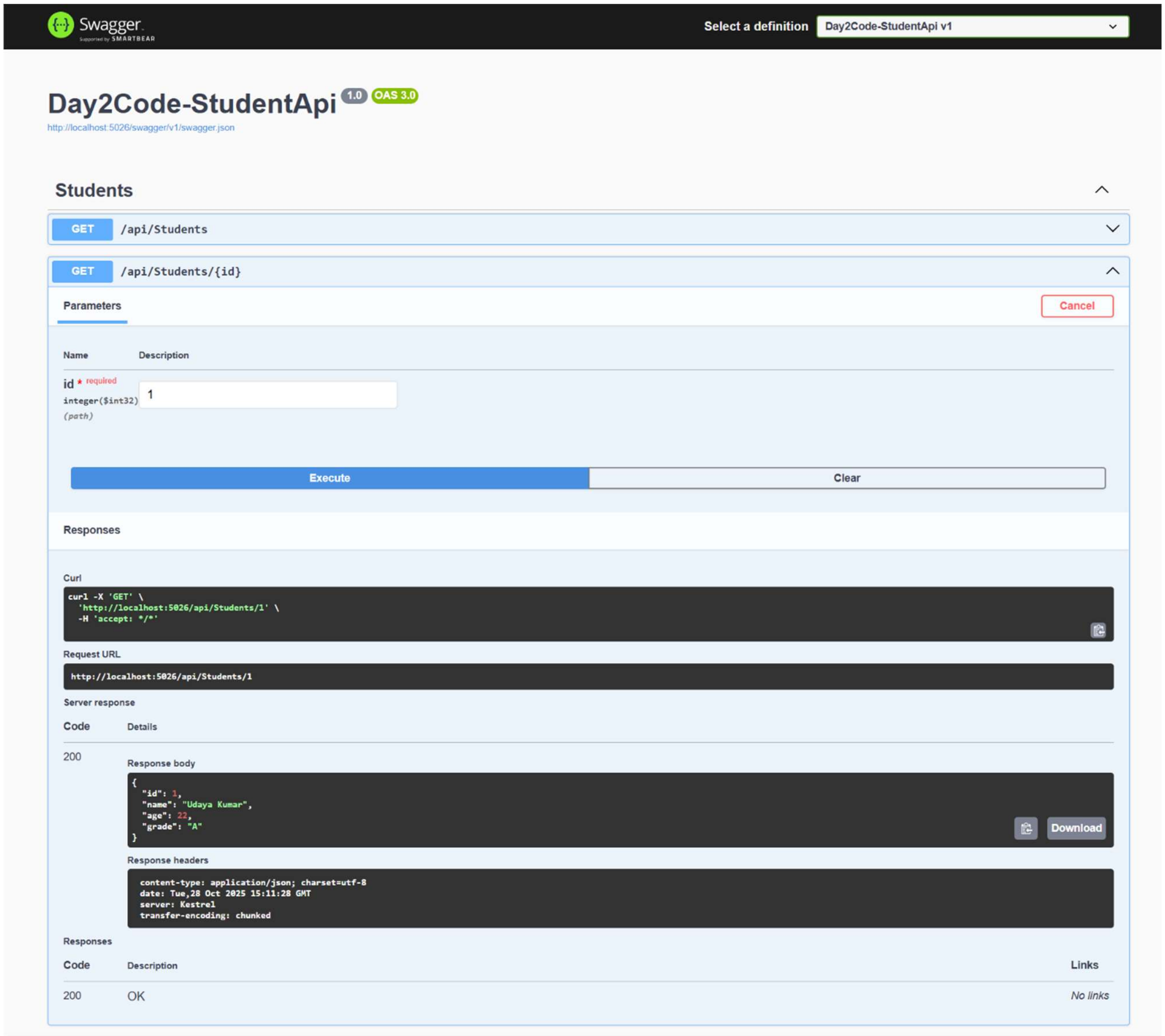
```
content-type: application/json; charset=utf-8
date: Tue,28 Oct 2025 15:11:01 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	No links

/api/Students/{id}

Output : GET (All data from database)



Select a definition: Day2Code-StudentApi v1

Day2Code-StudentApi 1.0 OAS 3.0

http://localhost:5026/swagger/v1/swagger.json

Students

GET /api/Students

GET /api/Students/{id}

Parameters

Name	Description
id * required integer($int32) (path)	1

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5026/api/Students/1' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5026/api/Students/1
```

Server response

Code	Details
200	Response body

```
{
  "id": 1,
  "name": "Udaya Kumar",
  "age": 22,
  "grade": "A"
}
```

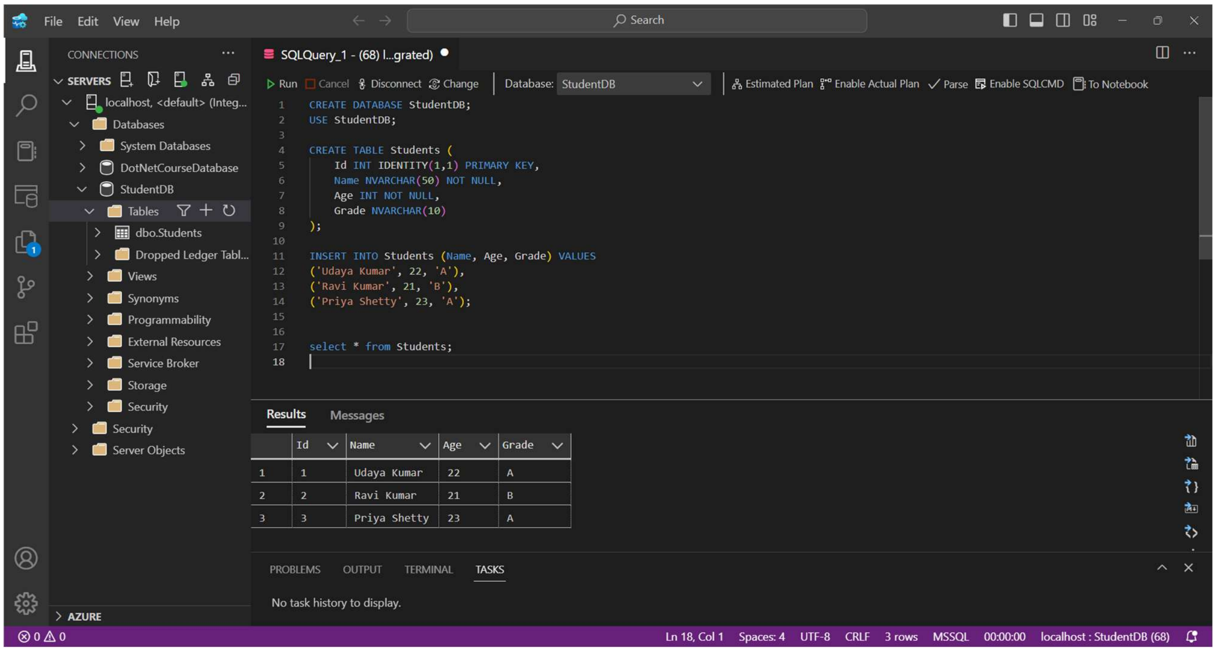
Response headers

```
content-type: application/json; charset=utf-8
date: Tue,28 Oct 2025 15:11:28 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	No links

Output : GET (Spacific data from database)



```sql
CREATE DATABASE StudentDB;
USE StudentDB;

CREATE TABLE Students (
    Id INT IDENTITY(1,1) PRIMARY KEY,
    Name NVARCHAR(50) NOT NULL,
    Age INT NOT NULL,
    Grade NVARCHAR(10)
);

INSERT INTO Students (Name, Age, Grade) VALUES
('Udaya Kumar', 22, 'A'),
('Ravi Kumar', 21, 'B'),
('Priya Shetty', 23, 'A');


select * from Students;

```

	Id	Name	Age	Grade
1	1	Udaya Kumar	22	A
2	2	Ravi Kumar	21	B
3	3	Priya Shetty	23	A

No task history to display.

Code : Database Code