DotNet-FSE Mandatory Hands-On

WEEK-4 **NAME:** Sri Ranjani Priya P

EXERCISE 1:

Demonstrate creation of a simple WebAPI – With Read, Write actions

CODE:(C#)
File: ValuesController.cs

```
using Microsoft.AspNetCore.Mvc;
using MyFirstApi1;
using System.Collections.Generic;
using System.Linq;

namespace MyFirstApi1.Controllers
{
   [Route("api/[controller]")]
   [ApiController]
   public class ValuesController : ControllerBase
   {
      static List<Product> products = new List<Product>
      {
         new Product { Id = 1, Name = "Book" },
         new Product { Id = 2, Name = "Laptop" },
         new Product { Id = 3, Name = "Keyboard" }
      };


      [HttpGet]
      public ActionResult<IEnumerable<Product>> Get()
      {
         return products;
      }


      [HttpGet("{id}")]
      public ActionResult<Product> Get(int id)
      {
         var product = products.FirstOrDefault(p => p.Id == id);
         if (product == null)
            return NotFound();
         return product;
      }
```

```
        [HttpPost]
    public ActionResult Post([FromBody] Product newProduct)
    {
        products.Add(newProduct);
        return Ok();
    }


    [HttpPut("{id}")]
    public ActionResult Put(int id, [FromBody] Product updatedProduct)
    {
        var product = products.FirstOrDefault(p => p.Id == id);
        if (product == null)
            return NotFound();

        product.Name = updatedProduct.Name;
        return Ok();
    }

    [HttpDelete("{id}")]
    public ActionResult Delete(int id)
    {
        var product = products.FirstOrDefault(p => p.Id == id);
        if (product == null)
            return NotFound();

        products.Remove(product);
        return Ok();
    }
  }
}
```

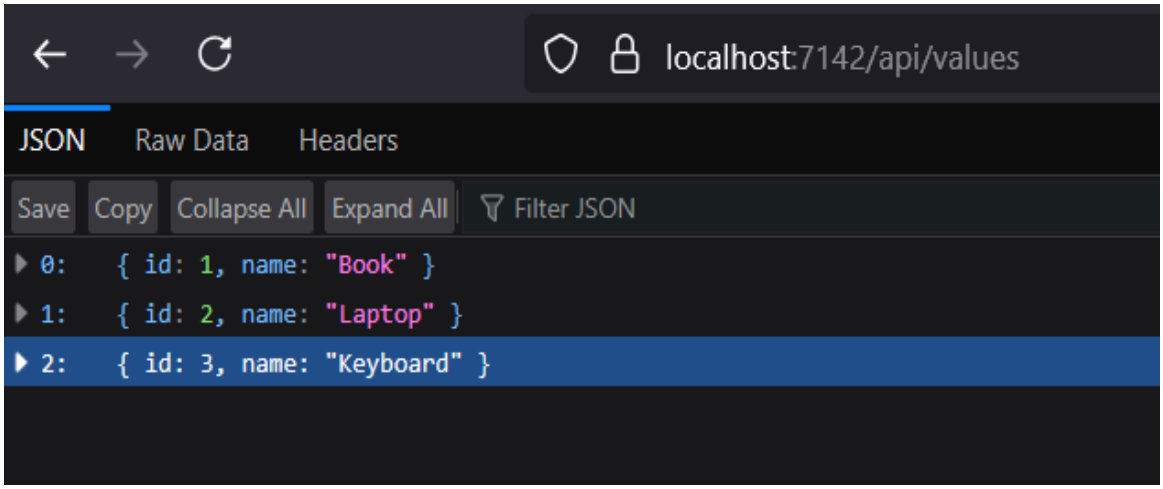
File: product.cs

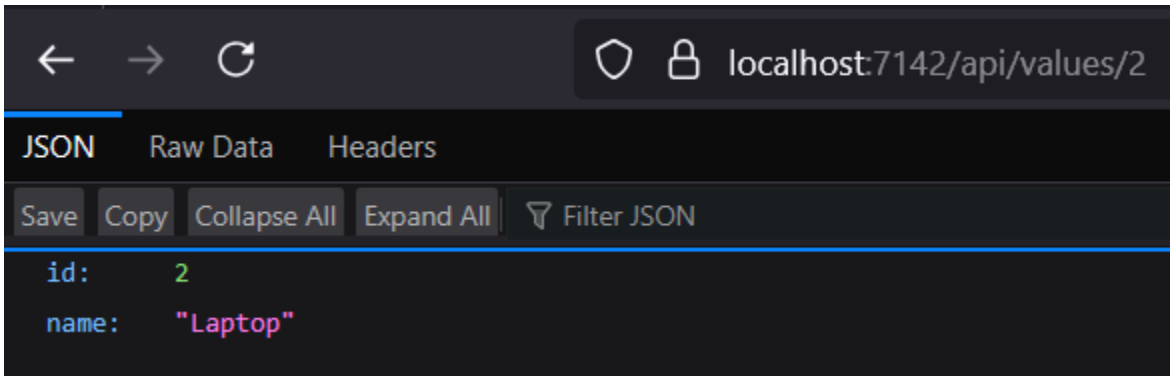
```
namespace MyFirstApi1
{
  public class Product
  {
    public int Id { get; set; }
    public string Name { get; set; }
  }

}
```

OUTPUT:

After executing the application - GET /api/values

```
1>MyFirstApi1 -> C:\Users\SEC\source\repos\MyFirstApi1\bin\Debug\net8.0\MyFirstApi1.dll
1>Done building project "MyFirstApi1.csproj".
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
========== Rebuild completed at 20:29 and took 01.702 seconds ==========
```





EXERCISE 2:

WebAPI with Swagger Integration and CRUD Operations

CODE:(C#)
File:Program.cs

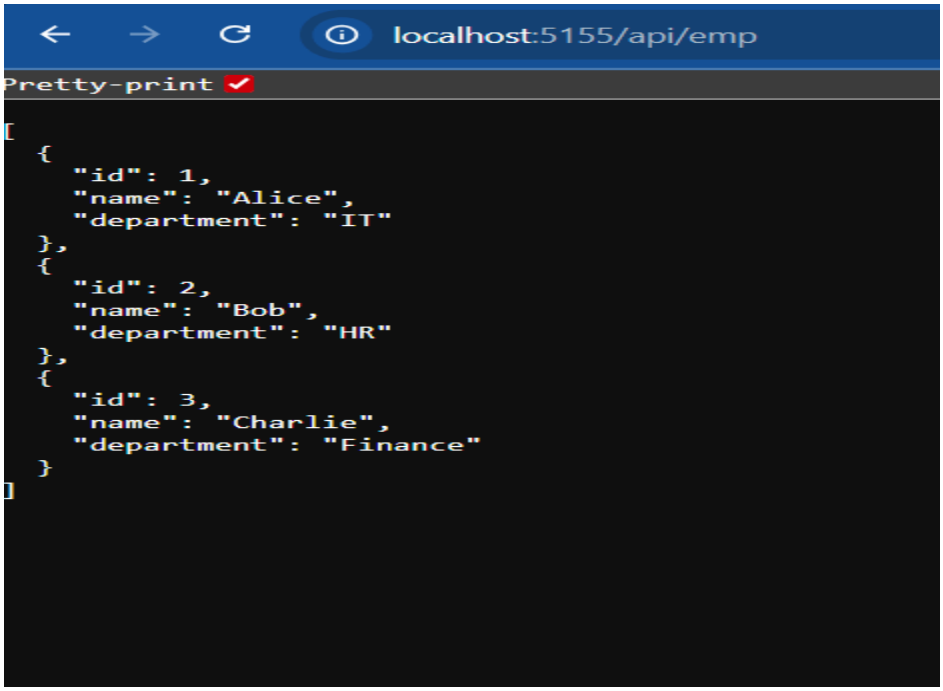
```
using Microsoft.AspNetCore.Mvc;
using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);


builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen(options =>
{
    options.SwaggerDoc("v1", new OpenApiInfo
    {
        Title = "Employee API",
        Version = "v1",
        Description = "API for managing employees",
        Contact = new OpenApiContact
        {
            Name = "Sriranjani",
            Email = "sriranjani2004@gmail.com",
            Url = new Uri("https://www.example.com")
        }
    });
});

var app = builder.Build();


app.UseSwagger();
app.UseSwaggerUI(options =>
{
    options.SwaggerEndpoint("/swagger/v1/swagger.json", "Employee API");
});


List<Employee> employees = new()
{
    new Employee { Id = 1, Name = "Alice", Department = "IT" },
    new Employee { Id = 2, Name = "Bob", Department = "HR" }
```

```
};


app.MapGet("/api/emp", () => employees)
  .WithTags("Employees");


app.MapGet("/api/emp/{id}", (int id) =>
{
   var emp = employees.FirstOrDefault(e => e.Id == id);
   return emp is not null ? Results.Ok(emp) : Results.NotFound();
})
.WithTags("Employees");


app.MapPost("/api/emp", ([FromBody] Employee emp) =>
{
   emp.Id = employees.Max(e => e.Id) + 1;
   employees.Add(emp);
   return Results.Created($"/api/emp/{emp.Id}", emp);
})
.WithTags("Employees");


app.MapPut("/api/emp/{id}", (int id, [FromBody] Employee updatedEmp) =>
{
   var emp = employees.FirstOrDefault(e => e.Id == id);
   if (emp is null) return Results.NotFound();
   emp.Name = updatedEmp.Name;
   emp.Department = updatedEmp.Department;
   return Results.Ok(emp);
})
.WithTags("Employees");


app.MapDelete("/api/emp/{id}", (int id) =>
{
   var emp = employees.FirstOrDefault(e => e.Id == id);
   if (emp is null) return Results.NotFound();
   employees.Remove(emp);
   return Results.Ok(emp);
})
.WithTags("Employees");
```

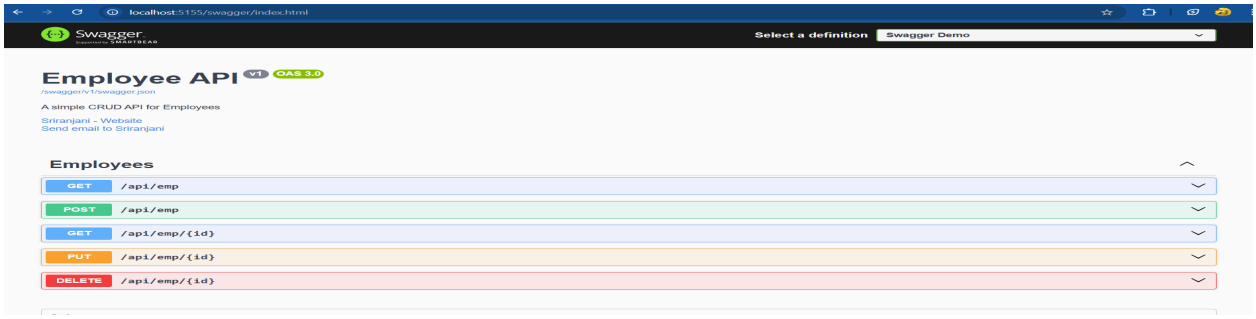
```
app.Run();


public class Employee
{
   public int Id { get; set; }
   public string Name { get; set; } = string.Empty;
   public string Department { get; set; } = string.Empty;}
```

OUTPUT:

URL: GET http://localhost:5155/api/emp



URL: http://localhost:5155/swagger



URL: http://localhost:5155/swagger-> POST New Employee

Curl

```
curl -X 'POST' \
  'http://localhost:5155/api/emp' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "id": 4,
  "name": "Bobby",
  "department": "Analyst"
}'
```

Request URL

```
http://localhost:5155/api/emp
```

Server response

Code	Details
201	Response body

```
{
  "id": 4,
  "name": "Bobby",
  "department": "Analyst"
}
```

Response headers

```
content-type: application/json; charset=utf-8
date: Sat,12 Jul 2025 16:01:49 GMT
location: /api/emp/4
server: Kestrel
transfer-encoding: chunked
```

After Posting New Employee

localhost:5155/api/emp

Pretty-print


```
[
  {
    "id": 1,
    "name": "Alice",
    "department": "IT"
  },
  {
    "id": 2,
    "name": "Bob",
    "department": "HR"
  },
  {
    "id": 3,
    "name": "Charlie",
    "department": "Finance"
  },
  {
    "id": 4,
    "name": "Bobby",
    "department": "Analyst"
  }
]
```

EXERCISE 3.A:

Web API using Custom Model Class

CODE:
Models/Department.cs

```
namespace SwaggerDemoAPI.Models
{
   public class Department
   {
      public int Id { get; set; }
      public string Name { get; set; }
   }
}
```

Models/Skill.cs

```
namespace SwaggerDemoAPI.Models
{
   public class Skill
   {
      public int Id { get; set; }
      public string Name { get; set; }
   }
}
```

Models/Employee.cs

```
namespace SwaggerDemoAPI.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; }
   }
}
```

Controllers/EmployeeController.cs

```
using Microsoft.AspNetCore.Mvc;
using SwaggerDemoAPI.Models;
using SwaggerDemoAPI.Filters;

[Route("api/[controller]")]
[ApiController]
[ServiceFilter(typeof(CustomAuthFilter))]
public class EmployeeController : ControllerBase
{
    private static List<Employee> _employees = new List<Employee>
    {
        new Employee
        {
            Id = 1,
            Name = "John",
            Salary = 50000,
            Permanent = true,
            Department = new Department { Id = 101, Name = "IT" },
            Skills = new List<Skill>
            {
                new Skill { Id = 1, Name = "C#" },
                new Skill { Id = 2, Name = "SQL" }
            },
            DateOfBirth = new DateTime(1990, 5, 24)
        }
    };

    [HttpGet("standard")]
    [ProducesResponseType(typeof(List<Employee>), 200)]
    public ActionResult<List<Employee>> GetStandard()
    {
        return Ok(_employees);
    }

    [HttpGet]
    [ProducesResponseType(500)]
    public IActionResult Get()
    {
        throw new Exception("Test exception");
    }
}
```

OUTPUT:
URL: https://localhost:7042/api/Employee/standard

localhost:7042/swagger/index.html

```
curl -X 'GET' \
  'https://localhost:7042/api/Employee/standard' \
  -H 'accept: text/plain' \
  -H 'Authorization: Bearer dummy-token'
```

Request URL

```
https://localhost:7042/api/Employee/standard
```

Server response

Code	Details
200	Response body


```
{
  "id": 1,
  "name": "John",
  "salary": 50000,
  "permanent": true,
  "department": {
    "id": 101,
    "name": "IT"
  },
  "skills": [
    {
      "id": 1,
      "name": "C#"
    },
    {
      "id": 2,
      "name": "SQL"
    }
  ],
  "dateOfBirth": "1990-05-24T00:00:00"
}
```


Download

Response headers

```
content-type: application/json; charset=utf-8
date: Sun,13 Jul 2025 13:43:31 GMT
server: Kestrel
```

Schemas

Department >

Employee >

Skill >

EXERCISE 3B:

Custom Authorization Filter

CODE:
Filters/CustomAuthFilter.cs

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Filters;

public class CustomAuthFilter : ActionFilterAttribute
{
    public override void OnActionExecuting(ActionExecutingContext context)
    {
        var hasHeader = context.HttpContext.Request.Headers.TryGetValue("Authorization", out
var token);

        if (!hasHeader)
        {
            context.Result = new BadRequestObjectResult("Invalid request - No Auth token");
            return;
        }

        if (!token.ToString().Contains("Bearer"))
        {
            context.Result = new BadRequestObjectResult("Invalid request - Token present but
Bearer unavailable");
        }
    }
}
```

OUTPUT:
Case 1: No Authorization Header

Case 2: Header Present, No 'Bearer'

Code	Details

400
Undocumented
	Error: response status is 400
	Response body
	`Invalid request - Token present but Bearer unavailable`
	Download
	Response headers
	`content-type: text/plain; charset=utf-8`
	`date: Sun,13 Jul 2025 13:47:19 GMT`
	`server: Kestrel`

Case 3: Valid Token Provided

localhost:7042/swagger/index.html

```
curl -X 'GET' \
  'https://localhost:7042/api/Employee/standard' \
  -H 'accept: text/plain' \
  -H 'Authorization: Bearer dummy-token'
```

Request URL

```
https://localhost:7042/api/Employee/standard
```

Server response

Code	Details
200	**Response body**


```
{
  "id": 1,
  "name": "John",
  "salary": 50000,
  "permanent": true,
  "department": {
    "id": 101,
    "name": "IT"
  },
  "skills": [
    {
      "id": 1,
      "name": "C#"
    },
    {
      "id": 2,
      "name": "SQL"
    }
  ],
  "dateOfBirth": "1990-05-24T00:00:00"
}
```


Download

Response headers

```
content-type: application/json; charset=utf-8
date: Sun,13 Jul 2025 13:43:31 GMT
server: Kestrel
```

EXERCISE 3C:

Custom Exception Filter

CODE:
Filters/CustomExceptionFilter.cs

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Filters;

public class CustomExceptionFilter : IExceptionFilter
{
    public void OnException(ExceptionContext context)
    {
        File.WriteAllText("exception_log.txt", context.Exception.ToString());

        context.Result = new ObjectResult("Internal Server Error occurred")
        {
            StatusCode = 500
        };
    }
}
```

Program.cs

```
using SwaggerDemoAPI.Filters;
using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers(options =>
{
    options.Filters.Add<CustomExceptionFilter>();});
builder.Services.AddScoped<CustomAuthFilter>();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new OpenApiInfo { Title = "SwaggerDemoAPI", Version = "v1" });

    c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme
    {
        In = ParameterLocation.Header,
        Description = "Enter 'Bearer' followed by space and token",
        Name = "Authorization",
        Type = SecuritySchemeType.ApiKey,
        Scheme = "Bearer"
```

```
    });

    c.AddSecurityRequirement(new OpenApiSecurityRequirement
    {
        {
            new OpenApiSecurityScheme
            {
                Reference = new OpenApiReference
                {
                    Type = ReferenceType.SecurityScheme,
                    Id = "Bearer"
                }
            },
            new string[] { }
        }
    });
});

var app = builder.Build();

app.UseHttpsRedirection();
app.UseSwagger();
app.UseSwaggerUI();
app.UseAuthorization();
app.MapControllers();
app.Run();
```

OUTPUT:

EXERCISE 4:

Web API PUT Operation Using FromBody and Validation

CODE:

FILE: Models/Employee.cs

```
namespace MyFirstApi1.Models
{
   public class Employee
   {
      public int Id { get; set; }
      public string? Name { get; set; }
      public string? Department { get; set; }
      public double Salary { get; set; }
   }
}
```

Controller – Controllers/EmployeeController.cs

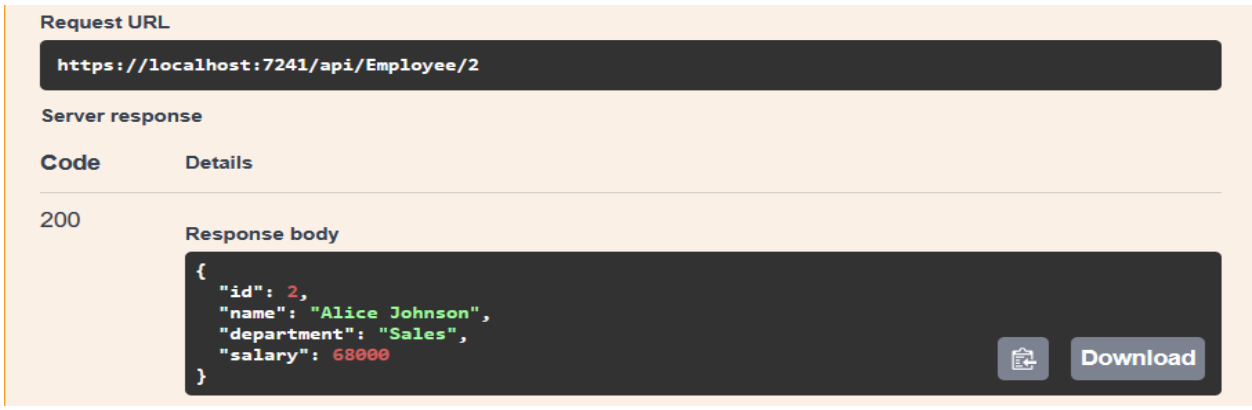
```
using Microsoft.AspNetCore.Mvc;
using MyFirstApi1.Models;
using System.Collections.Generic;
using System.Linq;

namespace MyFirstApi1.Controllers
{
   [Route("api/[controller]")]
   [ApiController]
   public class EmployeeController : ControllerBase
   {
      static List<Employee> employees = new List<Employee>
      {
         new Employee { Id = 1, Name = "John", Department = "HR", Salary = 45000 },
         new Employee { Id = 2, Name = "Alice", Department = "IT", Salary = 55000 },
         new Employee { Id = 3, Name = "Bob", Department = "Finance", Salary = 60000 }
      };

      [HttpPut("{id}")]
      public ActionResult<Employee> UpdateEmployee(int id, [FromBody] Employee
updatedEmployee)
      {
         if (id <= 0)
            return BadRequest("Invalid employee id");
```

```
            var employee = employees.FirstOrDefault(e => e.Id == id);
            if (employee == null)
                return BadRequest("Invalid employee id");

            if (updatedEmployee.Id != 0 && updatedEmployee.Id != id)
                return BadRequest("ID in body doesn't match ID in URL");

            employee.Name = updatedEmployee.Name;
            employee.Department = updatedEmployee.Department;
            employee.Salary = updatedEmployee.Salary;

            return Ok(employee);
        }
    }
}
```

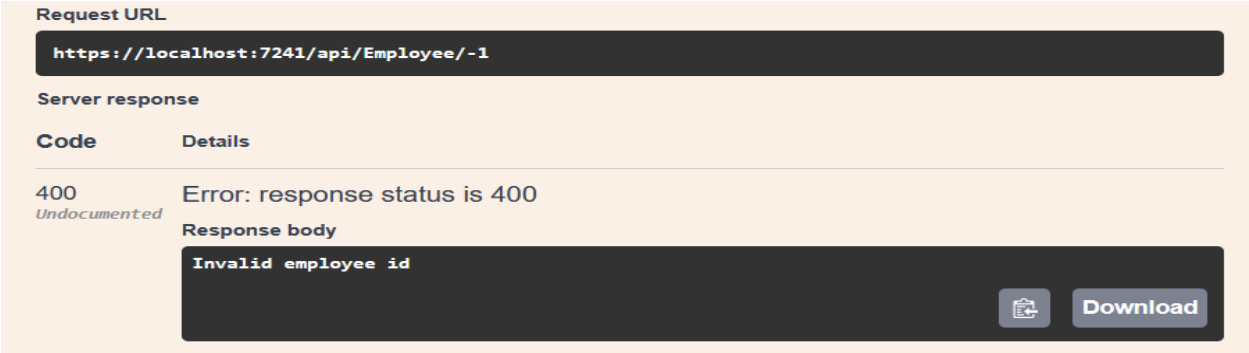
Program.cs (Default)

```
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();
app.UseSwagger();
app.UseSwaggerUI();
app.UseAuthorization();
app.MapControllers();
app.Run();
```

OUTPUT:



Test Case : ID = -1



EXERCISE 5:

Enabling CORS and Implementing Security in ASP.NET Core Web API using JWT

CODE:
Configure JWT Authentication in Program.cs

```
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.IdentityModel.Tokens;
using System.Text;

var builder = WebApplication.CreateBuilder(args);


var securityKey = "mysuperdupersecretkey1234567890!!";
var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(securityKey));

builder.Services.AddAuthentication(options =>
{
    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
})
.AddJwtBearer(options =>
{
    options.TokenValidationParameters = new TokenValidationParameters
    {
        ValidateIssuer = true,
        ValidateAudience = true,
        ValidateLifetime = true,
        ValidateIssuerSigningKey = true,
        ValidIssuer = "mySystem",
        ValidAudience = "myUsers",
        IssuerSigningKey = key
```

```
    };
});

builder.Services.AddCors(options =>
{
    options.AddPolicy("AllowAll", policy =>
        policy.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader());
});

builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

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

app.UseCors("AllowAll");
app.UseAuthentication();
app.UseAuthorization();

app.MapControllers();
app.Run();
```

AuthController1.cs — Token Generator:

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;

namespace SecureEmployeeApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class AuthController1 : ControllerBase
    {
        private readonly Dictionary<string, (string Password, string Role)> _users =
            new()
            {
```

```
            { "admin", ("12345", "Admin") },
            { "pocuser", ("12345", "POC") },
            { "normaluser", ("12345", "User") }
        };

    [HttpPost("Login")]
    [AllowAnonymous]
    public IActionResult Login([FromBody] LoginModel login)
    {
        if (_users.ContainsKey(login.Username) && _users[login.Username].Password ==
login.Password)
        {
            var role = _users[login.Username].Role;
            var token = GenerateJSONWebToken(login.Username, role);
            return Ok(new { token });
        }

        return Unauthorized("Invalid username or password");
    }

    private string GenerateJSONWebToken(string username, string userRole)
    {
        var securityKey = new SymmetricSecurityKey(
            Encoding.UTF8.GetBytes("mysuperdupersecretkey1234567890!!"));

        var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);

        var claims = new[]
        {
            new Claim(ClaimTypes.Name, username),
            new Claim(ClaimTypes.Role, userRole)
        };

        var token = new JwtSecurityToken(
            issuer: "mySystem",
            audience: "myUsers",
            claims: claims,
            expires: DateTime.Now.AddMinutes(2),
            signingCredentials: credentials);

        return new JwtSecurityTokenHandler().WriteToken(token);
    }

    public class LoginModel
    {
        public string Username { get; set; }
```

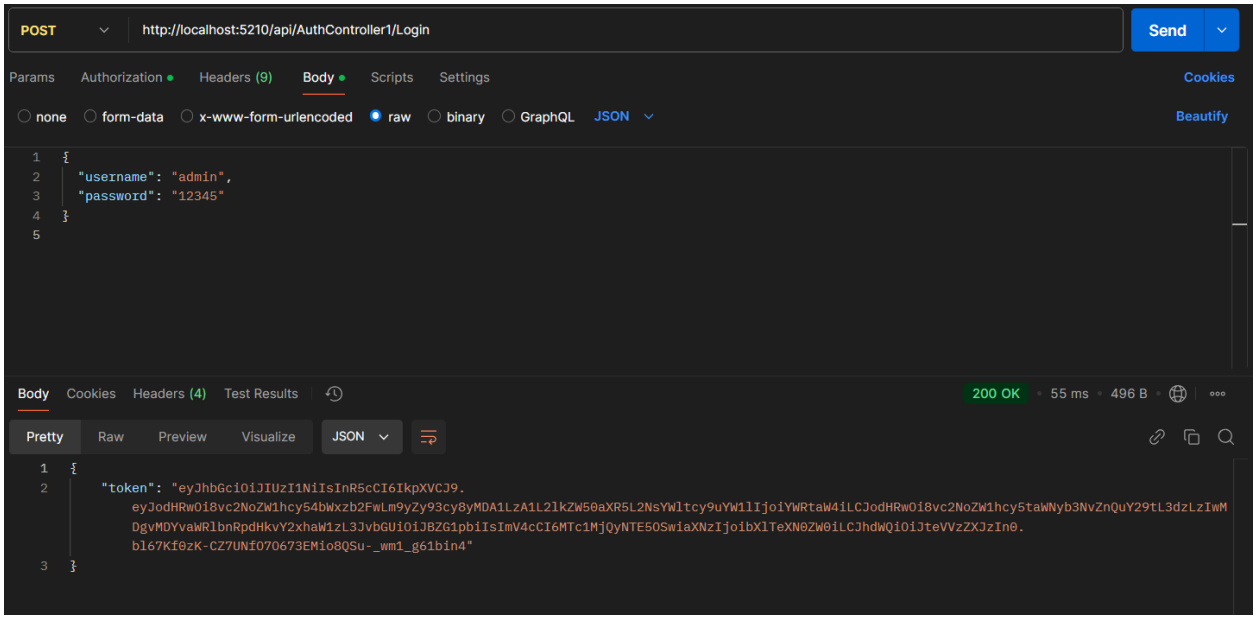
```
            public string Password { get; set; }
        }
    }
}
```

EmployeeController1.cs — Protected API:

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace SecureEmployeeApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    [Authorize(Roles = "Admin,POC")]
    public class EmployeeController1 : ControllerBase
    {
        [HttpGet]
        public IActionResult Get()
        {
            return Ok("You are authorized to view employee data (Controller1).");
        }
    }
}
```

OUTPUT:

POST http://localhost:5210/api/AuthController1/Login



Access Protected Data – GET
URL: GET http://localhost:5210/api/EmployeeController1