# WebApi Exercise Solutions

## Exercise - 1: First WebApi using .NET Core

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

* **RESTful Web Service**: A web service based on REST (Representational State Transfer) principles, using HTTP methods (GET, POST, etc.) to perform operations on resources (like products or users).
* **Web API**: A software interface that allows different systems to communicate over the internet using HTTP. It allows clients to access services/data from a server.
* **Microservice**: A small, independent service that performs a specific business function and communicates with other services via APIs. It helps in building scalable and modular applications.

### 2. Features of REST Architecture

* **Representational State Transfer**: Each resource is identified by a URI and can be represented in formats like JSON or XML.
* **Stateless**: No session is stored on the server between requests.
* **Messages**: Communication between client and server is done using standard HTTP messages.
* **Microservice Concept**: REST APIs support loosely coupled services which form microservices.
* **Flexible Response Format**: Unlike traditional web services (which use XML), REST APIs can return JSON, XML, or plain text.

### 3. Difference Between WebService & WebAPI

|  |  |
| --- | --- |
| **WebService** | **Web API** |
| Returns only XML | Can return JSON, XML, etc. |
| Requires more configuration | Lightweight and easy |
| Uses SOAP | Uses REST or HTTP |
| Works only over HTTP | Can work over HTTP, HTTPS |

### 4. Explain what is HttpRequest & HttpResponse

* **HttpRequest**: Sent by the client to request data or perform actions (e.g., GET /products).
* **HttpResponse**: Sent by the server in reply to the client’s request, containing data and status code (e.g., 200 OK, or error message).

### 5. List the types of Action Verbs

|  |  |
| --- | --- |
| **HTTP Verb** | **Purpose** |
| **GET** | Retrieve data (Read) |
| **POST** | Create new resource |
| **PUT** | Update existing resource |
| **DELETE** | Remove resource |

Each is used by adding an attribute like [HttpGet], [HttpPost], etc. in the controller methods.

### 6. List the types of HttpStatusCodes used in WebAPI

|  |  |  |
| --- | --- | --- |
| **Status Code** | **Meaning** | **When to Use** |
| **200 OK** | Success | For successful GET, PUT |
| **201 Created** | New resource created | After successful POST |
| **400 BadRequest** | Client sent bad data | Validation fails |
| **401 Unauthorized** | No login or token | Auth required |
| **404 Not Found** | Resource missing | Wrong ID in GET/DELETE |
| **500 Internal Server Error** | Server crashed | Unhandled exception |

### 7. Demonstrate creation of a simple WebAPI - With Read, Write actions

**Steps to create:**

1. Create a new ASP.NET Core Web API project
2. Add a model (e.g., Product)
3. Add a controller (ProductsController) using [ApiController] and route [Route("products")]
4. Implement methods using [HttpGet], [HttpPost], etc.
5. Run the API and test using Postman

### 8. Structure of a Web API

* **Controller**: Handles API requests (e.g., ProductsController)
* **ApiController Inheritance**: Controller should inherit from ControllerBase and have [ApiController] attribute
* **Action Verbs**: [HttpGet], [HttpPost], etc. map to HTTP methods
* **Action Methods**: Functions that handle logic for each route (e.g., GetAll(), Create())

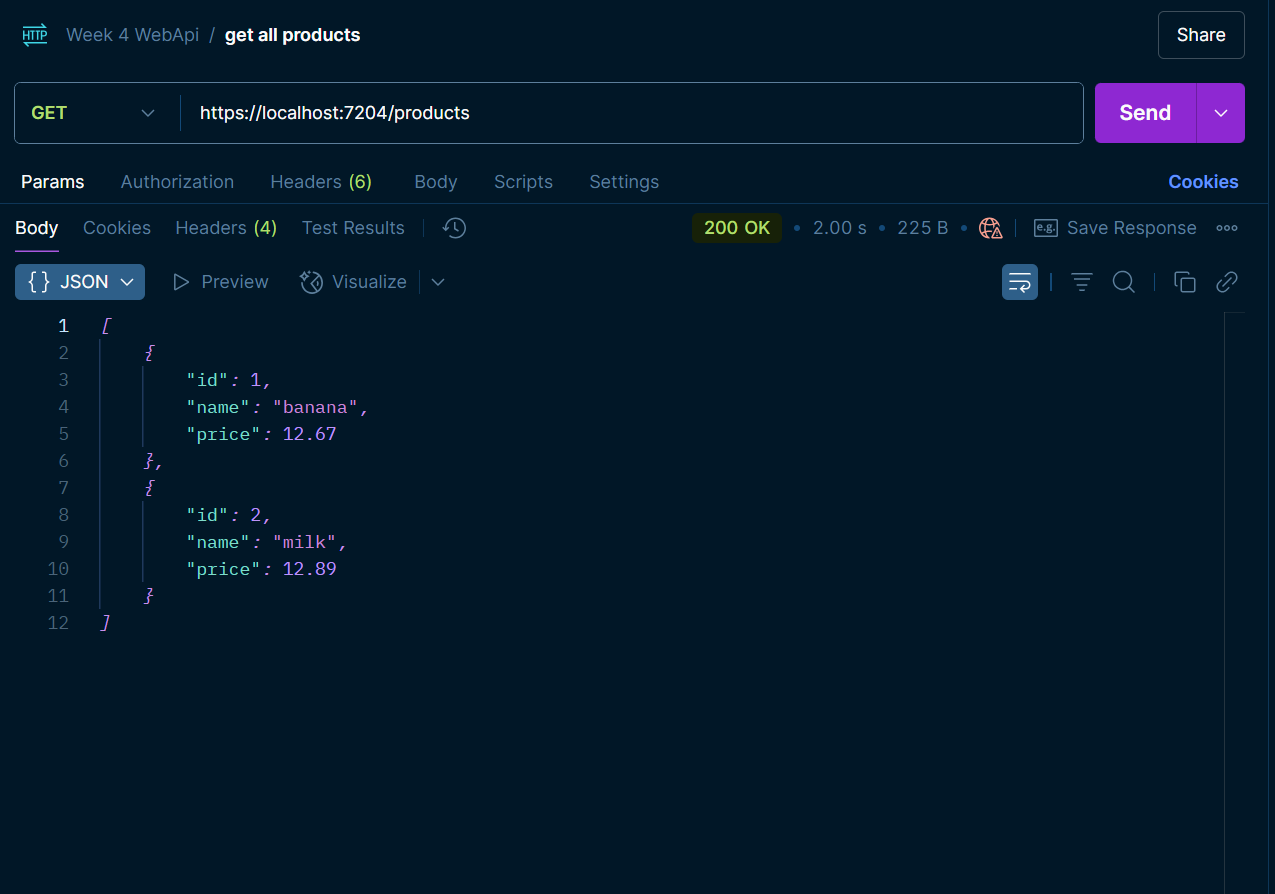
### 9. Explain the types of Configuration files of WebAPI

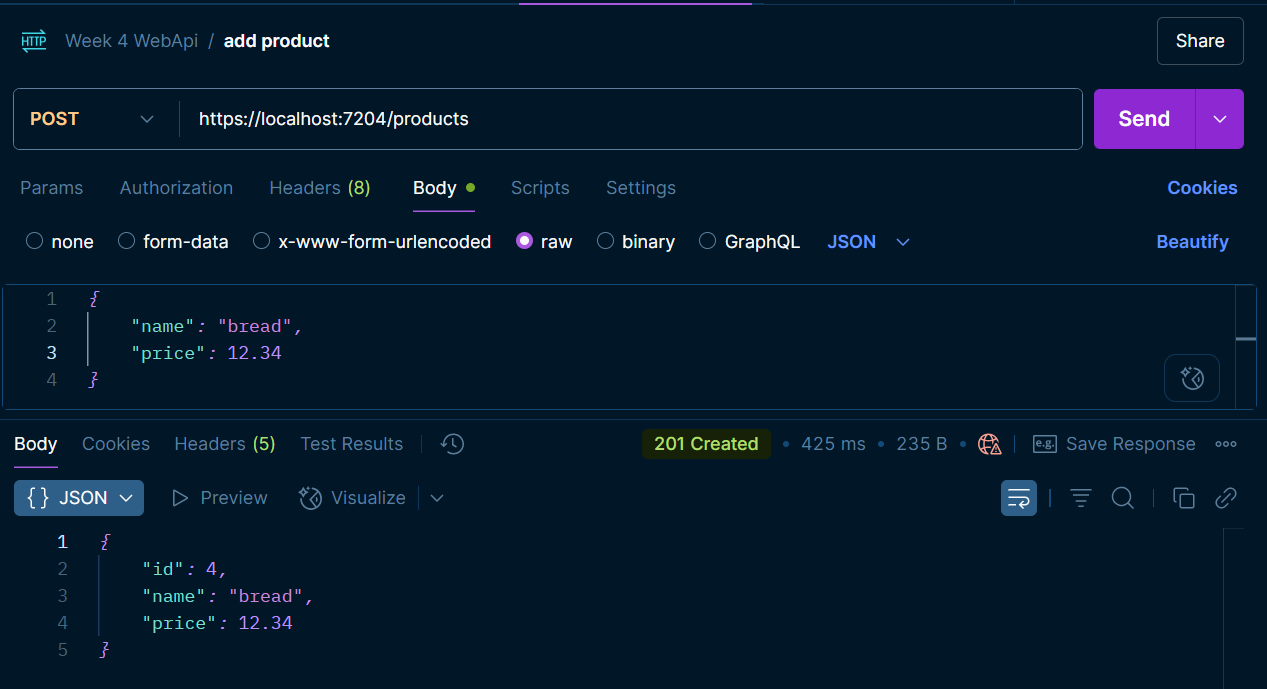
|  |  |
| --- | --- |
| **File** | **Purpose** |
| Program.cs | Register services, middleware, routing, Swagger |
| appsettings.json | Store app configuration like DB connection strings |
| launchSettings.json | Debugging setup like port and profile |
| Web.config (in .NET Framework) | Older format for IIS settings and connection strings |
| Route.config (used in .NET 4.5 WebAPI) | Setup custom routes in legacy projects |

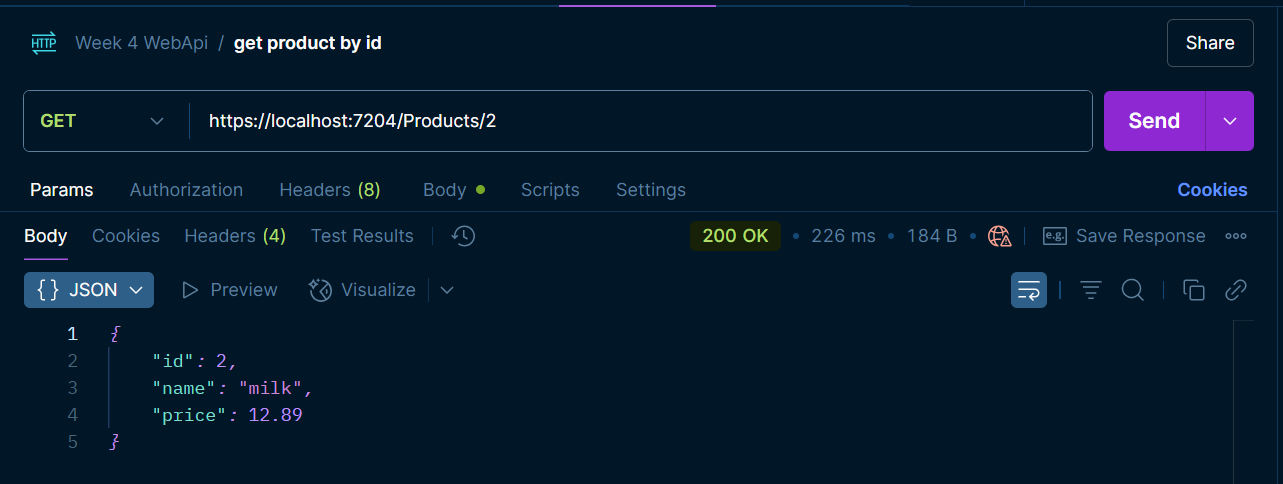
***Code:*** *ProductController.cs*

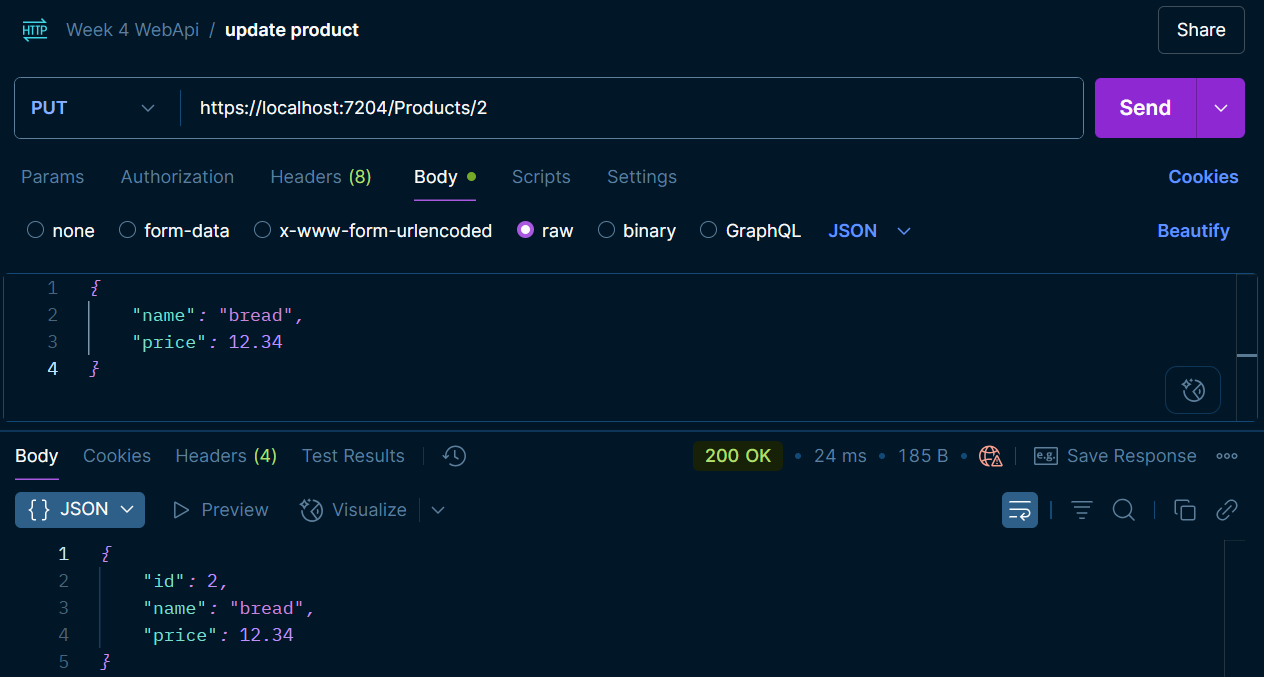
|  |
| --- |
| using Microsoft.AspNetCore.Mvc;  using Microsoft.EntityFrameworkCore;  using MyFirstWebAPI.Data;  using MyFirstWebAPI.DTO;  using MyFirstWebAPI.Entity;  using System.Collections.Generic;  using System.Linq;    namespace MyFirstWebAPI.Controller  {  [ApiController]  [Route("[controller]")]  public class ProductsController : ControllerBase  {  private readonly AppDbContext \_context;    public ProductsController(AppDbContext appDbContext)  {  \_context = appDbContext;  }    [HttpGet]  public IActionResult GetAll() => Ok(\_context.Products.ToList());    [HttpGet("{id}")]  public IActionResult GetById(int id)  {  var product = \_context.Products.Find(id);  return product == null ? NotFound() : Ok(product);  }    [HttpPost]  public IActionResult Create(ProductCreateDto productDto)  {  var product = new Product()  {  Name = productDto.Name,  Price = productDto.Price  };    \_context.Products.Add(product);  \_context.SaveChanges();    return CreatedAtAction(nameof(GetById), new { id = product.Id }, product);  }    [HttpPut("{id}")]  public IActionResult Update(int id, ProductCreateDto updatedProduct)  {  var product = \_context.Products.Find(id);  if (product == null) return NotFound();  product.Name = updatedProduct.Name;  product.Price = updatedProduct.Price;  \_context.SaveChanges();  return Ok(product);  }    [HttpDelete("{id}")]  public IActionResult Delete(int id)  {  var product = \_context.Products.Find(id);  if (product == null) return NotFound();  \_context.Products.Remove(product);  \_context.SaveChanges();  return NoContent();  }  }  } |

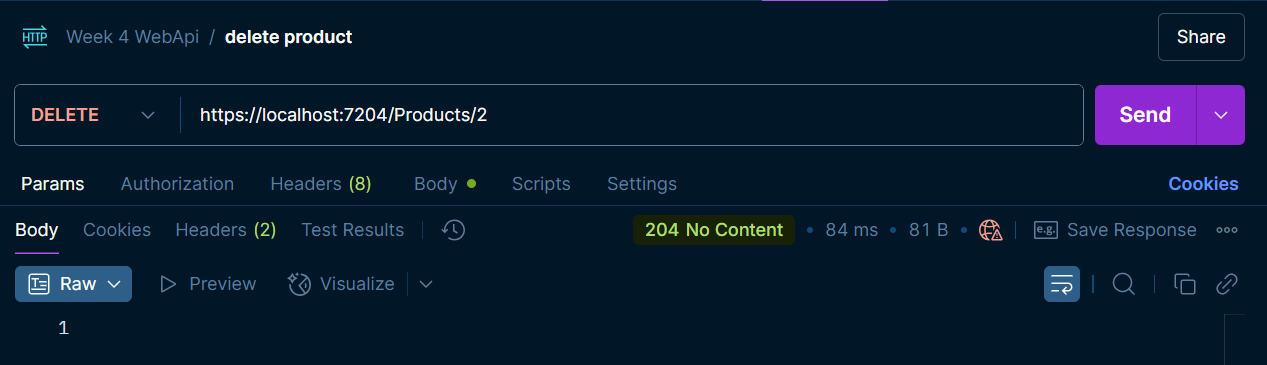
***Output:***

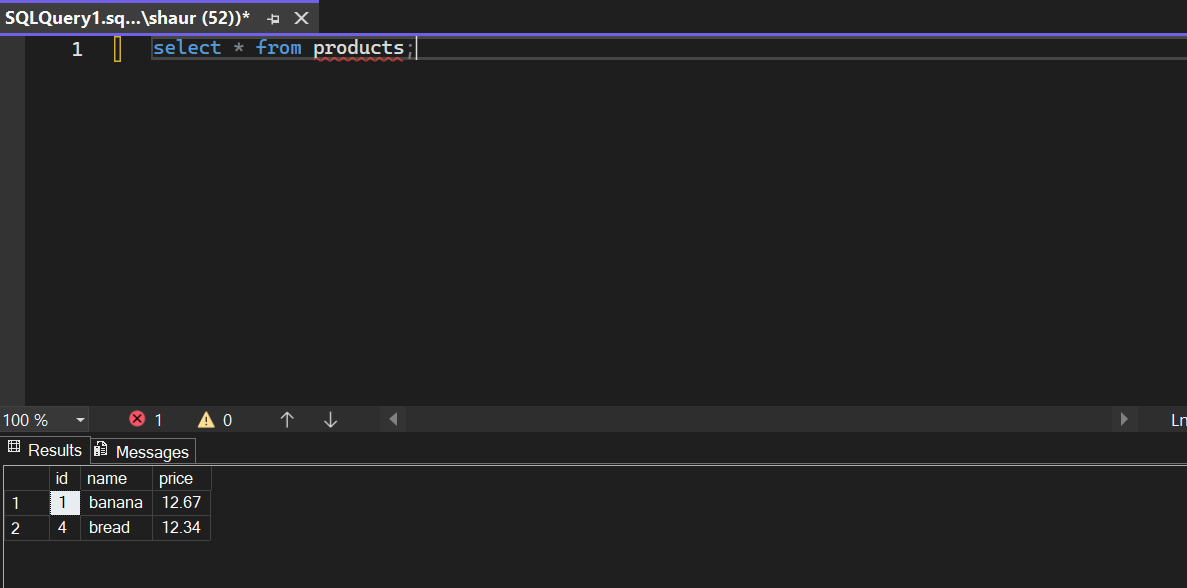












## Exercise – 2: Web API using .NET Core and Swagger

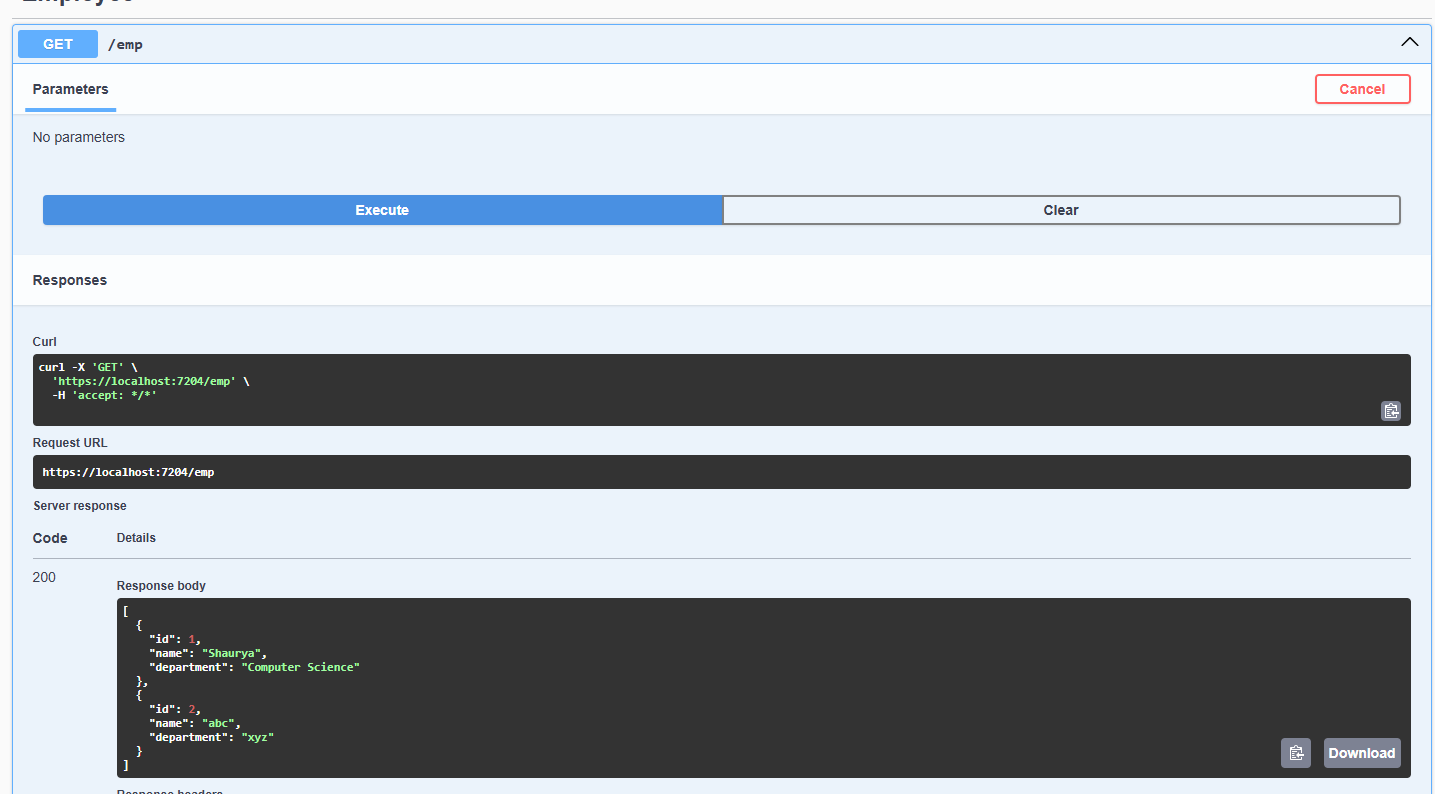
*Code: EmployeeController.cs*

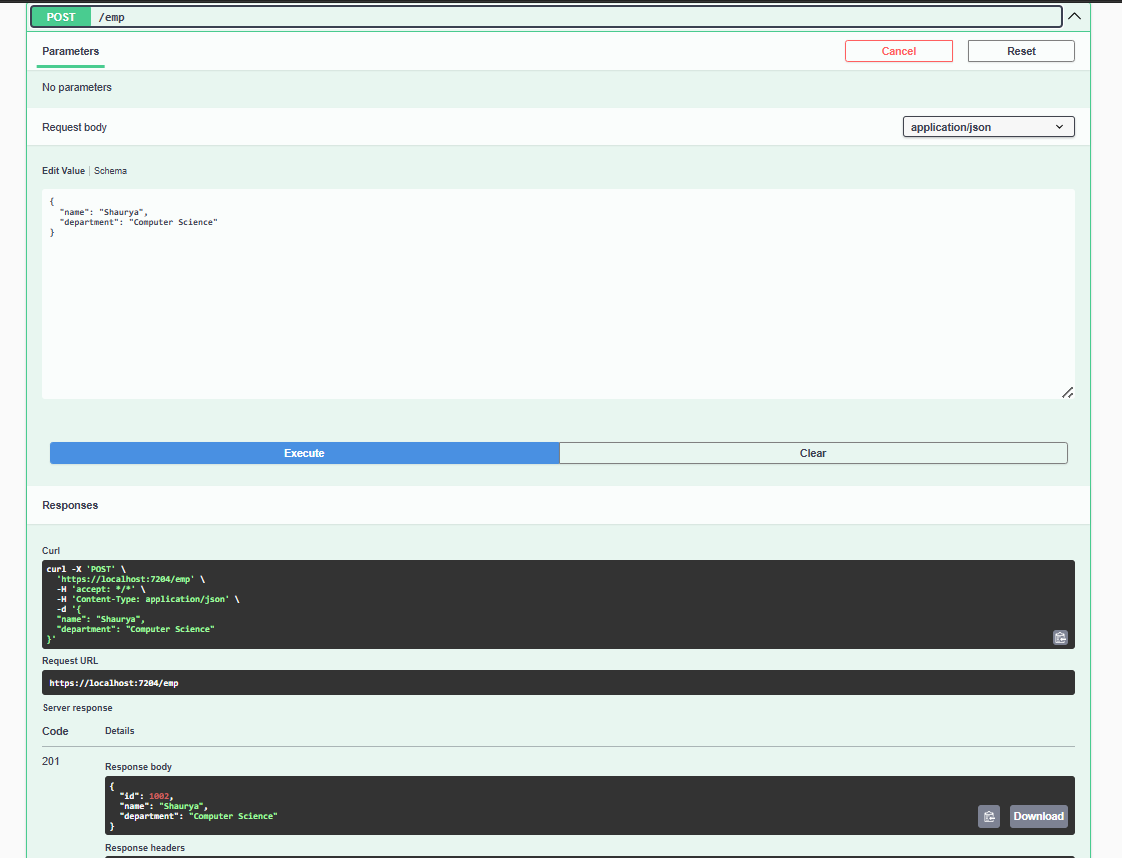
|  |
| --- |
| using Microsoft.AspNetCore.Mvc;  using MyFirstWebAPI.Data;  using MyFirstWebAPI.DTO;  using MyFirstWebAPI.Entity;    namespace MyFirstWebAPI.Controllers  {  [ApiController]  [Route("emp")]  public class EmployeeController : ControllerBase  {  private readonly AppDbContext \_context;    public EmployeeController(AppDbContext context)  {  \_context = context;  }    [HttpGet]  [ActionName("ListAll")]  [ProducesResponseType(StatusCodes.Status200OK)]  public IActionResult GetAll()  {  return Ok(\_context.Employees.ToList());  }    [HttpGet("{id}")]  [ActionName("FindById")]  public IActionResult GetById(int id)  {  var employee = \_context.Employees.Find(id);  if (employee == null) return NotFound();  return Ok(employee);  }    [HttpPost]  [ActionName("CreateNew")]  [ProducesResponseType(StatusCodes.Status201Created)]  [ProducesResponseType(StatusCodes.Status400BadRequest)]  public IActionResult Create(EmployeeAddDto employeeDto)  {  Employee employee = new Employee  {  Name = employeeDto.name,  Department = employeeDto.department  };  \_context.Employees.Add(employee);  \_context.SaveChanges();  return CreatedAtAction("FindById", new { id = employee.Id }, employee);  }  }  } |

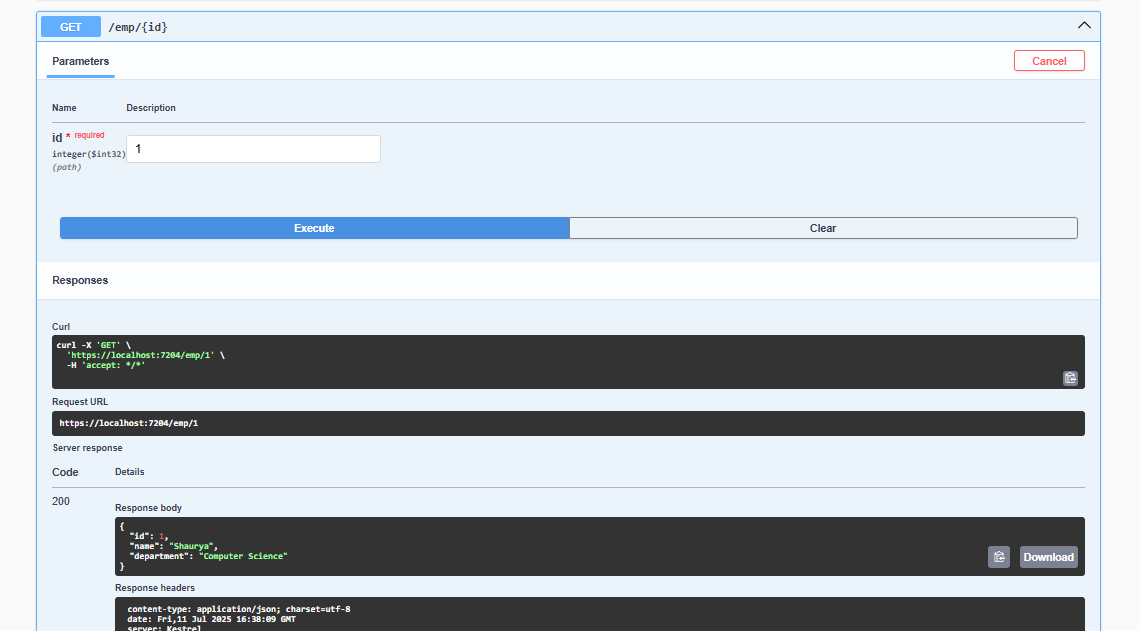
*Program.cs*

|  |
| --- |
| using Microsoft.EntityFrameworkCore;  using Microsoft.OpenApi.Models;  using MyFirstWebAPI.Data;    var builder = WebApplication.CreateBuilder(args);    builder.Services.AddControllers();  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")  }  });  });    builder.Services.AddDbContext<AppDbContext>(options =>  options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));    var app = builder.Build();    if (app.Environment.IsDevelopment())  {  app.UseSwagger();  app.UseSwaggerUI();  }    app.UseHttpsRedirection();  app.UseAuthorization();  app.MapControllers();  app.Run(); |

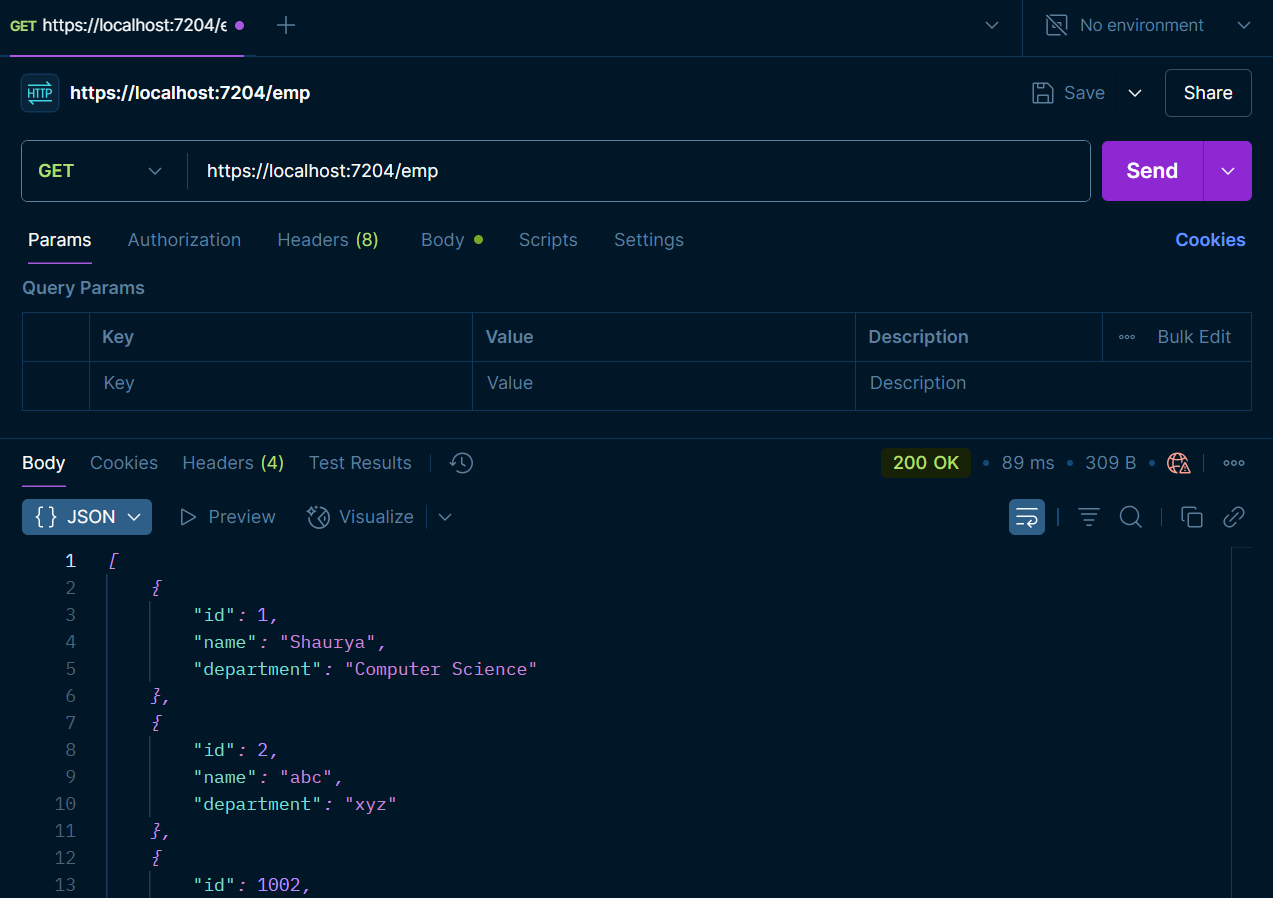
Output (Swagger)

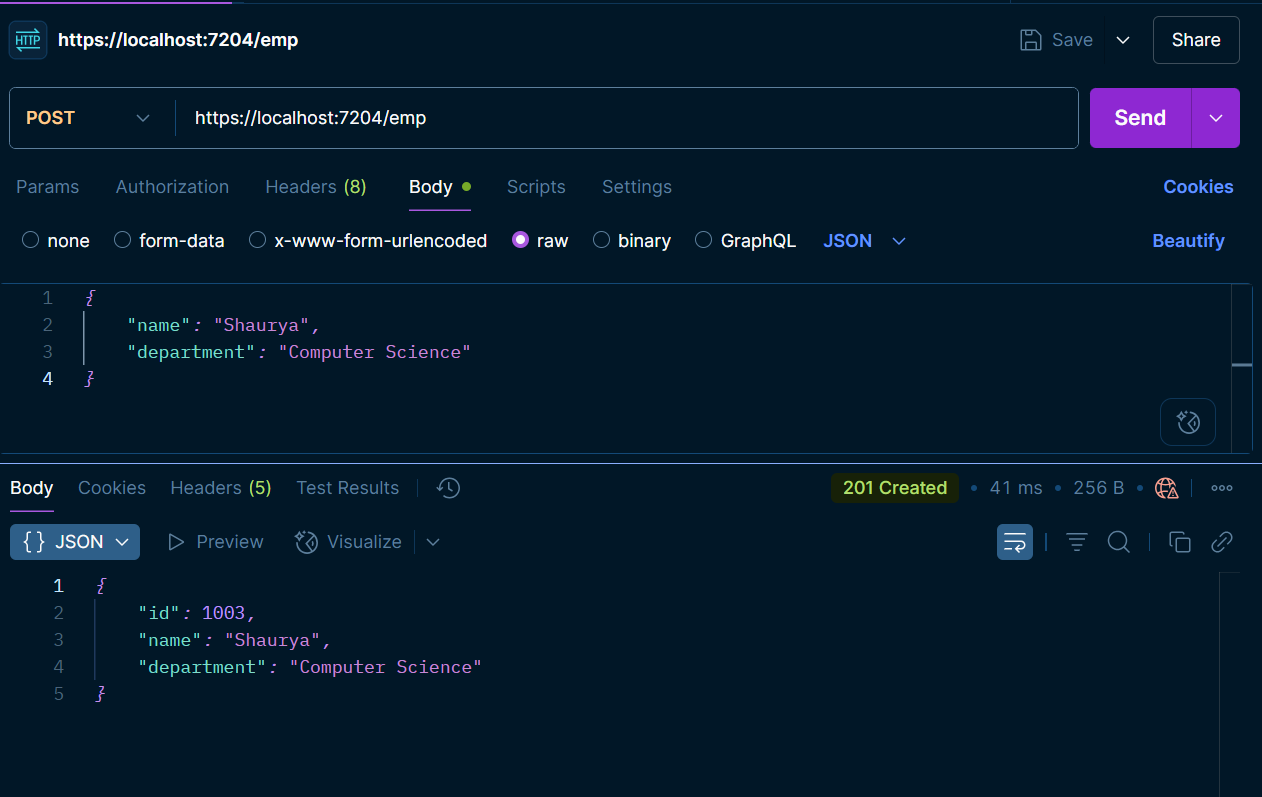


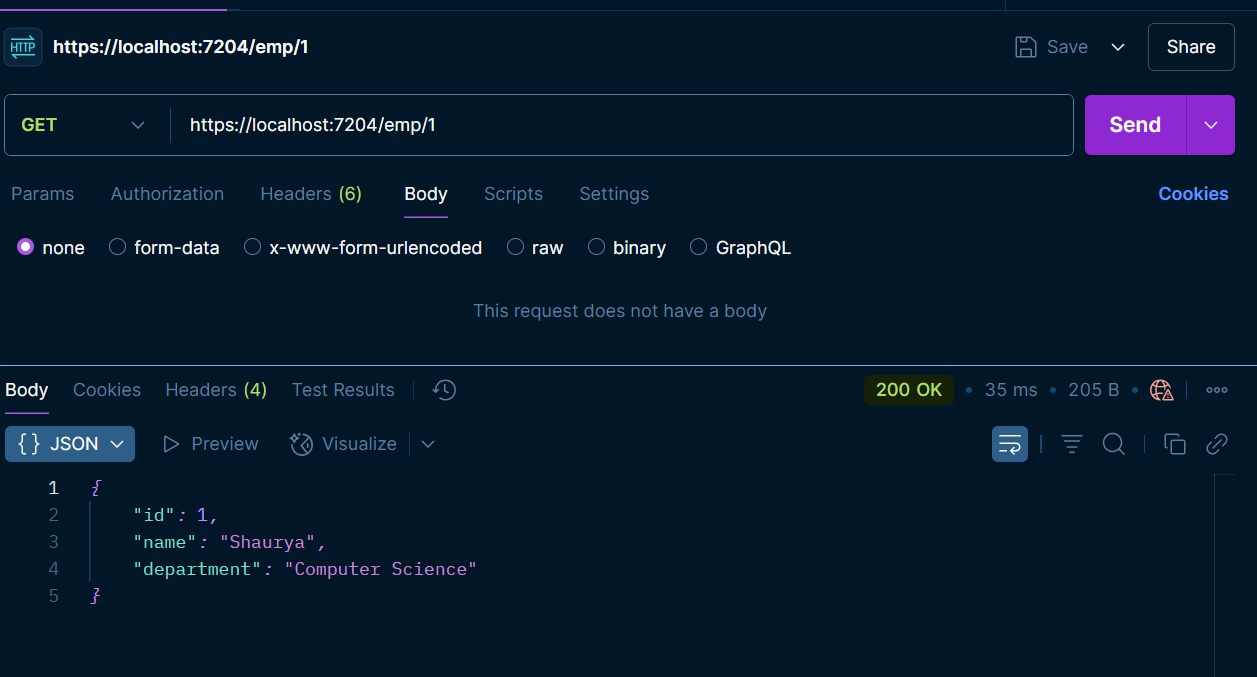




Output (Postman)







## Exercise – 3: Creating Controllers and Authentication

### Answers to Assignment Objectives

### **1. Demonstrate creation of an Action method to return list of custom class entity**

**Answer:**

An action method named Get() was created in EmployeeController, which returns a List<Employee> using a private method GetStandardEmployeeList(). It is annotated with [HttpGet] and ProducesResponseType(200).

[HttpGet]  
[ProducesResponseType(typeof(List<Employee>), 200)]  
public ActionResult<List<Employee>> Get()  
{  
 return Ok(\_employees);  
}

### **2. Model class creation, Use AllowAnonymous attribute, Use HttpGet action method**

**Answer:**

* Employee model was created with nested Department and list of Skill
* AllowAnonymous was applied to the POST method so that it bypasses authentication.
* The GET method uses [HttpGet] to expose an endpoint.

[HttpPost]  
[AllowAnonymous]  
public ActionResult<Employee> Post([FromBody] Employee employee) { ... }

### **3. Explain the usage of FromBody attribute**

**Answer:**

[FromBody] is used to bind complex objects (like Employee) from the request **body**, not from query string or URL. This is essential when sending full JSON objects in POST/PUT requests.

public ActionResult<Employee> Post([FromBody] Employee employee)

* Here, employee is populated from JSON sent in the request body.
* Without [FromBody], ASP.NET might not bind it correctly.

### **4. Read the model object from request, other than the query string parameter**

**Answer:**

Both POST and PUT methods read the Employee object from the **request body**, not query string.

Example:

{  
 "id": 3,  
 "name": "Charlie",  
 ...  
}

It’s parsed into a C# object using [FromBody].

### **5. Demonstrate Custom Filter**

**Answer:**

Two custom filters were created:

* CustomAuthFilter: Intercepts requests to check for the Authorization header.
* CustomExceptionFilter: Captures unhandled exceptions and logs them to a file.

These filters were applied to the controller ([CustomAuthFilter]) and globally in Program.cs.

### **6. Usage of ActionFilterAttribute, OnActionExecuting method to intercept the request**

**Answer:**

CustomAuthFilter inherits from ActionFilterAttribute and overrides OnActionExecuting. It checks:

* Is the Authorization header present?
* Does it contain the word "Bearer"?

If not, it short-circuits the request with a BadRequestResult.

public override void OnActionExecuting(ActionExecutingContext context)  
{  
 if (!context.HttpContext.Request.Headers.TryGetValue("Authorization", out var token))  
 context.Result = new BadRequestObjectResult("Invalid request - No Auth token");  
 else if (!token.ToString().Contains("Bearer"))  
 context.Result = new BadRequestObjectResult("Invalid request - Token present but Bearer unavailable");  
}

### **7. Create filter for Custom exception - Install WebApiCompatShim package**

**Answer:**

A CustomExceptionFilter was created implementing IExceptionFilter. It logs the full exception to a file (error\_log.txt) and returns a 500 response.

public void OnException(ExceptionContext context)  
{  
 File.WriteAllText("error\_log.txt", context.Exception.ToString());  
 context.Result = new ObjectResult("Something went wrong") { StatusCode = 500 };  
}

Also, the Microsoft.AspNetCore.Mvc.WebApiCompatShim package was installed as per the requirement using:

dotnet add package Microsoft.AspNetCore.Mvc.WebApiCompatShim

### Code: All Models (Employee.cs, Department.cs, Skills.cs)

|  |
| --- |
| namespace EmployeeWebApi.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<Skills> Skills { get; set; }  public DateTime DateOfBirth { get; set; }  }  }  namespace EmployeeWebApi.Models  {  public class Department  {  public int Id { get; set; }  public string Name { get; set; }  }  }  namespace EmployeeWebApi.Models  {  public class Skills  {  public int Id { get; set; }  public string Name { get; set; }  }  } |

Controller: EmployeeController.cs

|  |
| --- |
| using EmployeeWebApi.Filters;  using EmployeeWebApi.Models;  using Microsoft.AspNetCore.Authorization;  using Microsoft.AspNetCore.Mvc;    namespace EmployeeApi.Controllers;    [ApiController]  [Route("api/[controller]")]  [CustomAuthFilter]  public class EmployeeController : ControllerBase  {  private readonly List<Employee> \_employees;    public EmployeeController()  {  \_employees = GetStandardEmployeeList();  }    private List<Employee> GetStandardEmployeeList()  {  return new List<Employee>  {  new Employee  {  Id = 1,  Name = "Alice",  Salary = 50000,  Permanent = true,  DateOfBirth = new DateTime(1990, 5, 1),  Department = new Department { Id = 1, Name = "HR" },  Skills = new List<Skills> { new Skills { Id = 1, Name = "Communication" } }  },  new Employee  {  Id = 2,  Name = "Bob",  Salary = 60000,  Permanent = false,  DateOfBirth = new DateTime(1992, 8, 12),  Department = new Department { Id = 2, Name = "IT" },  Skills = new List<Skills> { new Skills { Id = 2, Name = "C#" } }  }  };  }    [HttpGet]  [ProducesResponseType(typeof(List<Employee>), 200)]  [ProducesResponseType(500)]  public ActionResult<List<Employee>> Get()  {  return Ok(\_employees);  }      [HttpPost]  [AllowAnonymous]  public ActionResult<Employee> Post([FromBody] Employee employee)  {  \_employees.Add(employee);  return Ok(employee);  }    [HttpPut("{id}")]  public ActionResult<Employee> Put(int id, [FromBody] Employee employee)  {  var emp = \_employees.FirstOrDefault(e => e.Id == id);  if (emp == null) return NotFound();    emp.Name = employee.Name;  emp.Salary = employee.Salary;  emp.Permanent = employee.Permanent;  emp.Department = employee.Department;  emp.Skills = employee.Skills;  emp.DateOfBirth = employee.DateOfBirth;    return Ok(emp);  }  } |

*Filters: (CustomAuthFilter.cs, CustomExceptionFilter.cs)*

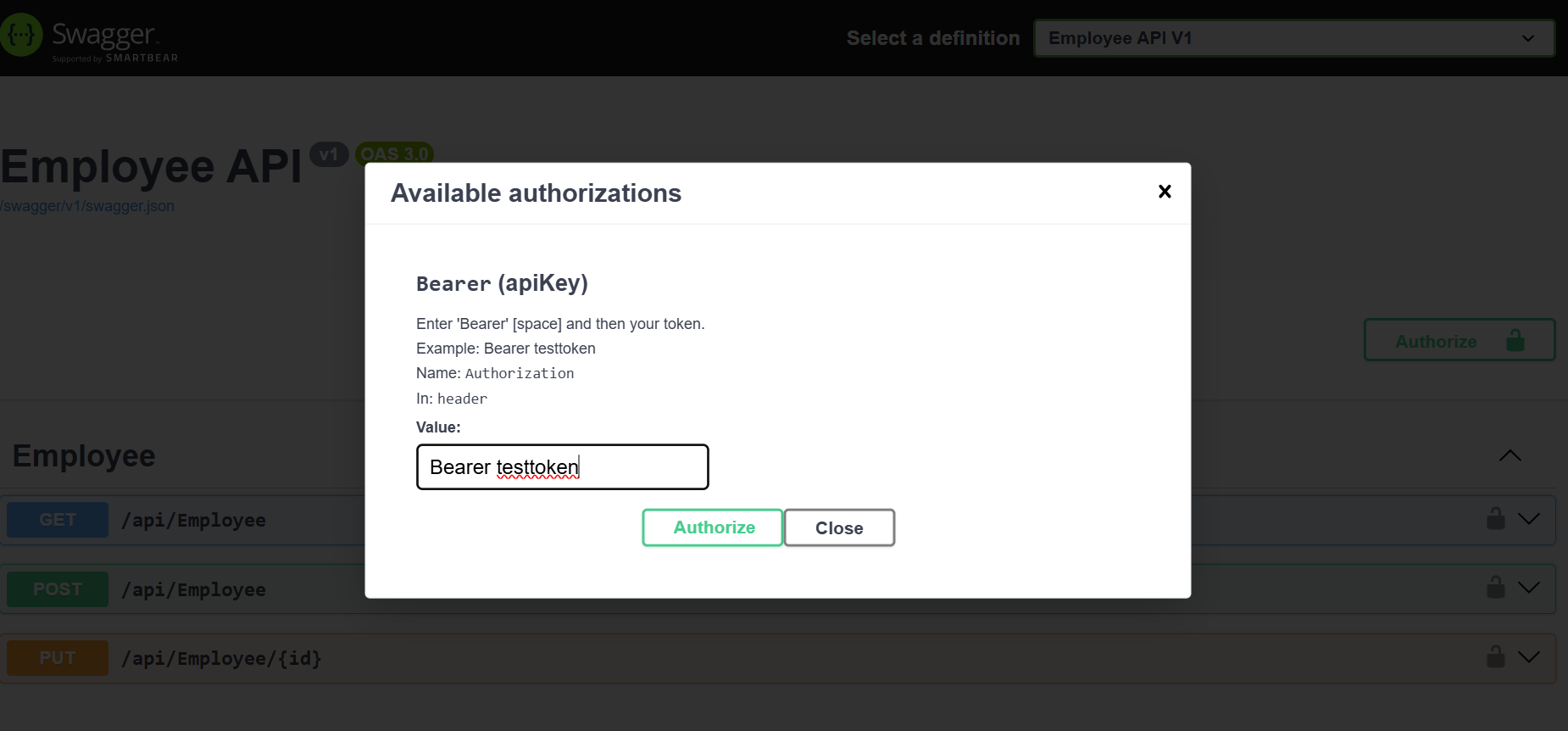
|  |
| --- |
| using Microsoft.AspNetCore.Mvc;  using Microsoft.AspNetCore.Mvc.Filters;    namespace EmployeeWebApi.Filters  {  public class CustomAuthFilter : ActionFilterAttribute  {  public override void OnActionExecuting(ActionExecutingContext context)  {  if(!context.HttpContext.Request.Headers.TryGetValue("Authorization", out var token))  {  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");  return;  }    base.OnActionExecuting(context);  }  }  }  using Microsoft.AspNetCore.Mvc;  using Microsoft.AspNetCore.Mvc.Filters;    namespace EmployeeWebApi.Filters  {  public class CustomExceptionFilter : IExceptionFilter  {  public void OnException(ExceptionContext context)  {  var error = context.Exception.ToString();  File.WriteAllText("error\_log.txt", error);    context.Result = new ObjectResult("Something went wrong")  {  StatusCode = 500  };  }  }  } |

*Program.cs*

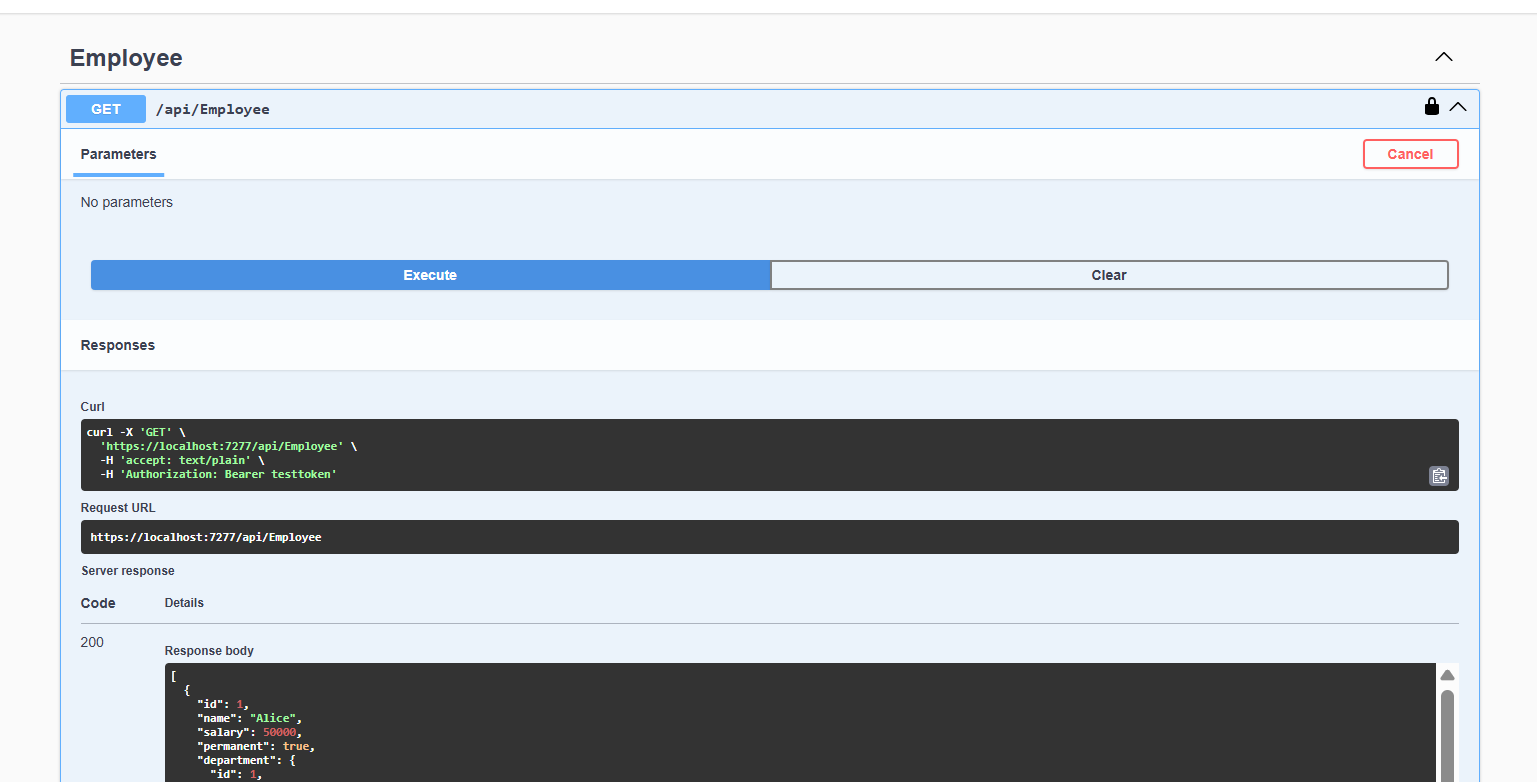
|  |
| --- |
| using EmployeeWebApi.Filters;  using Microsoft.OpenApi.Models;    var builder = WebApplication.CreateBuilder(args);    builder.Services.AddControllers(options =>  {  options.Filters.Add<CustomExceptionFilter>();  });    builder.Services.AddEndpointsApiExplorer();  builder.Services.AddSwaggerGen(c =>  {  c.SwaggerDoc("v1", new OpenApiInfo { Title = "Employee API", Version = "v1" });    c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme  {  Name = "Authorization",  Type = SecuritySchemeType.ApiKey,  Scheme = "Bearer",  BearerFormat = "JWT",  In = ParameterLocation.Header,  Description = "Enter 'Bearer' [space] and then your token.\n\nExample: Bearer testtoken"  });  c.AddSecurityRequirement(new OpenApiSecurityRequirement  {  {  new OpenApiSecurityScheme  {  Reference = new OpenApiReference { Type = ReferenceType.SecurityScheme, Id = "Bearer" }  },  Array.Empty<string>()  }  });  });    var app = builder.Build();    app.UseHttpsRedirection();  app.UseRouting();    app.UseSwagger();  app.UseSwaggerUI(c =>  {  c.SwaggerEndpoint("/swagger/v1/swagger.json", "Employee API V1");  });    app.UseAuthorization();  app.MapControllers();    app.Run(); |

*Output:*

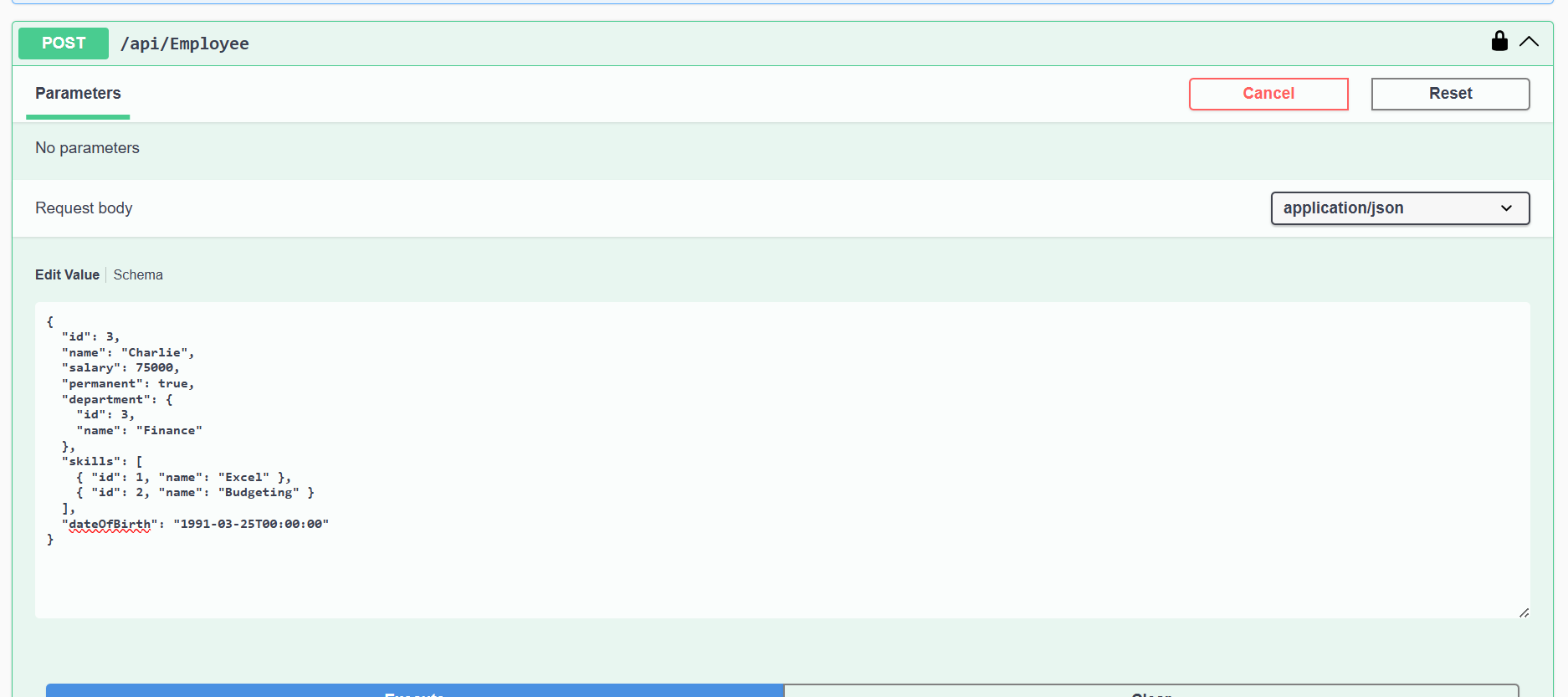
*Adding the Authorization*

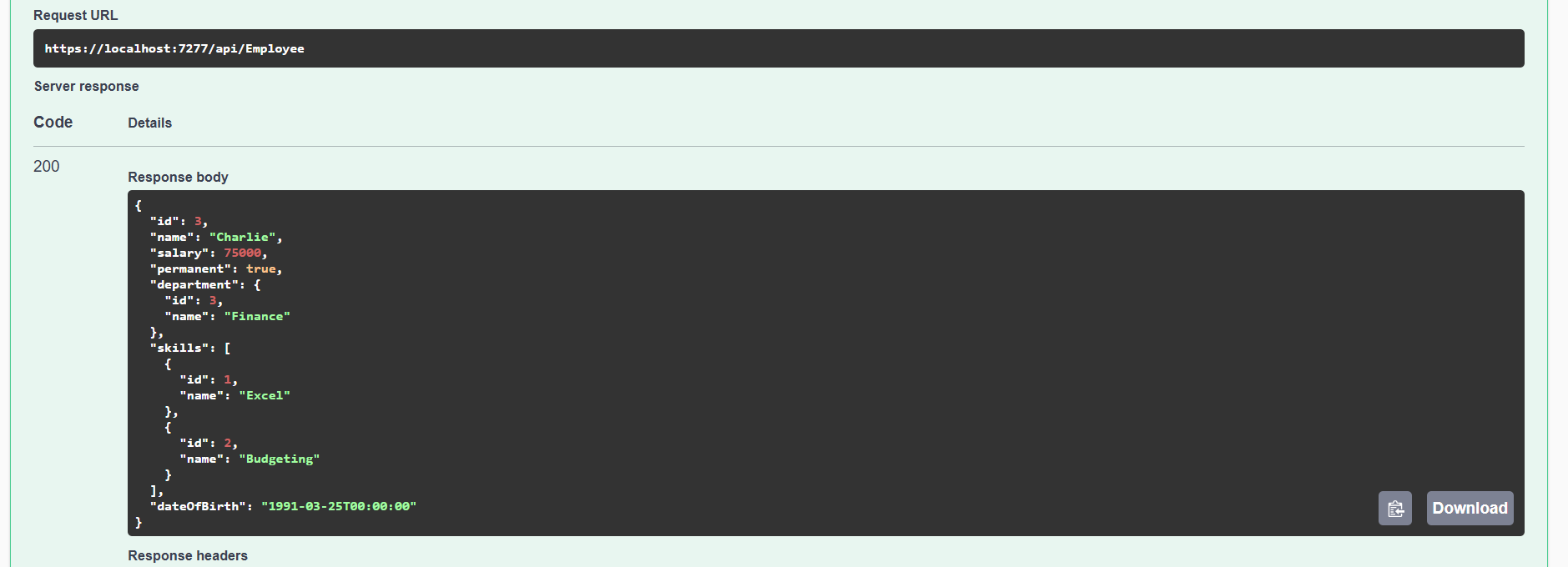


*Getting all the employees*

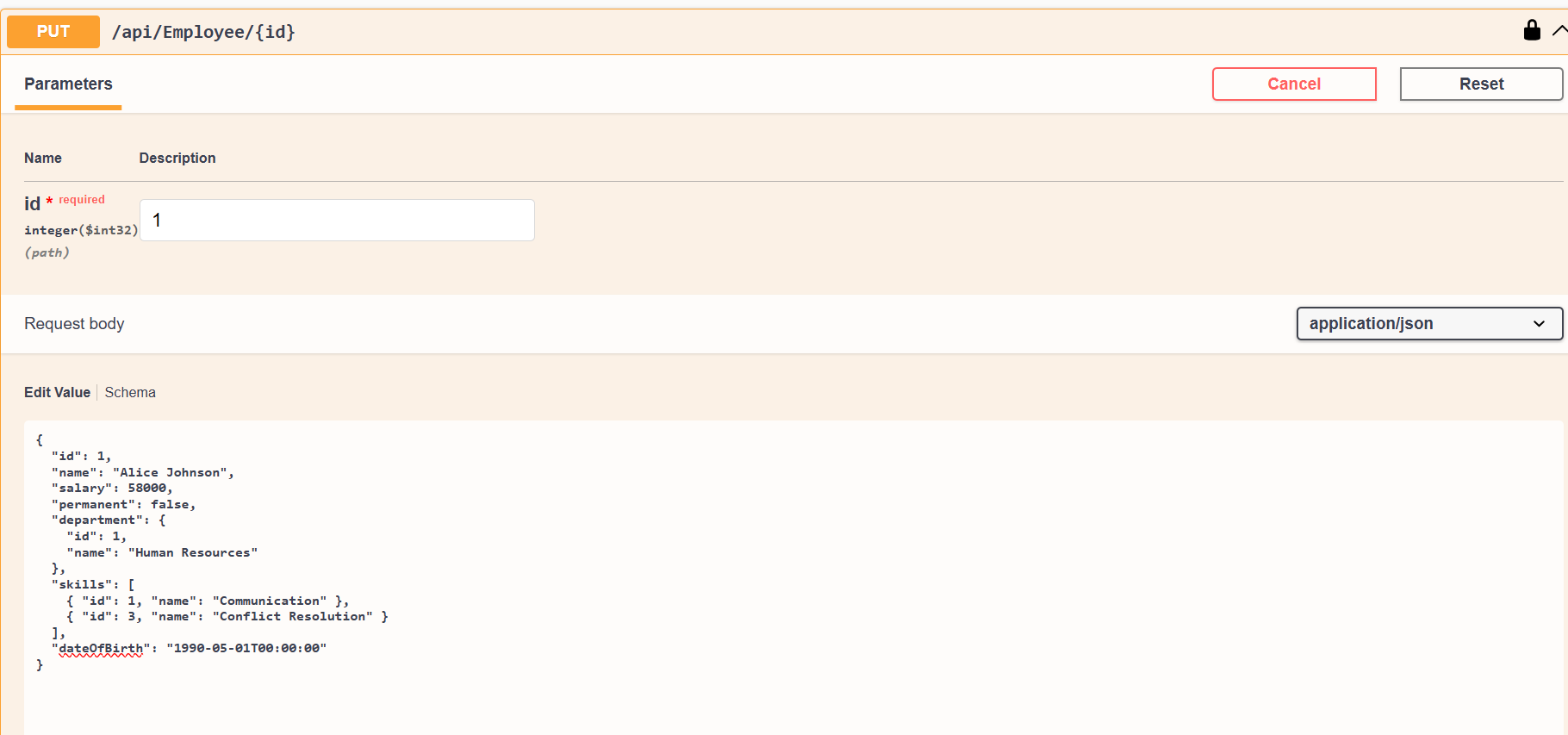


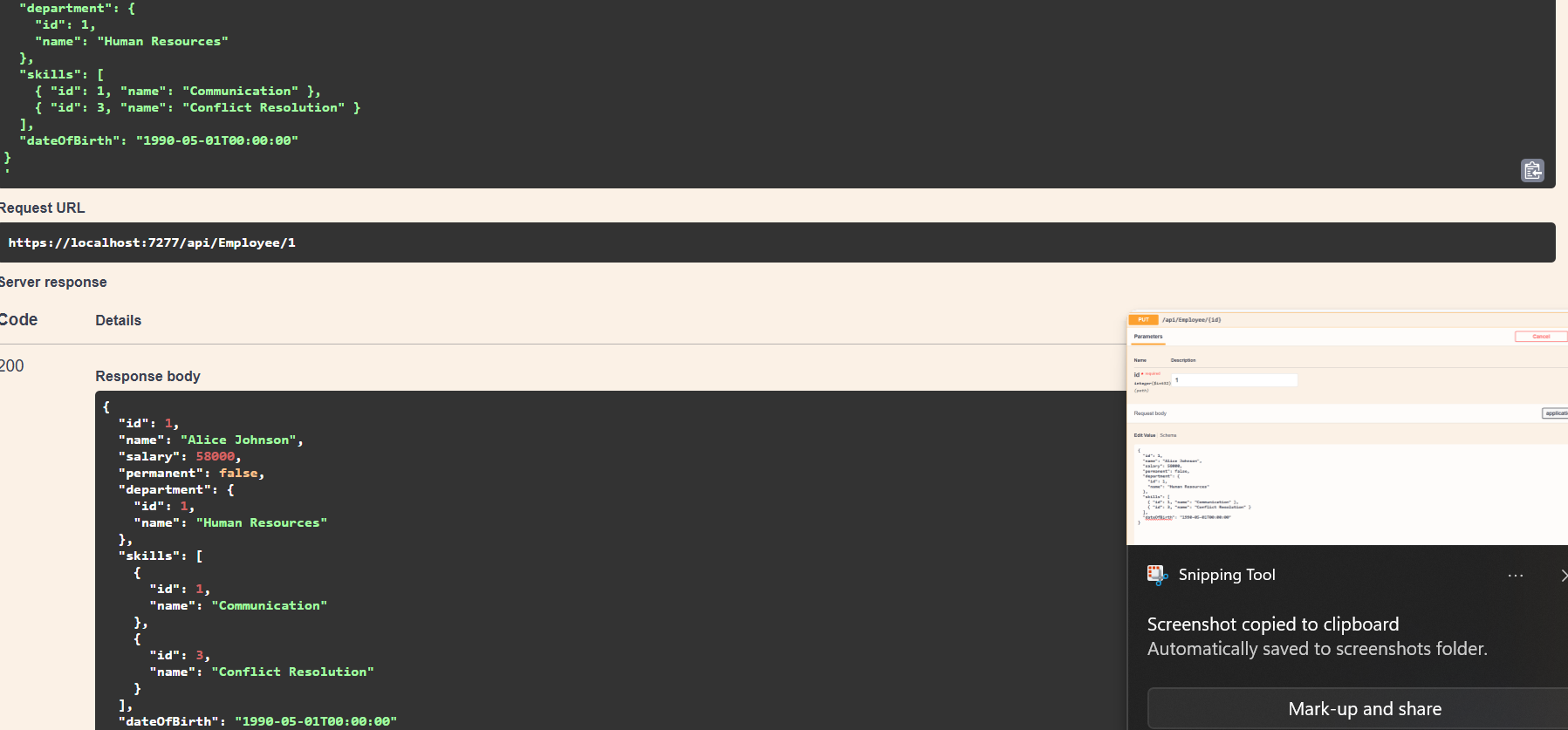
*Adding a new Employee*





*Updating the employee info*



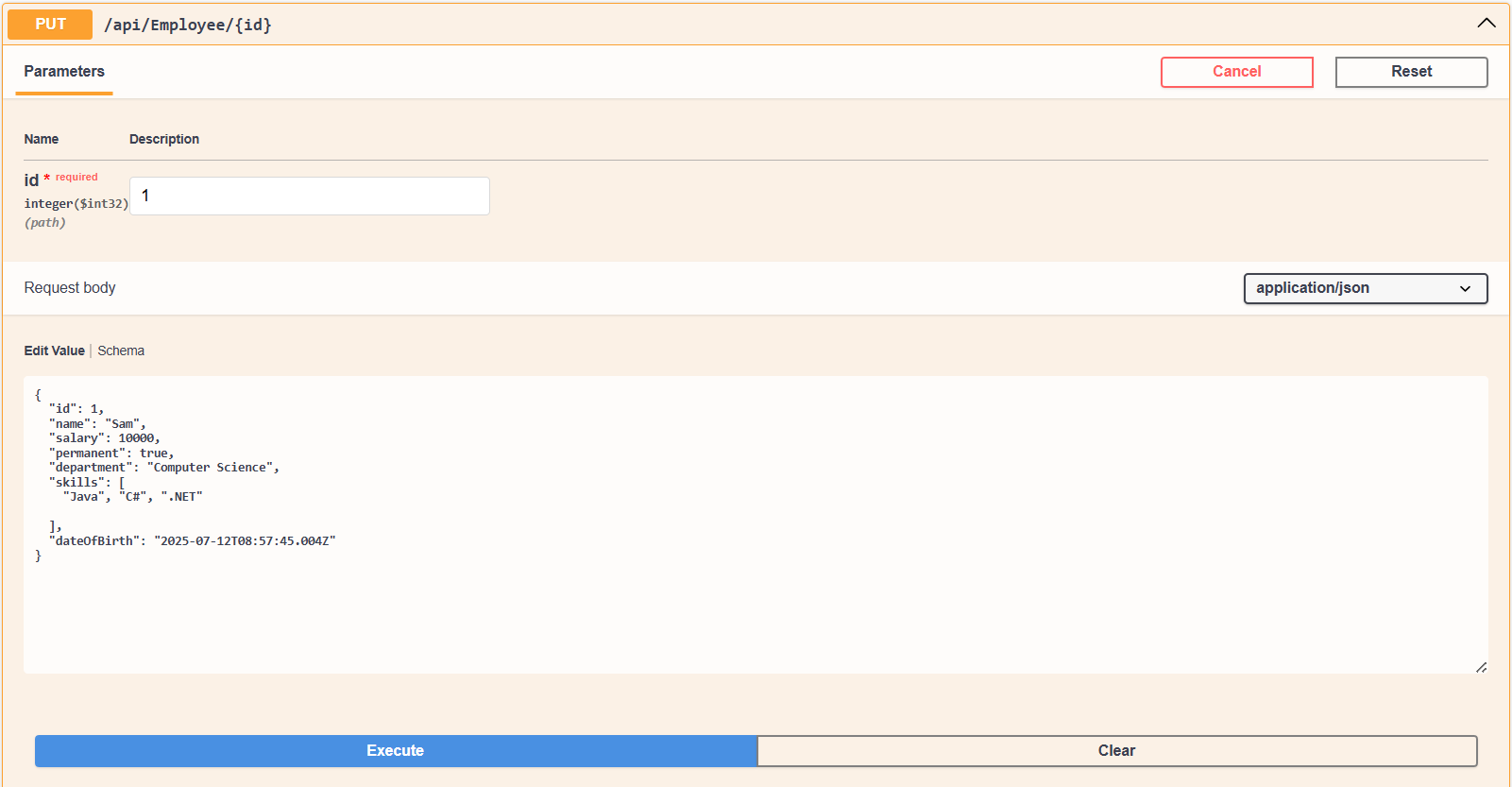


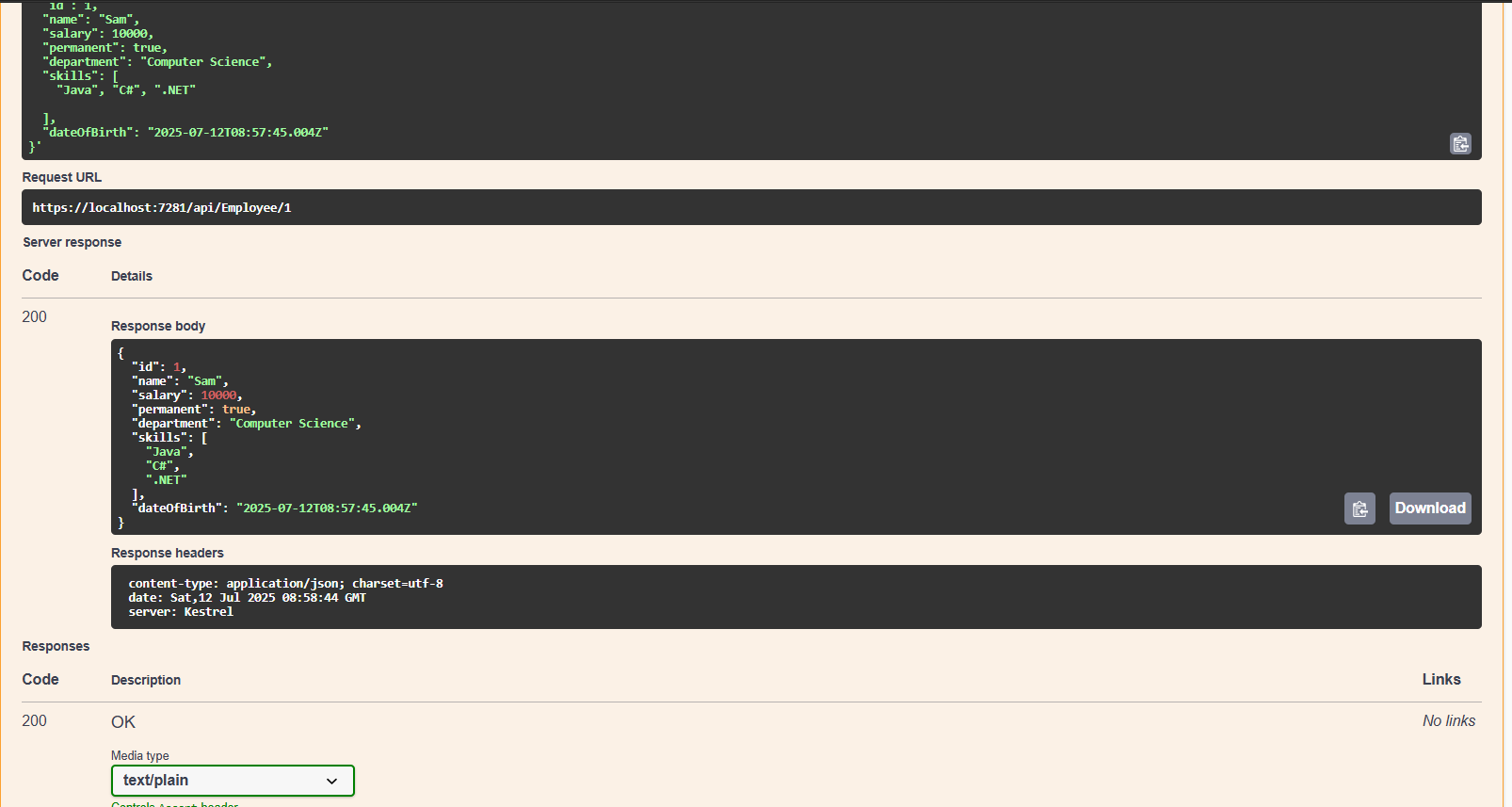
## Exercise – 4: Web API CRUD Operation

*Code: EmployeeCrudController.cs*

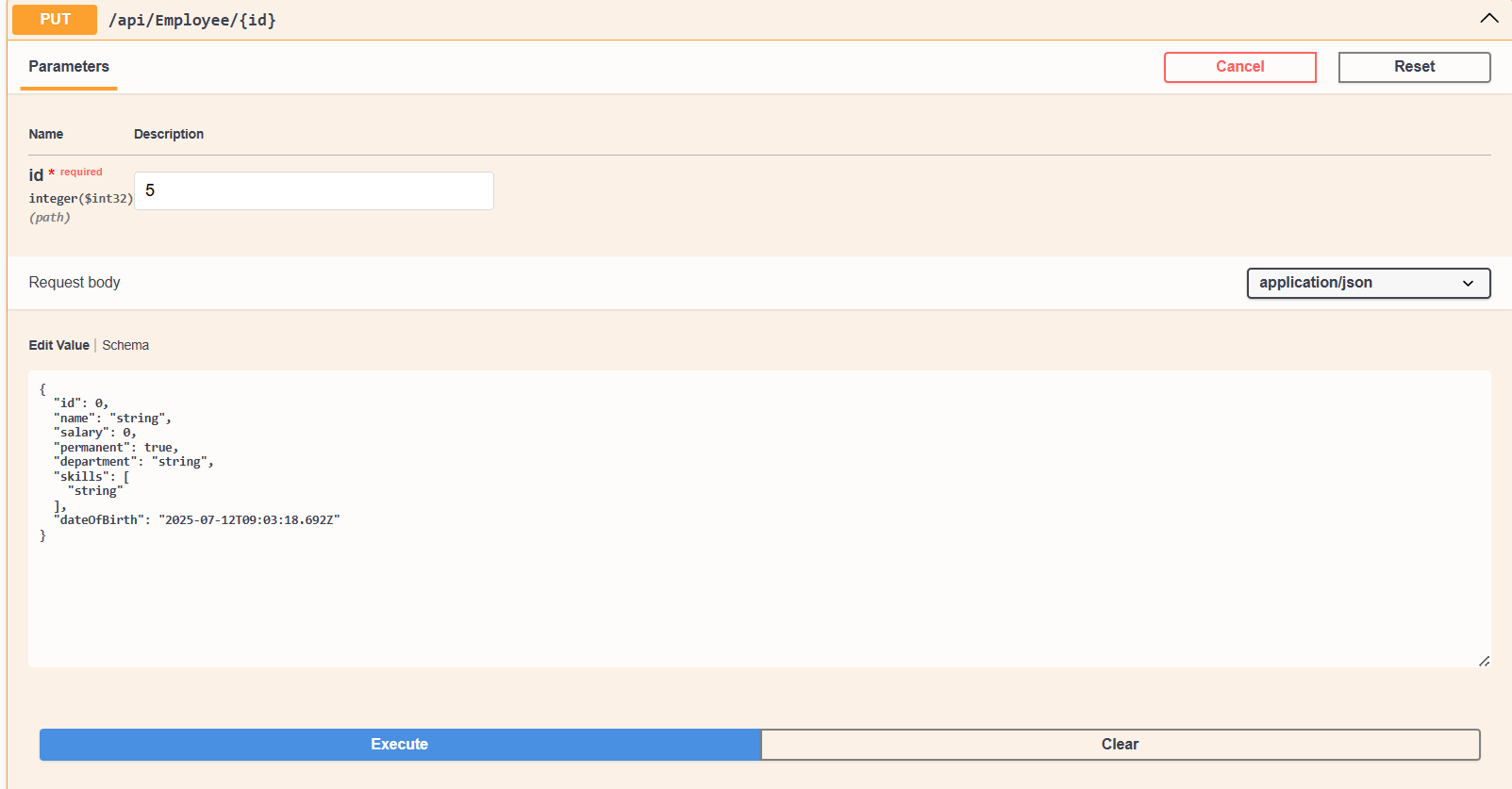
|  |
| --- |
| using EmployeeWebApi.Models;  using Microsoft.AspNetCore.Mvc;    namespace EmployeeCrudApi.Controllers  {  [ApiController]  [Route("api/[controller]")]  public class EmployeeController : ControllerBase  {  private static List<Employee> employees = new List<Employee>  {  new Employee  {  Id = 1,  Name = "Shaurya Sharma",  Salary = 50000,  Permanent = true,  Department = "Engineering",  Skills = new List<string> { "C#", "SQL" },  DateOfBirth = new DateTime(2000, 1, 15)  },  new Employee  {  Id = 2,  Name = "John Doe",  Salary = 60000,  Permanent = false,  Department = "HR",  Skills = new List<string> { "Communication", "Recruitment" },  DateOfBirth = new DateTime(1995, 10, 5)  }  };    [HttpPut("{id}")]  [ProducesResponseType(StatusCodes.Status200OK)]  [ProducesResponseType(StatusCodes.Status400BadRequest)]  public ActionResult<Employee> Update(int id, [FromBody] Employee updatedEmployee)  {  if (id <= 0)  return BadRequest("Invalid employee id");    var existingEmployee = employees.FirstOrDefault(e => e.Id == id);  if (existingEmployee == null)  return BadRequest("Invalid employee id");    existingEmployee.Name = updatedEmployee.Name;  existingEmployee.Salary = updatedEmployee.Salary;  existingEmployee.Permanent = updatedEmployee.Permanent;  existingEmployee.Department = updatedEmployee.Department;  existingEmployee.Skills = updatedEmployee.Skills;  existingEmployee.DateOfBirth = updatedEmployee.DateOfBirth;    return Ok(existingEmployee);  }  }  } |

*Output: Valid Input given*





*Output: Invalid Input given*





## Exercise – 5: JSON Web Token

### **1. What is CORS?**

**CORS (Cross-Origin Resource Sharing)** is a browser security feature that blocks requests from one origin to another unless explicitly allowed.

### **2. What is the use of CORS in Web API?**

CORS allows your Web API to accept requests from different domains — like enabling your frontend (localhost:4200) to access the API running on (localhost:5000).

### **3. What is Bearer Token Authentication?**

It's a security mechanism where a client sends a **token** in the Authorization header (Bearer <token>). The server validates this token to allow access to protected resources.

### **4. What is the difference between [Authorize] and [AllowAnonymous]?**

* [Authorize]: Restricts access to authenticated users (with valid JWT).
* [AllowAnonymous]: Overrides [Authorize] to **allow public access** (no token needed).

### **5. How to generate a token?**

A token is generated in the controller using a JwtSecurityToken, signing it with a secret key and adding claims like user ID and role. It is returned from the AuthController.

*Code: (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 JwtAuthWebApi.Controllers  {  [ApiController]  [Route("api/[controller]")]  [AllowAnonymous]  public class AuthController : ControllerBase  {  [HttpGet]  public IActionResult GetToken()  {  var token = GenerateJSONWebToken(1, "Admin");  return Ok(new {Token = token});  }    private string GenerateJSONWebToken(int userId, string userRole)  {  var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretkeyfortokenvalidation1234"));    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);  }  }  } |

*Code: (EmployeeController.cs)*

|  |
| --- |
| using JwtAuthWebApi.Models;  using Microsoft.AspNetCore.Authorization;  using Microsoft.AspNetCore.Mvc;    namespace JwtAuthWebApi.Controllers  {  [ApiController]  [Route("api/[controller]")]  [Authorize(Roles = "Admin,POC")]  public class EmployeeController : ControllerBase  {  [HttpGet]  public ActionResult<List<Employee>> Get()  {  return Ok(new List<Employee>  {  new Employee { Id = 1, Name = "Shaurya Sharma" }  });  }  }  } |

*Code: (Employee.cs)*

|  |
| --- |
| namespace JwtAuthWebApi.Models  {  public class Employee  {  public int Id { get; set; }  public string Name { get; set; }  }  } |

*Code: (Program.cs)*

|  |
| --- |
| using Microsoft.AspNetCore.Authentication.JwtBearer;  using Microsoft.IdentityModel.Tokens;  using System.Text;    var builder = WebApplication.CreateBuilder(args);    builder.Services.AddControllers();  builder.Services.AddEndpointsApiExplorer();  builder.Services.AddSwaggerGen();    string securityKey = "mysuperdupersecretkeyfortokenvalidation1234";  var symmetricSecurityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(securityKey));    builder.Services.AddAuthentication(x =>  {  x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;  x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;  x.DefaultSignInScheme = JwtBearerDefaults.AuthenticationScheme;  })  .AddJwtBearer(JwtBearerDefaults.AuthenticationScheme, x =>  {  x.TokenValidationParameters = new TokenValidationParameters  {  ValidateIssuer = true,  ValidateAudience = true,  ValidateLifetime = true,  ValidateIssuerSigningKey = true,  ValidIssuer = "mySystem",  ValidAudience = "myUsers",  IssuerSigningKey = symmetricSecurityKey  };  });    builder.Services.AddCors();    var app = builder.Build();    app.UseCors(policy => policy.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader());    app.UseAuthentication();  app.UseAuthorization();    if (app.Environment.IsDevelopment())  {  app.UseSwagger();  app.UseSwaggerUI();  }    app.MapControllers();  app.Run(); |

*Output:*

