1. **First Web Api using .Net core**

ValueControllers.cs:

using Microsoft.AspNetCore.Mvc;

namespace MyFirstWebApi.Controllers

{

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

[ApiController]

public class ValuesController : ControllerBase

{

// GET: api/<ValuesController>

[HttpGet]

public IEnumerable<string> Get()

{

return new string[] { "value1", "value2" };

}

// GET api/<ValuesController>/5

[HttpGet("{id}")]

public string Get(int id)

{

return "value";

}

// POST api/<ValuesController>

[HttpPost]

public void Post([FromBody] string value)

{

}

// PUT api/<ValuesController>/5

[HttpPut("{id}")]

public void Put(int id, [FromBody] string value)

{

}

// DELETE api/<ValuesController>/5

[HttpDelete("{id}")]

public void Delete(int id)

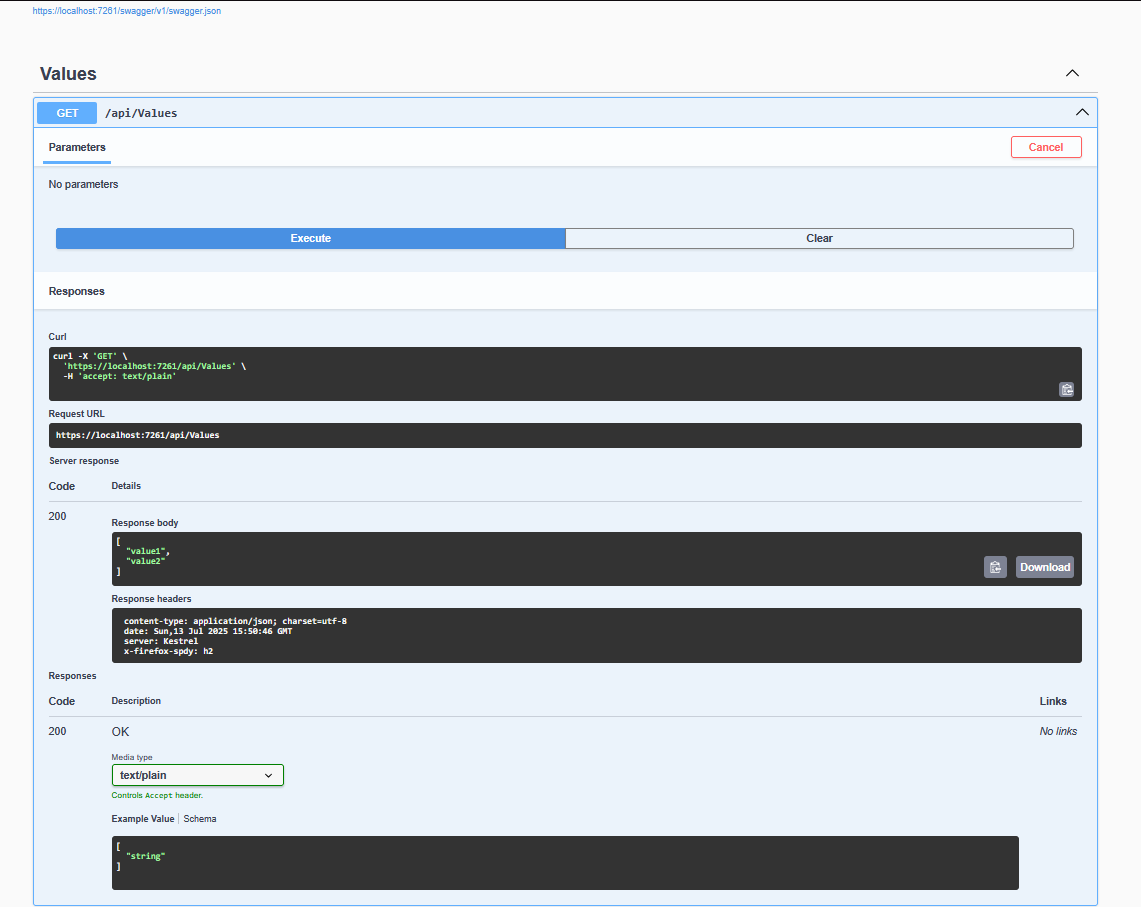
{

}

}

}

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

**2.Web Api using .Net core with Swagger**

EmployeeController.cs:

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Http;

using System.Collections.Generic;

namespace Lab1\_FirstWebApi.Controllers

{

[ApiController]

[Route("api/emp")]

public class EmployeeController : ControllerBase

{

[HttpGet]

[ActionName("GetAll")]

[ProducesResponseType(StatusCodes.Status200OK)]

public IEnumerable<string> GetAll()

{

return new string[] { "john", "doe", "Alice" };

}

[HttpGet("byid")]

[ActionName("GetById")]

[ProducesResponseType(StatusCodes.Status200OK)]

public string GetById()

{

return "Deb";

}

}

}

Program.cs:  
  
using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

// Add Swagger with metadata

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "TBD",

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")

}

});

c.ResolveConflictingActions(apiDescriptions => apiDescriptions.First());

});

var app = builder.Build();

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

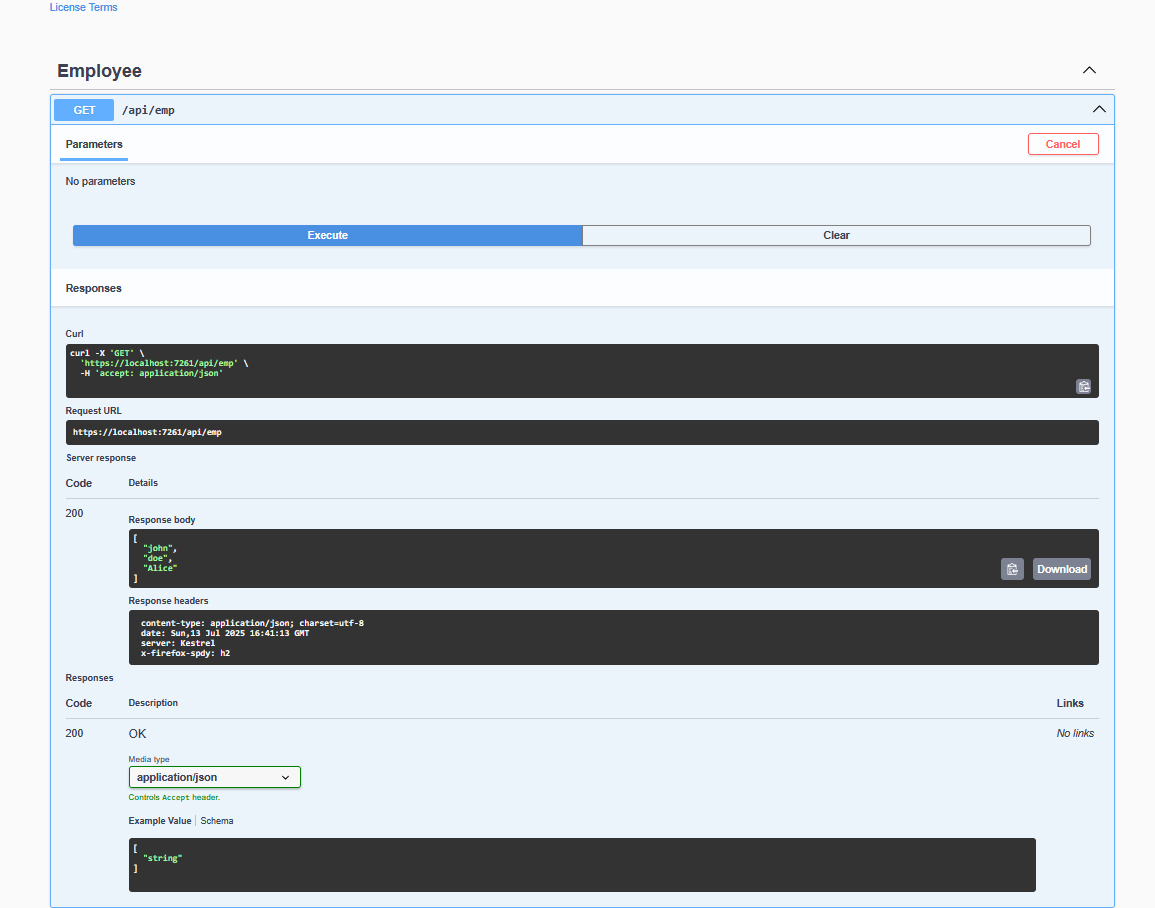
app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

OUTPUT:

**A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.**

**3. Web Api using custom model class, Create a Custom action filter for Authorization and Custom Exception filter.**

Program.cs:

using Microsoft.OpenApi.Models;

using MyFirstWebApi.Filters;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers(options =>

{

options.Filters.Add<CustomExceptionFilter>();

});

// Add Swagger with metadata

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddScoped<CustomAuthFilter>();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "TBD",

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")

}

});

c.ResolveConflictingActions(apiDescriptions => apiDescriptions.First());

});

var app = builder.Build();

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

EmployeeController.cs:

using MyFirstWebApi.Filters;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using MyFirstWebApi.Models;

namespace MyFirstWebApi.Controllers

{

[ApiController]

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

[AllowAnonymous]

[ServiceFilter(typeof(CustomAuthFilter))]

public class EmployeeController : ControllerBase

{

private List<Employee> employeeList;

public EmployeeController()

{

employeeList = GetStandardEmployeeList();

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "Deb",

Salary = 60000,

Permanent = true,

Department = new Department { Id = 1, Name = "IT" },

Skills = new List<Skill>

{

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

new Skill { Id = 2, Name = "ASP.NET Core" }

},

DateOfBirth = new DateTime(1998, 05, 20)

}

};

}

[HttpGet("standard")]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public ActionResult<List<Employee>> GetStandard()

{

// Uncomment this to test exception filter:

// throw new Exception("Forced exception for testing");

return Ok(employeeList);

}

[HttpPost]

public IActionResult Post([FromBody] Employee emp)

{

return Ok($"Employee received: {emp.Name}");

}

}

}

CustomAuthFilter.cs:

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace MyFirstWebApi.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

var hasAuthHeader = context.HttpContext.Request.Headers.TryGetValue("Authorization", out var token);

if (!hasAuthHeader)

{

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");

}

}

}

}

CustomExceptionFilter.cs:

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System;

using System.IO;

namespace MyFirstWebApi.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

string message = $"Exception: {context.Exception.Message}";

var logPath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "error\_log.txt");

File.AppendAllText(logPath, message + Environment.NewLine);

context.Result = new ObjectResult("An unexpected error occurred")

{

StatusCode = 500

};

}

}

}

Employee.cs:

using MyFirstWebApi.Models;

using System;

using System.Collections.Generic;

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; }

public Department Department { get; set; }

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

public DateTime DateOfBirth { get; set; }

}

}

Department.cs:

namespace MyFirstWebApi.Models

{

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

}

}

Skill.cs:

namespace MyFirstWebApi.Models

{

public class Skill

{

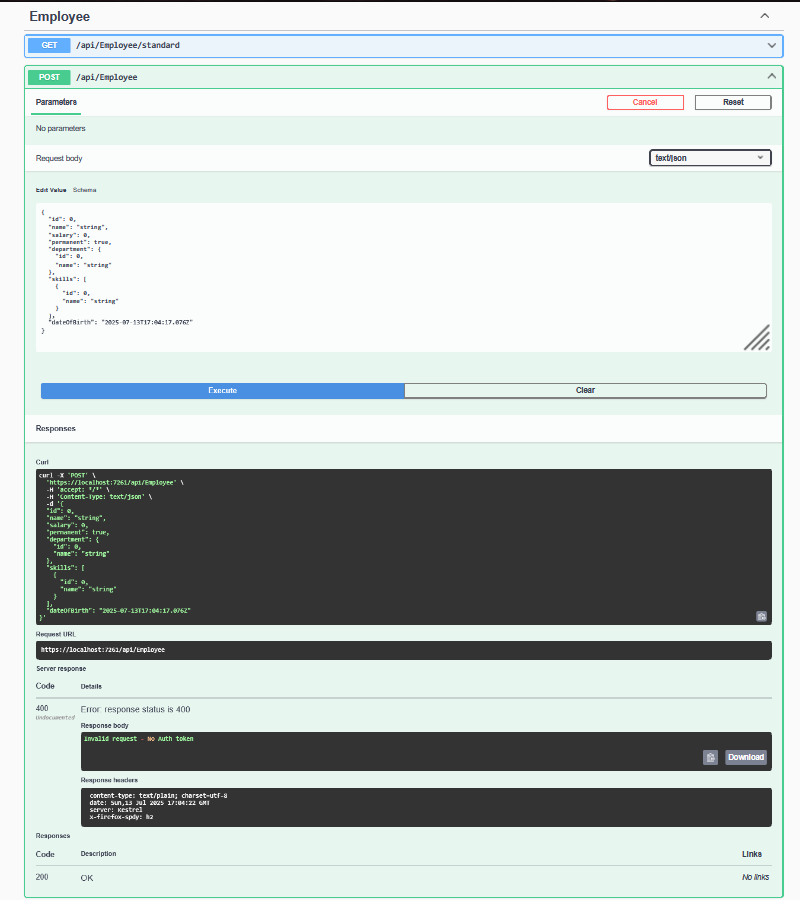
public int Id { get; set; }

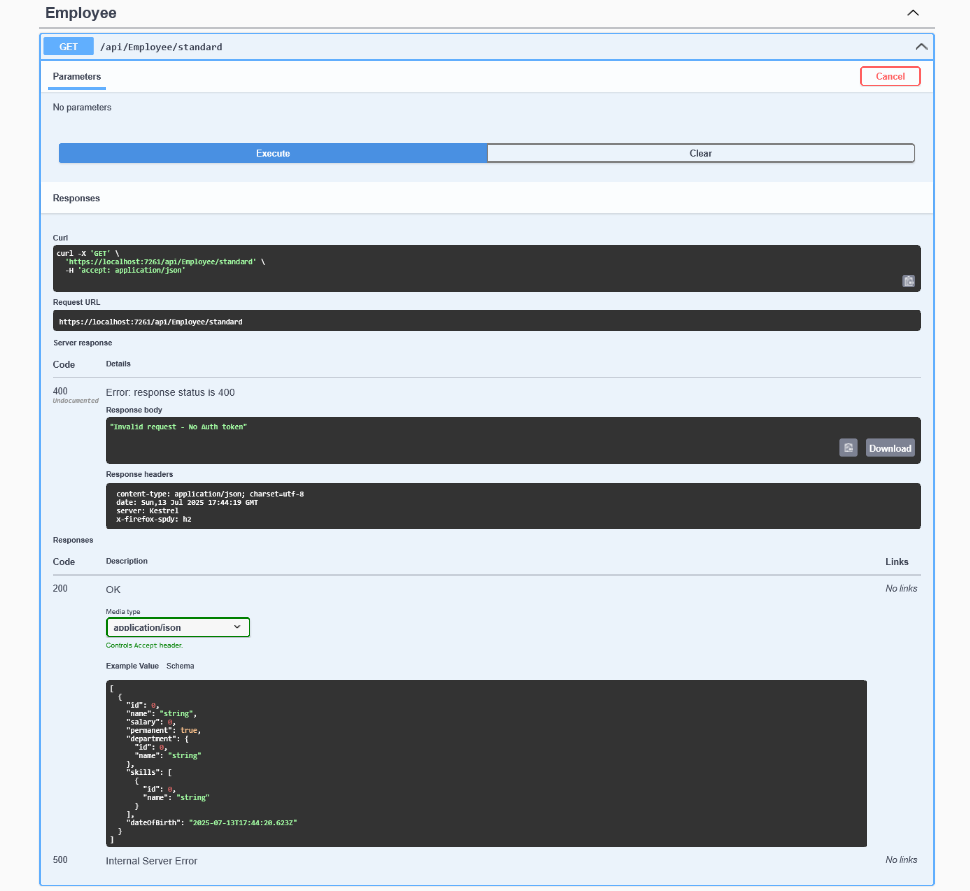
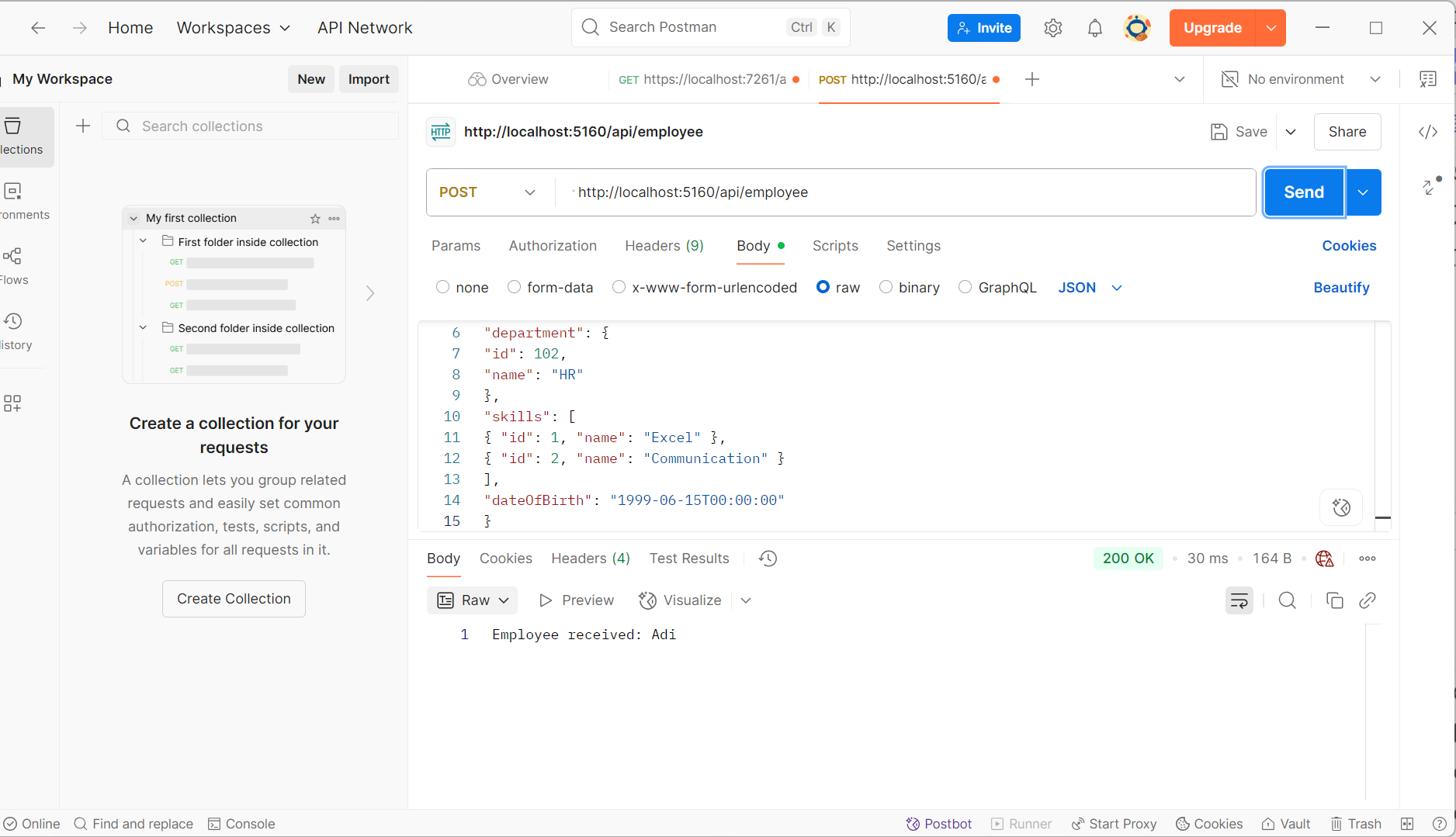
public string Name { get; set; }

}

}

OUTPUT:

**A screenshot of a computer

AI-generated content may be incorrect.**

**4. Web Api CRUD operation**

EmployeeController.cs:

using MyFirstWebApi.Filters;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using MyFirstWebApi.Models;

namespace MyFirstWebApi.Controllers

{

[ApiController]

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

[AllowAnonymous]

[ServiceFilter(typeof(CustomAuthFilter))]

public class EmployeeController : ControllerBase

{

private List<Employee> employeeList;

public EmployeeController()

{

// Fake database: hardcoded employee list

employeeList = new List<Employee>

{

new Employee

{

Id = 1,

Name = "Deb",

Salary = 60000,

Permanent = true,

Department = new Department { Id = 1, Name = "IT" },

Skills = new List<Skill>

{

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

new Skill { Id = 2, Name = "ASP.NET Core" }

},

DateOfBirth = new DateTime(2002, 10, 17)

},

new Employee

{

Id = 2,

Name = "Adi",

Salary = 50000,

Permanent = false,

Department = new Department { Id = 2, Name = "HR" },

Skills = new List<Skill>

{

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

new Skill { Id = 4, Name = "Communication" }

},

DateOfBirth = new DateTime(2001, 8, 5)

}

};

}

[HttpPut("{id}")]

public ActionResult<Employee> Put(int id, [FromBody] Employee updatedEmp)

{

// Validate input

if (id <= 0)

return BadRequest("Invalid employee id");

// Find employee by id

var employee = employeeList.FirstOrDefault(e => e.Id == id);

if (employee == null)

return BadRequest("Invalid employee id");

// Update employee data

employee.Name = updatedEmp.Name;

employee.Salary = updatedEmp.Salary;

employee.Permanent = updatedEmp.Permanent;

employee.Department = updatedEmp.Department;

employee.Skills = updatedEmp.Skills;

employee.DateOfBirth = updatedEmp.DateOfBirth;

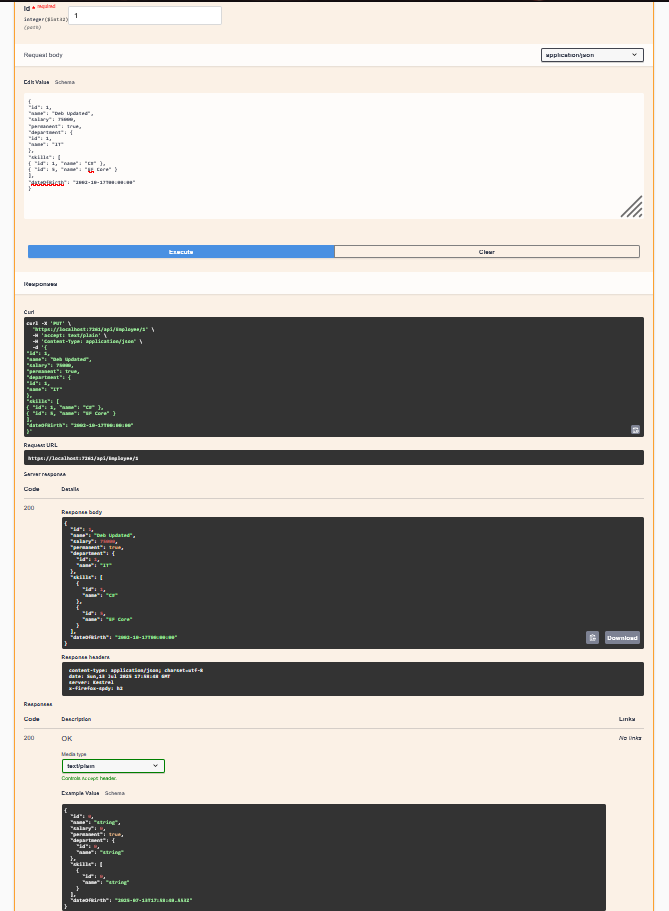
return Ok(employee);

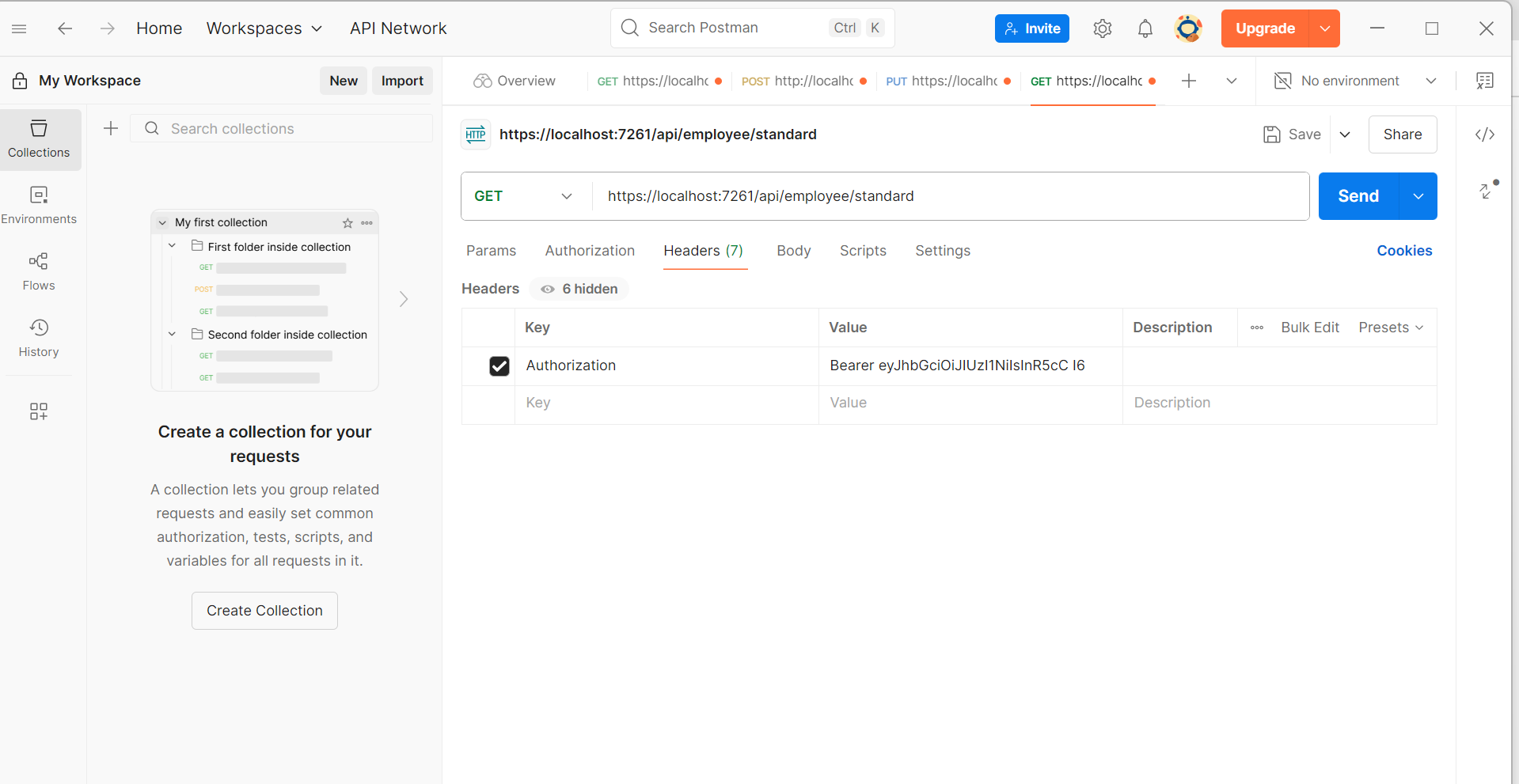
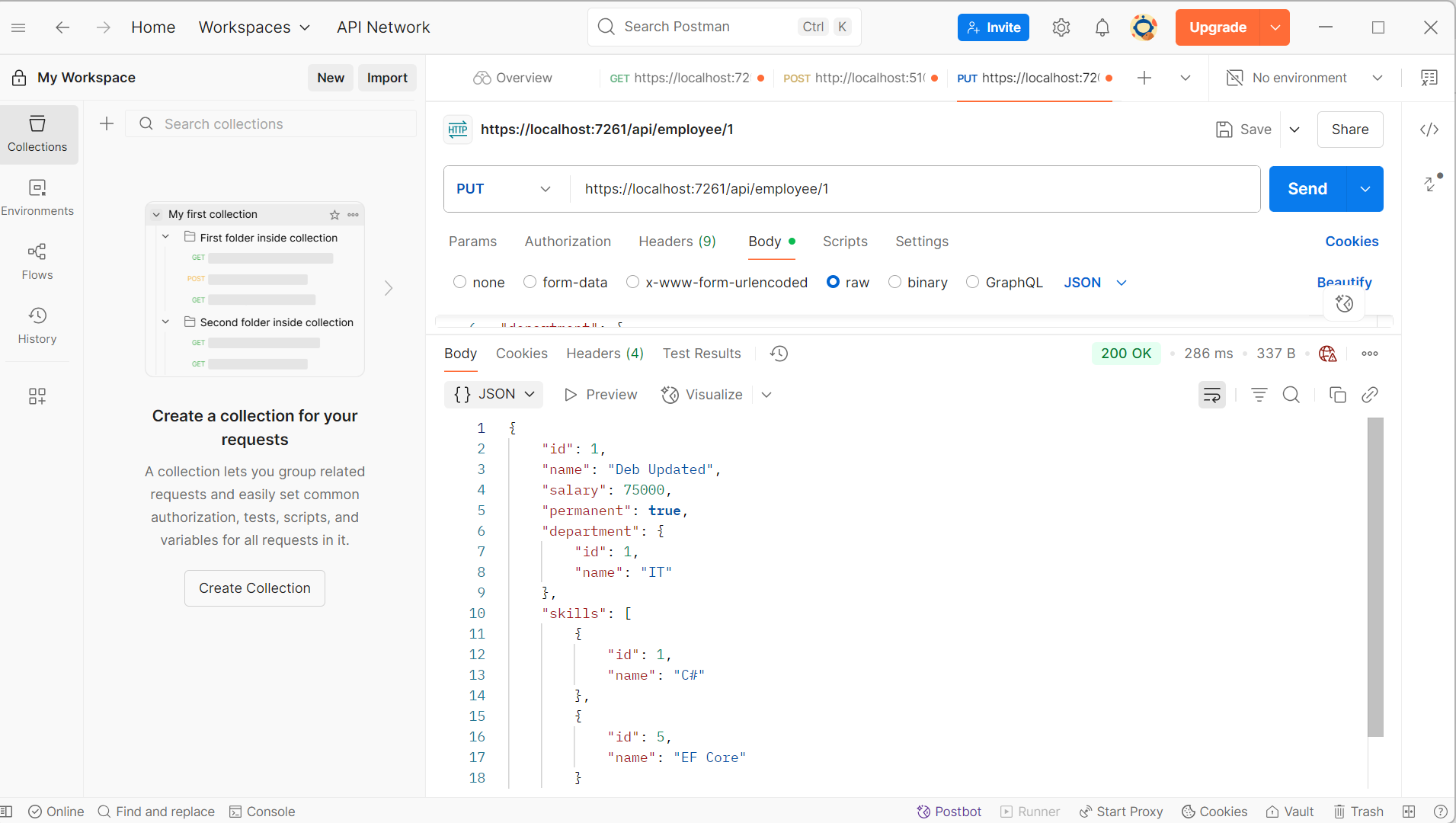
}

}

}

OUTPUT:

****

****

**5. JsonWebToken ,Use the JWT generated thru the AuthController to be used in POSTMAN request. Check for JWT expiration, Add the roles to be authorized in the Authorize attribute.**

AuthController.cs:

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 MyFirstWebApi.Controllers

{

[ApiController]

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

[AllowAnonymous]

public class AuthController : ControllerBase

{

[HttpGet("token")]

public IActionResult GetToken()

{

try

{

var token = GenerateJSONWebToken(101, "Admin");

return Ok(new { token });

}

catch (Exception ex)

{

return StatusCode(500, $"Token generation failed: {ex.Message}");

}

}

private string GenerateJSONWebToken(int userId, string userRole)

{

var key = "mysuperdupersecrettokenkey1234567890";

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

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

var claims = new List<Claim>

{

new Claim(ClaimTypes.Role, userRole),

new Claim("UserId", userId.ToString())

};

var token = new JwtSecurityToken(

issuer: "mySystem",

audience: "myUsers",

claims: claims,

expires: DateTime.Now.AddMinutes(2),

signingCredentials: credentials);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

}

Program.cs:

using Microsoft.OpenApi.Models;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

using MyFirstWebApi.Filters;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers(options =>

{

options.Filters.Add<CustomExceptionFilter>();

});

// Add Swagger with metadata

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddScoped<CustomAuthFilter>();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "TBD",

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")

}

});

c.ResolveConflictingActions(apiDescriptions => apiDescriptions.First());

});

string securityKey = "mysuperdupersecrettokenkey1234567890"; // 32+ chars

var symmetricKey = 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 = symmetricKey

};

});

var app = builder.Build();

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

OUTPUT:

