**Author - Utsav Saxena**  **Superset Id - 6361856**

**WEEK 4 - WEB-API HANDSON**

**Module - 1.WebApi HandsOn**

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

**CODE**

**/controllers/valueController.cs**

[ApiController]

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

public class ValuesController : ControllerBase

{

[HttpGet]

public IActionResult Get() => Ok(new[] { "Utsav", "6361856" });

[HttpPost]

public IActionResult Post([FromBody] string value) => Ok($"Received: {value}");

}

**/program.cs**

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

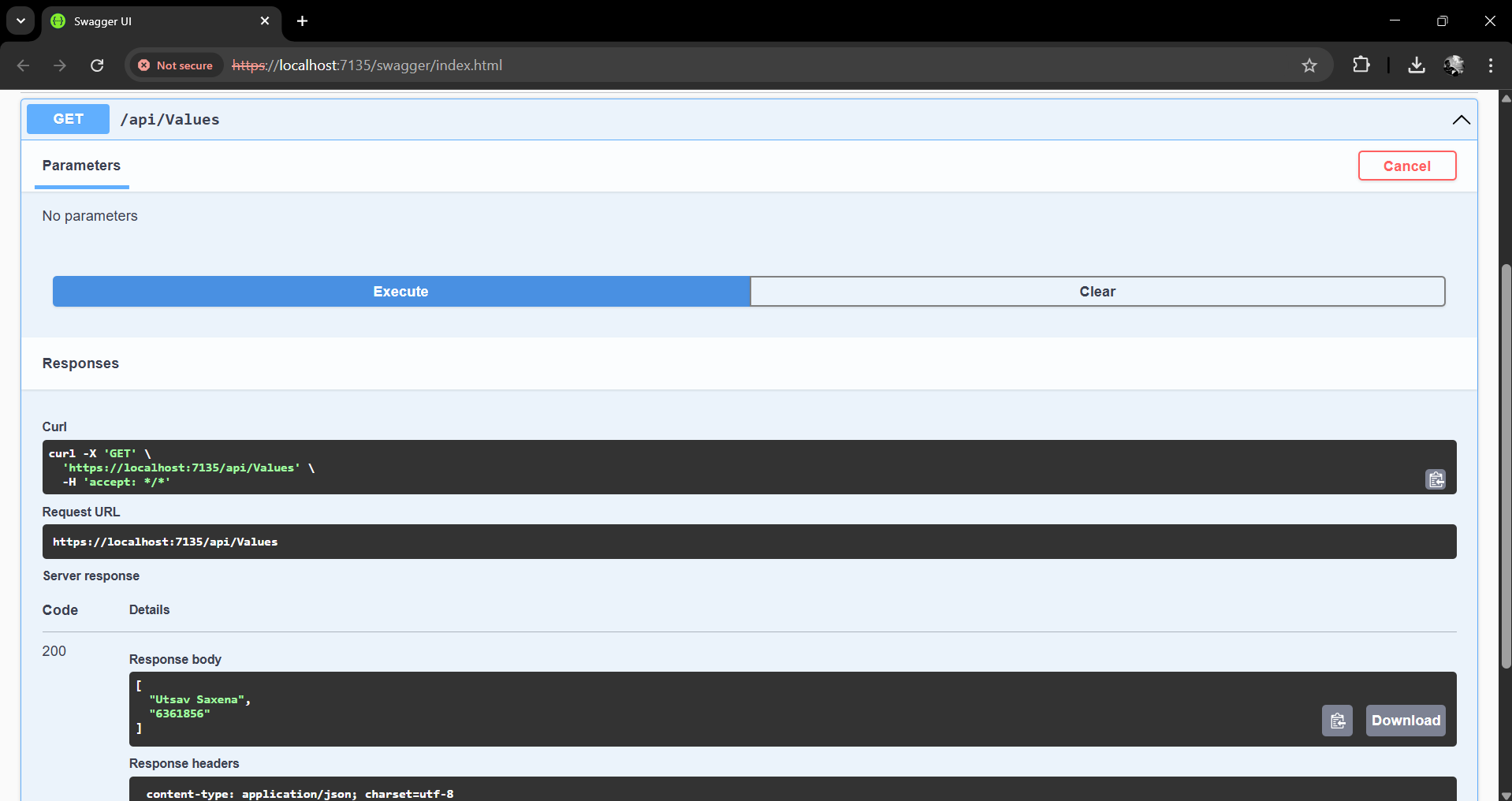
app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

**OUTPUT**

****

**Module - 2.WebApi HandsOn**

**Question 1:Web Api using .Net core with Swagger**

**CODE**

**/program.cs**

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new Microsoft.OpenApi.Models.OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "TBD",

TermsOfService = new Uri("https://example.com/terms"),

Contact = new Microsoft.OpenApi.Models.OpenApiContact

{

Name = "John Doe",

Email = "john@xyzmail.com",

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

},

License = new Microsoft.OpenApi.Models.OpenApiLicense

{

Name = "License Terms",

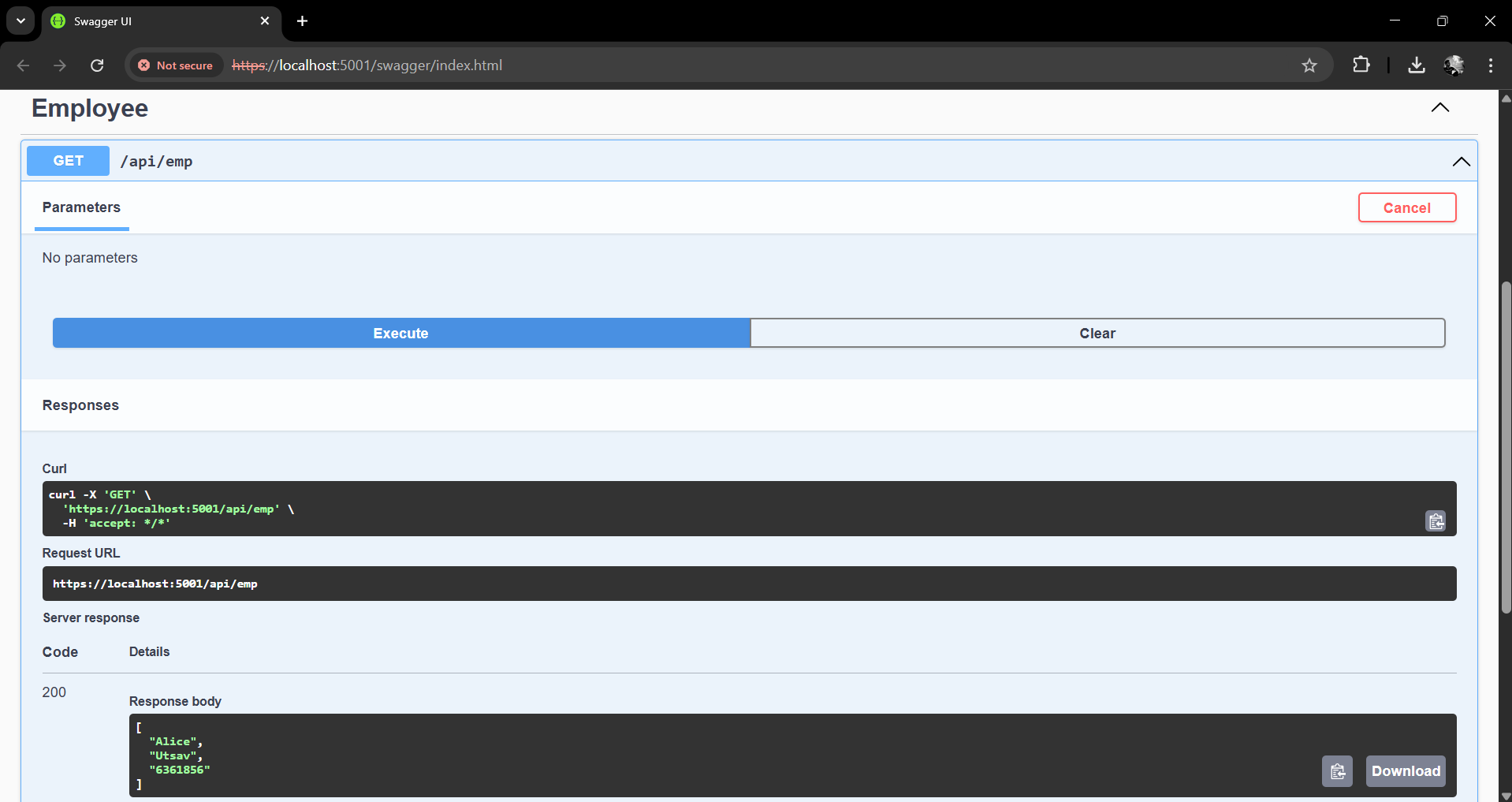
Url = new Uri("https://www.example.com")

}

});

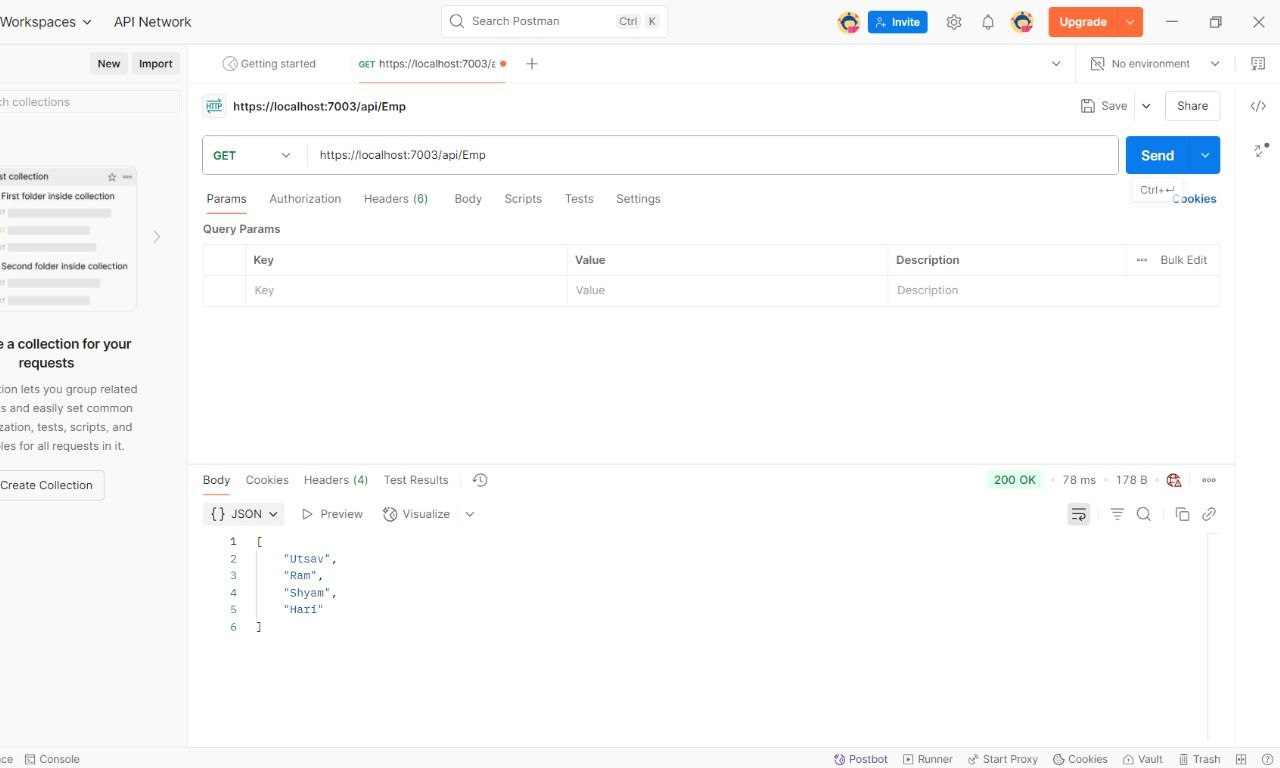
});

**OUTPUT**

****

**Question 2:Test the GET action method using POSTMAN**

**OUTPUT**

****

**Question 3:Modify the Controller name in the Route attribute of the Employee controller to ‘Emp’ and check its access thru POSTMAN**

**Answer**

Yes, if we use exact modified route.

**Module - 3.WebApi HandsOn**

**Question 1:Web Api using custom model class**

**CODE**

**/Models/Employee.cs**

namespace YourProject.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; }

}

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

}

public class Skill

{

public int Id { get; set; }

public string Name { get; set; }

}

}

**/Controllers/EmployeeController.cs**

using Microsoft.AspNetCore.Mvc;

using YourProject.Models;

using YourProject.Filters;

namespace YourProject.Controllers

{

[ApiController]

[Route("api/emp")]

[ServiceFilter(typeof(CustomAuthFilter))]

public class EmployeeController : ControllerBase

{

private static List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John Doe",

Salary = 50000,

Permanent = true,

DateOfBirth = new DateTime(1990, 1, 1),

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

Skills = new List<Skill>

{

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

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

}

}

};

}

[HttpGet]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public ActionResult<List<Employee>> Get()

{

throw new Exception("Simulated Exception");

}

[HttpPost]

public ActionResult Post([FromBody] Employee emp)

{

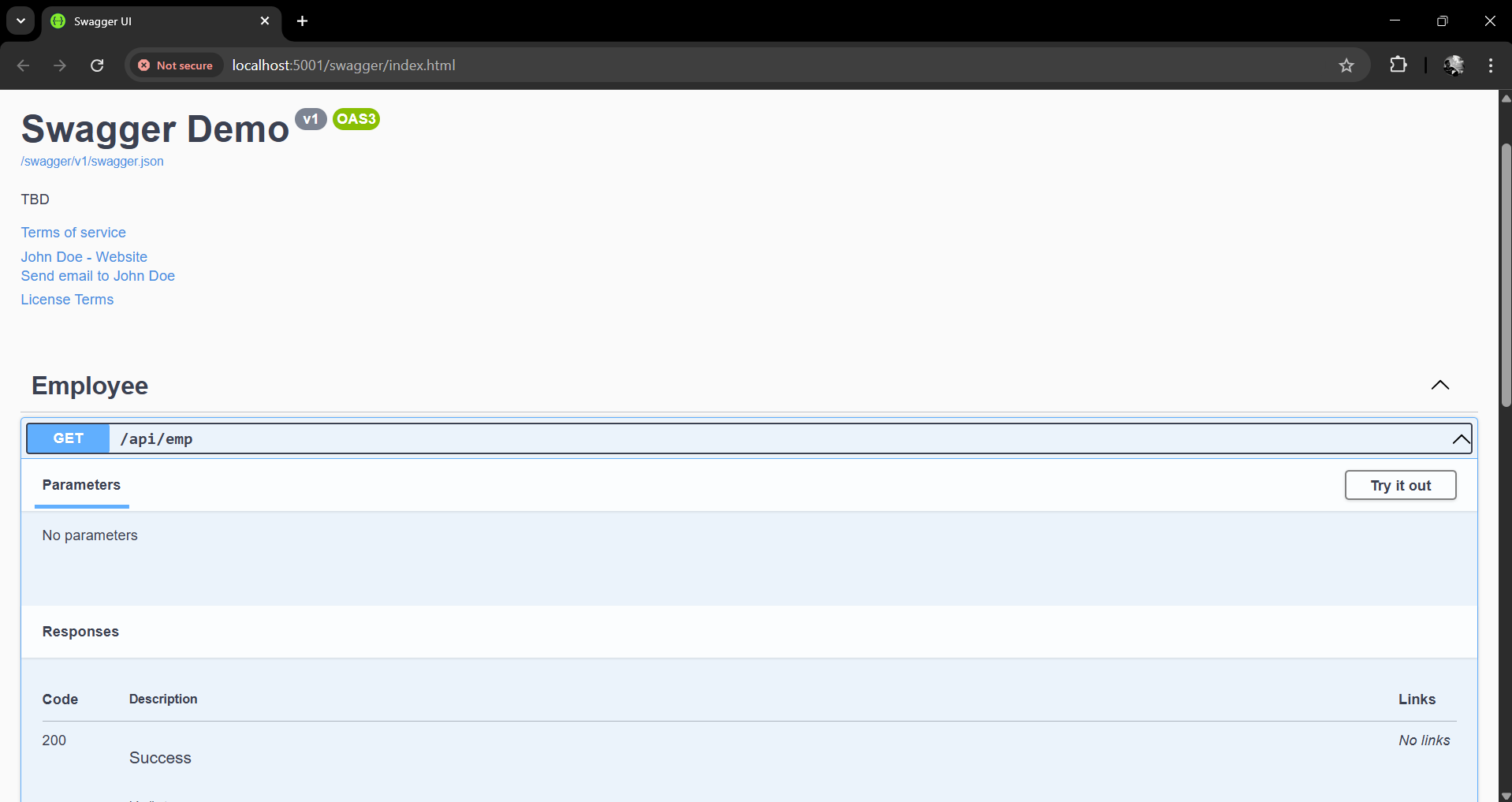
return Ok($"Received: {emp.Name}");

}

}

}

**OUTPUT**

****

**Question 2:Create a Custom action filter for Authorization.**

**CODE**

**/Filter/CustomActionFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace YourProject.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

var headers = context.HttpContext.Request.Headers;

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

{

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

return;

}

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

if (!token.Contains("Bearer"))

{

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

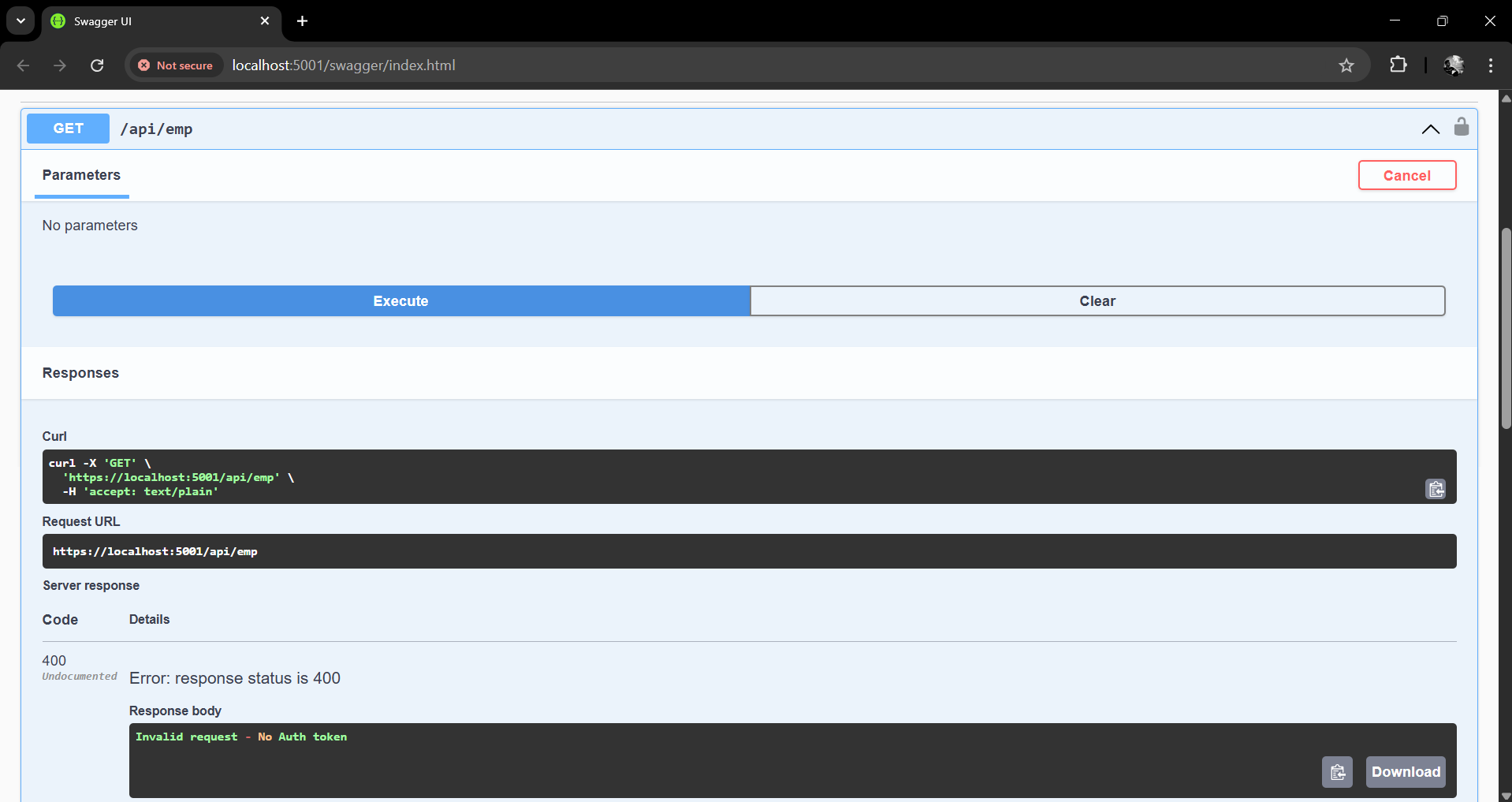
}

}

}

}

**OUTPUT**

****

**Question 3:Create a Custom Exception filter**

**CODE**

**/Filter/CustomExceptionFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace YourProject.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

var exception = context.Exception;

File.AppendAllText("errors.txt", $"{DateTime.Now}: {exception.Message}{Environment.NewLine}");

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

{

StatusCode = 500

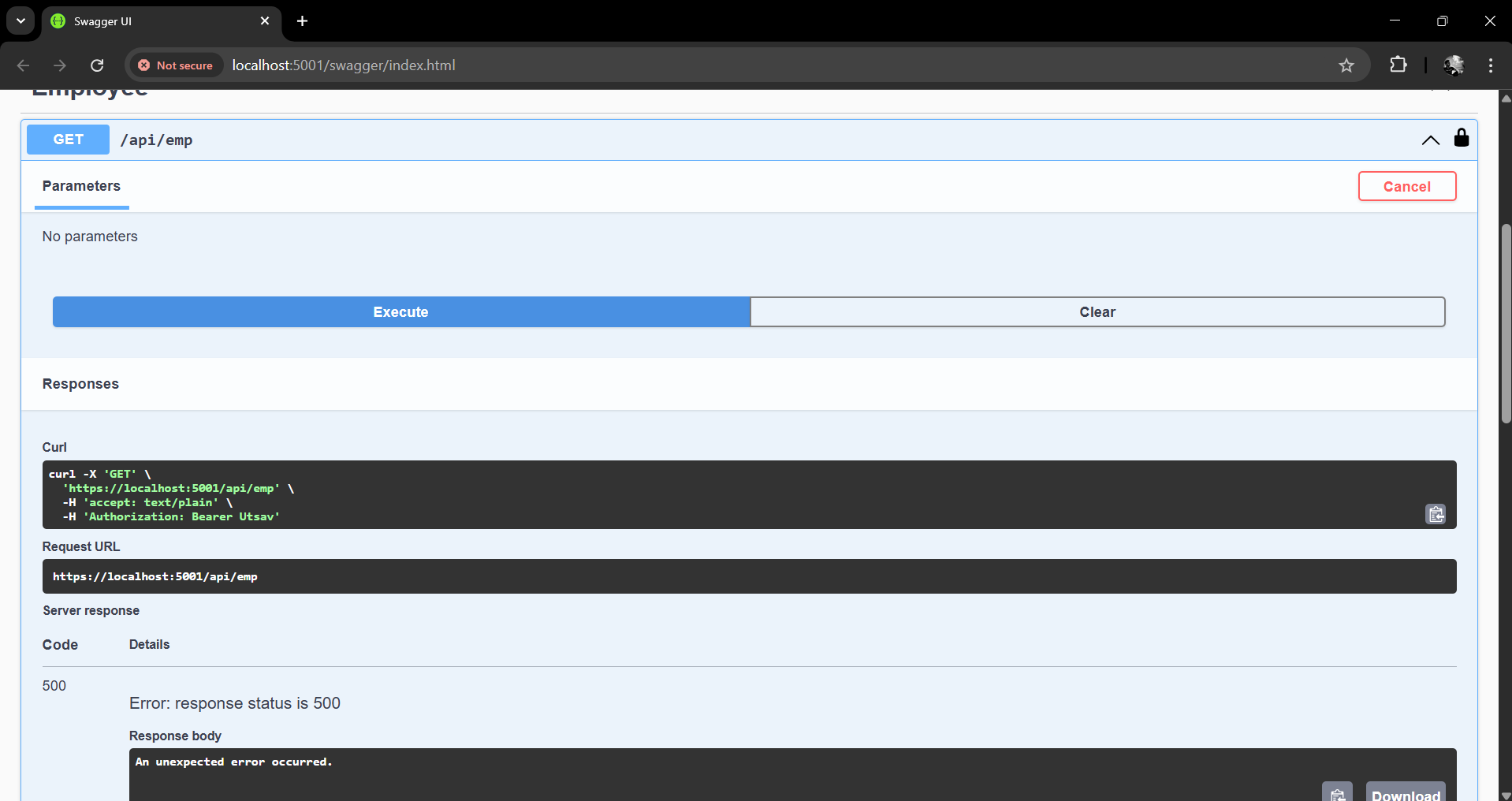
};

}

}

}

**OUTPUT**

****

**Module - 4.WebApi HandsOn**

**Question 1:Web Api CRUD operation**

**CODE**

**/Controller/EmployeeController.cs**

using Microsoft.AspNetCore.Mvc;

using YourProject.Models;

using YourProject.Filters;

namespace YourProject.Controllers

{

[ApiController]

[Route("api/emp")]

[ServiceFilter(typeof(CustomAuthFilter))]

public class EmployeeController : ControllerBase

{

// Simulate DB with static list

private static List<Employee> employees = new List<Employee>

{

new Employee

{

Id = 1,

Name = "John Doe",

Salary = 50000,

Permanent = true,

DateOfBirth = new DateTime(1990, 1, 1),

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

Skills = new List<Skill> { new Skill { Id = 1, Name = "C#" } }

},

new Employee

{

Id = 2,

Name = "Jane Smith",

Salary = 60000,

Permanent = false,

DateOfBirth = new DateTime(1992, 2, 2),

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

Skills = new List<Skill> { new Skill { Id = 2, Name = "Excel" } }

}

};

// GET all employees

[HttpGet]

[ProducesResponseType(StatusCodes.Status200OK)]

public ActionResult<List<Employee>> Get()

{

return Ok(employees);

}

// PUT: Update an employee

[HttpPut("{id}")]

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

{

if (id <= 0)

return BadRequest("Invalid employee id");

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

if (emp == null)

return BadRequest("Invalid employee id");

// Update the employee details

emp.Name = updatedEmp.Name;

emp.Salary = updatedEmp.Salary;

emp.Permanent = updatedEmp.Permanent;

emp.DateOfBirth = updatedEmp.DateOfBirth;

emp.Department = updatedEmp.Department;

emp.Skills = updatedEmp.Skills;

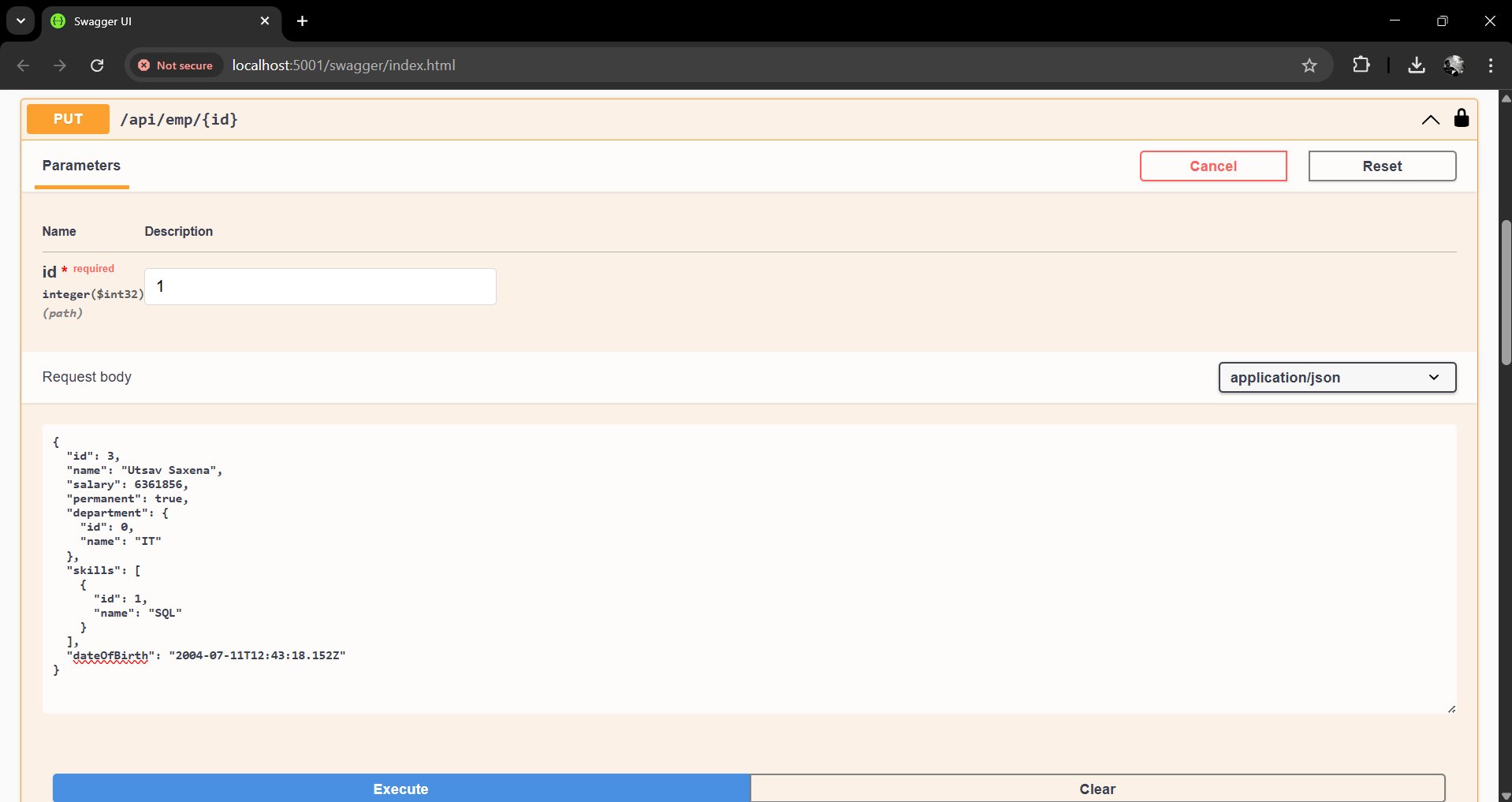
return Ok(emp);

}

}

}

**OUTPUT**

****

****

**Module - 5.WebApi HandsOn**

**Question 1:JsonWebToken**

**CODE**

**/program.cs**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

string securityKey = "mysuperdupersecret";

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

builder.Services.AddAuthentication(x =>

{

x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(x =>

{

x.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = "mySystem",

ValidAudience = "myUsers",

IssuerSigningKey = symmetricSecurityKey

};

});

**/Controllers/AuthCOntroller.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace YourProject.Controllers

{

[ApiController]

[Route("api/auth")]

[AllowAnonymous]

public class AuthController : ControllerBase

{

[HttpGet("token")]

public IActionResult GetToken()

{

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

return Ok(token);

}

private string GenerateJSONWebToken(int userId, string userRole)

{

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

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 expires in 2 mins

signingCredentials: credentials

);

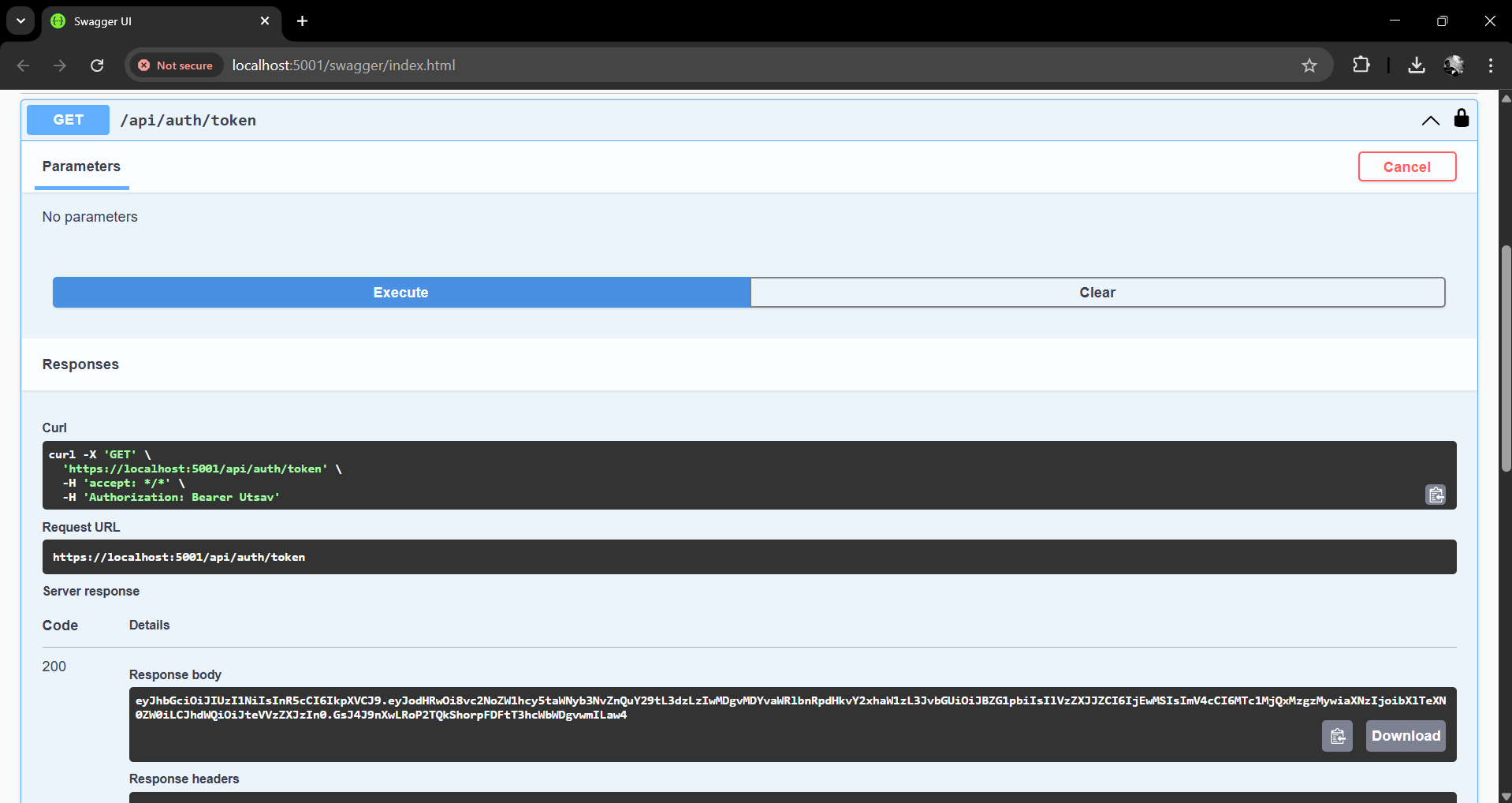
return new JwtSecurityTokenHandler().WriteToken(token);

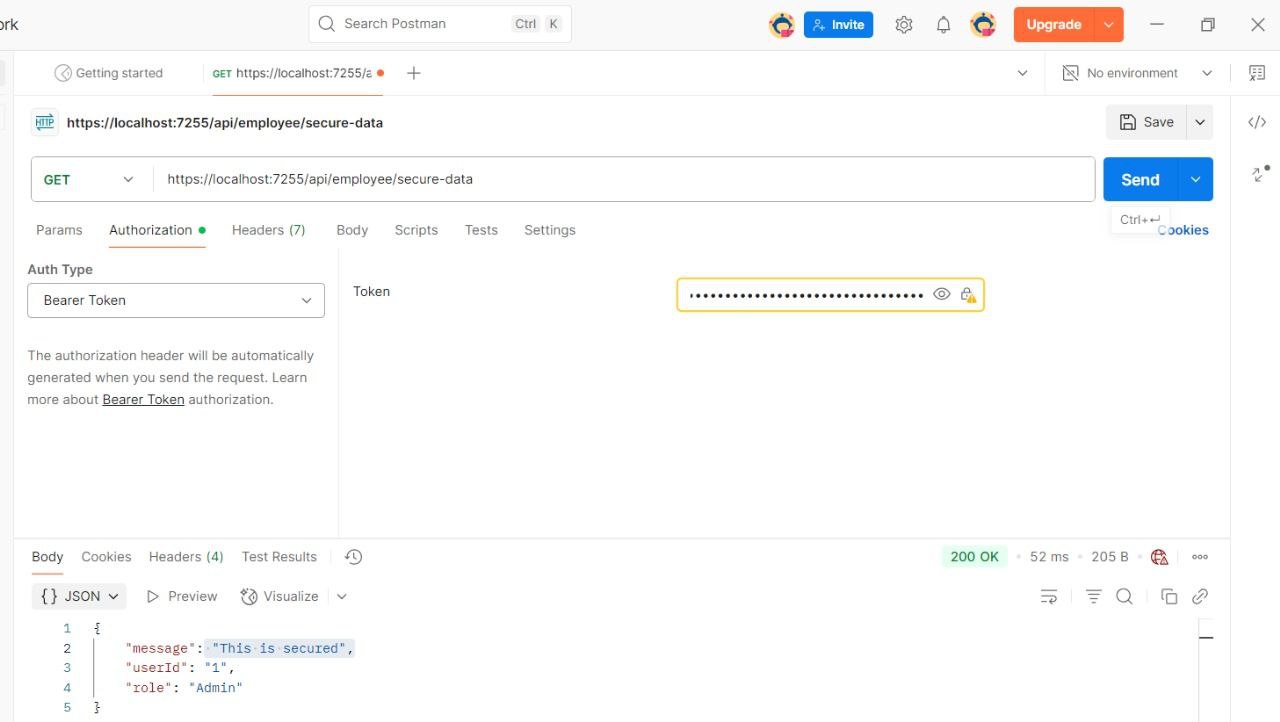
}

}

}

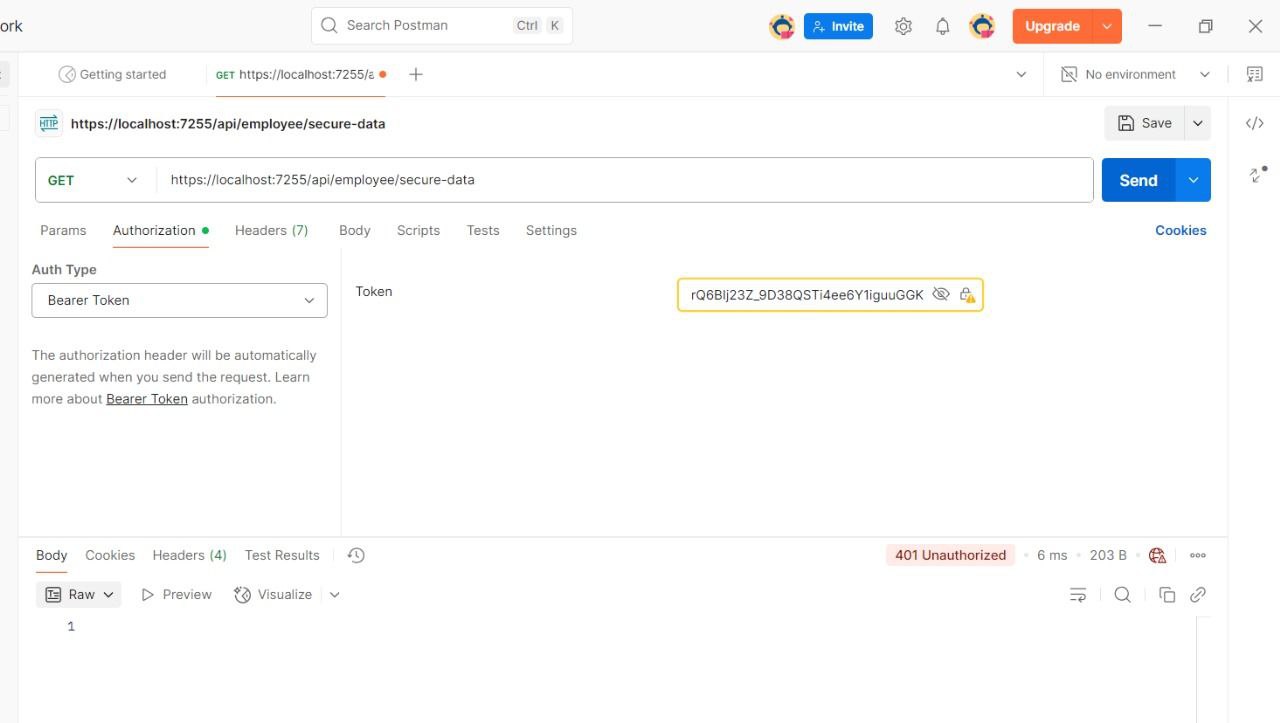
**OUTPUT**

****

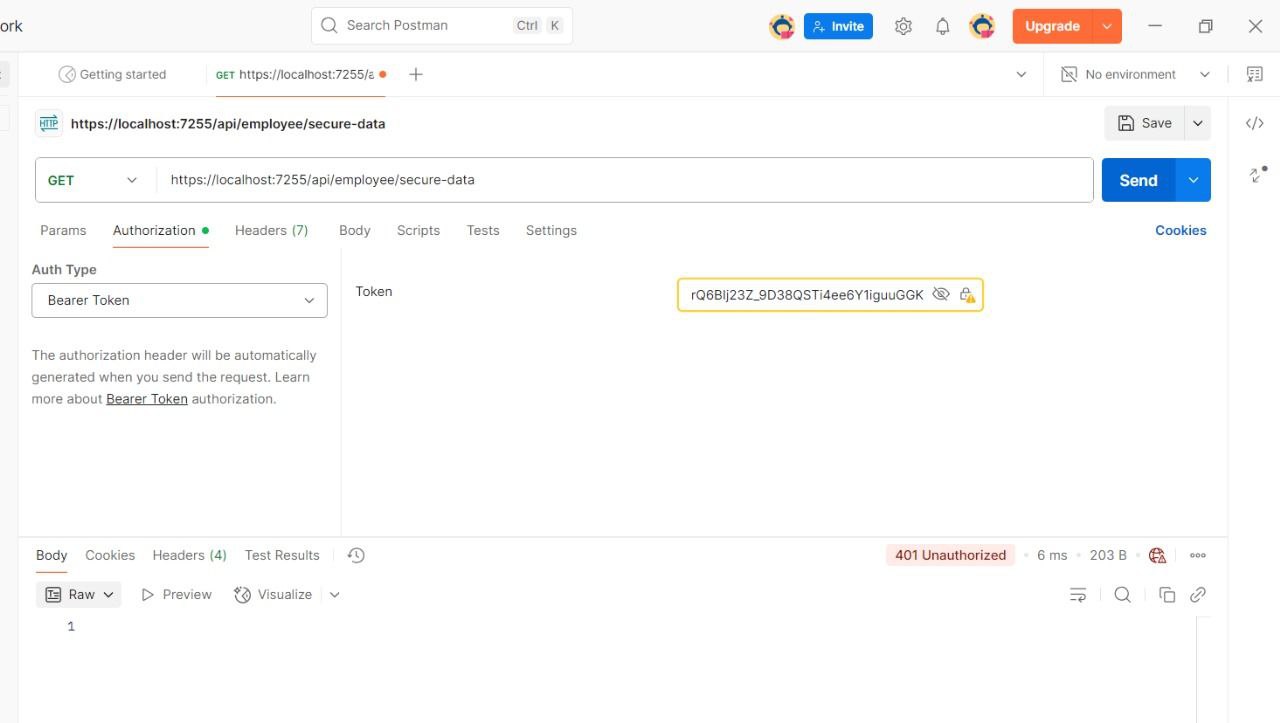
****

**Question 2:Use the JWT generated thru the AuthController to be used in POSTMAN request.**

**OUTPUT**

****

**Question 3:Check for JWT expiration**

****

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*