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

Product.cs

using Microsoft.AspNetCore.Mvc;

using myWebApi.Models;

using System.Collections.Generic;

using System.Linq;

namespace myWebApi.Controllers

{

[ApiController]

[Route("[controller]")]

public class ProductController : ControllerBase

{

// In-memory list for demo purposes

private static readonly List<Product> products = new List<Product>

{

new Product { Id = 1, Name = "Pen", Description = "Blue ink pen", Price = 10.5M },

new Product { Id = 2, Name = "Notebook", Description = "100 pages", Price = 25M }

};

[HttpGet]

public ActionResult<IEnumerable<Product>> Get()

{

return Ok(products);

}

[HttpGet("{id}")]

public ActionResult<Product> Get(int id)

{

var product = products.FirstOrDefault(p => p.Id == id);

if (product == null)

return NotFound();

return Ok(product);

}

[HttpPost]

public ActionResult Post([FromBody] Product product)

{

product.Id = products.Max(p => p.Id) + 1;

products.Add(product);

return CreatedAtAction(nameof(Get), new { id = product.Id }, product);

}

[HttpPut("{id}")]

public ActionResult Put(int id, [FromBody] Product updatedProduct)

{

var existing = products.FirstOrDefault(p => p.Id == id);

if (existing == null)

return NotFound();

existing.Name = updatedProduct.Name;

existing.Description = updatedProduct.Description;

existing.Price = updatedProduct.Price;

return NoContent();

}

[HttpDelete("{id}")]

public ActionResult Delete(int id)

{

var product = products.FirstOrDefault(p => p.Id == id);

if (product == null)

return NotFound();

products.Remove(product);

return NoContent();

}

}

}

ProductController.cs

using Microsoft.AspNetCore.Mvc;

using myWebApi.Models;

using System.Collections.Generic;

using System.Linq;

namespace myWebApi.Controllers

{

[ApiController]

[Route("[controller]")]

public class ProductController : ControllerBase

{

// In-memory list for demo purposes

private static readonly List<Product> products = new List<Product>

{

new Product { Id = 1, Name = "Pen", Description = "Blue ink pen", Price = 10.5M },

new Product { Id = 2, Name = "Notebook", Description = "100 pages", Price = 25M }

};

[HttpGet]

public ActionResult<IEnumerable<Product>> Get()

{

return Ok(products);

}

[HttpGet("{id}")]

public ActionResult<Product> Get(int id)

{

var product = products.FirstOrDefault(p => p.Id == id);

if (product == null)

return NotFound();

return Ok(product);

}

[HttpPost]

public ActionResult Post([FromBody] Product product)

{

product.Id = products.Max(p => p.Id) + 1;

products.Add(product);

return CreatedAtAction(nameof(Get), new { id = product.Id }, product);

}

[HttpPut("{id}")]

public ActionResult Put(int id, [FromBody] Product updatedProduct)

{

var existing = products.FirstOrDefault(p => p.Id == id);

if (existing == null)

return NotFound();

existing.Name = updatedProduct.Name;

existing.Description = updatedProduct.Description;

existing.Price = updatedProduct.Price;

return NoContent();

}

[HttpDelete("{id}")]

public ActionResult Delete(int id)

{

var product = products.FirstOrDefault(p => p.Id == id);

if (product == null)

return NotFound();

products.Remove(product);

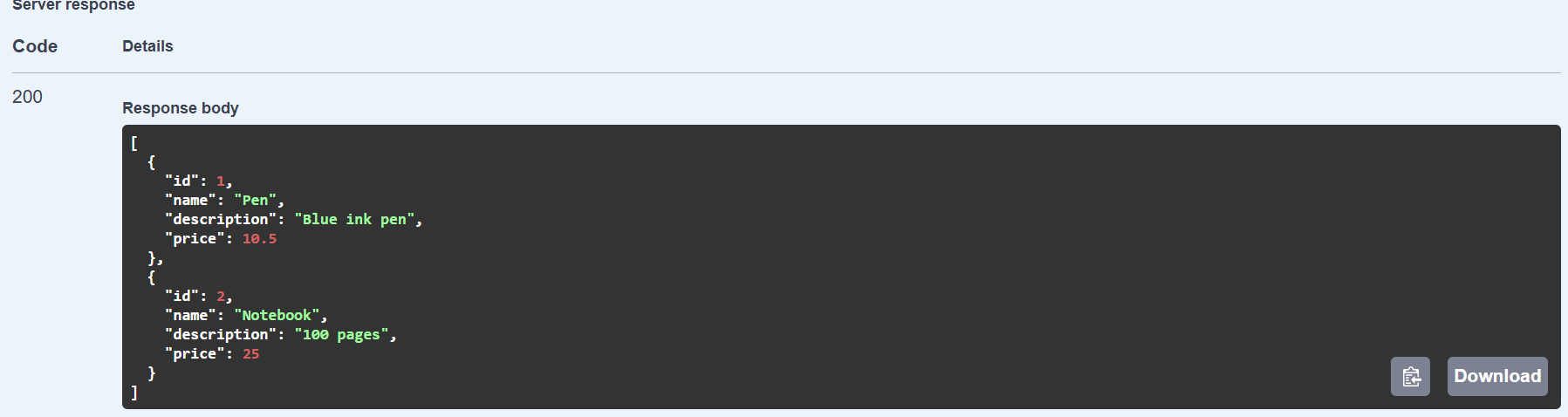
return NoContent();

}

}

}

Get Response



1. Web Api using .Net core with Swagger

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 = "TBD",

TermsOfService = new Uri("https://www.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.UseDeveloperExceptionPage();

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

c.RoutePrefix = "swagger"; // Open Swagger UI at /swagger

});

}

app.UseAuthorization();

app.MapControllers();

app.Run();

EmployeController.cs

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

namespace MySwaggerDemo.Controllers

{

[ApiController]

[Route("emp")] // Step 3 instruction

public class EmployeeController : ControllerBase

{

[HttpGet]

public IEnumerable<object> Get()

{

return new List<object>

{

new { Id = 1, Name = "Alice", Role = "HR" },

new { Id = 2, Name = "Bob", Role = "Developer" }

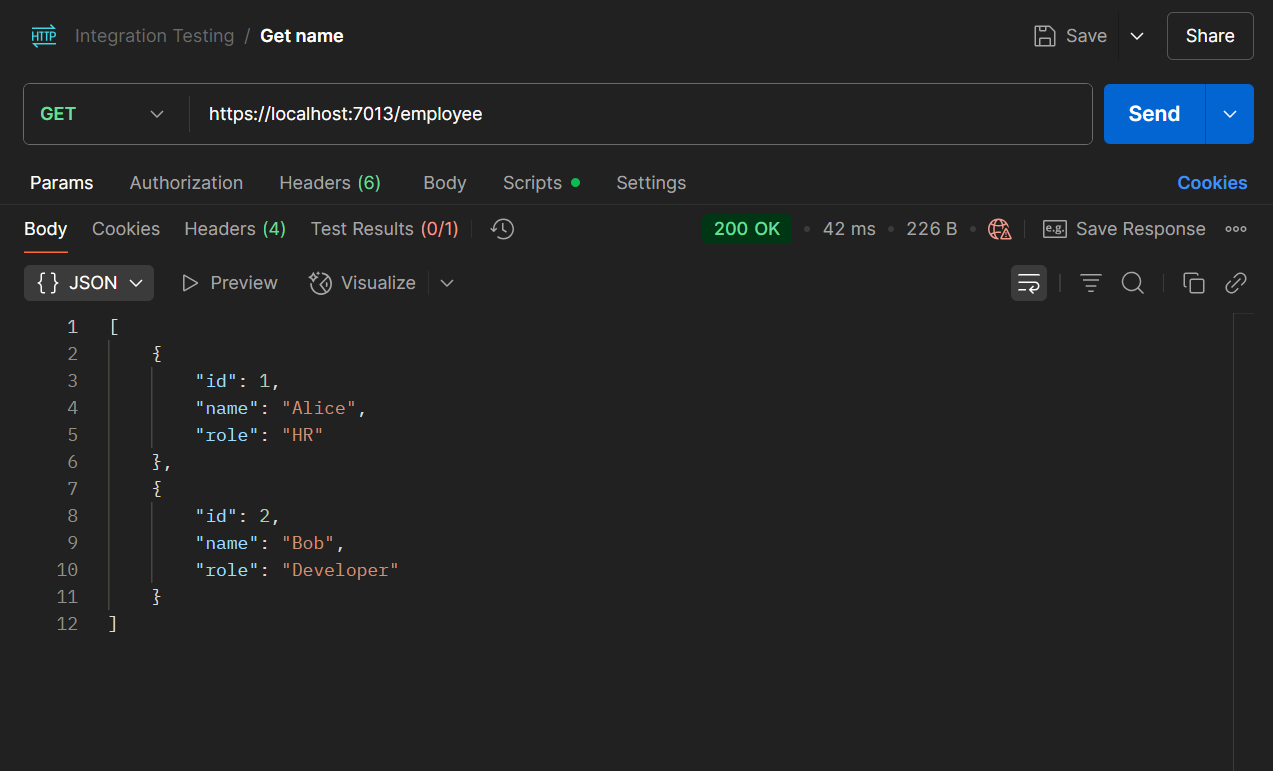
};

}

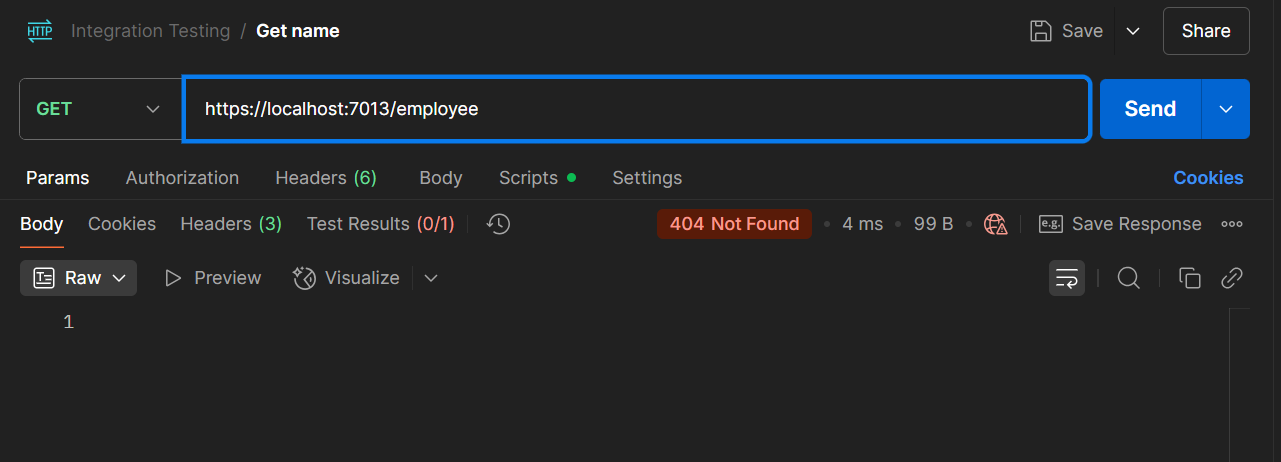
}

}

Output of GET API using Postman



Output when route is changed to ”emp”



1. **WebApi\_Handson**

**1)Web Api using custom model class**

**Models**

**Department.cs**

namespace MyWebApi.Models

{

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

}

}

**Skill.cs**

namespace MyWebApi.Models

{

public class Skill

{

public int Id { get; set; }

public string Name { get; set; }

}

}

**Employee.cs**

using System;

using System.Collections.Generic;

namespace MyWebApi.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; }

}

}

**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 = "TBD",

TermsOfService = new Uri("https://www.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.UseDeveloperExceptionPage();

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

c.RoutePrefix = "swagger"; // Open Swagger UI at /swagger

});

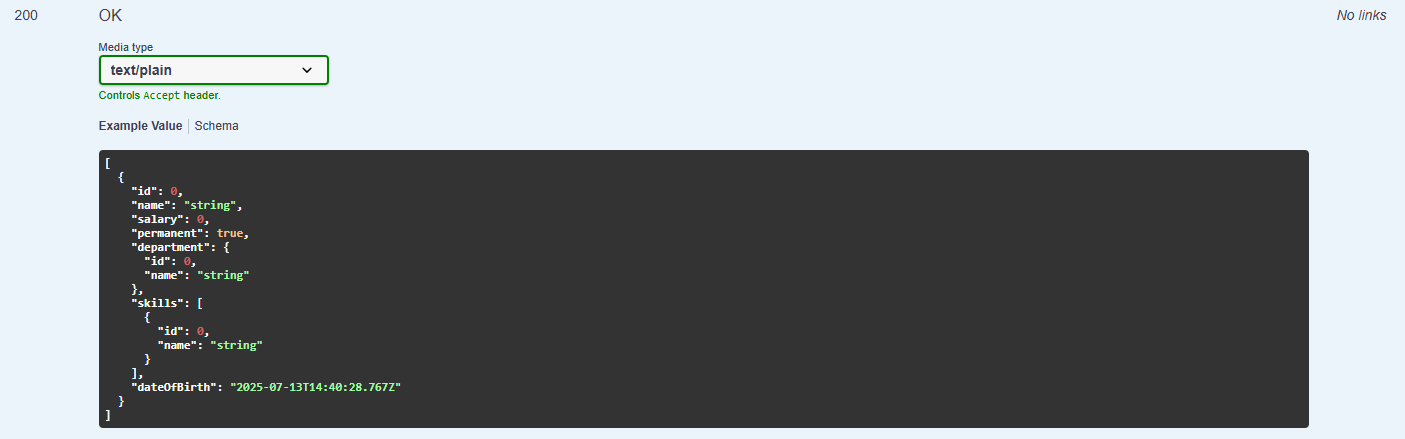
}

app.UseAuthorization();

app.MapControllers();

app.Run();

**Swagger Output for GET mthod**

****

**2)Create a Custom action filter for Authorization.**

**CustomAuthFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace MyWebApi.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

var headers = context.HttpContext.Request.Headers;

// Check if Authorization header exists

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

{

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

return;

}

// Check if it contains the word 'Bearer'

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

if (!token.Contains("Bearer", StringComparison.OrdinalIgnoreCase))

{

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

}

}

}

}

**Program.cs**

using MyWebApi.Filters;

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

// Register controller support

builder.Services.AddControllers();

// Register the custom authorization filter

builder.Services.AddScoped<CustomAuthFilter>();

// Register Swagger generator

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "Employee Web API with Authorization Filter",

Contact = new OpenApiContact

{

Name = "John Doe",

Email = "john@xyzmail.com",

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

},

License = new OpenApiLicense

{

Name = "License Terms",

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

}

});

});

var app = builder.Build();

// Swagger UI middleware

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

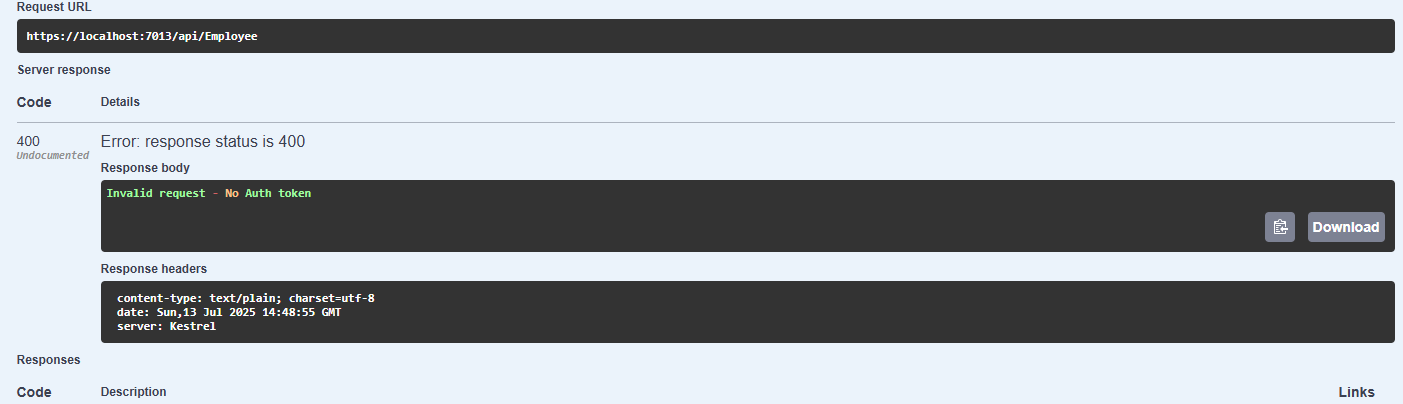
}

app.UseAuthorization();

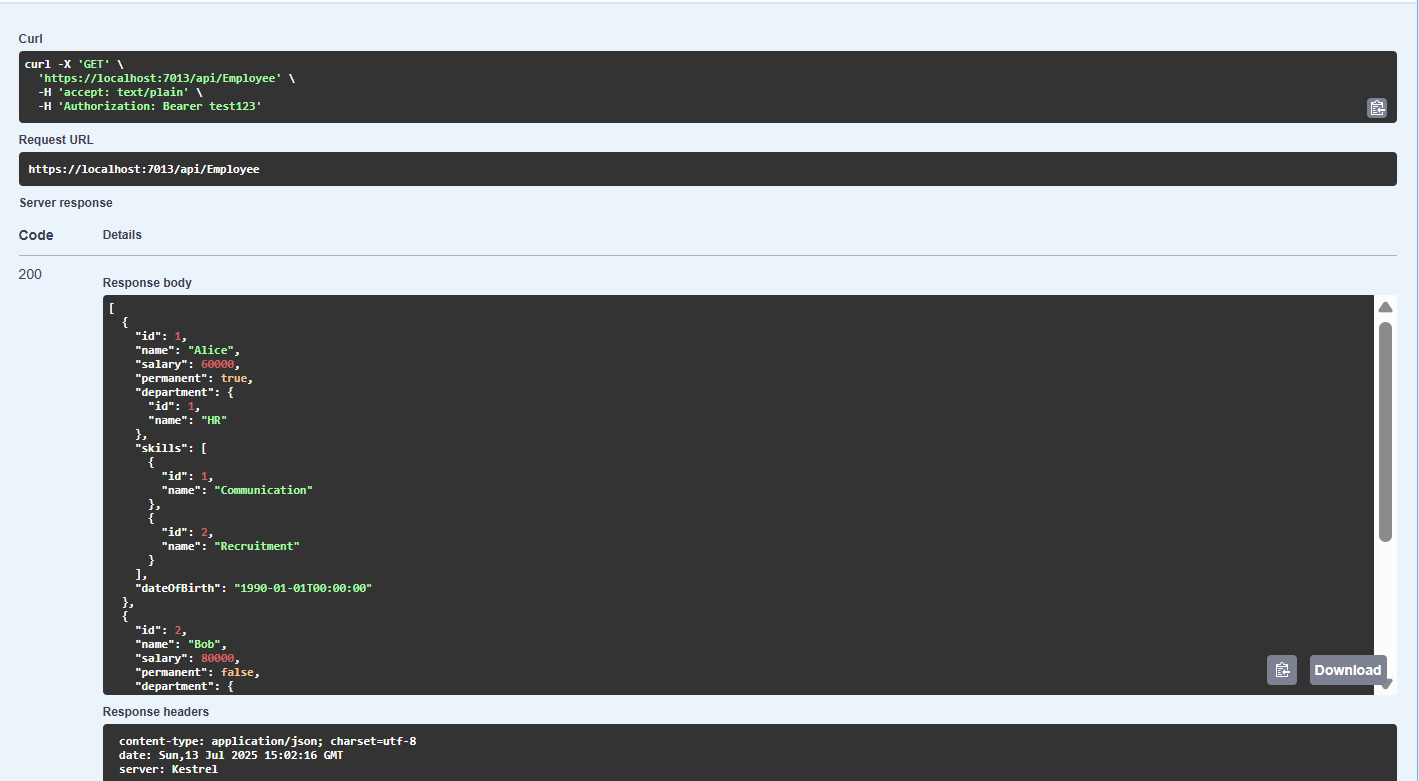
app.MapControllers();

app.Run();

Output without Authorization token



Output with authoriztion token



1. Custom Exception filter

**CustomExceptionFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System;

using System.IO;

namespace MyWebApi.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

private readonly string logFilePath = "exception\_log.txt";

public void OnException(ExceptionContext context)

{

var exception = context.Exception;

var message = $"[{DateTime.Now}] Exception: {exception.Message}\nStackTrace: {exception.StackTrace}\n\n";

// Log to file

File.AppendAllText(logFilePath, message);

// Set response

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

{

StatusCode = 500

};

context.ExceptionHandled = true;

}

}

}

**Program.cs**

using Microsoft.OpenApi.Models;

using MyWebApi.Filters;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddScoped<CustomAuthFilter>();

builder.Services.AddScoped<CustomExceptionFilter>();

// Swagger configuration

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "Employee API with Filters"

});

c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

Name = "Authorization",

Type = SecuritySchemeType.ApiKey,

Scheme = "Bearer",

BearerFormat = "JWT",

In = ParameterLocation.Header,

Description = "Enter 'Bearer' followed by space and your token."

});

c.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Id = "Bearer",

Type = ReferenceType.SecurityScheme

}

},

Array.Empty<string>()

}

});

});

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseAuthorization();

app.MapControllers();

app.Run();

EmployeeController.cs

using Microsoft.AspNetCore.Mvc;

using MyWebApi.Models;

using MyWebApi.Filters;

namespace MyWebApi.Controllers

{

[ApiController]

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

[ServiceFilter(typeof(CustomAuthFilter)), ServiceFilter(typeof(CustomExceptionFilter))]

public class EmployeeController : ControllerBase

{

private static List<Employee> \_employees;

public EmployeeController()

{

if (\_employees == null || !\_employees.Any())

{

\_employees = GetStandardEmployeeList();

}

}

[HttpGet]

[ProducesResponseType(typeof(List<Employee>), 200)]

[ProducesResponseType(500)]

public ActionResult<List<Employee>> Get()

{

// Simulate an exception to test CustomExceptionFilter

throw new Exception("This is a test exception from the GET method.");

}

[HttpGet("standard")]

[ProducesResponseType(typeof(Employee), 200)]

public ActionResult<Employee> GetStandard()

{

return \_employees.FirstOrDefault();

}

[HttpPost]

[ProducesResponseType(201)]

public IActionResult Post([FromBody] Employee employee)

{

\_employees.Add(employee);

return CreatedAtAction(nameof(GetStandard), new { id = employee.Id }, employee);

}

[HttpPut("{id}")]

[ProducesResponseType(204)]

[ProducesResponseType(404)]

public IActionResult Put(int id, [FromBody] Employee updated)

{

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

if (emp == null)

return NotFound();

emp.Name = updated.Name;

emp.Salary = updated.Salary;

emp.Permanent = updated.Permanent;

emp.Department = updated.Department;

emp.Skills = updated.Skills;

emp.DateOfBirth = updated.DateOfBirth;

return NoContent();

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "Alice",

Salary = 60000,

Permanent = true,

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

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "Communication" },

new Skill { Id = 2, Name = "Recruitment" }

},

DateOfBirth = new DateTime(1990, 1, 1)

},

new Employee

{

Id = 2,

Name = "Bob",

Salary = 80000,

Permanent = false,

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

Skills = new List<Skill>

{

new Skill { Id = 3, Name = "C#" },

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

},

DateOfBirth = new DateTime(1985, 7, 12)

}

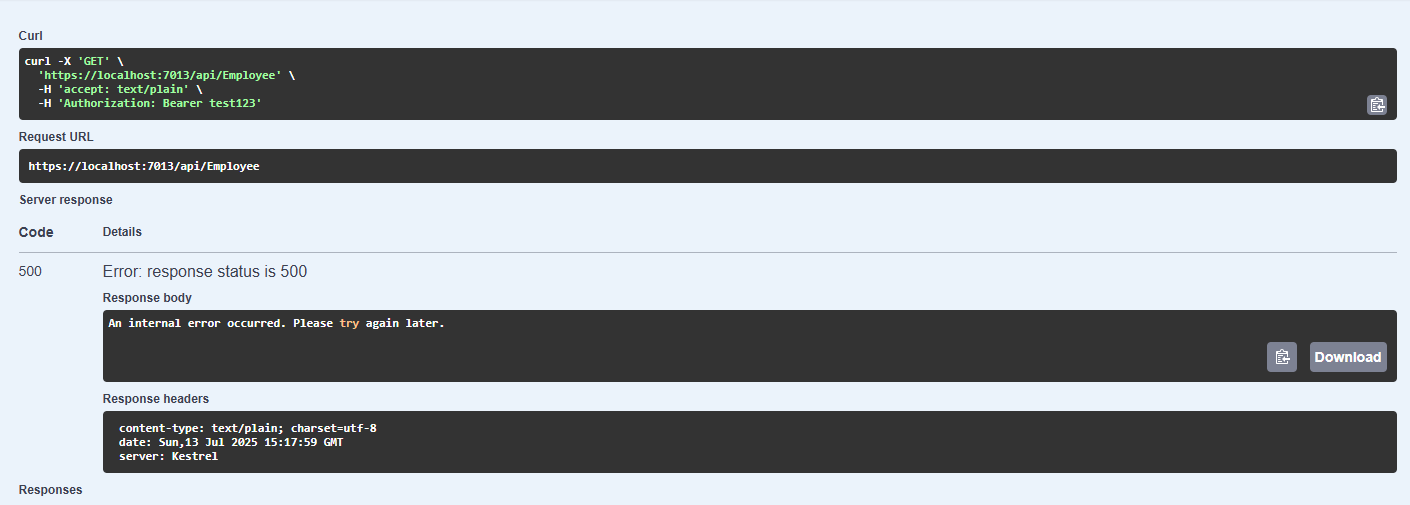
};

}

}

}

OUTPUT



1. WebApi HandsOn

**EmployeeController.cs**

using Microsoft.AspNetCore.Mvc;

using MyWebApi.Models;

using MyWebApi.Filters;

namespace MyWebApi.Controllers

{

[ApiController]

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

[ServiceFilter(typeof(CustomAuthFilter)), ServiceFilter(typeof(CustomExceptionFilter))]

public class EmployeeController : ControllerBase

{

private static List<Employee> \_employees;

public EmployeeController()

{

if (\_employees == null || !\_employees.Any())

{

\_employees = GetStandardEmployeeList();

}

}

[HttpGet]

[ProducesResponseType(typeof(List<Employee>), 200)]

[ProducesResponseType(500)]

public ActionResult<List<Employee>> Get()

{

return \_employees;

}

[HttpGet("standard")]

[ProducesResponseType(typeof(Employee), 200)]

public ActionResult<Employee> GetStandard()

{

return \_employees.FirstOrDefault();

}

[HttpPost]

[ProducesResponseType(201)]

public IActionResult Post([FromBody] Employee employee)

{

\_employees.Add(employee);

return CreatedAtAction(nameof(GetStandard), new { id = employee.Id }, employee);

}

[HttpPut("{id}")]

[ProducesResponseType(typeof(Employee), 200)]

[ProducesResponseType(400)]

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

{

if (id <= 0)

{

return BadRequest("Invalid employee id");

}

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

if (emp == null)

{

return BadRequest("Invalid employee id");

}

emp.Name = updated.Name;

emp.Salary = updated.Salary;

emp.Permanent = updated.Permanent;

emp.Department = updated.Department;

emp.Skills = updated.Skills;

emp.DateOfBirth = updated.DateOfBirth;

return Ok(emp);

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "Alice",

Salary = 60000,

Permanent = true,

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

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "Communication" },

new Skill { Id = 2, Name = "Recruitment" }

},

DateOfBirth = new DateTime(1990, 1, 1)

},

new Employee

{

Id = 2,

Name = "Bob",

Salary = 80000,

Permanent = false,

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

Skills = new List<Skill>

{

new Skill { Id = 3, Name = "C#" },

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

},

DateOfBirth = new DateTime(1985, 7, 12)

}

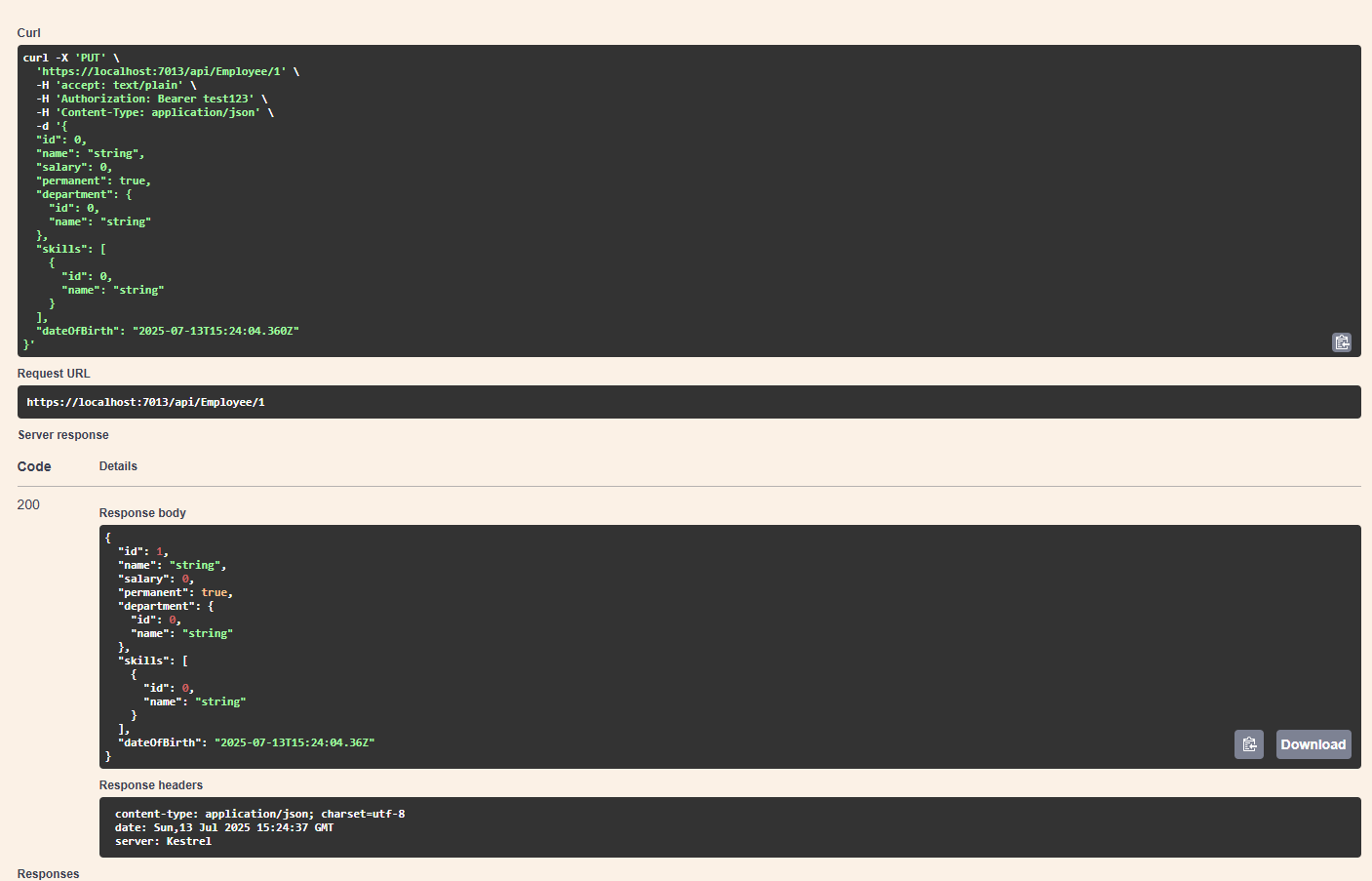
};

}

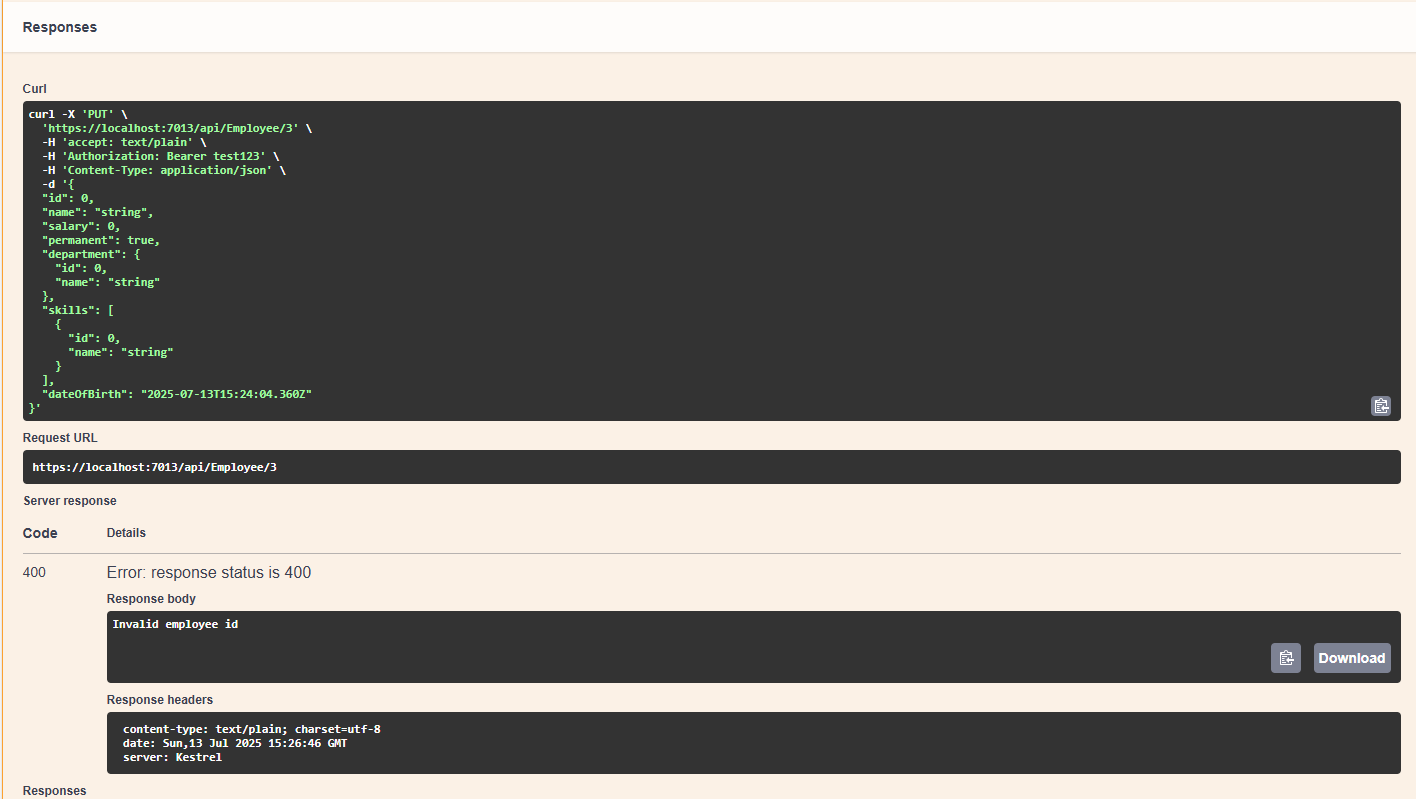
}

}

Output with valid employee id



Output with invalid employee id



5)WebApi Handon

1. **JsonWebToken**

**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 MyWebApi.Controllers

{

[ApiController]

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

[AllowAnonymous]

public class AuthController : ControllerBase

{

[HttpGet("token")]

public IActionResult GetToken()

{

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

return Ok(new { token });

}

private string GenerateJSONWebToken(int userId, string userRole)

{

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretkeyforjwttoken123"));

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(10),

signingCredentials: credentials

);

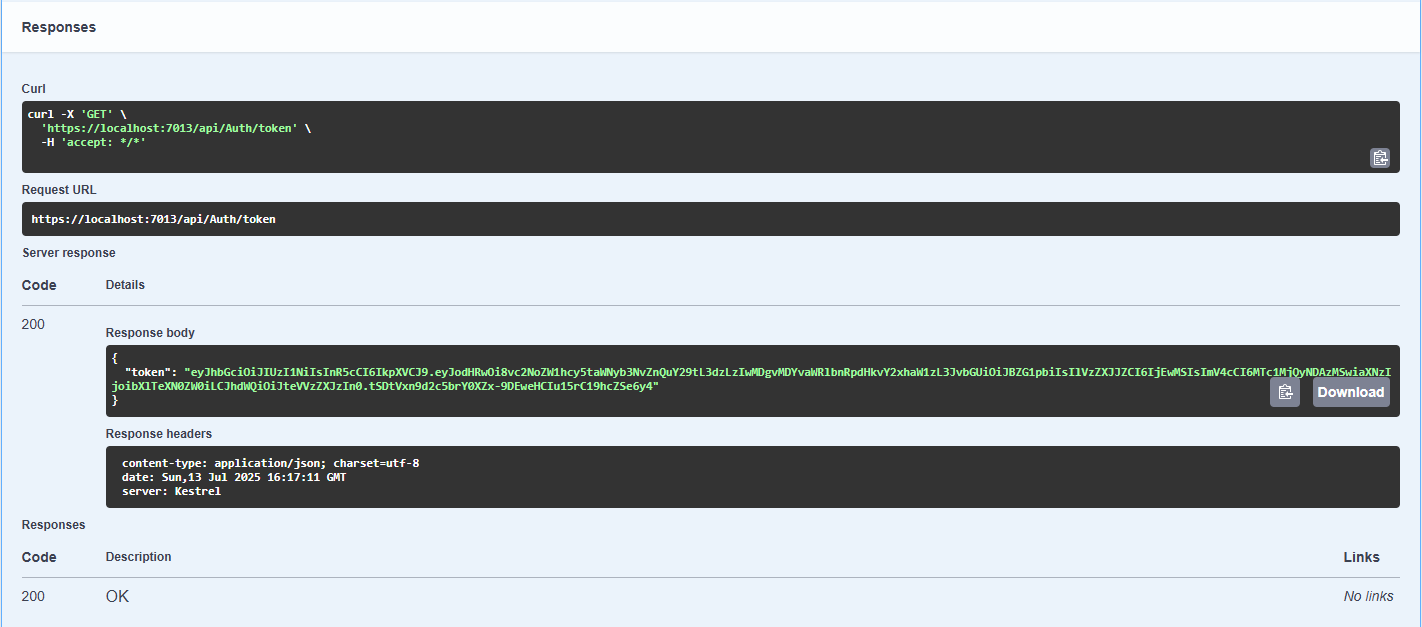
return new JwtSecurityTokenHandler().WriteToken(token);

}

}

}

Generated authentication Token



1. **Use the JWT generated thru the AuthController to be used in POSTMAN request.**

**EmployeeController.cs**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using MyWebApi.Models;

namespace MyWebApi.Controllers

{

[ApiController]

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

[Authorize] // ✅ JWT Authentication only, no custom filter

public class EmployeeController : ControllerBase

{

private static List<Employee> \_employees = new List<Employee>

{

new Employee

{

Id = 1,

Name = "Alice",

Salary = 70000,

Permanent = true,

Department = new Department { Id = 101, Name = "HR" },

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "Communication" },

new Skill { Id = 2, Name = "Leadership" }

},

DateOfBirth = new DateTime(1990, 5, 15)

},

new Employee

{

Id = 2,

Name = "Bob",

Salary = 85000,

Permanent = false,

Department = new Department { Id = 102, Name = "IT" },

Skills = new List<Skill>

{

new Skill { Id = 3, Name = "C#" },

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

},

DateOfBirth = new DateTime(1988, 9, 30)

}

};

[HttpGet]

[ProducesResponseType(typeof(List<Employee>), 200)]

[ProducesResponseType(401)]

public ActionResult<List<Employee>> GetEmployees()

{

return Ok(\_employees);

}

}

}

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 MyWebApi.Controllers

{

[ApiController]

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

[AllowAnonymous]

public class AuthController : ControllerBase

{

[HttpGet("token")]

public IActionResult GetToken()

{

var token = GenerateJSONWebToken(1, "Admin");

return Ok(new { token });

}

private string GenerateJSONWebToken(int userId, string userRole)

{

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretkeyforjwttoken123")); // Must be at least 256 bits

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), // ✅ Token valid for 2 minutes

signingCredentials: credentials

);

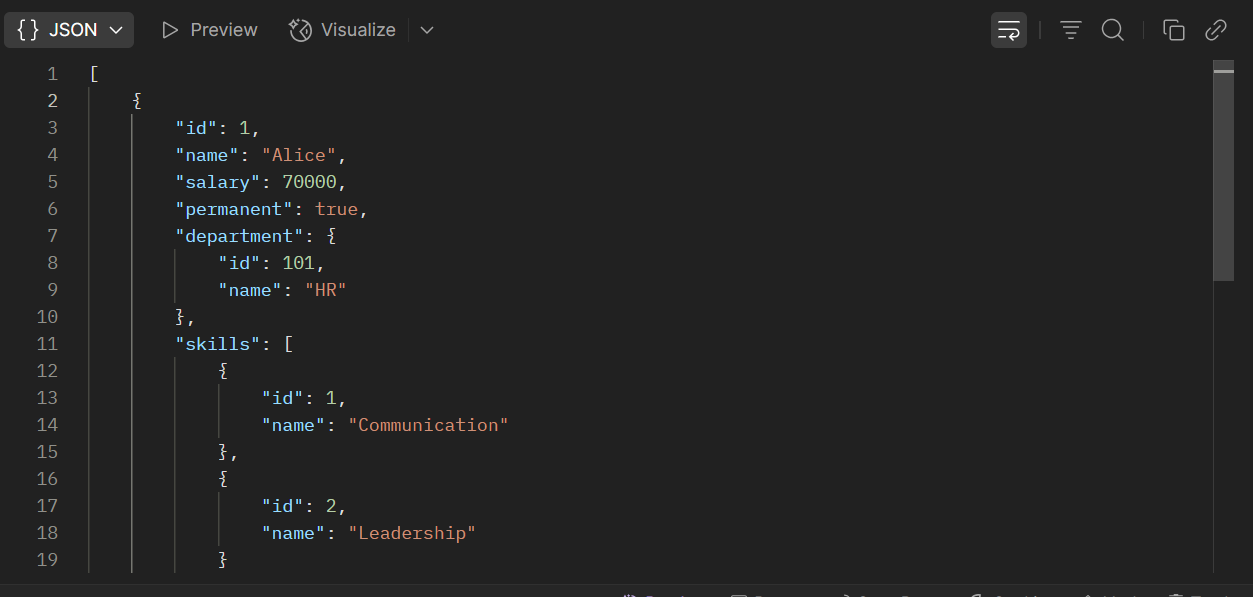
return new JwtSecurityTokenHandler().WriteToken(token);

}

}

}

Output After Auth token



1. Add the roles to be authorized in the Authorize attribute.

AuthController.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace MyWebApi.Controllers

{

[ApiController]

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

public class AuthController : ControllerBase

{

[HttpGet("token")]

public IActionResult GetToken([FromQuery] string role = "Admin")

{

var token = GenerateJSONWebToken(1, role);

return Ok(new { token });

}

private string GenerateJSONWebToken(int userId, string userRole)

{

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretkeyforjwttoken123"));

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

);

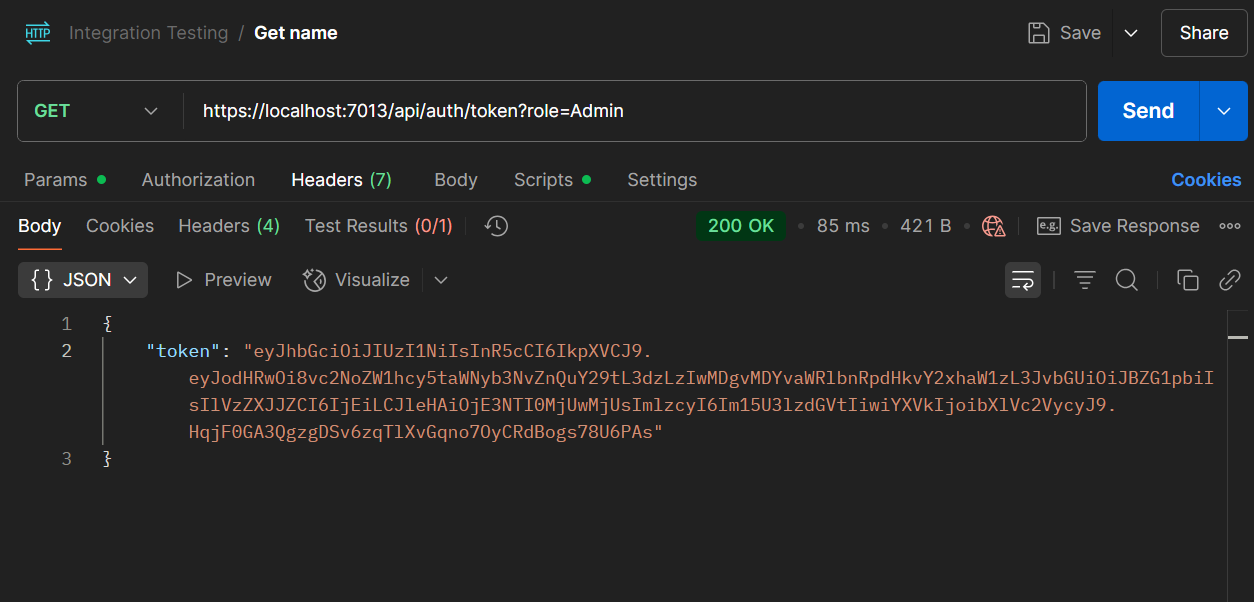
return new JwtSecurityTokenHandler().WriteToken(token);

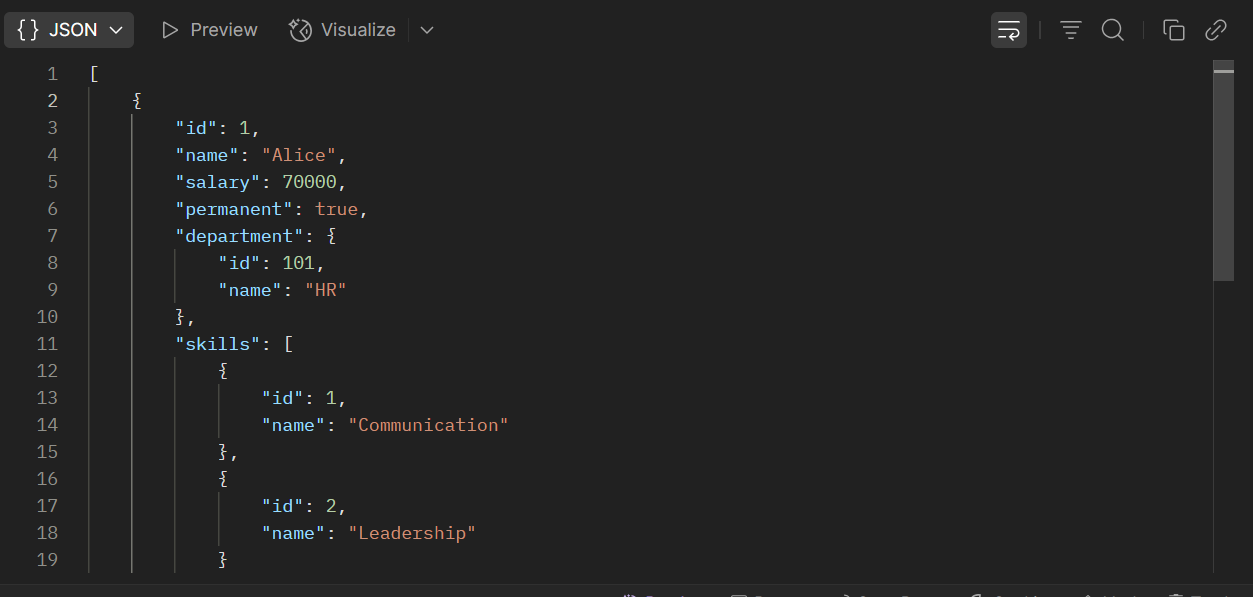
}

}

}

Output





OUTPUT

