

Day 6 – Stored Procedures and API Integration with ASP.NET Core Date: 21-10-2025

Objective: Learn to call and manage SQL Server stored procedures from ASP.NET Core APIs. Integrate backend logic securely and efficiently using ADO.NET or EF Core

1. Introduction to Stored Procedures

A **stored procedure** is a precompiled collection of one or more SQL statements stored in a database.

Instead of sending multiple SQL queries from the application, the client can call the procedure by name, passing parameters if needed.

Purpose: To improve performance, security, and maintainability of database operations.

2. Why Use Stored Procedures in ASP.NET Core Projects

Feature	Explanation
Performance	Stored procedures are precompiled and optimized by SQL Server.
Security	Prevents SQL injection as parameters are handled safely.
Reusability	Common database logic can be reused across applications.
Maintainability	Easier to update database logic without modifying application code.
Separation of Concerns	Keeps business logic separate from data access logic.

3. Structure of a Stored Procedure

Syntax:

```
CREATE PROCEDURE procedure_name
```

```
    @parameter1 datatype,
```

```
    @parameter2 datatype = default_value
```

```
AS
```

```
BEGIN
```

```
    -- SQL Statements
```

```
END;
```

Example:

```
CREATE PROCEDURE sp_GetStudents
```

```
AS
```

```
BEGIN
```

```
    SELECT * FROM Students;
```

```
END;
```

To execute:

```
EXEC sp_GetStudents;
```

4. Creating Stored Procedures for StudentCourseDB

Below are examples of CRUD stored procedures used in the Student Management API.

1. Get All Students

```
CREATE PROCEDURE sp_GetStudents
```

```
AS
```

```
BEGIN
```

```
    SELECT s.Id, s.Name, s.Age, s.Grade, c.CourseName
```

```
    FROM Students s
```

```
    INNER JOIN Courses c ON s.CourseId = c.CourseId;
```

```
END;
```

2. Get Student by ID

```
CREATE PROCEDURE sp_GetStudentById
```

```
    @Id INT
```

```
AS
```

```
BEGIN
```

```
    SELECT * FROM Students WHERE Id = @Id;
```

```
END;
```

3. Add Student

```
CREATE PROCEDURE sp_AddStudent
```

```
    @Name NVARCHAR(100),
```

```
    @Age INT,
```

```
    @Grade NVARCHAR(10),
```

```
    @CourseId INT
```

```
AS
BEGIN
    INSERT INTO Students (Name, Age, Grade, CourseId)
    VALUES (@Name, @Age, @Grade, @CourseId);
END;
```

4. Update Student

```
CREATE PROCEDURE sp_UpdateStudent
    @Id INT,
    @Name NVARCHAR(100),
    @Age INT,
    @Grade NVARCHAR(10),
    @CourseId INT
AS
BEGIN
    UPDATE Students
    SET Name = @Name, Age = @Age, Grade = @Grade, CourseId = @CourseId
    WHERE Id = @Id;
END;
```

5. Delete Student

```
CREATE PROCEDURE sp_DeleteStudent
    @Id INT
AS
BEGIN
    DELETE FROM Students WHERE Id = @Id;
END;
```

5. Integrating Stored Procedures in ASP.NET Core (EF Core)

There are multiple ways to call stored procedures in ASP.NET Core using EF Core. The most common methods are:

1. Using FromSqlRaw()

Example to call stored procedure that returns data:

```
var students = _context.Students
    .FromSqlRaw("EXEC sp_GetStudents")
    .ToList();
```

2. Using Database.ExecuteSqlRaw()

Used for stored procedures that do not return data (like Insert/Update/Delete).

Example:

```
_context.Database.ExecuteSqlRaw(
    "EXEC sp_AddStudent @p0, @p1, @p2, @p3",
    new object[] { name, age, grade, courseId });
```

3. Using SqlParameter for Safety

```
var idParam = new SqlParameter("@Id", 2);

var student = _context.Students
    .FromSqlRaw("EXEC sp_GetStudentById @Id", idParam)
    .ToList();
```

6. API Controller Integration Example

Example StudentsController methods using stored procedures:

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

    public StudentsController(StudentCourseContext context)
    {
        _context = context;
    }
}
```

```

}

[HttpGet]
public IActionResult GetAllStudents()
{
    var students = _context.Students.FromSqlRaw("EXEC sp_GetStudents").ToList();
    return Ok(students);
}

[HttpGet("{id}")]
public IActionResult GetStudentById(int id)
{
    var idParam = new SqlParameter("@Id", id);
    var student = _context.Students.FromSqlRaw("EXEC sp_GetStudentById @Id",
idParam).FirstOrDefault();
    return Ok(student);
}

[HttpPost]
public IActionResult AddStudent(Student s)
{
    _context.Database.ExecuteSqlRaw(
        "EXEC sp_AddStudent @p0, @p1, @p2, @p3",
        s.Name, s.Age, s.Grade, s.CourseId);
    return Ok("Student added successfully.");
}

[HttpPut("{id}")]
public IActionResult UpdateStudent(int id, Student s)
{
    _context.Database.ExecuteSqlRaw(
        "EXEC sp_UpdateStudent @p0, @p1, @p2, @p3, @p4",
        id, s.Name, s.Age, s.Grade, s.CourseId);
    return Ok("Student updated successfully.");
}

```

```

    }

    [HttpDelete("{id}")]
    public IActionResult DeleteStudent(int id)
    {
        _context.Database.ExecuteSqlRaw("EXEC sp_DeleteStudent @p0", id);
        return Ok("Student deleted successfully.");
    }
}

```

7. Advantages of Using Stored Procedures in EF Core

1. **Performance** – SQL Server precompiles and caches the query plan.
2. **Security** – Parameters prevent SQL injection.
3. **Simpler Maintenance** – Database logic can be changed independently.
4. **Reduced Data Transfer** – Only the required data is sent back.
5. **Logging and Auditing** – Stored procedures can include tracking logic.

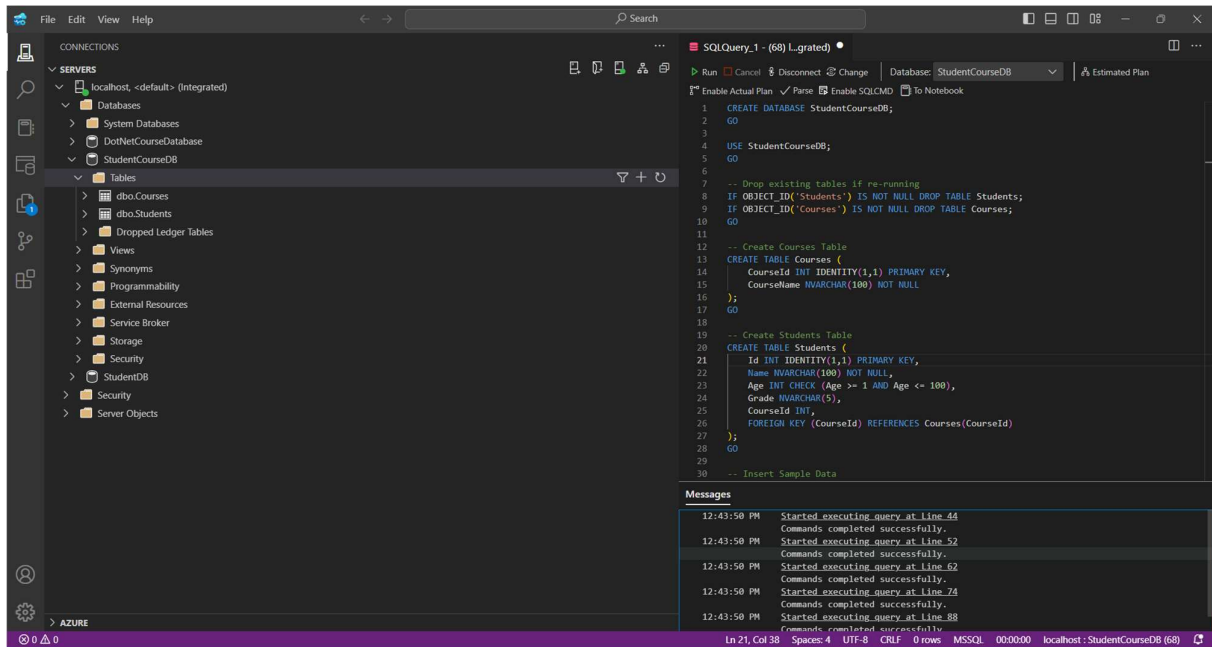
8. Common Issues & Fixes

Issue	Reason	Solution
Invalid object name	Database context connected to wrong DB	Check connection string
Procedure not found	Procedure not created or name mismatch	Verify stored procedure name in SQL Server
Parameter count mismatch	Incorrect number of parameters	Match parameters exactly as defined
No data returned	Result set not mapped	Check your entity model and SQL SELECT columns

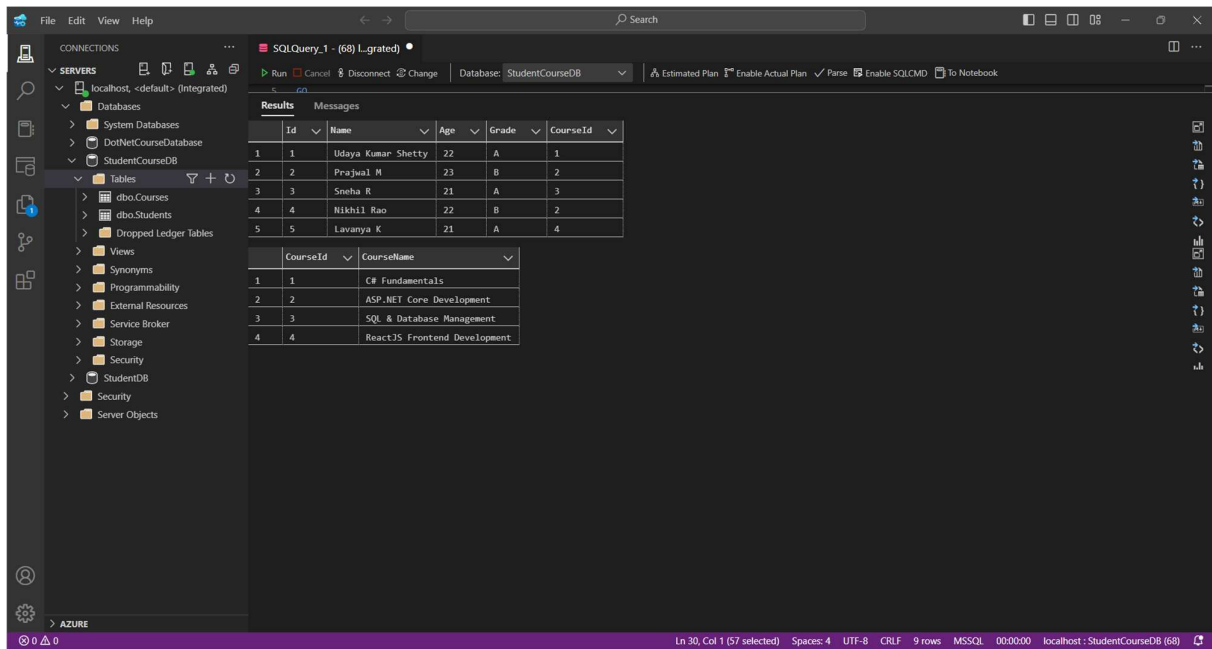
Mini Exercise

- Create a stored procedure to retrieve students **above a certain age**.
- Call it in your controller using FromSqlRaw.

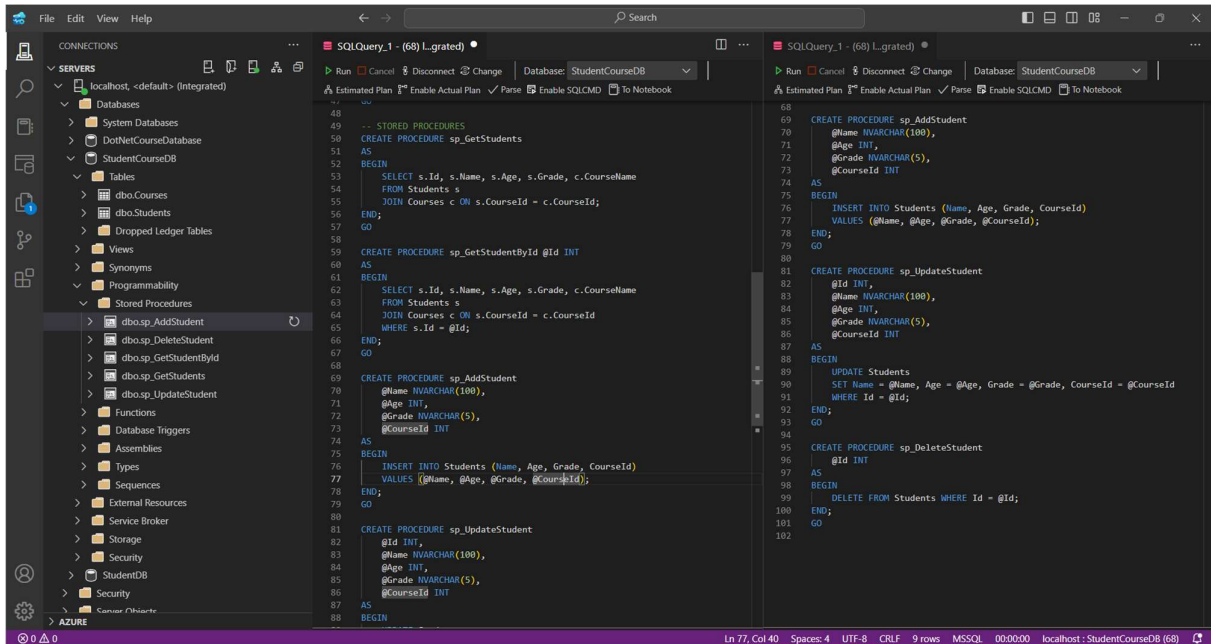
Snapshots :



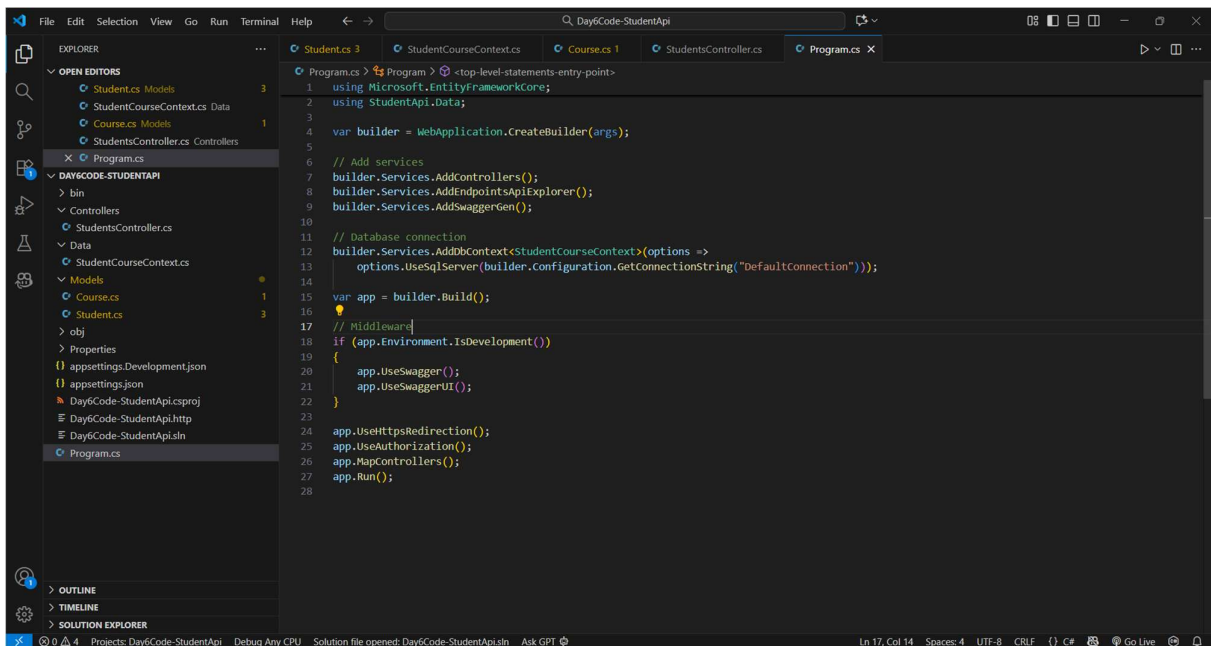
StudentCourseDB created successfully in SQL Server.



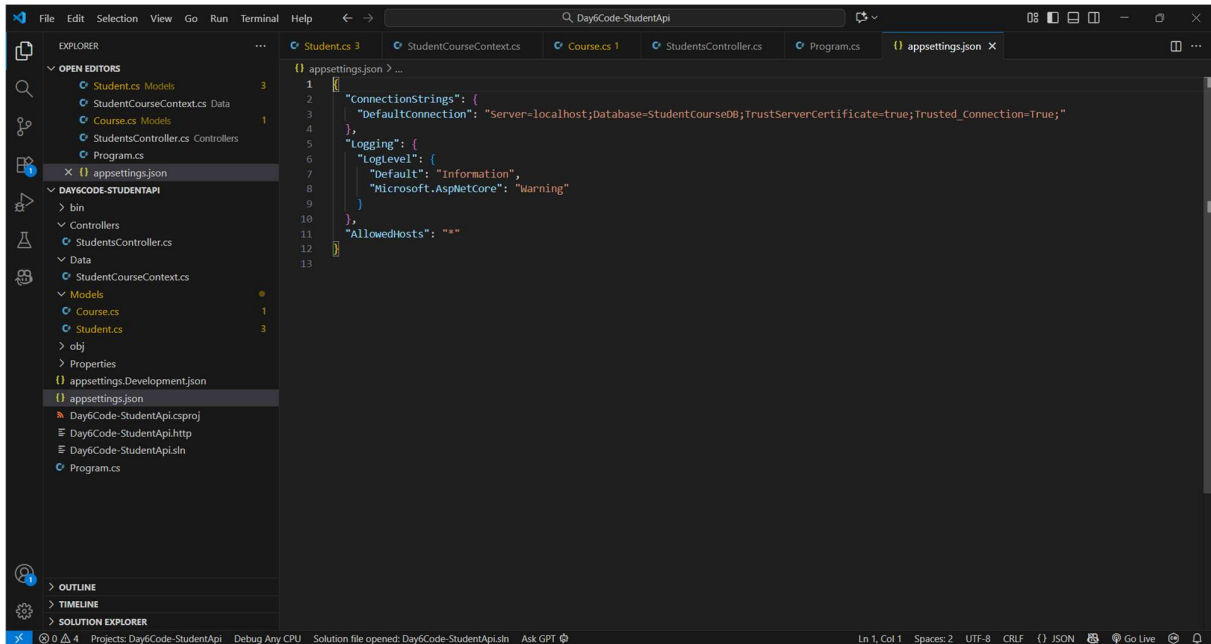
Students and Courses tables created with sample data in SSMS.



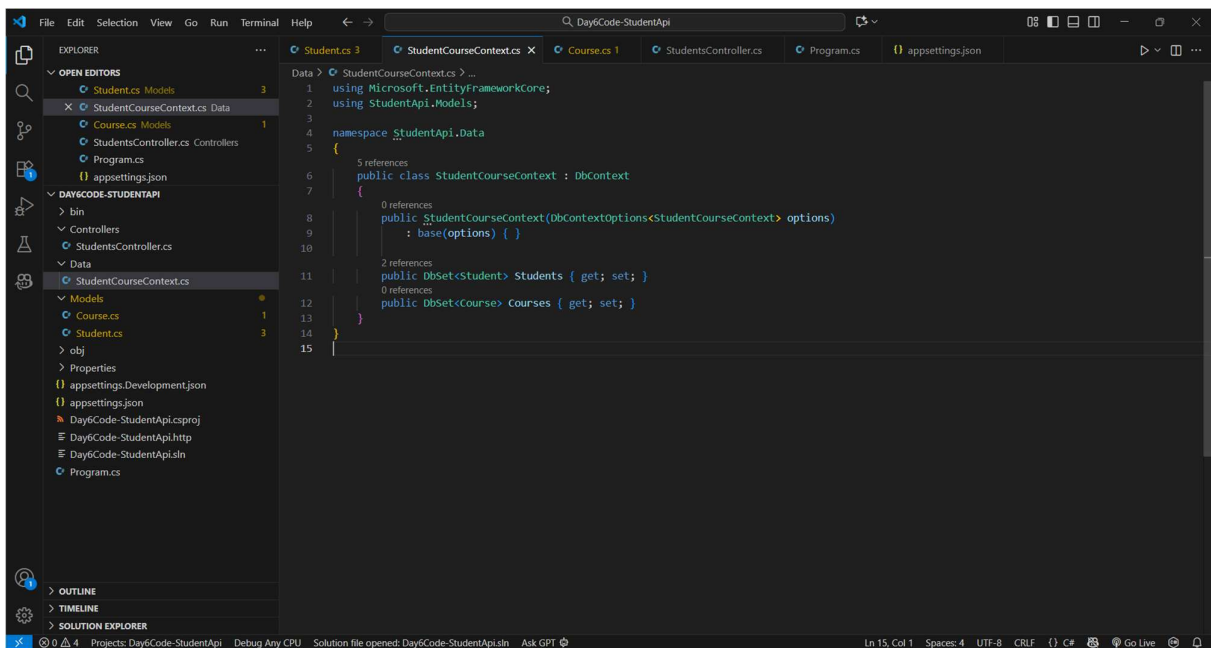
All stored procedures (CRUD) created and visible in SSMS under Programmability → Stored Procedures.



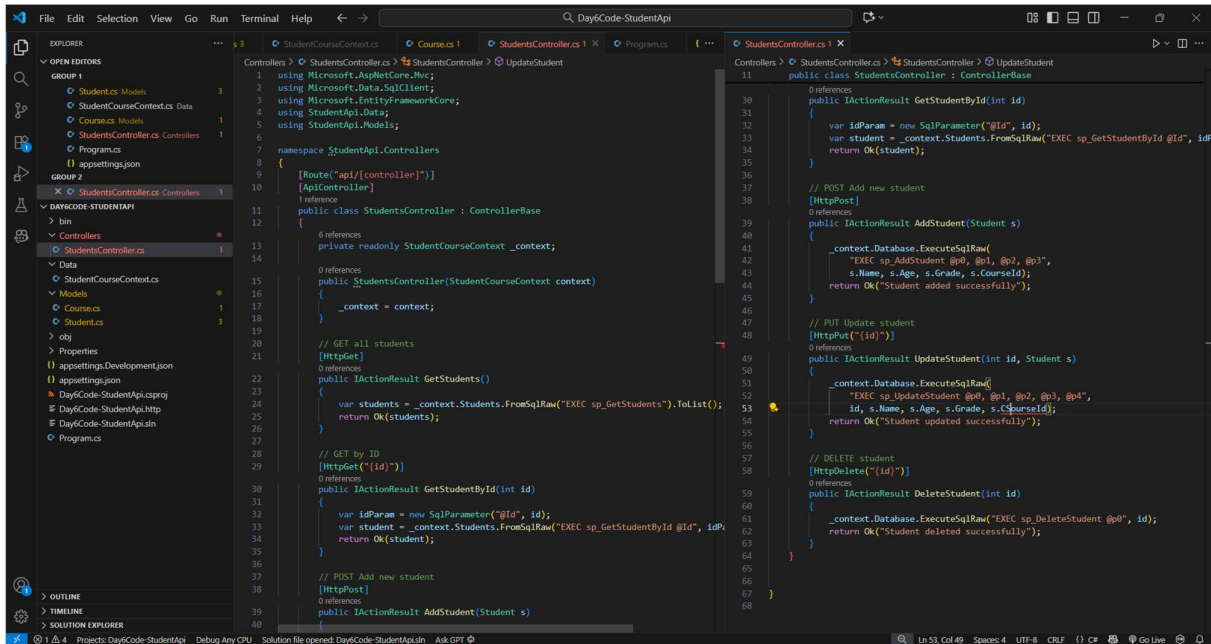
ASP.NET Core Web API project structure showing Controllers, Models, Data, and Program.cs files.



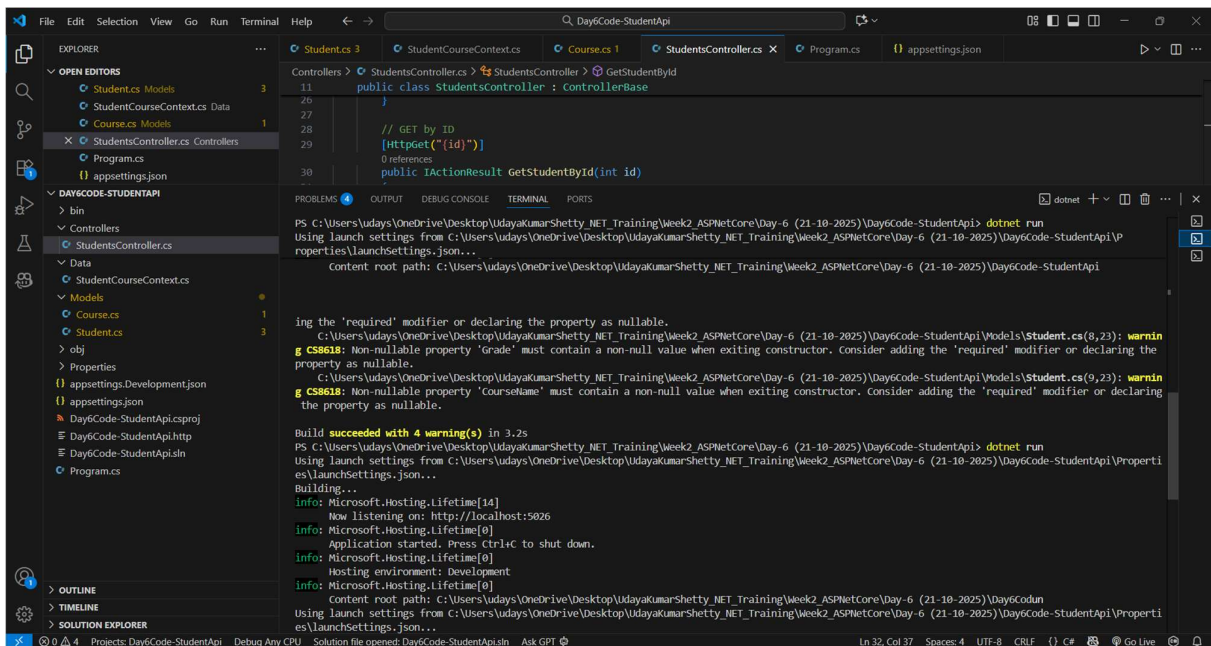
Connection string configured in appsettings.json to link API with SQL Server database.




StudentCourseContext.cs showing DbSet<Student> and DbSet<Course> for EF Core integration.



StudentsController configured to call SQL stored procedures using FromSqlRaw and ExecuteSqlRaw.



EF Core migration and database update commands executed successfully in terminal.

 Swagger
powered by SMARTBEAR

Select a definition Day2Code-StudentApi v1

Day6Code-StudentApi 1.0 OAS 3.0

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

Students

GET /api/Students

Parameters

No parameters

Execute Clear

Responses

Curl

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

Request URL

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

Server response

Code

Details

200

Response body

```
{
  "id": 1,
  "name": "Udaya Kumar Shetty",
  "age": 22,
  "grade": "A",
  "courseName": "C# Fundamentals",
  "courseId": 1
},
{
  "id": 2,
  "name": "Prajwal M",
  "age": 23,
  "grade": "B",
  "courseName": "ASP.NET Core Development",
  "courseId": 2
},
{
  "id": 3,
  "name": "Sneha B",
  "age": 21,
  "grade": "A",
  "courseName": "SQL & Database Management",
  "courseId": 3
},
{
  "id": 4,
  "name": "Sneha B",
  "age": 21,
  "grade": "A",
  "courseName": "SQL & Database Management",
  "courseId": 3
}
]
```

Response headers

```
content-type: application/json; charset=utf-8
date: Wed, 29 Oct 2025 07:41:20 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	No links

POST /api/Students

GET /api/Students/{id}


PUT /api/Students/{id}

DELETE /api/Students/{id}

Schemas

Student >

GET /api/students tested successfully in Swagger fetching student data from stored procedure.

 **Swagger**
OpenAPI 3.0

Select a definition **Day2Code-StudentApi v1**

Day6Code-StudentApi ^{1.0} **OAS 3.0**

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

Students

GET /api/Students

POST /api/Students

Parameters

No parameters

Request body

application/json

Edit Value | Schema

```
{  "id": 6,  "name": "Raj",  "age": 21,  "grade": "A",  "courseName": "java",  "courseId": 1}
```

Execute

Clear

Responses

Curl

```
curl -X 'POST' \  "http://localhost:5026/api/Students" \  -H 'accept: */*' \  -H 'Content-Type: application/json' \  -d '{  "id": 6,  "name": "Raj",  "age": 21,  "grade": "A",  "courseName": "java",  "courseId": 1}'
```

Request URL

http://localhost:5026/api/Students

Server response

Code	Details
200	<div><div>Response body</div><div>Student added successfully</div><div>Download</div></div> <div><div>Response headers</div><div>content-type: text/plain; charset=utf-8 date: Wed, 29 Oct 2025 07:42:21 GMT server: Kestrel transfer-encoding: chunked</div></div>

Responses

Code	Description	Links
200	OK	No links

GET /api/Students/{id}


PUT /api/Students/{id}

DELETE /api/Students/{id}

Schemas

Student >

POST /api/students adds a new student record to SQL database.

 **Swagger**
OpenAPI 3.0

Select a definition Day2Code-StudentApi v1

Day6Code-StudentApi 1.0 OAS 3.0

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

Students

GET

/api/Students

POST

/api/Students

GET

/api/Students/{id}

PUT

/api/Students/{id}

DELETE

/api/Students/{id}

Parameters

Cancel

Name	Description
id <small>required</small>	<input type="text" value="6"/>
<small>integer(int32)</small>	<small>(path)</small>

Execute

Clear

Responses

Curl

```
curl -X 'DELETE' \
  'http://localhost:5026/api/Students/6' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5026/api/Students/6
```

Server response

Code	Details
200	<div><div>Response body</div><div>Student deleted successfully</div><div>Download</div></div> <div><div>Response headers</div><div>content-type: text/plain; charset=utf-8 date: Wed, 29 Oct 2025 07:42:44 GMT server: Kestrel transfer-encoding: chunked</div></div>

Responses

Code	Description	Links
200	OK	No links

Schemas

Student >

DELETE /api/students/{id} successfully deletes a record using stored procedure.