# **WEEK 4: ASP.NET Core 8.0 Web API**

## **Create a .Net core web application with API template. Use the option to create controller with Read Write permissions. Notice the ValuesController creation with Action methods corresponding to the Action verbs. On creation of the Web API, execute the application and check if the GET action method result is returned as expected.**

### **Code:**

**StudentController.cs**

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

namespace MyFirstWebAPI.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    public class StudentController : ControllerBase

    {

        private static List<string> students = new() { "Priya", "John", "Sara" };

        [HttpGet]

        public ActionResult<IEnumerable<string>> GetAll()

        {

            return Ok(students);

        }

        [HttpPost]

        public ActionResult AddStudent([FromBody] string name)

        {

            if (string.IsNullOrWhiteSpace(name))

                return BadRequest("Name cannot be empty");

            students.Add(name);

            return Ok("Student added!");

        }

        [HttpPut("{index}")]

        public ActionResult UpdateStudent(int index, [FromBody] string name)

        {

            if (index < 0 || index >= students.Count)

                return BadRequest("Invalid index");

            students[index] = name;

            return Ok("Student updated!");

        }

        [HttpDelete("{index}")]

        public ActionResult DeleteStudent(int index)

        {

            if (index < 0 || index >= students.Count)

                return BadRequest("Invalid index");

            students.RemoveAt(index);

            return Ok("Student deleted!");

        }

    }

}

**Program.cs**

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

var app = builder.Build();

app.UseRouting();

app.UseEndpoints(endpoints =>

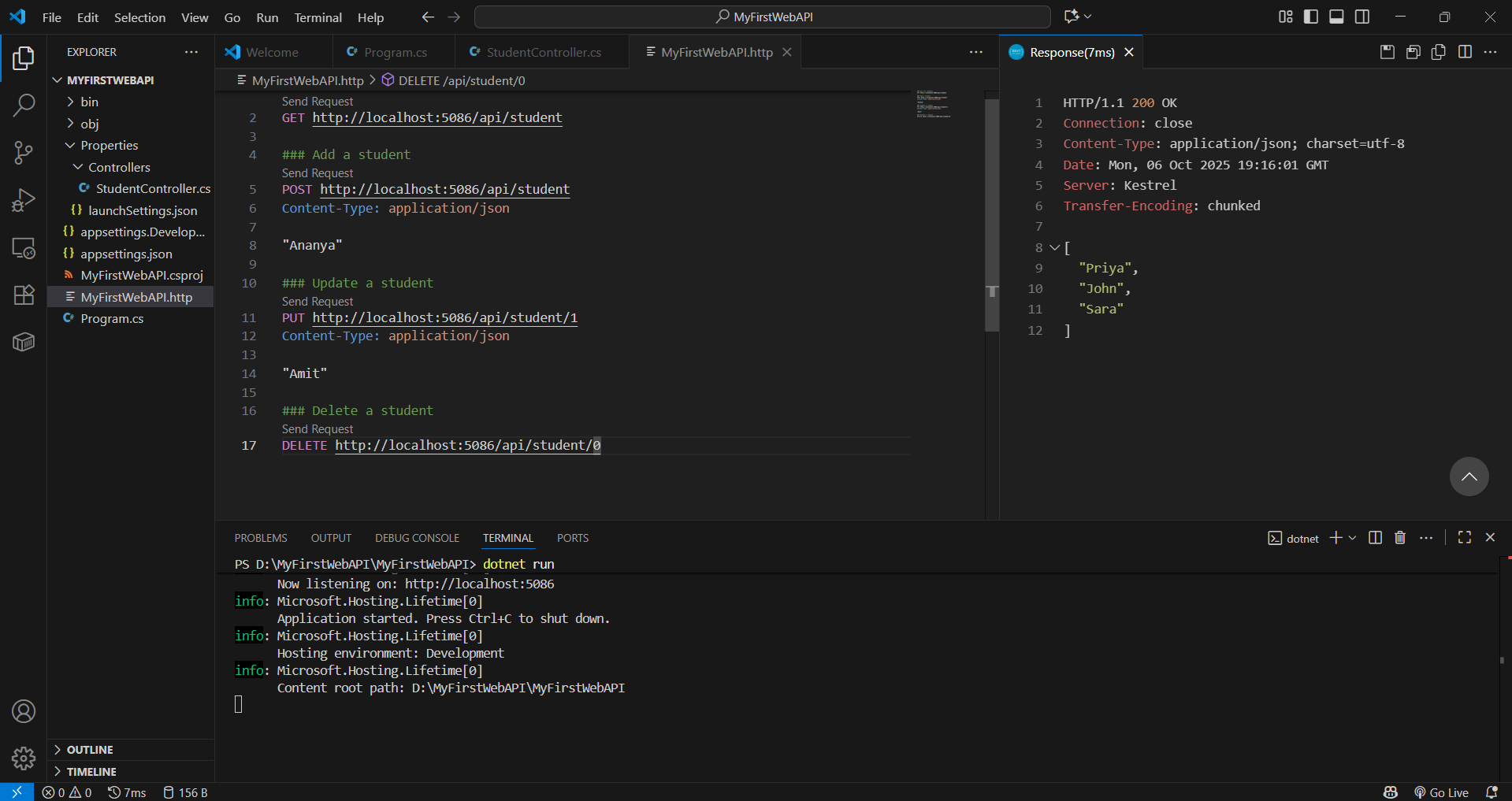
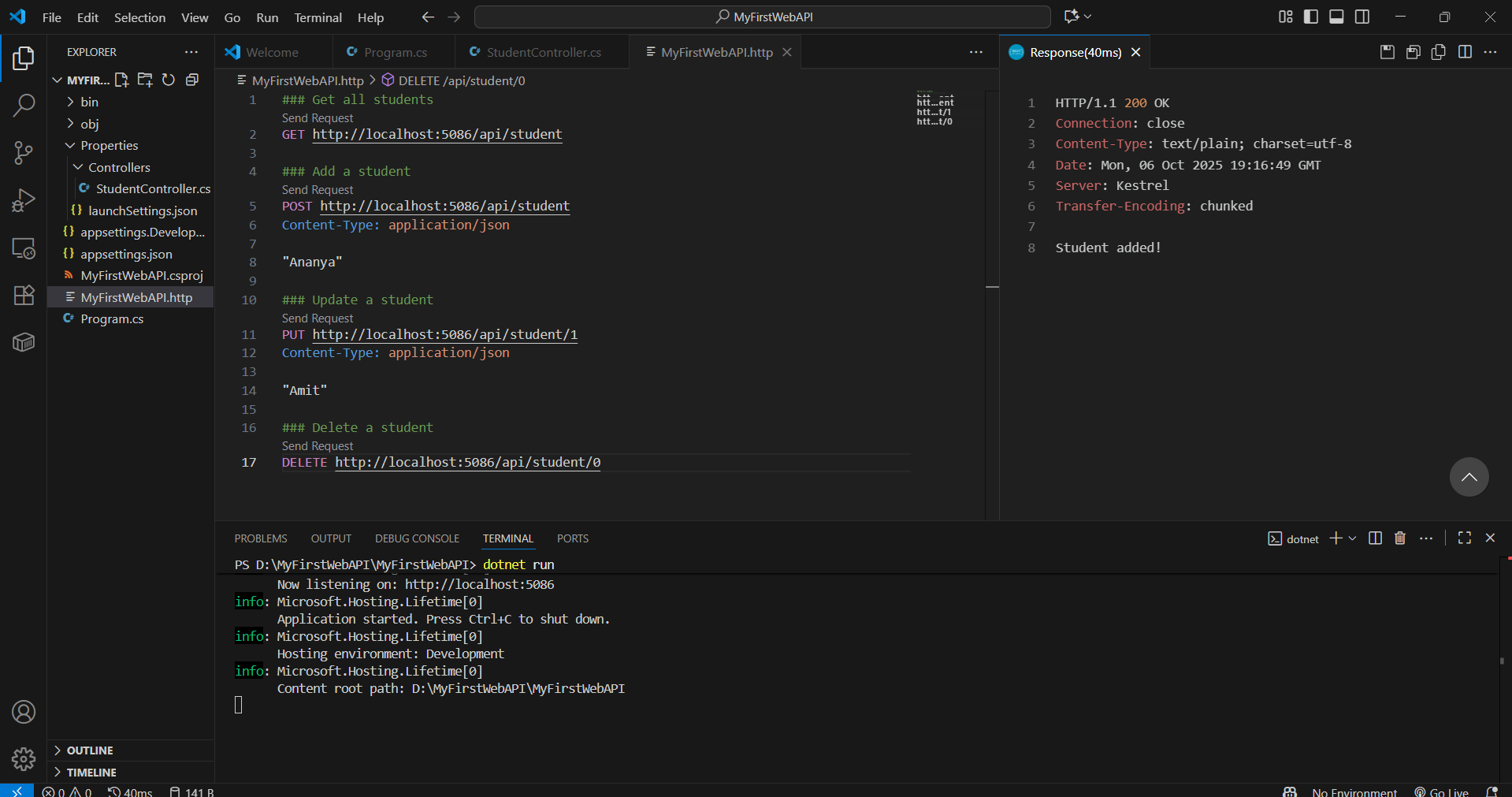
{

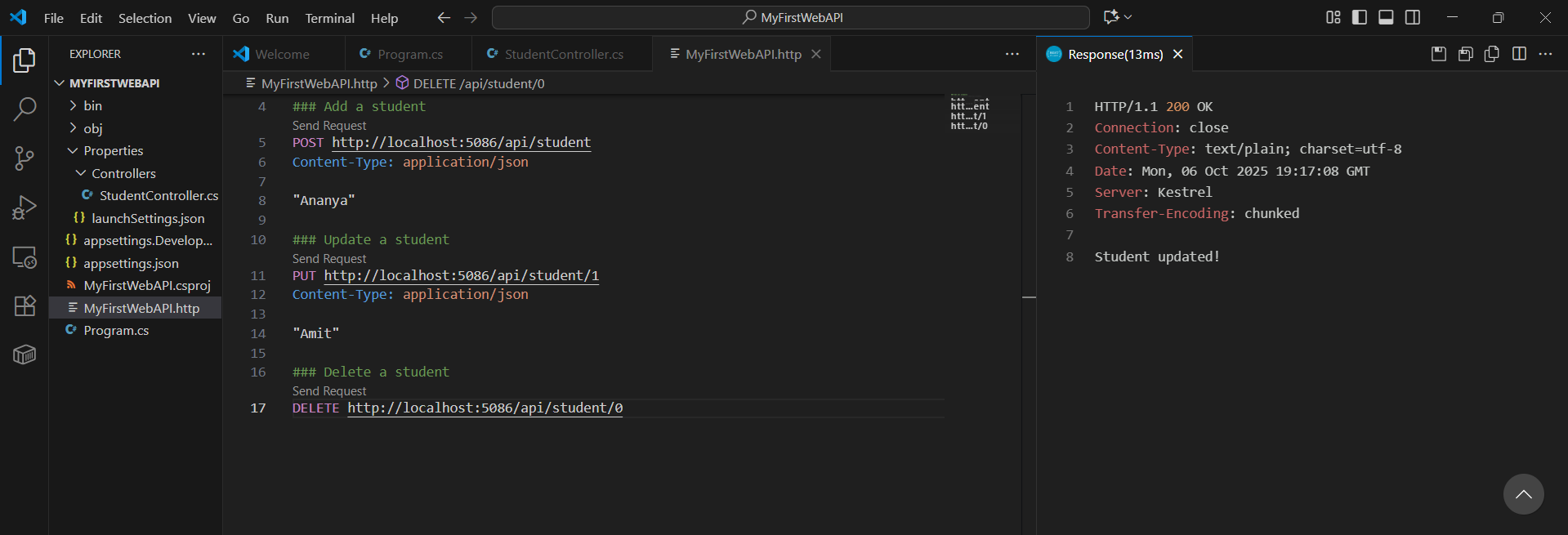
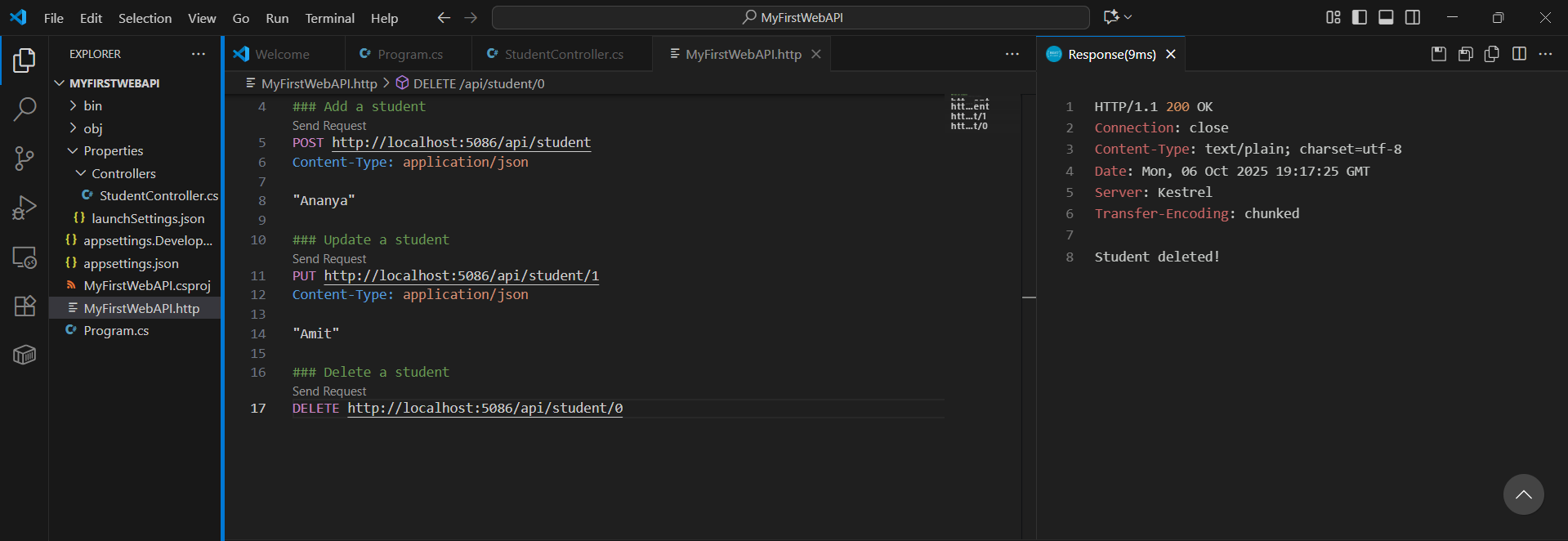
    endpoints.MapControllers();

});

app.Run();

### **Output:**





## **Use Swagger and Postman to Test**

### **Code:**

**StudentController.cs:**

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

namespace MyFirstWebAPI.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    public class StudentController : ControllerBase

    {

        private static List<string> students = new() { "Priya", "John", "Sara" };

        [HttpGet]

        [ProducesResponseType(StatusCodes.Status200OK)]

        [ProducesResponseType(StatusCodes.Status500InternalServerError)]

        public ActionResult<IEnumerable<string>> GetAll()

        {

            return Ok(students);

        }

        [HttpPost]

        public ActionResult AddStudent([FromBody] string name)

        {

            if (string.IsNullOrWhiteSpace(name))

                return BadRequest("Name cannot be empty");

            students.Add(name);

            return Ok("Student added!");

        }

        [HttpPut("{index}")]

        public ActionResult UpdateStudent(int index, [FromBody] string name)

        {

            if (index < 0 || index >= students.Count)

                return BadRequest("Invalid index");

            students[index] = name;

            return Ok("Student updated!");

        }

        [HttpDelete("{index}")]

        public ActionResult DeleteStudent(int index)

        {

            if (index < 0 || index >= students.Count)

                return BadRequest("Invalid index");

            students.RemoveAt(index);

            return Ok("Student deleted!");

        }

    }

}

**Program.cs**:

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddSwaggerGen(c =>

{

    c.SwaggerDoc("v1", new OpenApiInfo

    {

        Title = "Swagger Demo",

        Version = "v1",

        Description = "Demo API using Swagger UI",

        TermsOfService = new Uri("https://example.com/terms"),

        Contact = new OpenApiContact

        {

            Name = "John Doe",

            Email = "john@xyzmail.com",

            Url = new Uri("https://www.example.com")

        },

        License = new OpenApiLicense

        {

            Name = "License Terms",

            Url = new Uri("https://www.example.com")

        }

    });

});

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

    app.UseSwagger();

    app.UseSwaggerUI(c =>

    {

        c.SwaggerEndpoint("/swagger/v1/swagger.json", "Swagger Demo v1");

    });

}

app.UseRouting();

app.UseEndpoints(endpoints =>

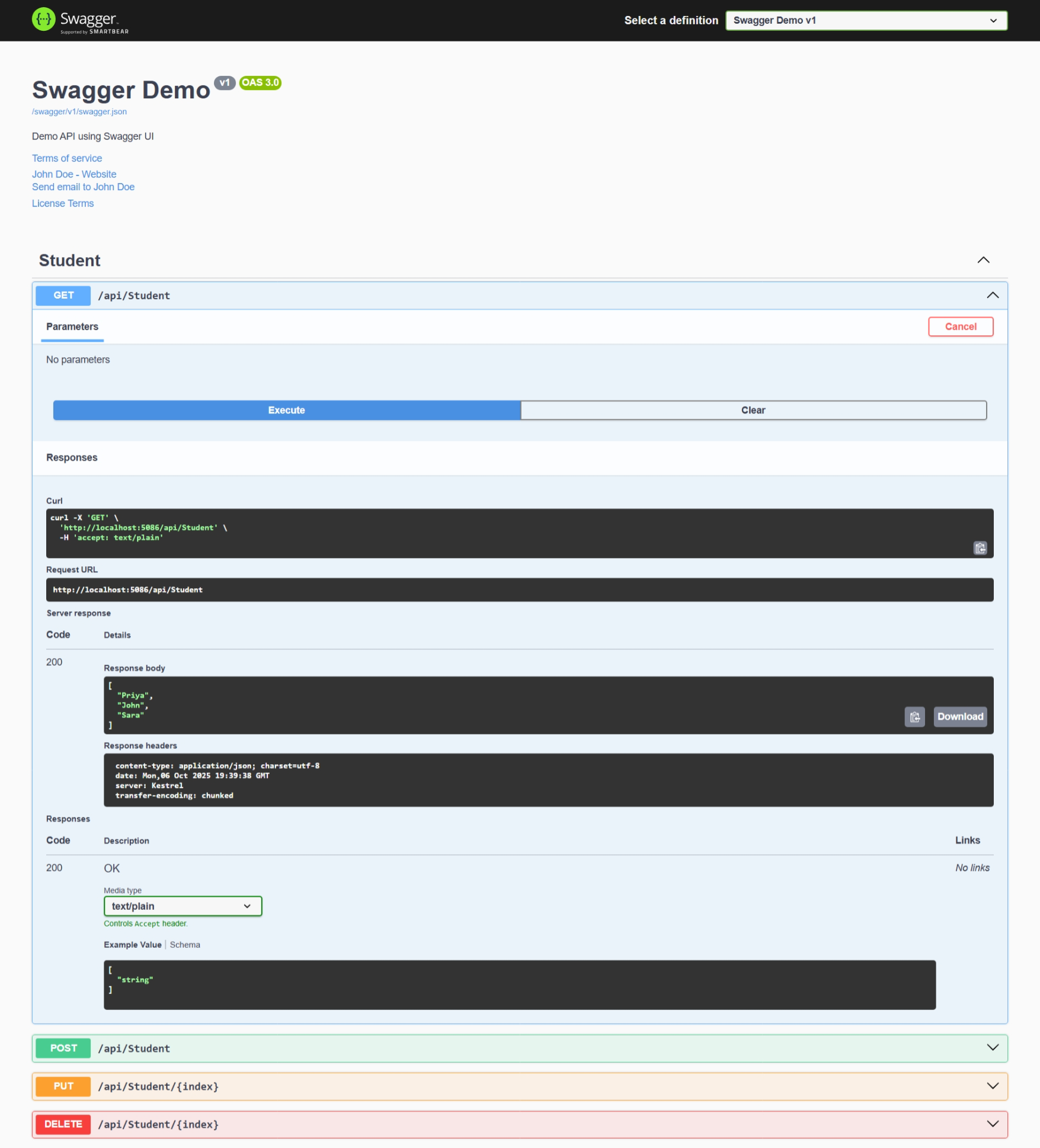
{

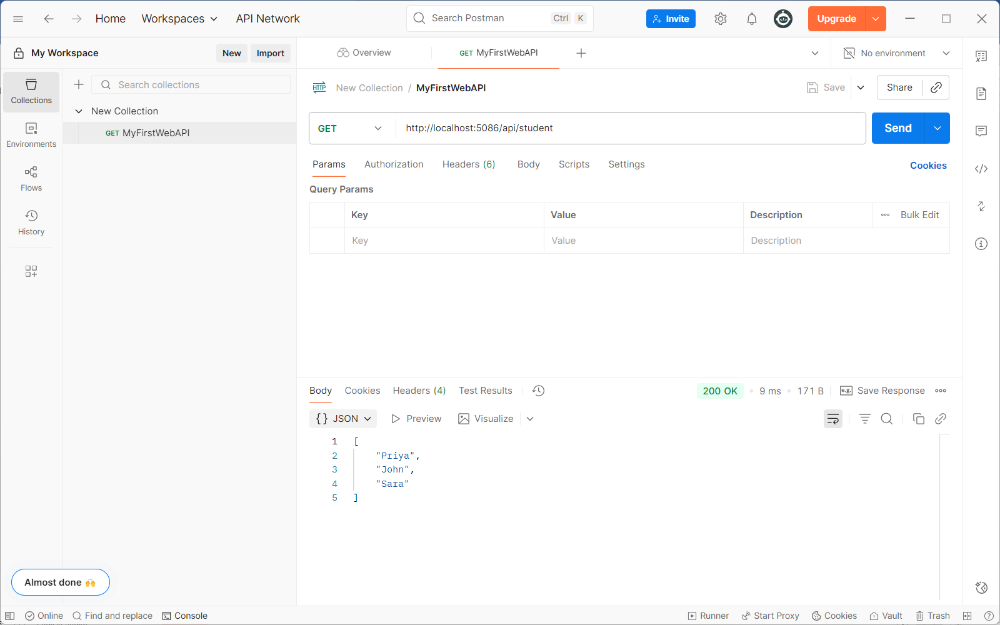
    endpoints.MapControllers();

});

app.Run();

### **Output:**





## **ASP.NET Core Web API – Custom Model, Filters & Exception Handling**

### **Code:**

**Controllers/Employee.cs**

using Microsoft.AspNetCore.Mvc;

using WebApiDemo.Models;

using WebApiDemo.Filters;

using System.Collections.Generic;

using System;

using Microsoft.AspNetCore.Authorization;

namespace WebApiDemo.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    [CustomAuthFilter]

    public class EmployeeController : ControllerBase

    {

        private readonly List<Employee> \_employees;

        public EmployeeController()

        {

            \_employees = GetStandardEmployeeList();

        }

        private List<Employee> GetStandardEmployeeList()

        {

            return new List<Employee>()

            {

                new Employee

                {

                    Id = 1,

                    Name = "Alice",

                    Salary = 60000,

                    Permanent = true,

                    DateOfBirth = new DateTime(1995, 4, 12),

                    Department = new Department { Id = 101, DeptName = "IT" },

                    Skills = new List<Skill>

                    {

                        new Skill{ Id = 1, SkillName = "C#" },

                        new Skill{ Id = 2, SkillName = "SQL" }

                    }

                },

                new Employee

                {

                    Id = 2,

                    Name = "Bob",

                    Salary = 50000,

                    Permanent = false,

                    DateOfBirth = new DateTime(1998, 7, 22),

                    Department = new Department { Id = 102, DeptName = "HR" },

                    Skills = new List<Skill>

                    {

                        new Skill{ Id = 3, SkillName = "Excel" }

                    }

                }

            };

        }

        [HttpGet]

        [AllowAnonymous]

        [ProducesResponseType(StatusCodes.Status200OK)]

        public ActionResult<List<Employee>> Get()

        {

            return Ok(\_employees);

        }

        [HttpGet("{id}")]

        [ProducesResponseType(StatusCodes.Status500InternalServerError)]

        public ActionResult<Employee> GetById(int id)

        {

            if (id == 999)

                throw new Exception("Custom exception triggered!");

            var emp = \_employees.Find(e => e.Id == id);

            if (emp == null)

                return NotFound();

            return Ok(emp);

        }

        [HttpPost]

        public ActionResult<Employee> Create([FromBody] Employee employee)

        {

            \_employees.Add(employee);

            return Ok(employee);

        }

    }

}

**Filters/CustomAuthFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System.Linq;

namespace WebApiDemo.Filters

{

    public class CustomAuthFilter : ActionFilterAttribute

    {

        public override void OnActionExecuting(ActionExecutingContext context)

        {

            var headers = context.HttpContext.Request.Headers;

            if (!headers.ContainsKey("Authorization"))

            {

                context.Result = new BadRequestObjectResult("Invalid request - No Auth token");

                return;

            }

            var authHeader = headers["Authorization"].ToString();

            if (!authHeader.Contains("Bearer"))

            {

                context.Result = new BadRequestObjectResult("Invalid request - Token present but Bearer unavailable");

                return;

            }

            base.OnActionExecuting(context);

        }

    }

}

**Filters/CustomExceptionFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System;

using System.IO;

namespace WebApiDemo.Filters

{

    public class CustomExceptionFilter : IExceptionFilter

    {

        public void OnException(ExceptionContext context)

        {

            var exception = context.Exception;

            File.AppendAllText("logs.txt", $"{DateTime.Now}: {exception.Message}{Environment.NewLine}");

            context.Result = new ObjectResult("An error occurred. Please contact support.")

            {

                StatusCode = StatusCodes.Status500InternalServerError

            };

            context.ExceptionHandled = true;

        }

    }

}

**Models/Department.cs**

namespace WebApiDemo.Models

{

    public class Department

    {

        public int Id { get; set; }

        public string DeptName { get; set; }

    }

}

**Models/Employee.cs**

using System;

using System.Collections.Generic;

namespace WebApiDemo.Models

{

    public class Employee

    {

        public int Id { get; set; }

        public string Name { get; set; }

        public int Salary { get; set; }

        public bool Permanent { get; set; }

        public Department Department { get; set; }

        public List<Skill> Skills { get; set; }

        public DateTime DateOfBirth { get; set; }

    }

}

**Models/Skill.cs**

namespace WebApiDemo.Models

{

    public class Skill

    {

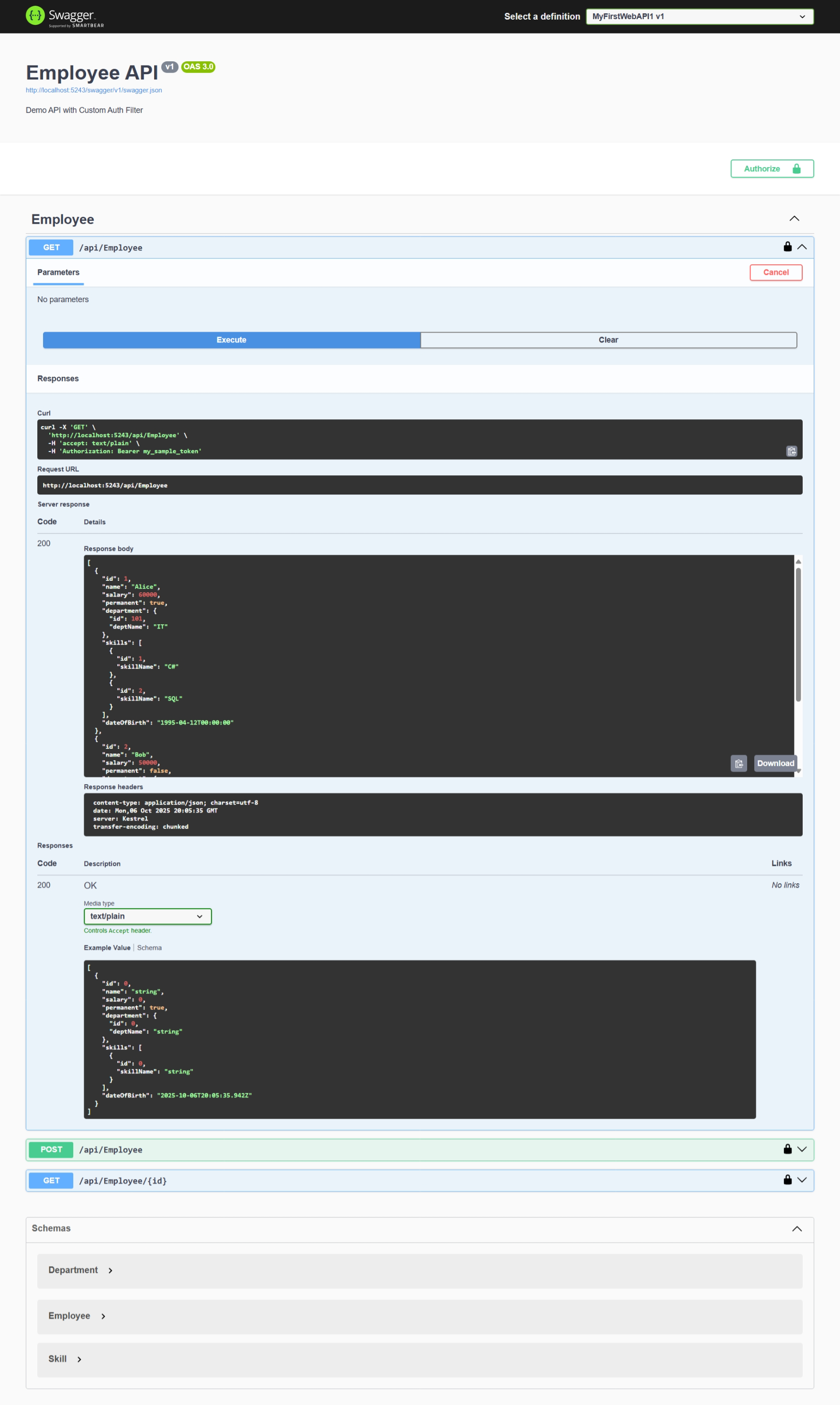
        public int Id { get; set; }

        public string SkillName { get; set; }

    }

}

### **Output:**



## **Web API CRUD Operations using .NET Core (Create, Update, Delete) with Swagger & Postman**

### **Code:**

**Models/Employee.cs**

namespace MyFirstWebAPI.Models

{

    public class Employee

    {

        public int Id { get; set; }

        public string Name { get; set; }

        public int Salary { get; set; }

        public bool Permanent { get; set; }

    }

}

**Controller/EmployeeController.cs**

using Microsoft.AspNetCore.Mvc;

using MyFirstWebAPI.Models;

using System.Collections.Generic;

using System.Linq;

namespace MyFirstWebAPI.Controllers

{

    [Route("api/[controller]")]

    [ApiController]

    public class EmployeeController : ControllerBase

    {

        private static List<Employee> employees = new List<Employee>()

        {

            new Employee { Id = 1, Name = "Priya", Salary = 60000, Permanent = true },

            new Employee { Id = 2, Name = "Rahul", Salary = 50000, Permanent = false },

            new Employee { Id = 3, Name = "Sneha", Salary = 55000, Permanent = true }

        };

        [HttpGet]

        [ProducesResponseType(StatusCodes.Status200OK)]

        public ActionResult<IEnumerable<Employee>> GetAll()

        {

            return Ok(employees);

        }

        [HttpPost]

        [ProducesResponseType(StatusCodes.Status201Created)]

        public ActionResult<Employee> Create([FromBody] Employee emp)

        {

            emp.Id = employees.Max(e => e.Id) + 1;

            employees.Add(emp);

            return CreatedAtAction(nameof(GetAll), new { id = emp.Id }, emp);

        }

        [HttpPut("{id}")]

        [ProducesResponseType(StatusCodes.Status200OK)]

        [ProducesResponseType(StatusCodes.Status400BadRequest)]

        public ActionResult<Employee> Update(int id, [FromBody] Employee emp)

        {

            if (id <= 0)

                return BadRequest("Invalid employee id");

            var existing = employees.FirstOrDefault(e => e.Id == id);

            if (existing == null)

                return BadRequest("Invalid employee id");

            existing.Name = emp.Name;

            existing.Salary = emp.Salary;

            existing.Permanent = emp.Permanent;

            return Ok(existing);

        }

        [HttpDelete("{id}")]

        [ProducesResponseType(StatusCodes.Status200OK)]

        [ProducesResponseType(StatusCodes.Status400BadRequest)]

        public ActionResult Delete(int id)

        {

            if (id <= 0)

                return BadRequest("Invalid employee id");

            var emp = employees.FirstOrDefault(e => e.Id == id);

            if (emp == null)

                return BadRequest("Invalid employee id");

            employees.Remove(emp);

            return Ok($"Employee with ID {id} deleted successfully");

        }

    }

}

**Program.cs**

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddAuthorization();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

    c.SwaggerDoc("v1", new OpenApiInfo

    {

        Title = "Employee API",

        Version = "v1",

        Description = "CRUD API with Swagger",

        Contact = new OpenApiContact

        {

            Name = "John Doe",

            Email = "john@xyzmail.com",

            Url = new Uri("https://www.example.com")

        }

    });

});

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

    app.UseSwagger();

    app.UseSwaggerUI(c =>

    {

        c.SwaggerEndpoint("/swagger/v1/swagger.json", "Employee API v1");

        c.RoutePrefix = string.Empty;

    });

}

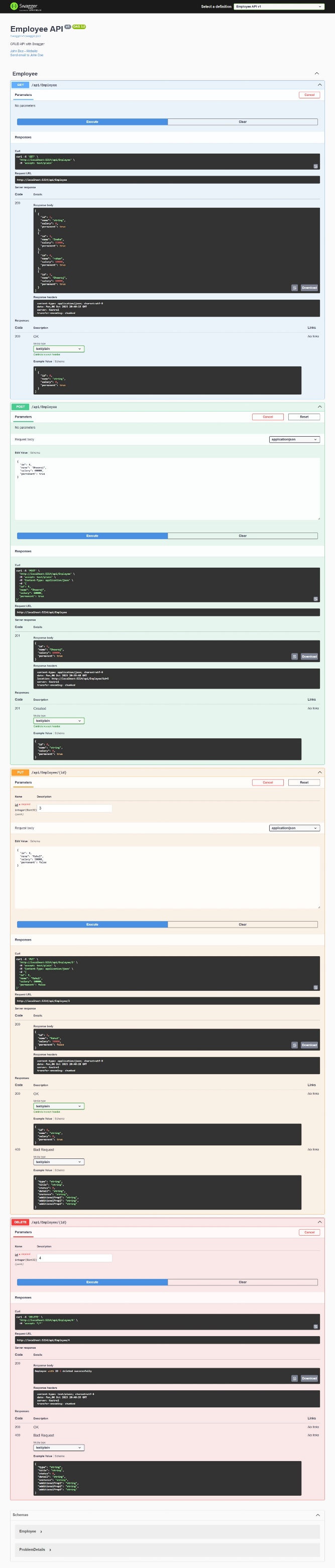
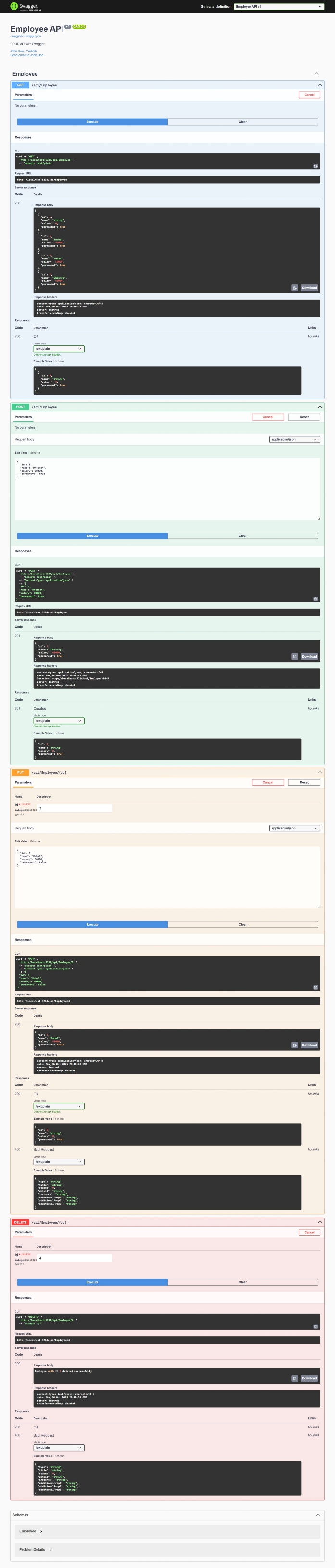
app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

### **Output:**



## **Implementing CORS + JWT (Bearer) Authentication in ASP.NET Core Web API**

### **Code:**

**AuthController.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

using MyFirstWebAPI4.Models;

using Microsoft.AspNetCore.Authorization;

namespace MyFirstWebAPI4.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    public class AuthController : ControllerBase

    {

        private readonly IConfiguration \_configuration;

        public AuthController(IConfiguration configuration)

        {

            \_configuration = configuration;

        }

        [HttpPost("login")]

        public IActionResult Login([FromBody] LoginModel login)

        {

            if (login == null || string.IsNullOrEmpty(login.Username) || string.IsNullOrEmpty(login.Password))

                return BadRequest("Invalid credentials");

            if (login.Username == "admin" && login.Password == "password")

            {

                var jwtKey = \_configuration["Jwt:Key"];

                if (string.IsNullOrEmpty(jwtKey))

                    return StatusCode(500, "JWT key missing in configuration");

                var key = Encoding.UTF8.GetBytes(jwtKey);

                var tokenHandler = new JwtSecurityTokenHandler();

                var tokenDescriptor = new SecurityTokenDescriptor

                {

                    Subject = new ClaimsIdentity(new[]

                    {

                        new Claim(ClaimTypes.Name, login.Username),

                        new Claim(ClaimTypes.Role, "Admin")

                    }),

                    Expires = DateTime.UtcNow.AddHours(1),

                    Issuer = \_configuration["Jwt:Issuer"],

                    Audience = \_configuration["Jwt:Audience"],

                    SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature)

                };

                var token = tokenHandler.CreateToken(tokenDescriptor);

                return Ok(new { token = tokenHandler.WriteToken(token) });

            }

            return Unauthorized("Username or password is incorrect");

        }

        [HttpGet("test")]

        [Microsoft.AspNetCore.Authorization.Authorize]

        public IActionResult TestEndpoint()

        {

            return Ok("You are successfully authorized!");

        }

    }

}

**Startup.cs**

using Microsoft.IdentityModel.Tokens;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using System.Text;

using Microsoft.OpenApi.Models;

namespace MyFirstWebAPI4

{

    public class Startup

    {

        public IConfiguration Configuration { get; }

        public Startup(IConfiguration configuration)

        {

            Configuration = configuration;

        }

        public void ConfigureServices(IServiceCollection services)

        {

            services.AddControllers();

            services.AddEndpointsApiExplorer();

                services.AddSwaggerGen(c =>

    {

        c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

        {

            Name = "Authorization",

            Type = SecuritySchemeType.Http,

            Scheme = "Bearer",

            BearerFormat = "JWT",

            In = ParameterLocation.Header,

            Description = "Enter 'Bearer' [space] and then your valid token.\nExample: Bearer abc123def456"

        });

        c.AddSecurityRequirement(new OpenApiSecurityRequirement

        {

            {

                new OpenApiSecurityScheme

                {

                    Reference = new OpenApiReference { Type = ReferenceType.SecurityScheme, Id = "Bearer" }

                },

                new string[] { }

            }

        });

    });

            services.AddCors(options =>

            {

                options.AddPolicy("AllowLocalhost",

                    builder =>

                    {

                        builder.AllowAnyOrigin()

                               .AllowAnyHeader()

                               .AllowAnyMethod();

                    });

            });

            var jwtKey = Configuration["Jwt:Key"];

            var key = Encoding.UTF8.GetBytes(jwtKey ?? throw new InvalidOperationException("JWT key not found in configuration"));

            services.AddAuthentication(x =>

            {

                x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

                x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

                x.DefaultSignInScheme = JwtBearerDefaults.AuthenticationScheme;

            })

            .AddJwtBearer(JwtBearerDefaults.AuthenticationScheme, x =>

            {

                x.TokenValidationParameters = new TokenValidationParameters

                {

                    ValidateIssuer = true,

                    ValidateAudience = true,

                    ValidateLifetime = true,

                    ValidateIssuerSigningKey = true,

                    ValidIssuer = Configuration["Jwt:Issuer"],

                    ValidAudience = Configuration["Jwt:Audience"],

                    IssuerSigningKey = new SymmetricSecurityKey(key)

                };

            });

        }

        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

        {

            if (env.IsDevelopment())

            {

                app.UseDeveloperExceptionPage();

                app.UseSwagger();

                app.UseSwaggerUI(c =>

                {

                    c.SwaggerEndpoint("/swagger/v1/swagger.json", "MyFirstWebAPI4 V1");

                });

            }

            app.UseHttpsRedirection();

            app.UseRouting();

            app.UseCors("AllowLocalhost");

            app.UseAuthentication();

            app.UseAuthorization();

            app.UseEndpoints(endpoints =>

            {

                endpoints.MapControllers();

            });

        }

    }

}

### **Output:**

