**Superset-Id: 6364597  
Name: SRINJOY PAUL**

**Week-5 WEB API HANDSON**

**1. WebApi\_Handson.docx**

## **1) Explain the concept of RESTful web service, Web API & Microservice**

**Answer:**  
- **RESTful Web Service** is an architectural style based on stateless communication and resource representation.  
- **Web API** allows communication between client and server over HTTP using standard verbs (GET, POST, etc.).  
- **Microservice** is a small, independently deployable service that performs a specific business function.

## 2) What are the Features of REST Architecture?

**Answer:**  
-**Representational State Transfer (REST)**: Resource-based design  
-**Stateless**: Each request from client to server must contain all information  
-**Messages**: Generally exchanged via HTTP in JSON or XML

## 3) Concept of Microservice

**Answer:**Microservice architecture is a design approach where a large application is built as a suite of small, independent services, each focused on a specific business capability. These services communicate with each other via APIs, usually over HTTP (REST) or messaging queues.

**Key Concepts of Microservices:**

**- Independent Deployment:**  Each microservice can be deployed and updated independently without affecting others.

**-Single Responsibility:**  Each service is designed to perform one business function (e.g., user service, payment service).  
-**Technology Agnostic:**  Different microservices can be developed using different programming languages and frameworks.

**-Decentralized Data Management:** Each microservice manages its own database (no shared schema). **-Lightweight Communication:** Services typically use REST APIs or messaging (RabbitMQ, Kafka) to interact. **-Scalability:** Individual services can be scaled independently based on demand.  
**-Resilience:** Failure in one service does not necessarily bring down the whole system.

## Difference between WebService & WebAPI, Not restricted to send XML as response

**Answer:**

| Aspect | WebService (SOAP-based) | WebAPI (RESTful) |
| --- | --- | --- |
| Response Format | Restricted to XML only | Not restricted — supports JSON, XML, plain text, HTML, etc. |
| Flexibility | Fixed to SOAP envelope using XML | Flexible — returns data based on client’s Accept header |
| Modern Usage | Less preferred for modern web or mobile apps (which use JSON) | Preferred for modern applications due to JSON-first approach |
| Serialization | Requires strict XML schema (XSD/WSDL) | Easily serializes/deserializes POCOs to/from JSON/XML |

## 5)Explain what is HttpRequest & HttpResponse

**Answer:**  
**-HttpRequest**: A message sent by the client to request data or perform an action (includes headers, URL, method, and body).  
  
-**HttpResponse**: The server's reply to the request, including a status code, headers, and possibly a response body (like JSON or HTML).

## 6) List the types of Action Verbs

**Answer:**

| HTTP Verb | Purpose | Typical Use Case |
| --- | --- | --- |
| **HttpGet** | Retrieve data from the server | Fetch list of products, user profile, etc. |
| **HttpPost** | Send new data to the server | Create a new record (e.g., new order) |
| **HttpPut** | Update existing data | Modify an existing item entirely |
| **HttpDelete** | Delete a specific resource | Remove a user, delete an item from cart |

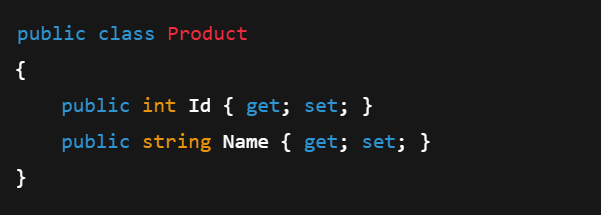
## 7) List the types of HttpStatusCodes used in WebAPI

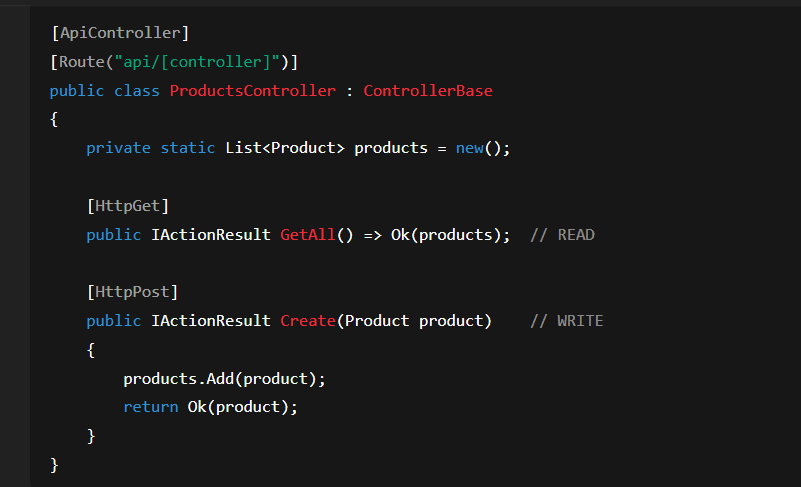
| Status Code | Name | Meaning |
| --- | --- | --- |
| **200** | **OK** | The request was successful. |
| **400** | **BadRequest** | The request is malformed or contains invalid data. |
| **401** | **Unauthorized** | The request lacks valid authentication credentials. |
| **500** | **InternalServerError** | An unexpected error occurred on the server. |

**Answer:**

**9)Demonstrate creation of a simple WebAPI - With Read, Write actions**

**Answer:  
Steps:  
1) Create a New Project:  
 In Visual Studio or CLI:**dotnet new webapi -n SampleApi  
  
2)**Create a Model:**

****

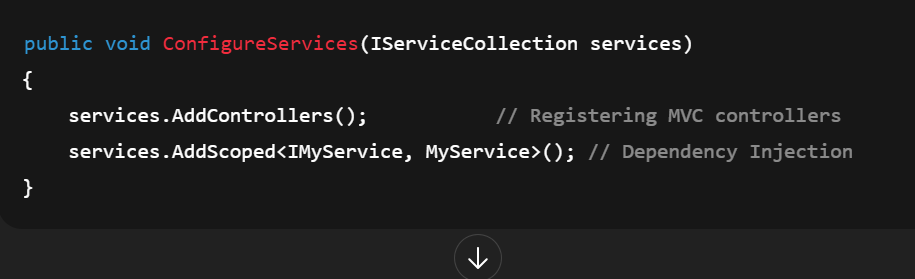
1. **Create a Controller**
2. **Run and Test using Swagger UI or Postman.**

| Component | Description |
| --- | --- |
| **Controller** | Inherits from ControllerBase or ApiController |
| **Route** | Defined using [Route("api/[controller]")] attribute |
| **Action Methods** | Decorated with action verbs like [HttpGet], [HttpPost], etc. |
| **Model** | Class representing data (e.g., Product, User) |

**10) Explain the types of Configuration files of WebAPI  
Answer:   
 1. Startup.cs (for .NET Core):** Acts as the entry point for app configuration.

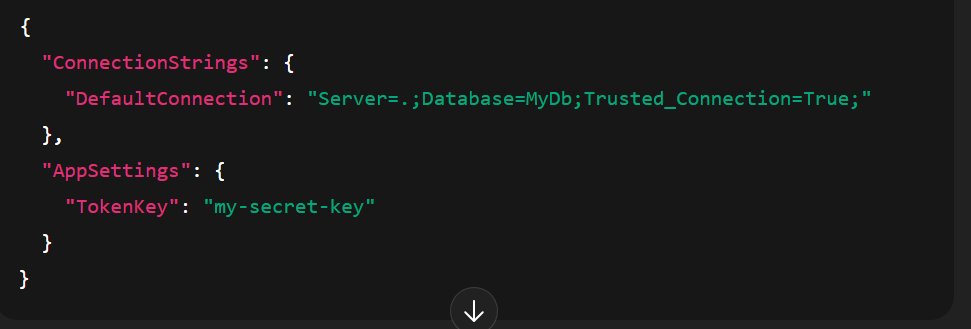
**Contains two main methods:**

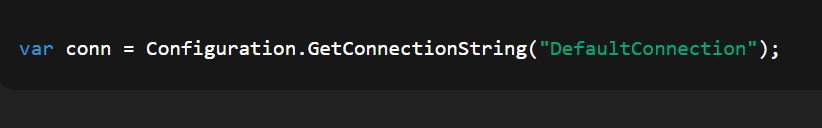
* ConfigureServices(IServiceCollection services) – Used for dependency injection (DI).
* Configure(IApplicationBuilder app, ...) – Used to set up middleware (like routing, CORS, etc.).



**2)appSettings.json:   
-**Used to store application settings, like connection strings, keys, and custom configure values.

-Replaces traditional web.config in .NET Core.





**3. launchSettings.json:**

- Used only in development to configure how the app starts (e.g., ports, environment).

- Contains profiles for IIS Express, Kestrel, etc.  


**4. Route.config (in .NET Framework 4.5)**

-A separate configure file for defining custom routing rules in Web Forms or legacy APIs.  
-Defines how URLs map to handlers/controllers.

**5) WebAPI.config / Web.config (in .NET Framework 4.5)**

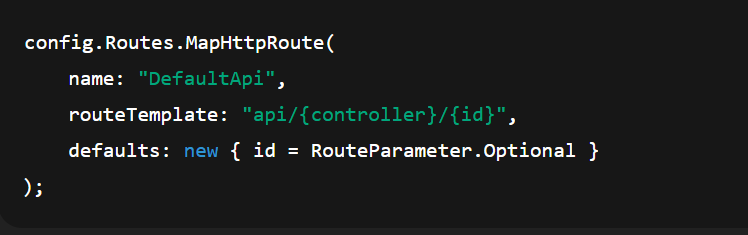
-Contains all major configurations:

-Routing (in WebApiConfig.cs)

-Dependency injection

-Handler and filter registrations

- Connection strings and app settings (within web.config XML)





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

**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}")]

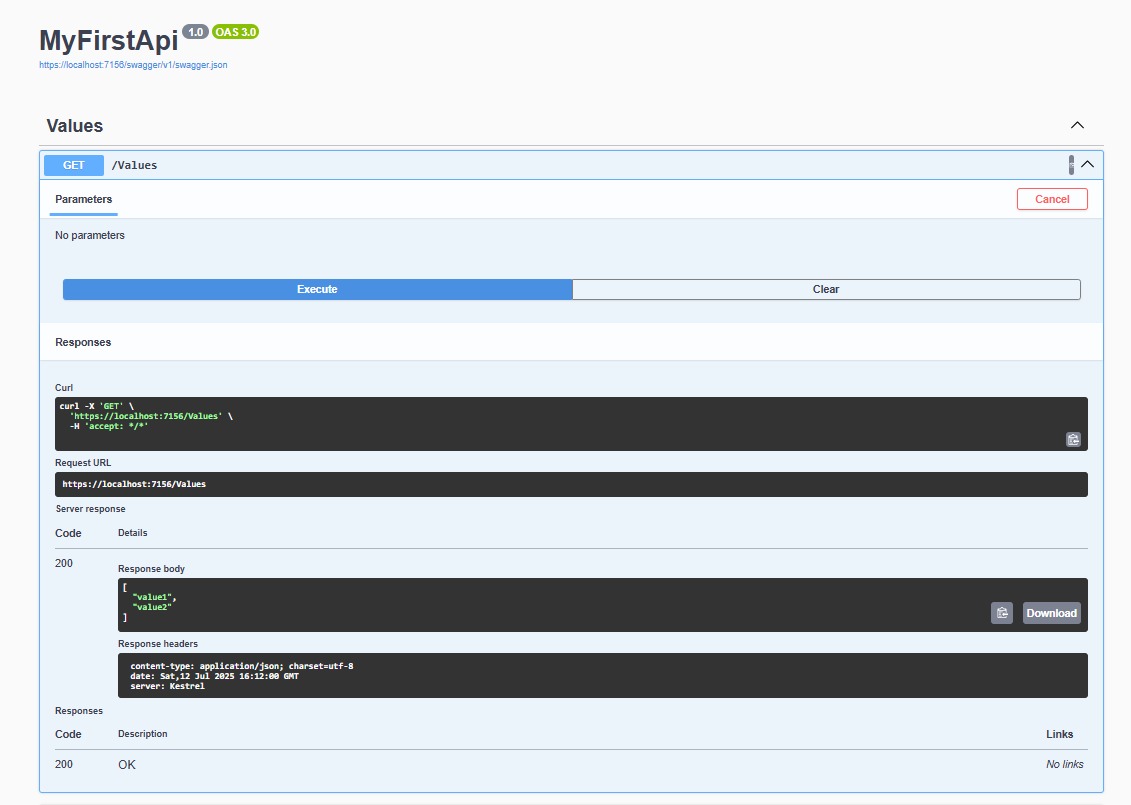
public void Delete(int id)

{

}

}

}

**Output:**

1. **WebApi\_Handson.docx  
     
   2)Swagger Integration & Postman**

**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"

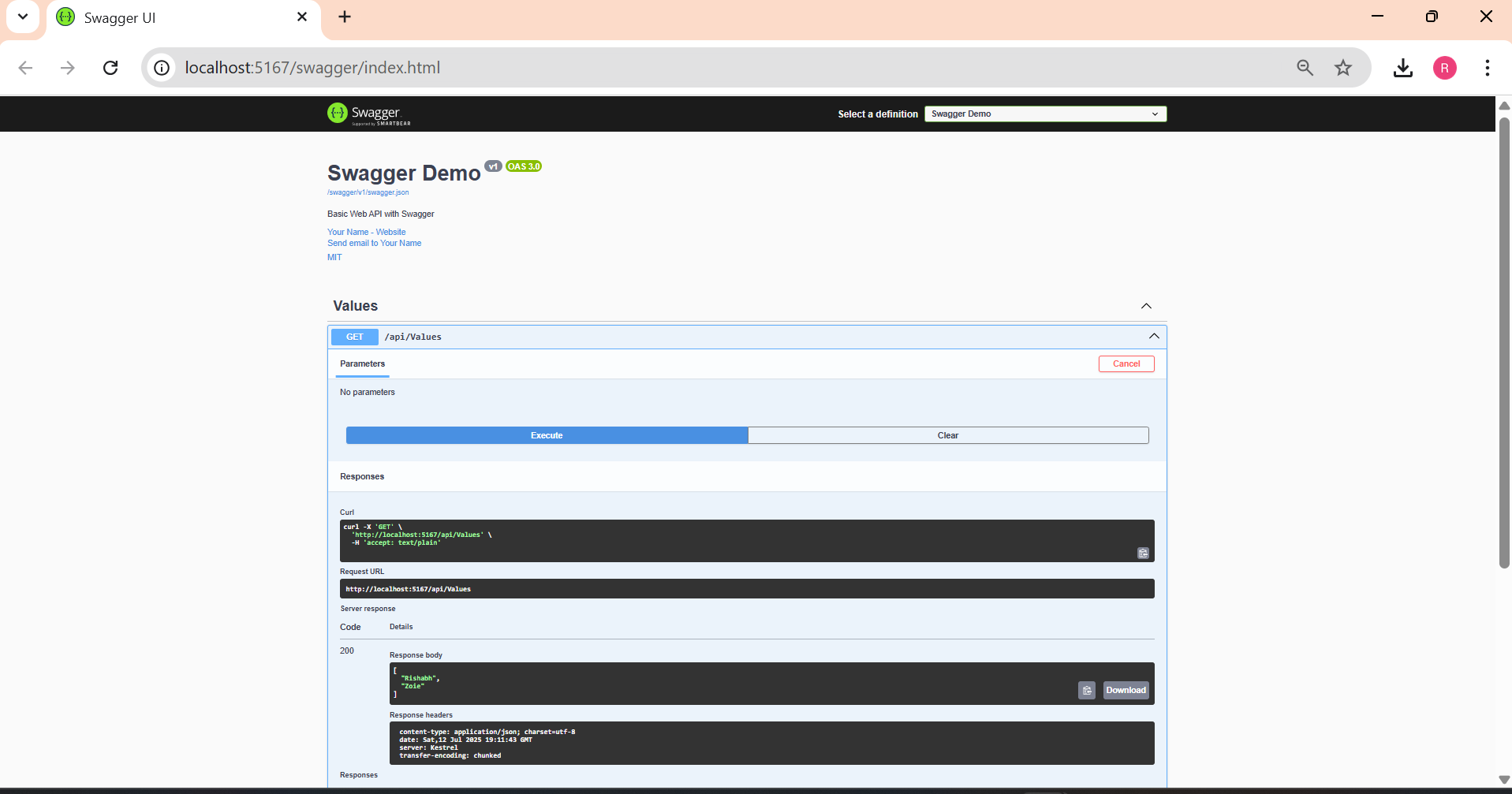
}

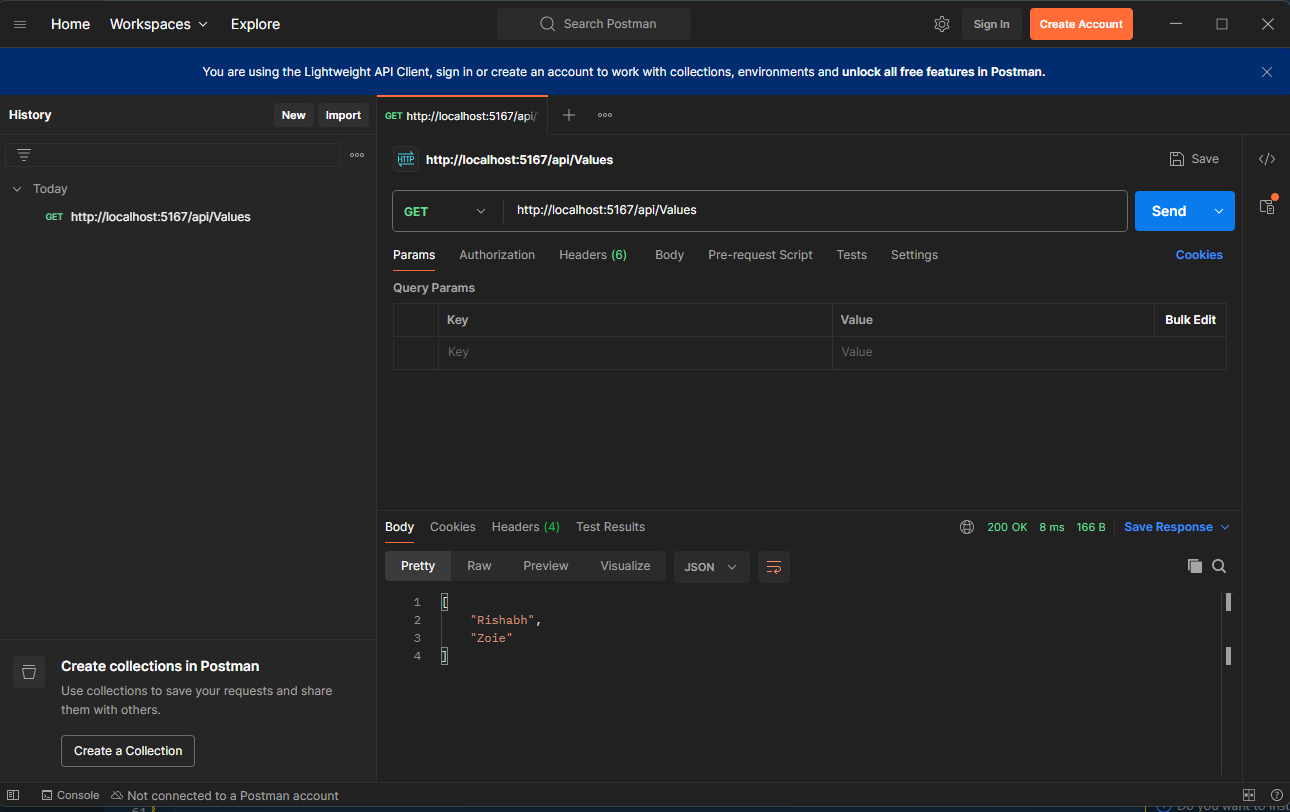
}

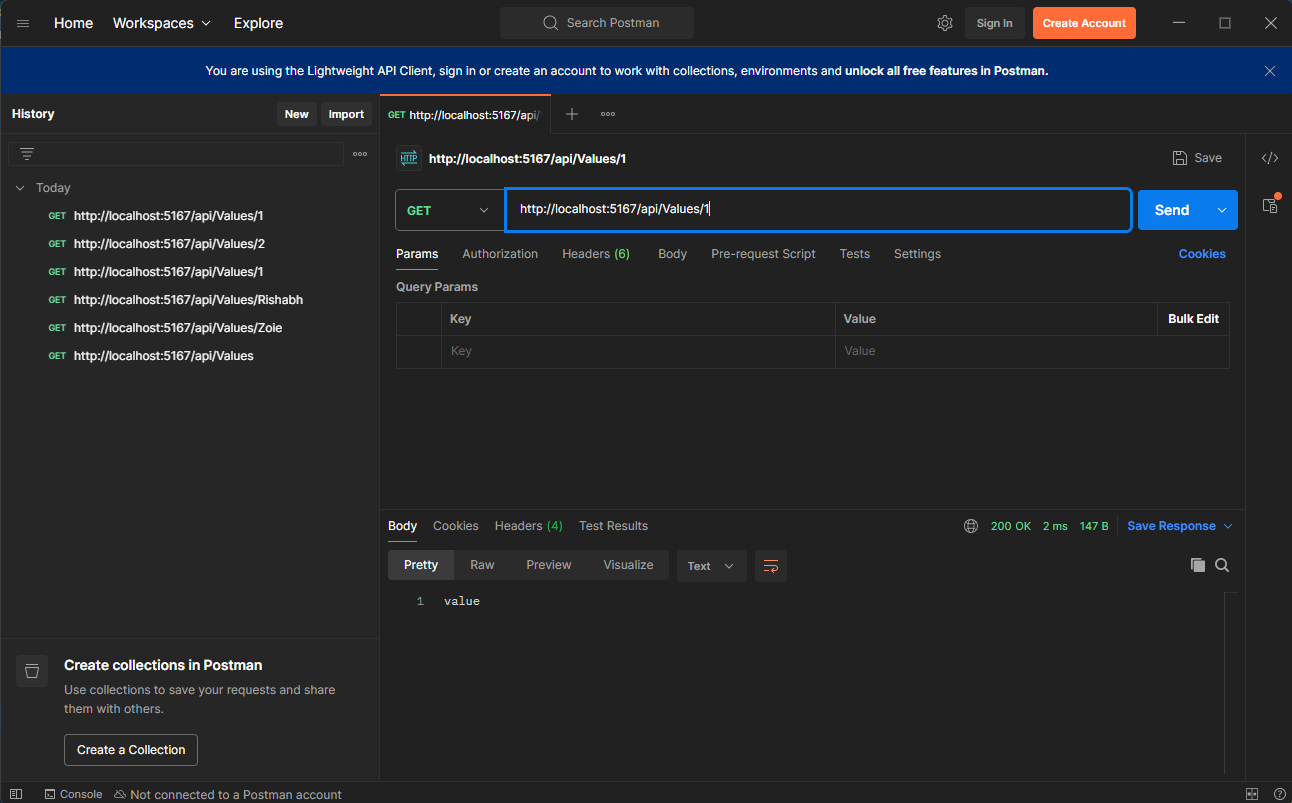
}

}

**Postman Output:**







1. **WebApi\_Handson.docx**
2. **Custom Models & Filters**

**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()

{

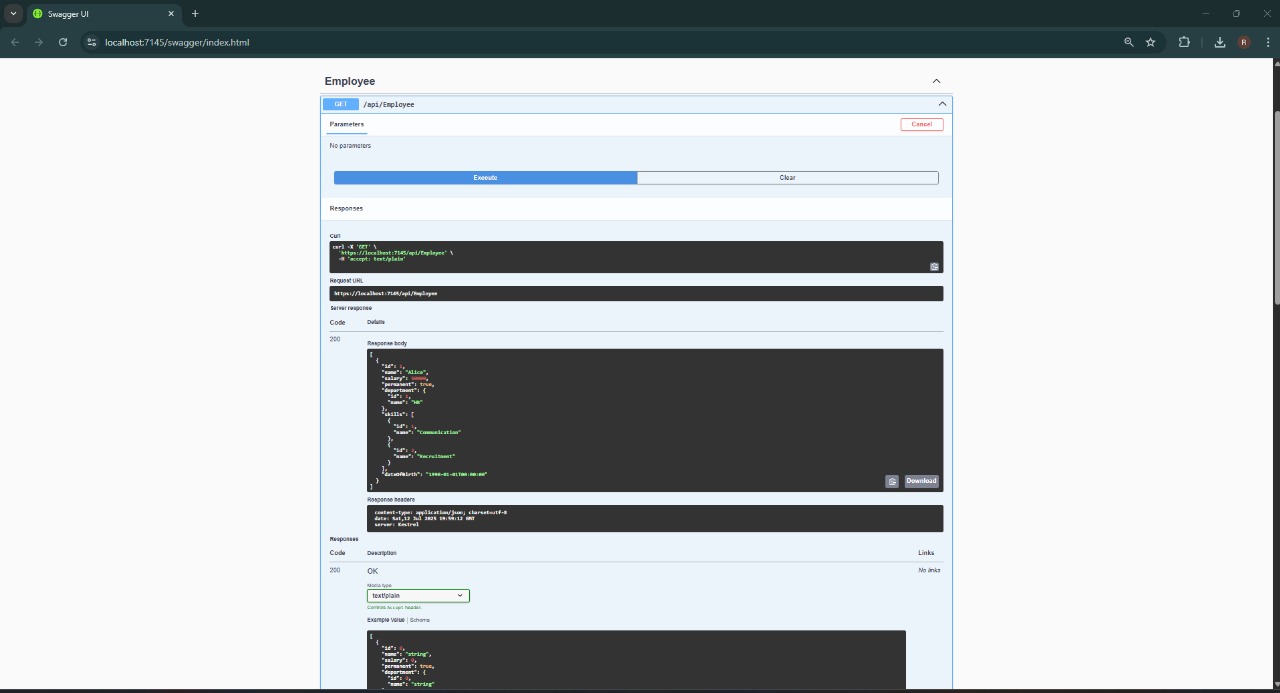
throw new Exception("Test exception");

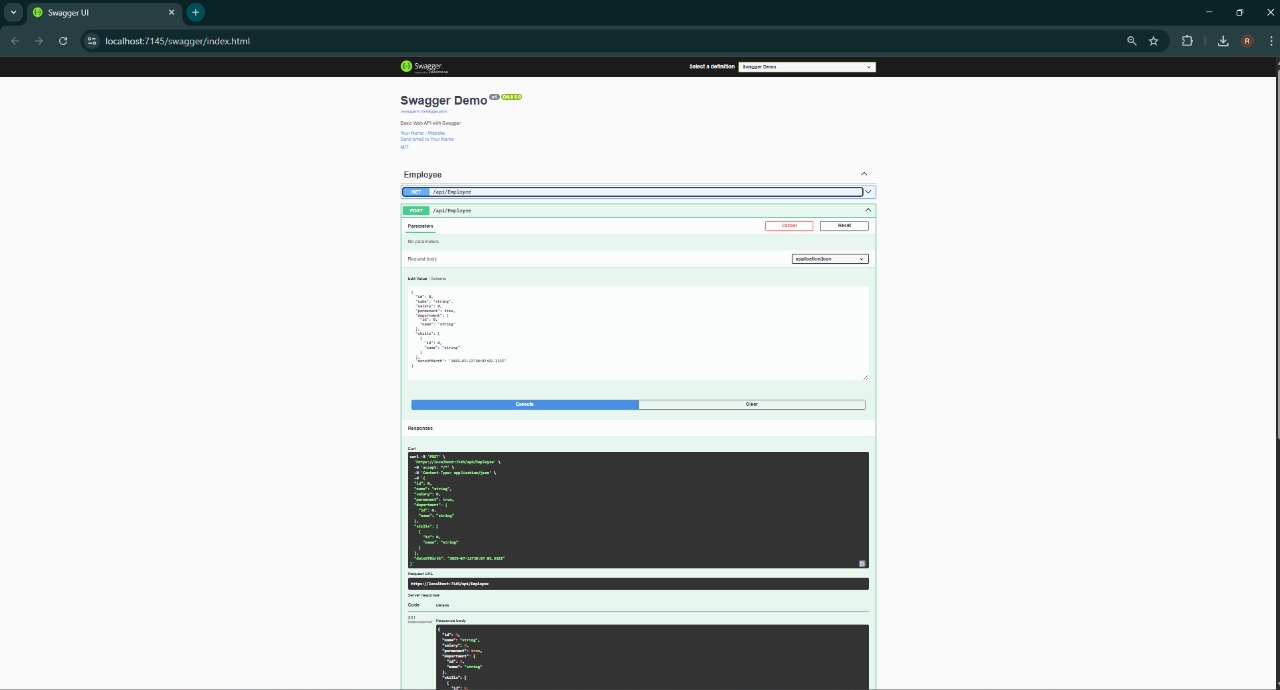
}

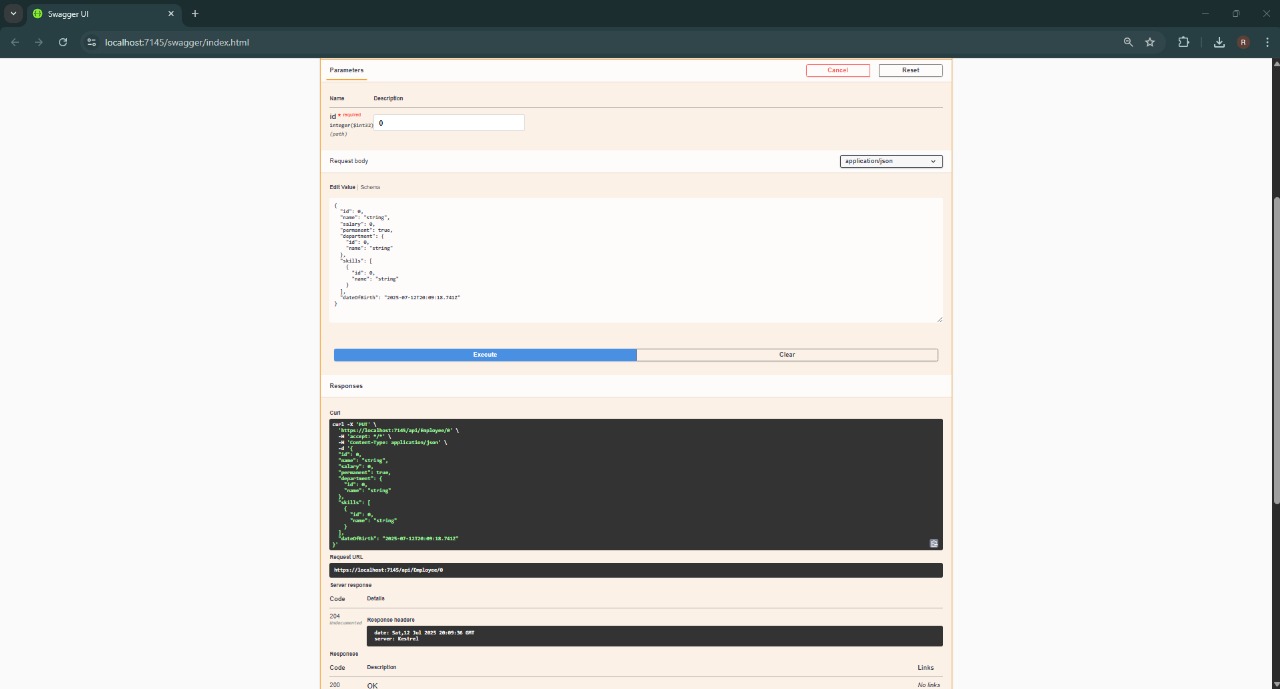
}

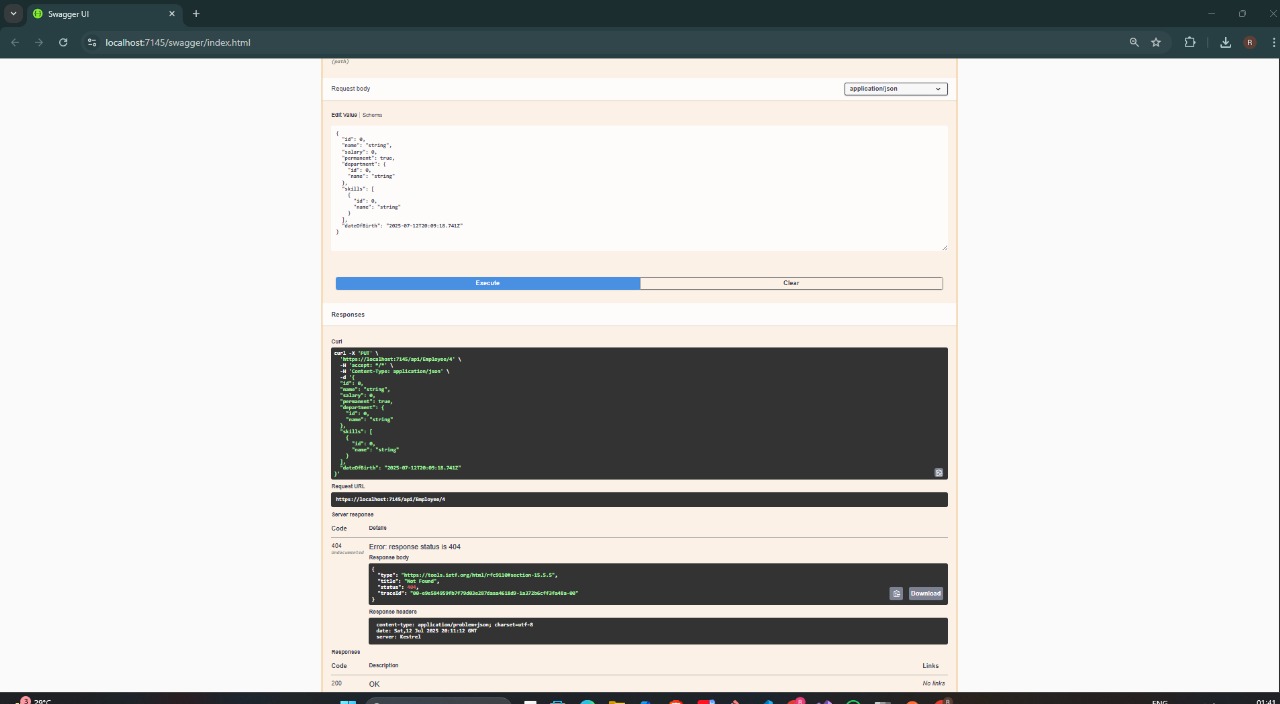
}

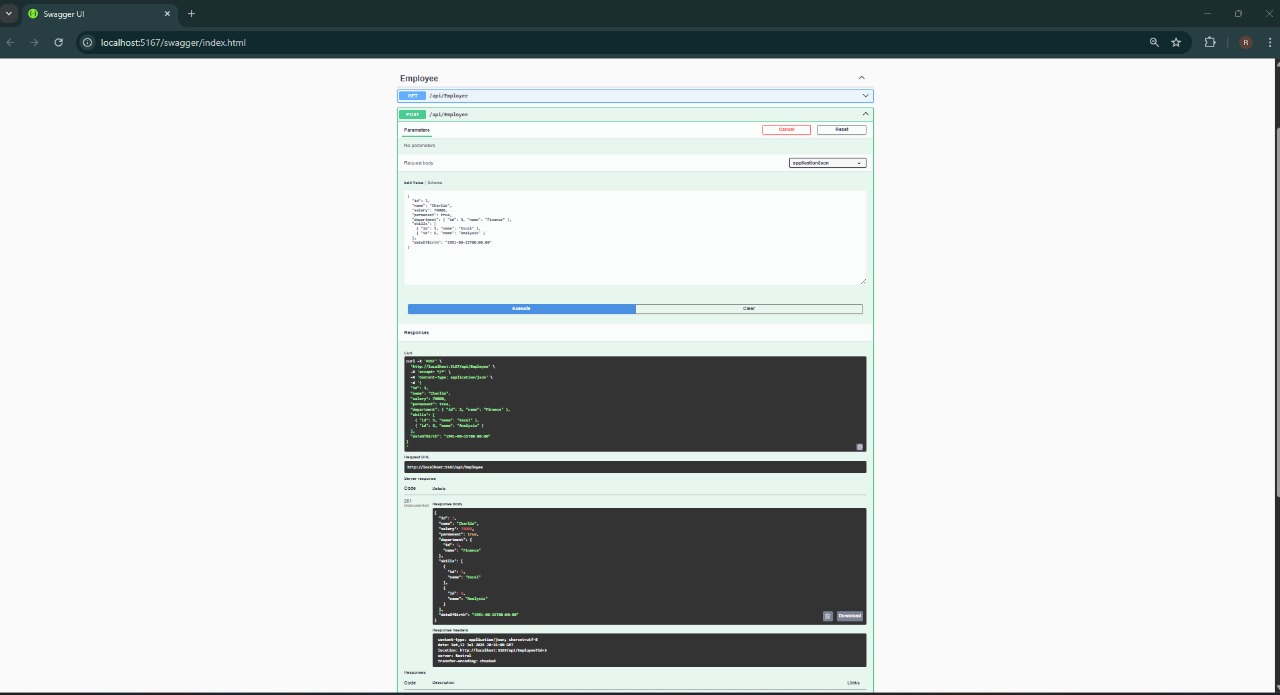
**Output:**











1. **WebApi\_Handson.docx**

**3)CRUD Operations**

**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");

});

}

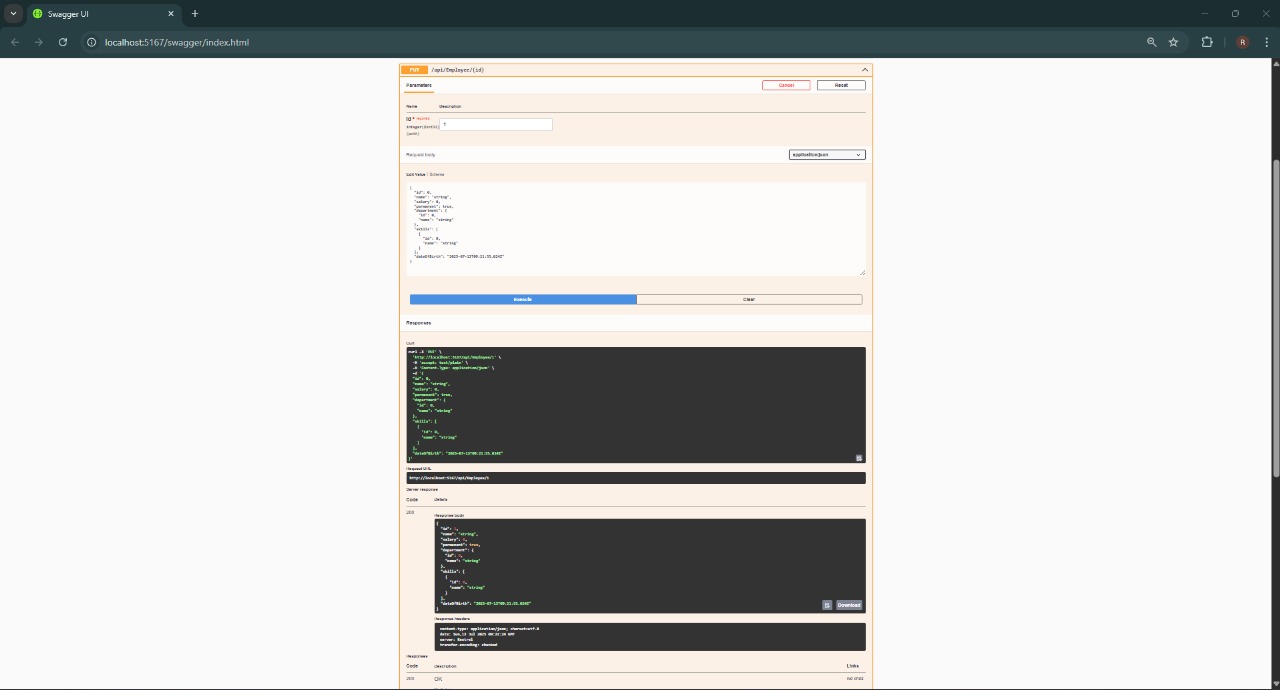
app.UseHttpsRedirection();

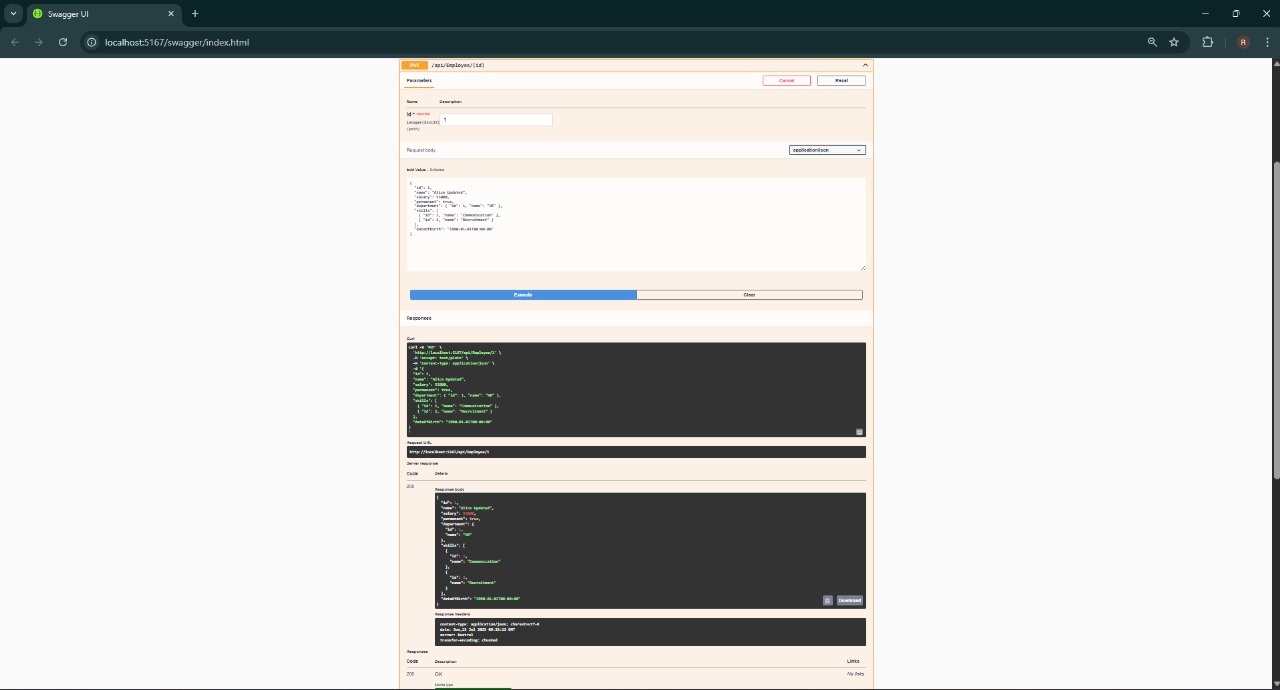
app.UseAuthorization();

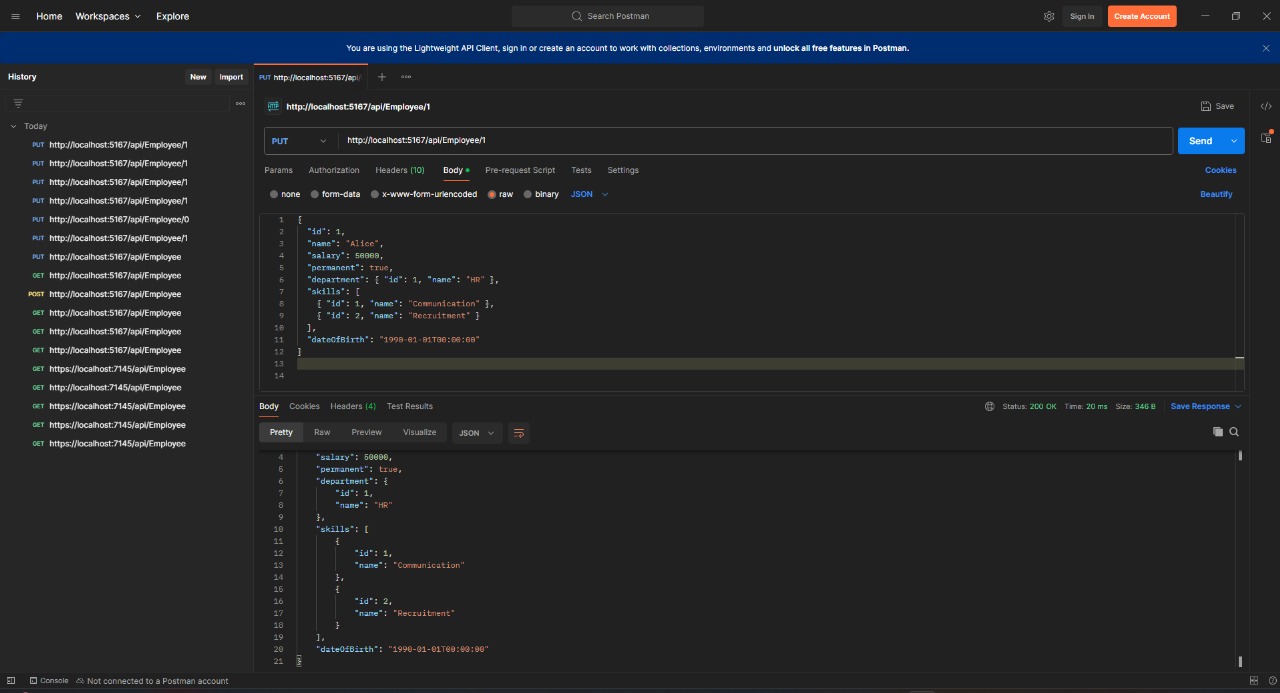
app.MapControllers();

app.Run();

**Output:**







1. **WebApi\_Handson.docx**

**4)JWT Authentication & Role-Based Access  
  
  
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:**

