ASP.NET Core Web API Hands-On Summary

# 1. RESTful Web API Basics (Ref: WebApi\_Handson.docx)

Concepts:

- REST, Statelessness, Microservices, HTTP Verbs (GET, POST, PUT, DELETE)

- Common status codes: 200 OK, 400 BadRequest, 401 Unauthorized, 500 InternalServerError

Steps:

1. Create project in Visual Studio using Web API template

2. Auto-generated `WeatherForecastController` with GET method

3. Run project and open Swagger

**Code:**

using Microsoft.AspNetCore.Mvc;

namespace WebApiDemo.Controllers

{

[ApiController]

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

public class ValuesController : ControllerBase

{

[HttpGet]

public ActionResult<IEnumerable<string>> Get()

{

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

}

[HttpGet("{id}")]

public ActionResult<string> Get(int id)

{

return "value";

}

[HttpPost]

public void Post([FromBody] string value)

{

}

[HttpPut("{id}")]

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

{

}

[HttpDelete("{id}")]

public void Delete(int id)

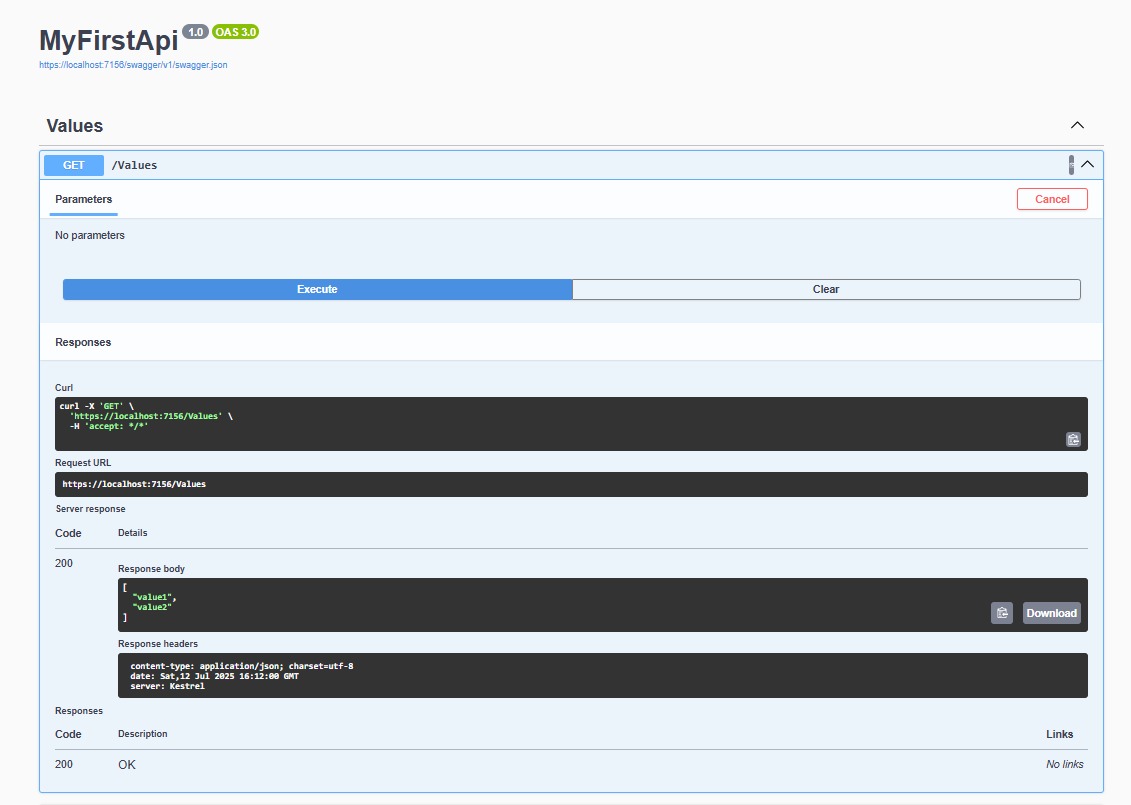
{

}

}

}

**Output:**

.

# Swagger Integration & Postman (Ref: 2. WebApi\_Handson.docx)

**Program.cs:**

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

// Add services

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddCors(options =>

{

options.AddPolicy("AllowAll", policy =>

{

policy.AllowAnyOrigin()

.AllowAnyHeader()

.AllowAnyMethod();

});

});

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "Basic Web API with Swagger",

Contact = new OpenApiContact

{

Name = "Your Name",

Email = "your@email.com",

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

},

License = new OpenApiLicense

{

Name = "MIT",

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

}

});

});

var app = builder.Build();

// Configure middleware

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

**ValuesController.cs:**

using Microsoft.AspNetCore.Mvc;

namespace WebApiDemo.Controllers

{

[ApiController]

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

public class ValuesController : ControllerBase

{

[HttpGet]

public ActionResult<IEnumerable<string>> Get()

{

return new string[] { "Aryan", "Zoie" };

}

[HttpGet("{id}")]

public ActionResult<string> Get(int id)

{

return "value";

}

[HttpPost]

public void Post([FromBody] string value)

{

}

[HttpPut("{id}")]

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

{

}

[HttpDelete("{id}")]

public void Delete(int id)

{

}

}

}

**launchSettings.json:**

{

"$schema": "http://json.schemastore.org/launchsettings.json",

"iisSettings": {

"windowsAuthentication": false,

"anonymousAuthentication": true,

"iisExpress": {

"applicationUrl": "http://localhost:10522",

"sslPort": 44388

}

},

"profiles": {

"http": {

"commandName": "Project",

"dotnetRunMessages": true,

"launchBrowser": true,

"launchUrl": "swagger",

"applicationUrl": "http://localhost:5167",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

},

"https": {

"commandName": "Project",

"dotnetRunMessages": true,

"launchBrowser": true,

"launchUrl": "swagger",

"applicationUrl": "https://localhost:7145;http://localhost:5167",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

},

"IIS Express": {

"commandName": "IISExpress",

"launchBrowser": true,

"launchUrl": "swagger",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

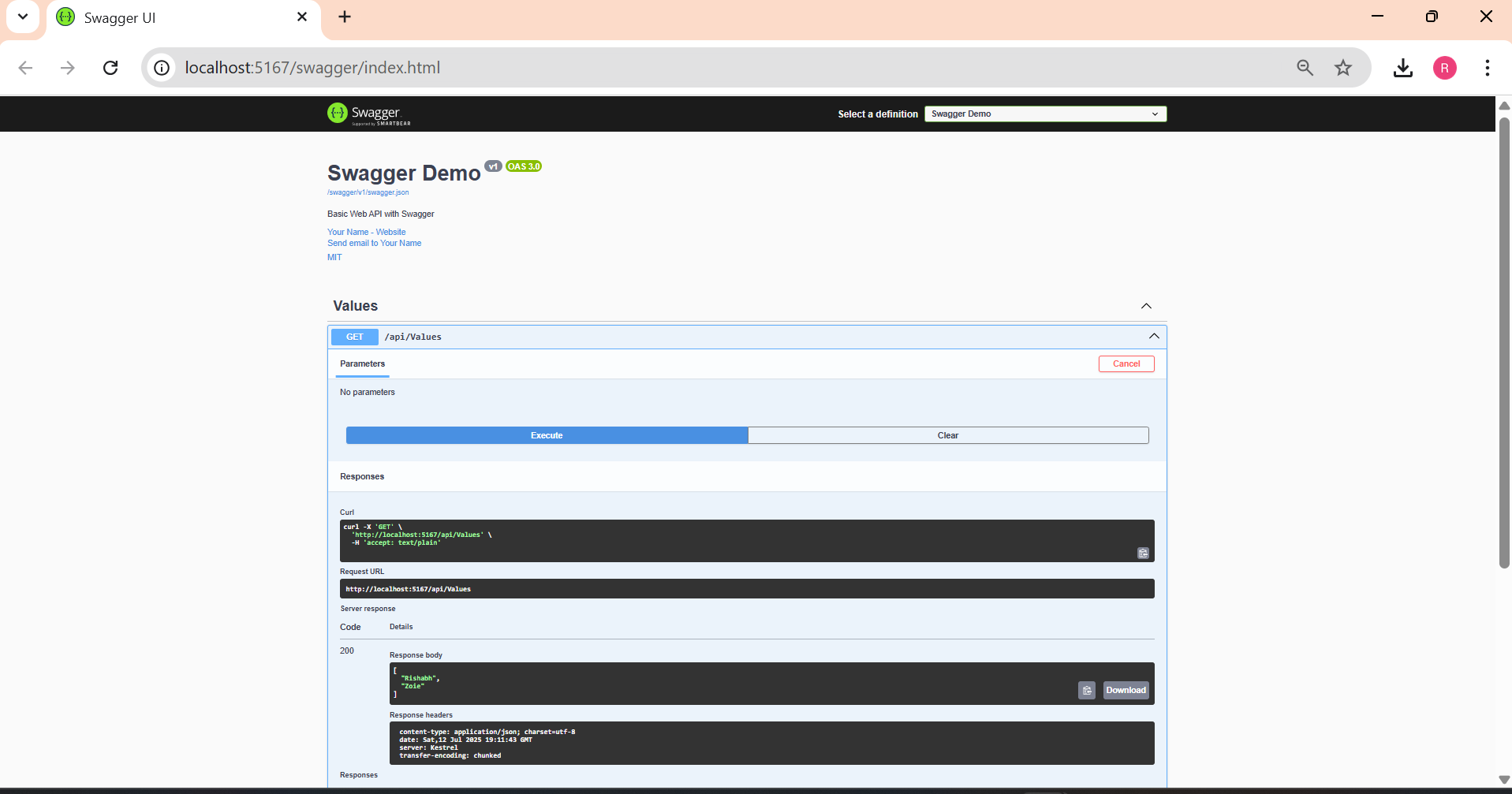
}

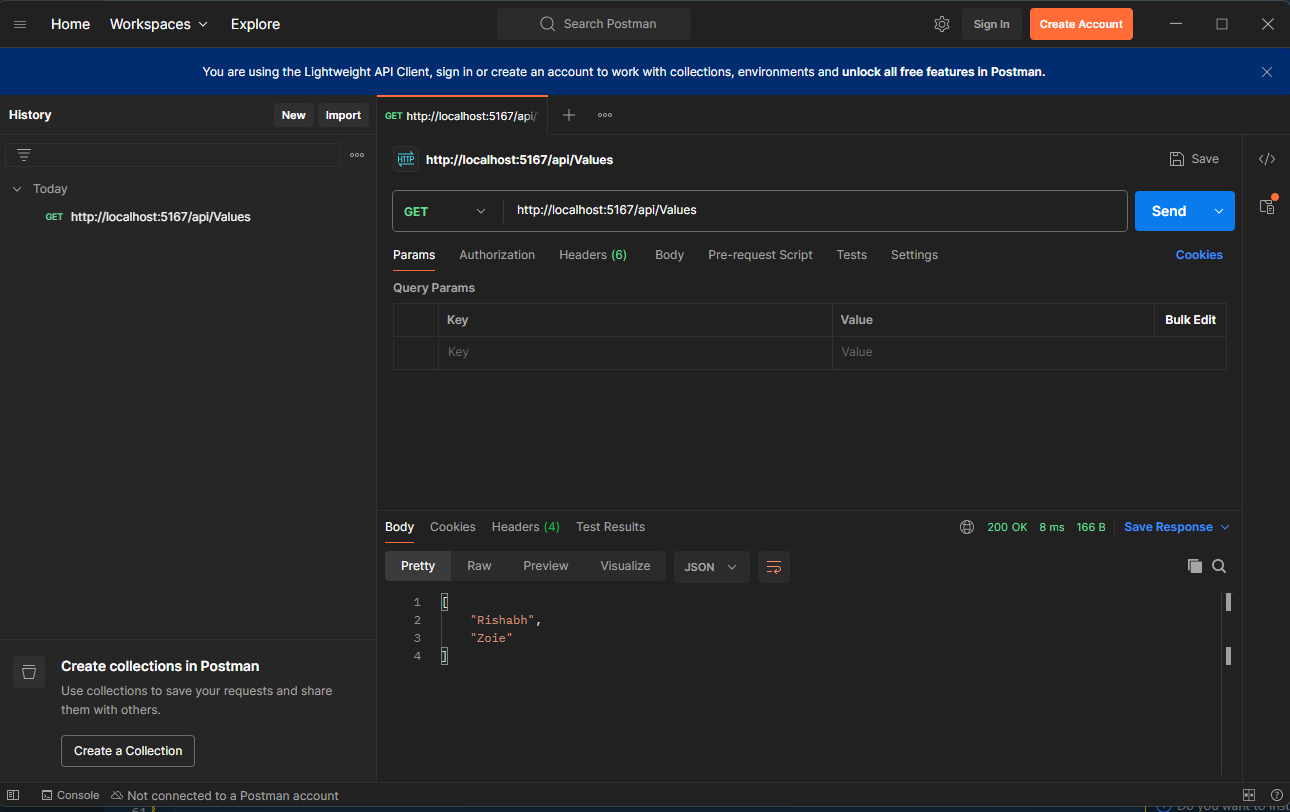
}

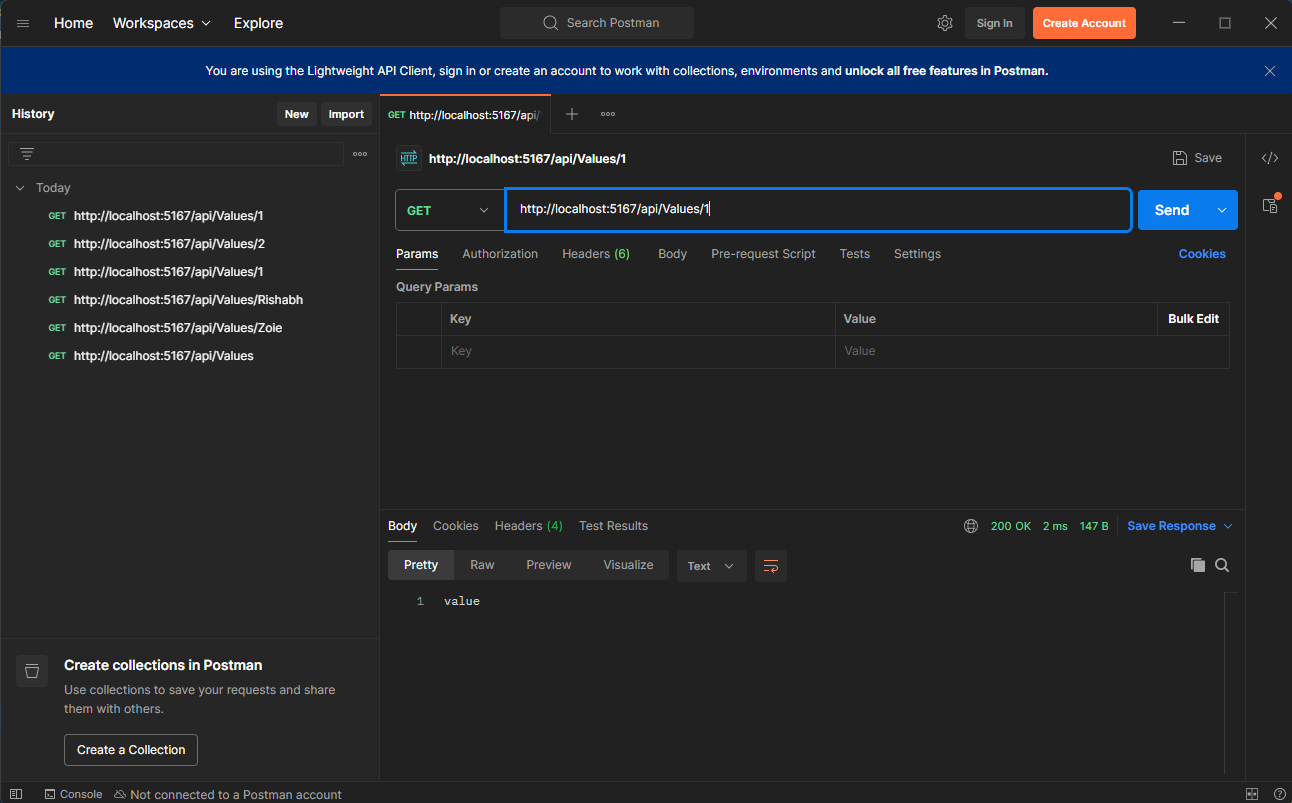
}

}

**Postman Output:**







# 3. Custom Models & Filters (Ref: 3. WebApi\_Handson.docx)

**EmployeeController.cs:**

using Microsoft.AspNetCore.Mvc;

using WebApiDemo.Models;

namespace WebApiDemo.Controllers

{

[ApiController]

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

public class EmployeeController : ControllerBase

{

private static List<Employee> \_employees = GetStandardEmployeeList();

private static List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "Alice",

Salary = 50000,

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)

}

};

}

[HttpGet]

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

public ActionResult<List<Employee>> Get()

{

return Ok(\_employees);

}

[HttpGet("standard")]

public ActionResult<List<Employee>> GetStandard()

{

return Ok(GetStandardEmployeeList());

}

[HttpPost]

public IActionResult Post([FromBody] Employee emp)

{

\_employees.Add(emp);

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

}

[HttpPut("{id}")]

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

{

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

if (existing == null) return NotFound();

return NoContent();

}

}

}

**Program.cs:**

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers(options =>

{

options.Filters.Add<CustomExceptionFilter>();

});

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddCors(options =>

{

options.AddPolicy("AllowAll", policy =>

{

policy.AllowAnyOrigin()

.AllowAnyHeader()

.AllowAnyMethod();

});

});

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "Basic Web API with Swagger",

Contact = new OpenApiContact

{

Name = "Your Name",

Email = "your@email.com",

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

},

License = new OpenApiLicense

{

Name = "MIT",

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

}

});

});

builder.Services.AddScoped<CustomAuthFilter>();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

**ValueController.cs:**

using Microsoft.AspNetCore.Mvc;

namespace WebApiDemo.Controllers

{

[ApiController]

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

public class ValuesController : ControllerBase

{

[HttpGet]

public ActionResult<IEnumerable<string>> Get()

{

return new string[] { "Rishabh", "Zoie" };

}

[HttpGet("{id}")]

public ActionResult<string> Get(int id)

{

return "value";

}

[HttpPost]

public void Post([FromBody] string value)

{

}

[HttpPut("{id}")]

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

{

}

[HttpDelete("{id}")]

public void Delete(int id)

{

}

[HttpGet("throw")]

public IActionResult ThrowException()

{

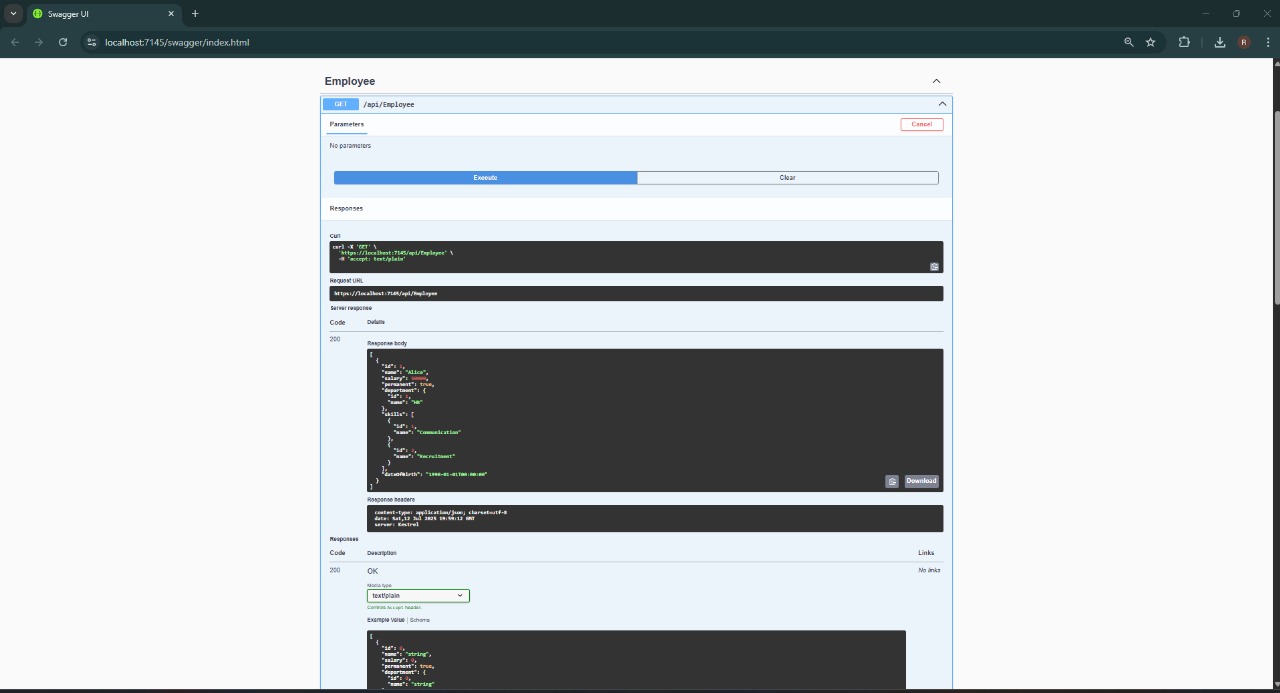
throw new Exception("Test exception");

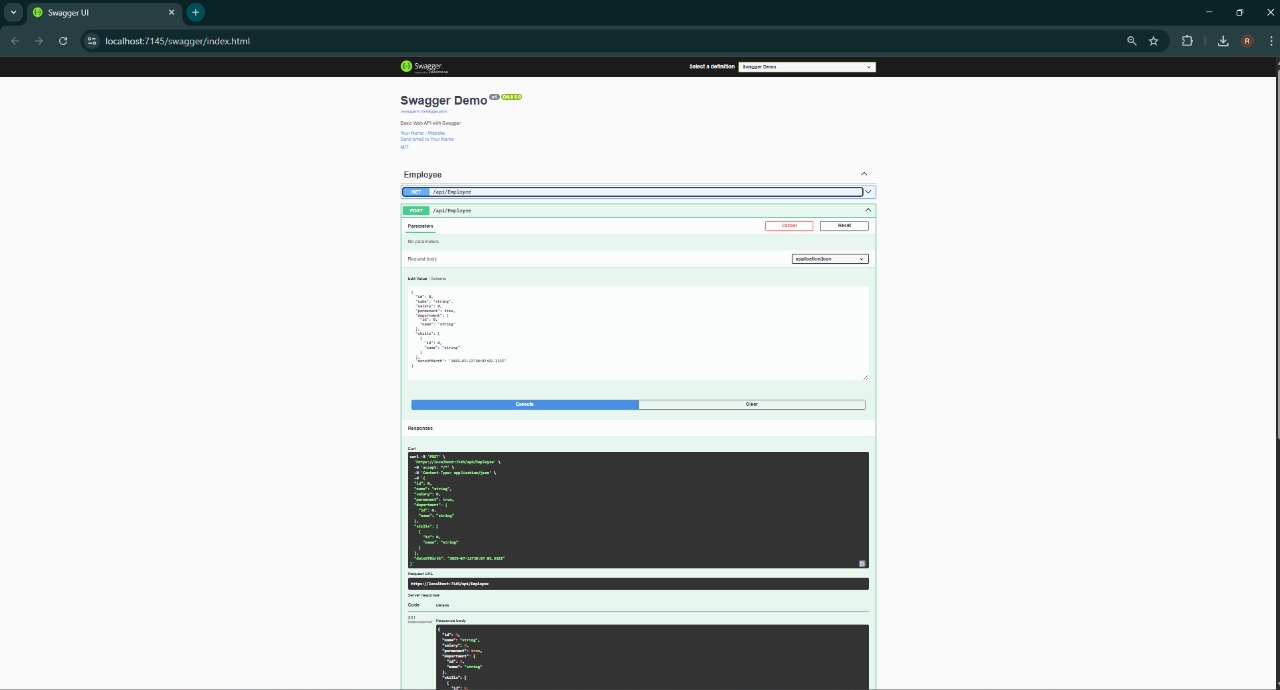
}

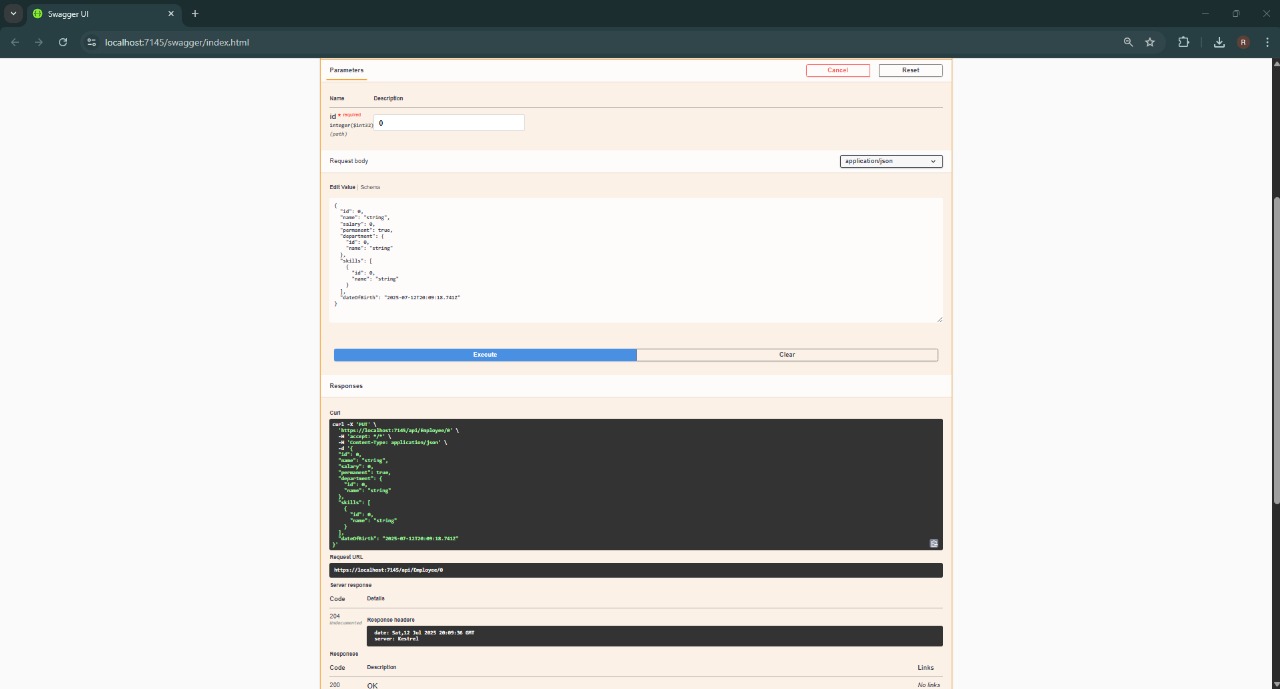
}

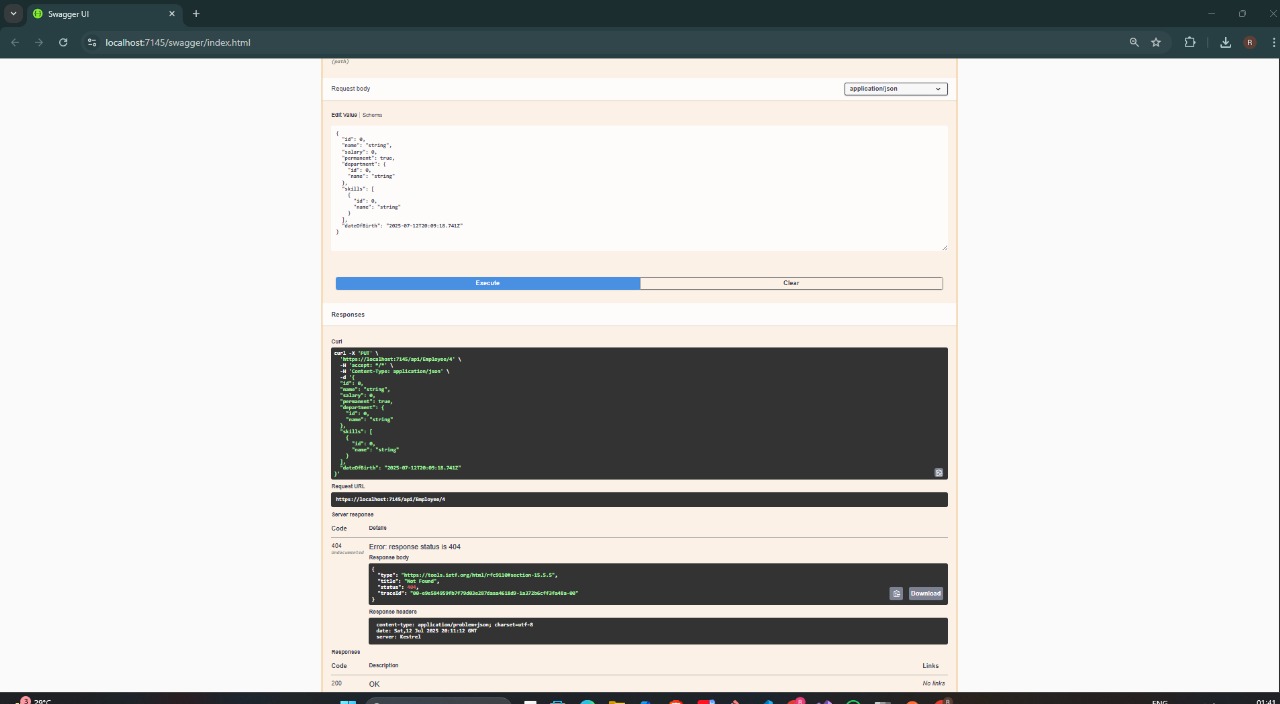
}

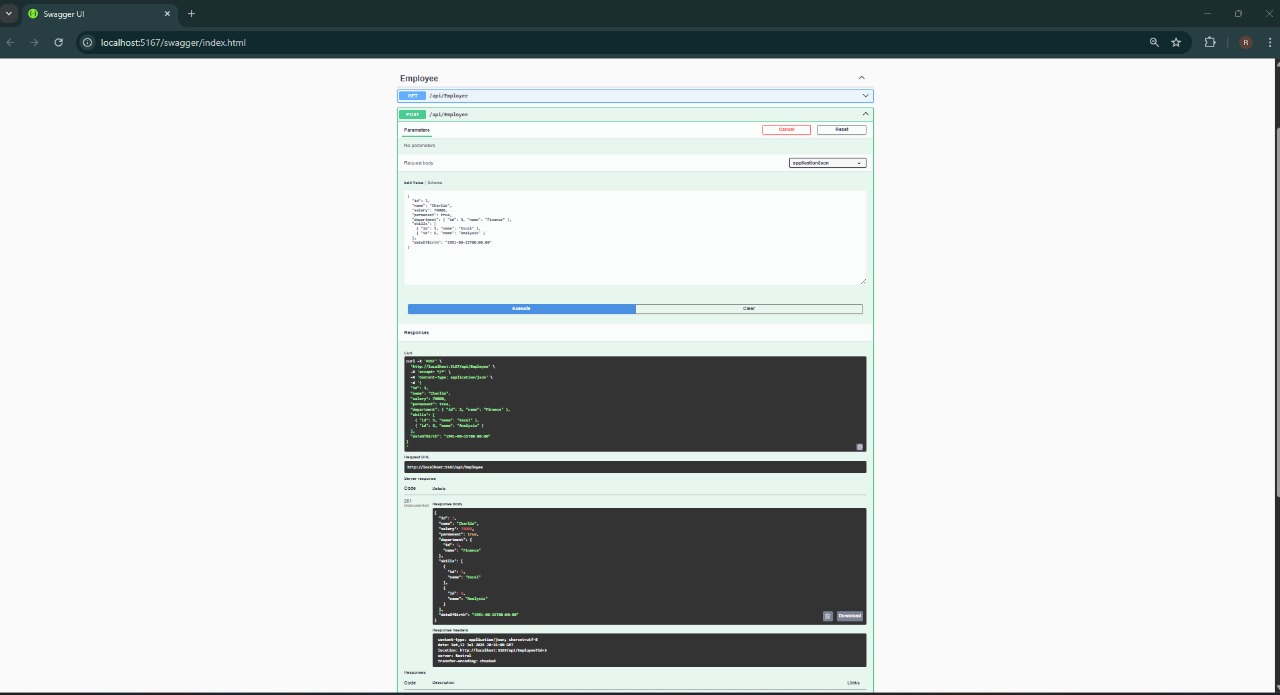
**Output:**











# CRUD Operations (Ref: 4. WebApi\_Handson.docx)

**EmployeeController.cs:**

using Microsoft.AspNetCore.Mvc;

using WebApiDemo.Models;

namespace WebApiDemo.Controllers

{

[ApiController]

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

public class EmployeeController : ControllerBase

{

private static List<Employee> \_employees = GetStandardEmployeeList();

private static List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "Alice",

Salary = 50000,

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)

}

};

}

[HttpGet]

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

public ActionResult<List<Employee>> Get()

{

return Ok(\_employees);

}

[HttpGet("standard")]

public ActionResult<List<Employee>> GetStandard()

{

return Ok(GetStandardEmployeeList());

}

[HttpPost]

public IActionResult Post([FromBody] Employee emp)

{

\_employees.Add(emp);

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

}

[HttpPut("{id}")]

public ActionResult<Employee> Put(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;

existing.Department = emp.Department;

existing.Skills = emp.Skills;

existing.DateOfBirth = emp.DateOfBirth;

return Ok(existing);

}

}

}

**Program.cs:**

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers(options =>

{

options.Filters.Add<CustomExceptionFilter>();

});

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddCors(options =>

{

options.AddPolicy("AllowAll", policy =>

{

policy.AllowAnyOrigin()

.AllowAnyHeader()

.AllowAnyMethod();

});

});

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "Basic Web API with Swagger",

Contact = new OpenApiContact

{

Name = "...",

Email = "yyyy@email.com",

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

},

License = new OpenApiLicense

{

Name = "MIT",

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

}

});

});

builder.Services.AddScoped<CustomAuthFilter>();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

}

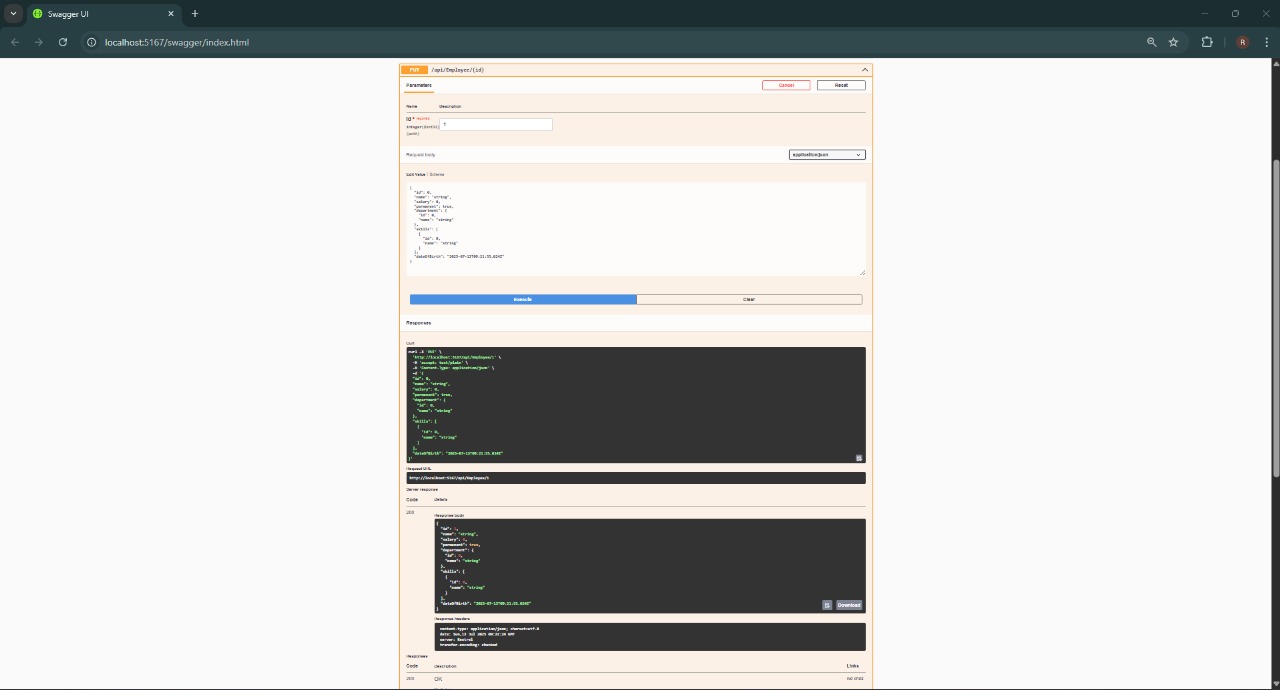
app.UseHttpsRedirection();

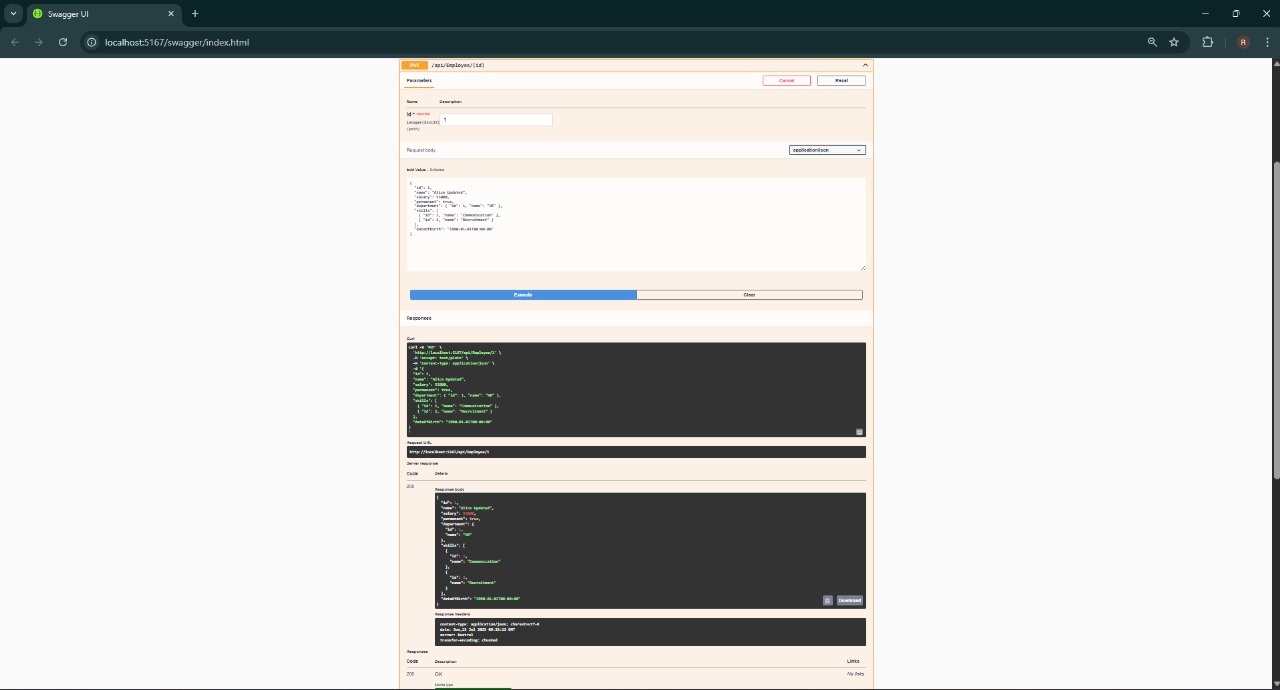
app.UseAuthorization();

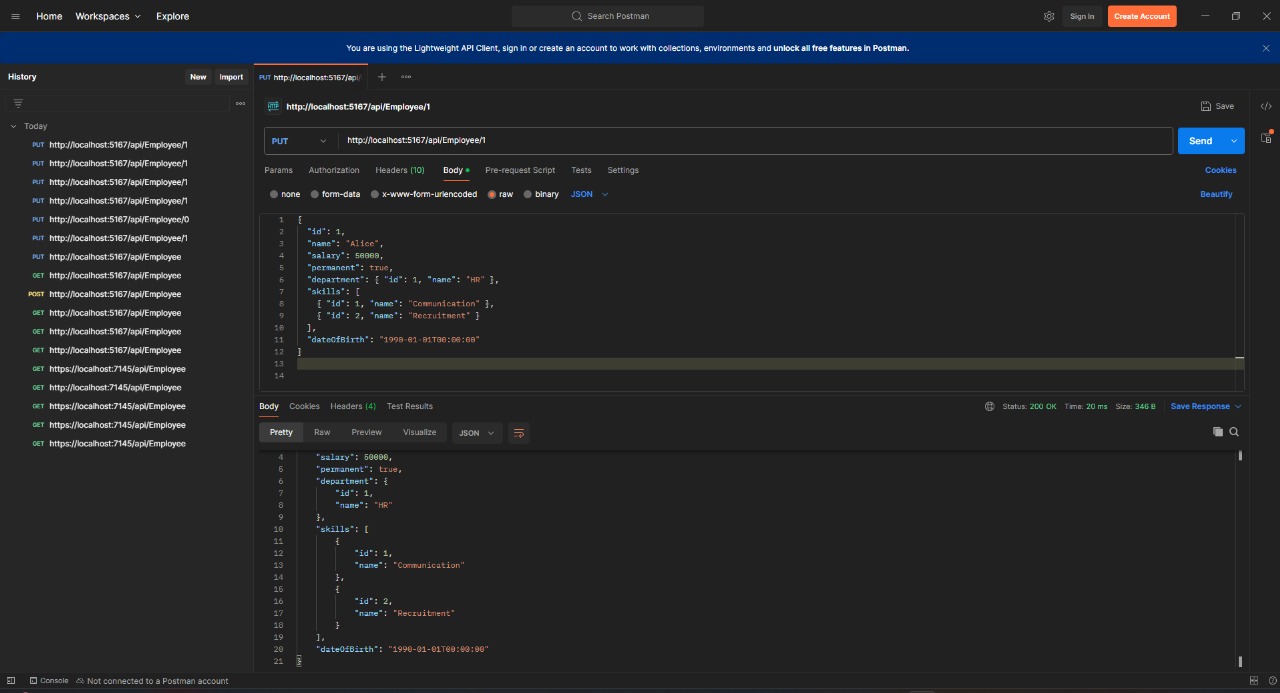
app.MapControllers();

app.Run();

**Output:**







# JWT Authentication & Role-Based Access (Ref: 5. WebApi\_Handson.docx)

**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 WebApiDemo.Controllers

{

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

[ApiController]

[AllowAnonymous]

public class AuthController : ControllerBase

{

[HttpGet("token")]

public IActionResult GetToken()

{

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretkey1234567890abcd"));

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

var claims = new[]

{

new Claim(ClaimTypes.Role, "Admin"),

new Claim("UserId", "1")

};

var token = new JwtSecurityToken(

issuer: "mySystem",

audience: "myUsers",

claims: claims,

expires: DateTime.Now.AddMinutes(2),

signingCredentials: credentials);

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

}

}

}

**EmployeeController.cs:**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using WebApiDemo.Models;

namespace WebApiDemo.Controllers

{

[Authorize(Roles = "Admin,POC")]

[ApiController]

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

public class EmployeeController : ControllerBase

{

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

{

new Employee

{

Id = 1,

Name = "Alice",

Salary = 50000,

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)

}

};

[HttpGet]

public ActionResult<List<Employee>> Get() => Ok(\_employees);

[HttpPost]

public IActionResult Post([FromBody] Employee emp)

{

\_employees.Add(emp);

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

}

[HttpPut("{id}")]

public ActionResult<Employee> Put(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;

existing.Department = emp.Department;

existing.Skills = emp.Skills;

existing.DateOfBirth = emp.DateOfBirth;

return Ok(existing);

}

}

}

**Program.cs:**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

string securityKey = "mysuperdupersecretkey1234567890abcd";

var symmetricSecurityKey = 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 = symmetricSecurityKey

};

});

builder.Services.AddControllers();

builder.Services.AddCors(options =>

{

options.AddPolicy("AllowAll", policy =>

policy.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader());

});

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo { Title = "JWT Demo", Version = "v1" });

});

var app = builder.Build();

app.UseHttpsRedirection();

app.UseCors("AllowAll");

app.UseAuthentication();

app.UseAuthorization();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.MapControllers();

app.Run();

**Output:**

