**. Folder Structure**

StudentManagement/

│

├── Models/

│ └── Student.cs

│

├── Services/

│ └── StudentService.cs

│

├── Controllers/

│ └── StudentController.cs

│

└── Program.cs

**2. Model – Student.cs**

namespace StudentManagement.Models

{

public class Student

{

public int StudentId { get; set; }

public string Name { get; set; } = string.Empty;

public int Age { get; set; }

public string Grade { get; set; } = string.Empty;

}

}

**3. Service – StudentService.cs**

using StudentManagement.Models;

using System.Collections.Generic;

using System.Linq;

namespace StudentManagement.Services

{

public class StudentService

{

private readonly List<Student> \_students;

public StudentService()

{

\_students = new List<Student>

{

new Student { StudentId = 1, Name = "Alice", Age = 18, Grade = "A" },

new Student { StudentId = 2, Name = "Bob", Age = 17, Grade = "B" },

new Student { StudentId = 3, Name = "Charlie", Age = 16, Grade = "C" }

};

}

public IEnumerable<Student> GetAllStudents()

{

return \_students;

}

public Student? GetStudentById(int id)

{

return \_students.FirstOrDefault(s => s.StudentId == id);

}

public Student CreateStudent(Student newStudent)

{

int nextId = \_students.Any() ? \_students.Max(s => s.StudentId) + 1 : 1;

newStudent.StudentId = nextId;

\_students.Add(newStudent);

return newStudent;

}

public bool UpdateStudent(int id, Student updatedStudent)

{

var student = \_students.FirstOrDefault(s => s.StudentId == id);

if (student == null)

return false;

student.Name = updatedStudent.Name;

student.Age = updatedStudent.Age;

student.Grade = updatedStudent.Grade;

return true;

}

public bool DeleteStudent(int id)

{

var student = \_students.FirstOrDefault(s => s.StudentId == id);

if (student == null)

return false;

\_students.Remove(student);

return true;

}

}

}

**4. Controller – StudentController.cs**

using Microsoft.AspNetCore.Mvc;

using StudentManagement.Models;

using StudentManagement.Services;

namespace StudentManagement.Controllers

{

[ApiController]

[Route("api/[controller]")]

public class StudentController : ControllerBase

{

private readonly StudentService \_studentService;

public StudentController(StudentService studentService)

{

\_studentService = studentService;

}

// GET: api/Student

[HttpGet]

public ActionResult<IEnumerable<Student>> GetAllStudents()

{

var students = \_studentService.GetAllStudents();

if (!students.Any())

return NoContent(); // 204 No Content

return Ok(students); // 200 OK

}

// GET: api/Student/{id}

[HttpGet("{id}")]

public ActionResult<Student> GetStudentById(int id)

{

var student = \_studentService.GetStudentById(id);

if (student == null)

return NotFound(); // 404 Not Found

return Ok(student); // 200 OK

}

// POST: api/Student

[HttpPost]

public ActionResult<Student> CreateStudent([FromBody] Student newStudent)

{

if (newStudent == null)

return BadRequest("Invalid student data."); // 400 Bad Request

var createdStudent = \_studentService.CreateStudent(newStudent);

return CreatedAtAction(nameof(GetStudentById), new { id = createdStudent.StudentId }, createdStudent); // 201 Created

}

// PUT: api/Student/{id}

[HttpPut("{id}")]

public IActionResult UpdateStudent(int id, [FromBody] Student updatedStudent)

{

if (updatedStudent == null)

return BadRequest("Invalid student data."); // 400 Bad Request

var isUpdated = \_studentService.UpdateStudent(id, updatedStudent);

if (!isUpdated)

return NotFound(); // 404 Not Found

return NoContent(); // 204 No Content

}

// DELETE: api/Student/{id}

[HttpDelete("{id}")]

public IActionResult DeleteStudent(int id)

{

var isDeleted = \_studentService.DeleteStudent(id);

if (!isDeleted)

return NotFound(); // 404 Not Found

return NoContent(); // 204 No Content

}

}

}

**5. Program.cs**

using StudentManagement.Services;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container

builder.Services.AddSingleton<StudentService>();

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.MapControllers();

app.Run();

**6. Running the Project**

Open the terminal in the project folder and run these commands:

cd StudentManagement

dotnet restore

dotnet build

dotnet run

Then, open in browser:

http://localhost:8080/swagger/index.html

You’ll see the Swagger UI showing the following endpoints:

* GET /api/Student – Get all students
* GET /api/Student/{id} – Get student by ID
* POST /api/Student – Add new student
* PUT /api/Student/{id} – Update student by ID
* DELETE /api/Student/{id} – Delete student by ID

**7. Expected HTTP Status Codes**

| **Operation** | **Success Code** | **Failure Code** | **Notes** |
| --- | --- | --- | --- |
| GET All Students | 200 OK / 204 NoContent | – | Returns list or empty result |
| GET by ID | 200 OK | 404 NotFound |  |
| POST | 201 Created | 400 BadRequest |  |
| PUT | 204 NoContent | 404 NotFound / 400 BadRequest |  |
| DELETE | 204 NoContent | 404 NotFound |  |