Day 3 – CRUD Operations Using Entity Framework Core **Date:** 18-10-2025

Objective: Learn how to perform Create, Read, Update, and Delete operations using Entity Framework Core. Build data-driven .NET apps with clean, maintainable code.

1. Introduction

The term **CRUD** stands for the four basic database operations - Create**, Read, Update, and Delete**.

In EF Core, CRUD operations are performed using the **DbContext** and **DbSet** classes, which represent your database and tables in object-oriented form.

Operation	SQL Equivalent	EF Core Method
Create	INSERT	Add(), AddAsync()
Read	SELECT	ToList(), Find(), Where()
Update	UPDATE	Update()
Delete	DELETE	Remove()

2. CRUD Architecture Overview

In an ASP.NET Core Web API project, CRUD operations usually follow a **three-layer structure**:

1. **Model Layer** → Represents database entities.
2. **Data Access Layer (Repository)** → Contains logic for communicating with the database using EF Core.
3. **Controller Layer** → Exposes endpoints (API routes) that call repository methods.

This separation ensures the code is **modular**, **testable**, and **maintainable**.

3. Read Operation (GET)

The **Read** operation retrieves data from the database.

Example:

```
var students = _context.Students.ToList();
```

- This fetches all rows from the Students table.
- EF Core translates it into:
- SELECT * FROM Students;

With Filter:

```
var student = _context.Students.FirstOrDefault(s => s.Id == id);
```

- This retrieves a specific student by ID.

4. Create Operation (POST)

The **Create** operation inserts new data into the database.

Example:

```
[HttpPost]
public async Task<IActionResult> AddStudent(Student student)
{
   _context.Students.Add(student);
   await _context.SaveChangesAsync();
   return CreatedAtAction(nameof(GetStudentById), new { id = student.Id }, student);
}
```

Explanation:

- Add() → Adds the new record to the change tracker.
- SaveChangesAsync() → Executes the INSERT query.
- CreatedAtAction() → Returns HTTP 201 status code with the new resource URI.

5. Update Operation (PUT)

The **Update** operation modifies existing data.

Example:

```
[HttpPut("{id}")]
public async Task<IActionResult> UpdateStudent(int id, Student student)
{
   if (id != student.Id)
      return BadRequest("Student ID mismatch");
   _context.Entry(student).State = EntityState.Modified;
   await _context.SaveChangesAsync();
   return Ok("Student updated successfully")  }
```

Explanation:

- Marks entity as Modified in the DbContext.
- When SaveChangesAsync() is called, EF Core executes an UPDATE statement.

6. Delete Operation (DELETE)

The **Delete** operation removes a record.

Example:

```
[HttpDelete("{id}")]
public async Task<IActionResult> DeleteStudent(int id)
{
    var student = await _context.Students.FindAsync(id);
    if (student == null)
        return NotFound("Student not found");
    _context.Students.Remove(student);
    await _context.SaveChangesAsync();
    return Ok("Student deleted successfully");
}
```

Explanation:

- FindAsync() retrieves the record.
- Remove() marks it for deletion.
- SaveChangesAsync() performs the DELETE operation in SQL.

7. Using Asynchronous Operations

All EF Core operations can be executed asynchronously for better performance:

Operation	Async Method
Add	AddAsync()
Save	SaveChangesAsync()
Find	FindAsync()
ToList	ToListAsync()

Example:

```
var students = await _context.Students.ToListAsync();
```

This prevents blocking the main thread during long-running database operations.

8. Repository Pattern (Optional for Larger Projects)

To keep the code modular, use a **repository pattern**.

IStudentRepository.cs

```
public interface IStudentRepository
{
    Task<IEnumerable<Student>> GetAllAsync();
    Task<Student> GetByIdAsync(int id);
    Task<Student> AddAsync(Student student);
    Task UpdateAsync(Student student);
    Task DeleteAsync(int id);
}
```

StudentRepository.cs

```
public class StudentRepository : IStudentRepository
{
    private readonly StudentDBContext _context;
    public StudentRepository(StudentDBContext context) => _context = context;

    public async Task<IEnumerable<Student>> GetAllAsync() => await
_context.Students.ToListAsync();
    public async Task<Student> GetByIdAsync(int id) => await _context.Students.FindAsync(id);
    public async Task<Student> AddAsync(Student student)
    {
        _context.Students.Add(student);
        await _context.SaveChangesAsync();
        return student;
    }
    public async Task UpdateAsync(Student student)
    {
        _context.Entry(student).State = EntityState.Modified;
        await _context.SaveChangesAsync();
```

```
    }
    public async Task DeleteAsync(int id)
    {
        var student = await _context.Students.FindAsync(id);
        if (student != null)
        {
            _context.Students.Remove(student);
            await _context.SaveChangesAsync();
        }
    }
}
```

The controller can then depend on this interface instead of the direct context, improving **testability**.

9. Error Handling and Validation

Always validate and handle exceptions during CRUD operations:

Example:

```
try
{
    await _context.SaveChangesAsync();
}
catch (DbUpdateException ex)
{
    return BadRequest("Database update failed: " + ex.Message);
}
```

Validation Example using Data Annotations:

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

10. HTTP Status Codes for CRUD APIs

Operation	Method	Success Code	Description
Create	POST	201	Resource created
Read	GET	200	Data retrieved
Update	PUT	200 / 204	Data updated
Delete	DELETE	200 / 204	Data deleted

11. Testing the API

Once all methods are implemented:

- Launch Swagger (https://localhost:****/swagger)
- Test:
    - **GET** → Fetch all students
    - **POST** → Add a new student
    - **PUT** → Update student
    - **DELETE** → Remove student

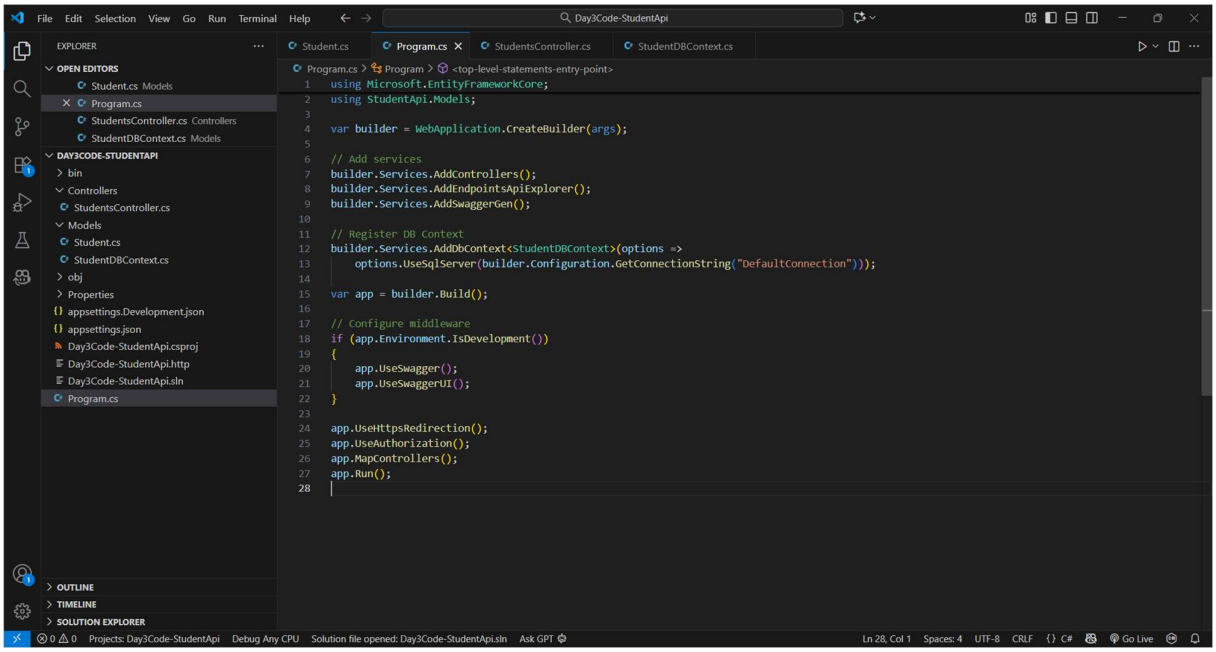
Take screenshots for documentation.

Mini Task for Day 3

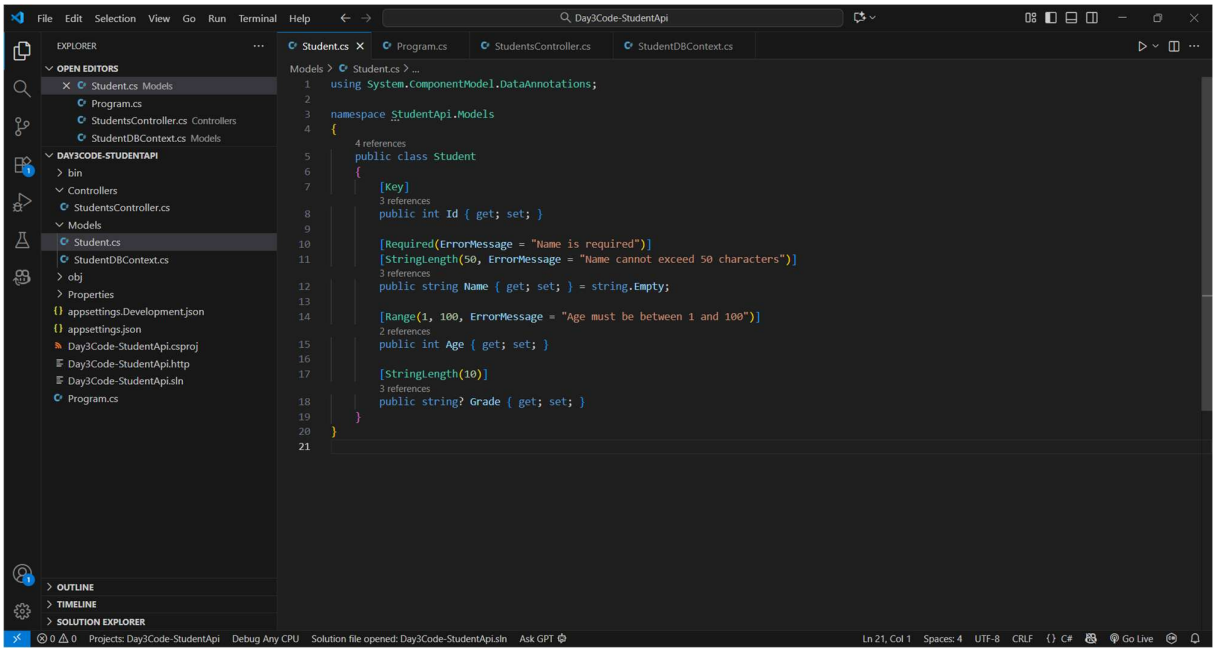
Build a Complete CRUD API for Students:

- Use SQL Server database StudentDB.
- Create StudentController with:
    - GET /api/students
    - GET /api/students/{id}
    - POST /api/students
    - PUT /api/students/{id}
    - DELETE /api/students/{id}
- Test all operations in Swagger.

Snapshots :



Code : Program.cs



Code : Student.cs

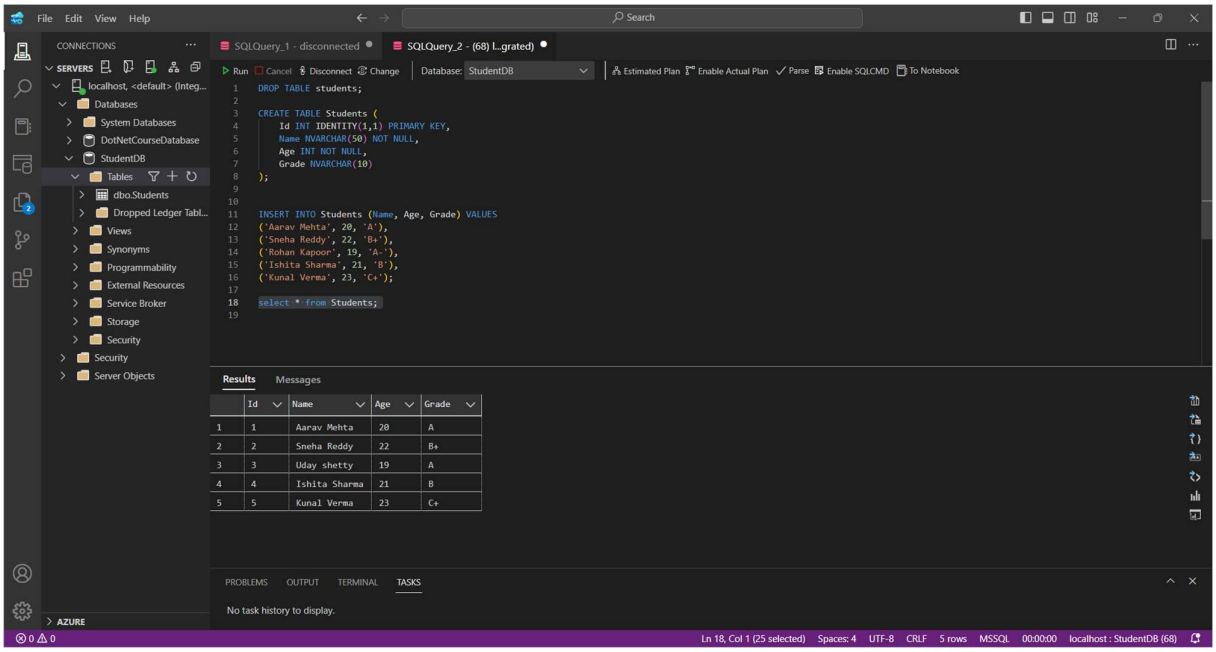
```
using Microsoft.EntityFrameworkCore;

namespace StudentApi.Models
{
    5 references
    public class StudentDBContext : DbContext
    {
        0 references
        public StudentDBContext(DbContextOptions<StudentDBContext> options)
            : base(options)
        {
        }

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

        0 references
        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

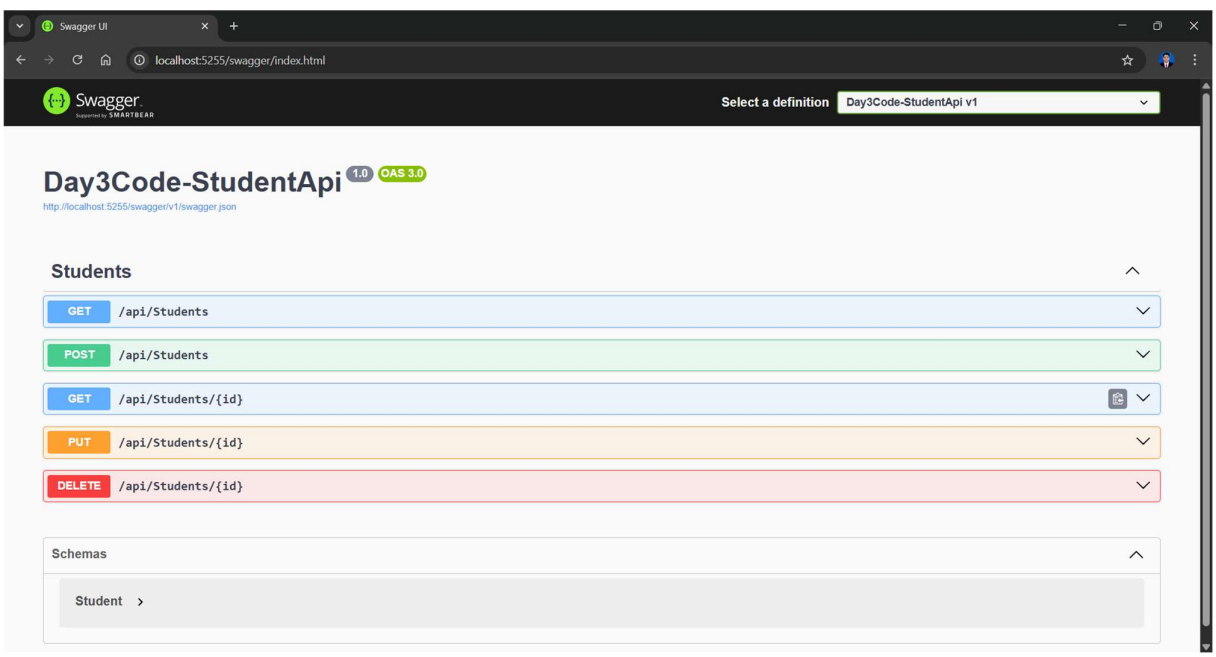
```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using StudentApi.Models;

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

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

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

        // GET: api/students/5
        [HttpGet("{id}")]
        1 reference
        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);
        }

        // POST: api/students
        [HttpPost]
        0 references
        public async Task<IActionResult> AddStudent([FromBody] Student student)
        {
            if (!ModelState.IsValid)
```

```
    public class StudentsController : ControllerBase
        public async Task<IActionResult> AddStudent([FromBody] Student student)
        }

        // PUT: api/students/5
        [HttpPut("{id}")]
        0 references
        public async Task<IActionResult> UpdateStudent(int id, [FromBody] Student student
        {
            if (id != student.Id)
                return BadRequest("ID mismatch");

            var existingStudent = await _context.Students.FindAsync(id);
            if (existingStudent == null)
                return NotFound("Student not found");

            existingStudent.Name = student.Name;
            existingStudent.Age = student.Age;
            existingStudent.Grade = student.Grade;

            _context.Entry(existingStudent).State = EntityState.Modified;
            await _context.SaveChangesAsync();

            return Ok("Student updated successfully");
        }

        // DELETE: api/students/5
        [HttpDelete("{id}")]
        0 references
        public async Task<IActionResult> DeleteStudent(int id)
        {
            var student = await _context.Students.FindAsync(id);
            if (student == null)
                return NotFound("Student not found");

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

            return Ok("Student deleted successfully");
        }
    }
}
```

Code : StudentController.cs



Code : Database Code



Output : All method

Swagger

Select a definition Day3Code-StudentApi v1

Day3Code-StudentApi 1.0 OAS 3.0

http://localhost:5255/swagger/v1/swagger.json

Students

GET /api/Students

Parameters Cancel

No parameters

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5255/api/Students' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5255/api/Students
```

Server response

Code	Details
200	Response body


```
    "grade": "A"
  },
  {
    "id": 2,
    "name": "Sneha Reddy",
    "age": 22,
    "grade": "B+"
  },
  {
    "id": 3,
    "name": "Rohan Kapoor",
    "age": 19,
    "grade": "A-"
  },
  {
    "id": 4,
    "name": "Ishita Sharma",
    "age": 21,
    "grade": "B"
  },
  {
    "id": 5,
    "name": "Kunal Verma",
    "age": 23,
    "grade": "C+"
  }
]
```


Download

Response headers

```
content-type: application/json; charset=utf-8
date: Tue,28 Oct 2025 16:58:38 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 ›

Output : GET (All data from database)

Swagger

Select a definition: Day3Code-StudentApi v1

Day3Code-StudentApi 1.0 OAS 3.0

http://localhost:5255/swagger/v1/swagger.json

Students

GET /api/Students

POST /api/Students

Parameters | Cancel

No parameters

Request body — application/json

Edit Value | Schema

```
{
  "id": 0,
  "name": "string",
  "age": 100,
  "grade": "string"
}
```

Execute | Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:5255/api/Students' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "id": 0,
  "name": "string",
  "age": 100,
  "grade": "string"
}'
```

Request URL

```
http://localhost:5255/api/Students
```

Server response

Code	Details
201 *Undocumented*	Response body

```
{
  "id": 7,
  "name": "string",
  "age": 100,
  "grade": "string"
}
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Tue,28 Oct 2025 17:00:06 GMT
location: http://localhost:5255/api/Students/7
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	*No links*

GET /api/Students/{id}

PUT /api/Students/{id}

DELETE /api/Students/{id}

Schemas

Student ›

Output : POST Method (Insert data)

Swagger

Select a definition Day3Code-StudentApi v1

Day3Code-StudentApi 1.0 OAS 3.0

http://localhost:5255/swagger/v1/swagger.json

Students

GET /api/Students

POST /api/Students

GET /api/Students/{id}

Parameters Cancel

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

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5255/api/Students/1' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5255/api/Students/1
```

Server response

Code	Details
200	Response body

```
{
  "id": 1,
  "name": "Aarav Mehta",
  "age": 20,
  "grade": "A"
}
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Tue,28 Oct 2025 17:00:36 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	No links

PUT /api/Students/{id}

DELETE /api/Students/{id}

Schemas

Student ›

Output : GET (Specific data from database)

Swagger

Select a definition: Day3Code-StudentApi v1

Day3Code-StudentApi

1.0 OAS 3.0

http://localhost:5255/swagger/v1/swagger.json

Students

GET /api/Students

POST /api/Students

GET /api/Students/{id}

PUT /api/Students/{id}

Parameters

Cancel Reset

Name	Description
id * required integer($int32) (path)	3

Request body

application/json

Edit Value | Schema

```
{
  "id": 3,
  "name": "Uday shetty",
  "age": 19,
  "grade": "A"
}
```

Execute Clear

Responses

Curl

```
curl -X 'PUT' \
  'http://localhost:5255/api/Students/3' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "id": 3,
  "name": "Uday shetty",
  "age": 19,
  "grade": "A"
}'
```

Request URL

http://localhost:5255/api/Students/3

Server response

Code	Details
200	Response body Student updated successfully Download Response headers content-type: text/plain; charset=utf-8 date: Tue,28 Oct 2025 17:02:03 GMT server: Kestrel transfer-encoding: chunked

Responses

Code	Description	Links
200	OK	No links

DELETE /api/Students/{id}

Schemas

Student >

Output : PUT (Update data)

Swagger

Select a definition Day3Code-StudentApi v1

Day3Code-StudentApi 1.0 OAS 3.0

http://localhost:5255/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 * required integer($int32) (path)	7

Execute Clear

Responses

Curl

```
curl -X 'DELETE' \
  'http://localhost:5255/api/Students/7' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5255/api/Students/7
```

Server response

Code	Details

200	Response body
	`Student deleted successfully`
	Download
	Response headers
	`content-type: text/plain; charset=utf-8`
	`date: Tue,28 Oct 2025 17:02:45 GMT`
	`server: Kestrel`
	`transfer-encoding: chunked`

Responses

Code	Description	Links
200	OK	No links

Schemas

Student ›

Output : DELETE (Delete data)