**WEEK ASSIGNMENT 4**

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

**Create a .Net core web application with API template. Use the option to create controller with Read Write permissions. Notice the ValuesController creation with Action methods corresponding to the Action verbs.**

**On creation of the Web API, execute the application and check if the GET action method result is returned as expected.**

**Controller:**

**using Microsoft.AspNetCore.Mvc;**

**using System.Collections.Generic;**

**namespace MyFirstWebApi.Controllers**

**{**

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

**[ApiController]**

**public class ValuesController : ControllerBase**

**{**

**[HttpGet]**

**public ActionResult<IEnumerable<string>> Get()**

**{**

**return Ok(new string[] { "value1", "value2" });**

**}**

**[HttpGet("{id}")]**

**public ActionResult<string> Get(int id)**

**{**

**return Ok("value " + id);**

**}**

**[HttpPost]**

**public ActionResult Post([FromBody] string value)**

**{**

**return Ok("Posted: " + value);**

**}**

**[HttpPut("{id}")]**

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

**{**

**return Ok($"Updated id {id} with value {value}");**

**}**

**[HttpDelete("{id}")]**

**public ActionResult Delete(int id)**

**{**

**return Ok($"Deleted id {id}");**

**}**

**}**

**}**

**MyFirstWebApi.csproj  
  
<Project Sdk="Microsoft.NET.Sdk.Web">**

**<PropertyGroup>**

**<OutputType>Exe</OutputType>**

**<TargetFramework>net9.0</TargetFramework>**

**</PropertyGroup>**

**</Project>**

**[Program.cs](http://program.cs)**

**using Microsoft.AspNetCore.Builder;**

**using Microsoft.Extensions.DependencyInjection;**

**using Microsoft.Extensions.Hosting;**

**var builder = WebApplication.CreateBuilder(args);**

***// Register controller services***

**builder.Services.AddControllers();**

**var app = builder.Build();**

***// Enable routing and controller mapping***

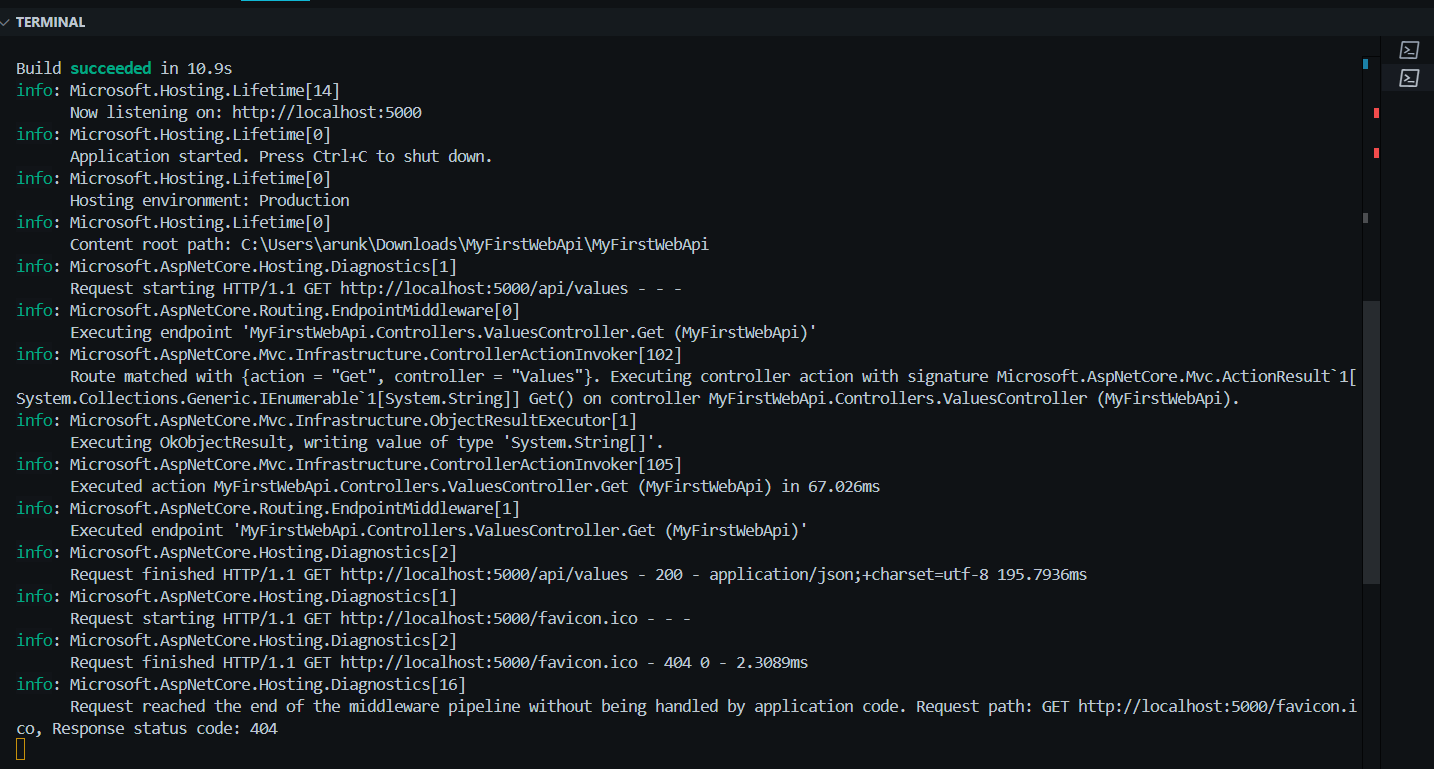
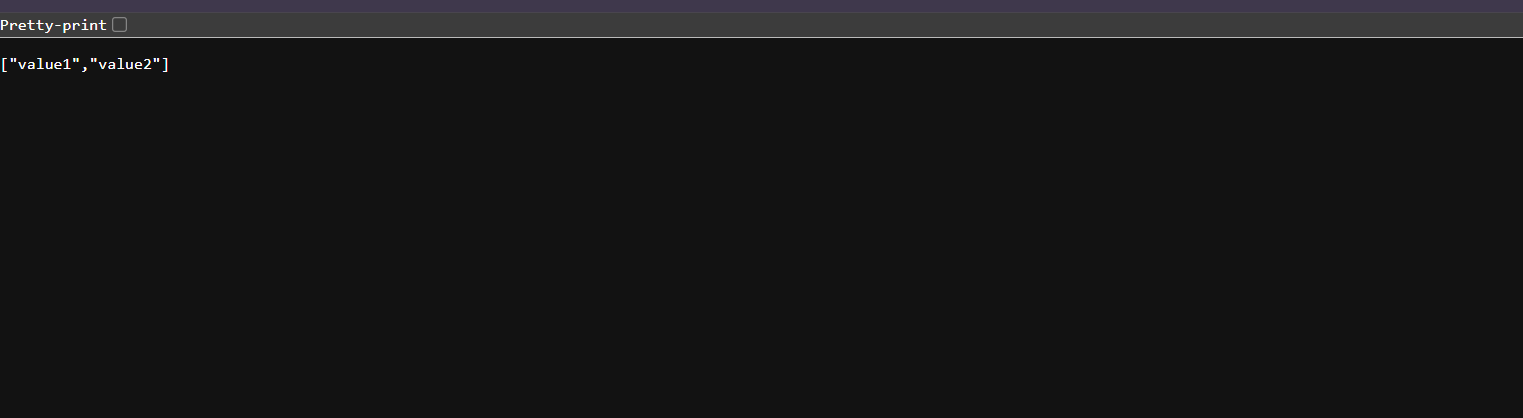
**app.UseRouting();**

**app.UseAuthorization();**

**app.MapControllers();**

**app.Run();**

**OUtPUT:**

**  
  
  
**

**2.** **Web Api using .Net core with Swagger**

**Create a .Net core web application with API template. (Use existing application if created). Install Swashbuckle.AspNetCore Nuget package. Post this do the following steps in [Startup.cs](http://startup.cs)**

**Program:**

**[EmployeeController.cs](http://employeecontroller.cs):**

**using Microsoft.AspNetCore.Mvc;**

**namespace SwaggerDemo.Controllers**

**{**

**[Route("api/emp")]**

**[ApiController]**

**public class EmployeeController : ControllerBase**

**{**

**static List<string> employees = new List<string> { "John", "Jane", "Alice" };**

**[HttpGet]**

**[ProducesResponseType(StatusCodes.Status200OK)]**

**[ProducesResponseType(StatusCodes.Status404NotFound)]**

**public IActionResult Get()**

**{**

**return Ok(employees);**

**}**

**[HttpPost]**

**public IActionResult Post([FromBody] string name)**

**{**

**employees.Add(name);**

**return Ok(employees);**

**}**

**}**

**}**

**[Program.cs](http://program.cs)**

**using Microsoft.OpenApi.Models;**

**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("http://example.com"),**

**Contact = new OpenApiContact**

**{**

**Name = "John Doe",**

**Email = "john@xyzmail.com",**

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

**},**

**License = new OpenApiLicense**

**{**

**Name = "License Terms",**

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

**}**

**});**

**});**

**var app = builder.Build();**

**app.UseSwagger();**

**app.UseSwaggerUI(c =>**

**{**

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

**});**

**app.UseAuthorization();**

**app.MapControllers();**

**app.Run();**

**<Project Sdk="Microsoft.NET.Sdk.Web">**

**<PropertyGroup>**

**<TargetFramework>net9.0</TargetFramework>**

**<Nullable>enable</Nullable>**

**<ImplicitUsings>enable</ImplicitUsings>**

**</PropertyGroup>**

**<ItemGroup>**

**<PackageReference Include="Swashbuckle.AspNetCore" Version="6.5.0" />**

**</ItemGroup>**

**</Project>**

**SwaggerDemo.csproj  
  
<Project Sdk="Microsoft.NET.Sdk.Web">**

**<PropertyGroup>**

**<TargetFramework>net9.0</TargetFramework>**

**<Nullable>enable</Nullable>**

**<ImplicitUsings>enable</ImplicitUsings>**

**</PropertyGroup>**

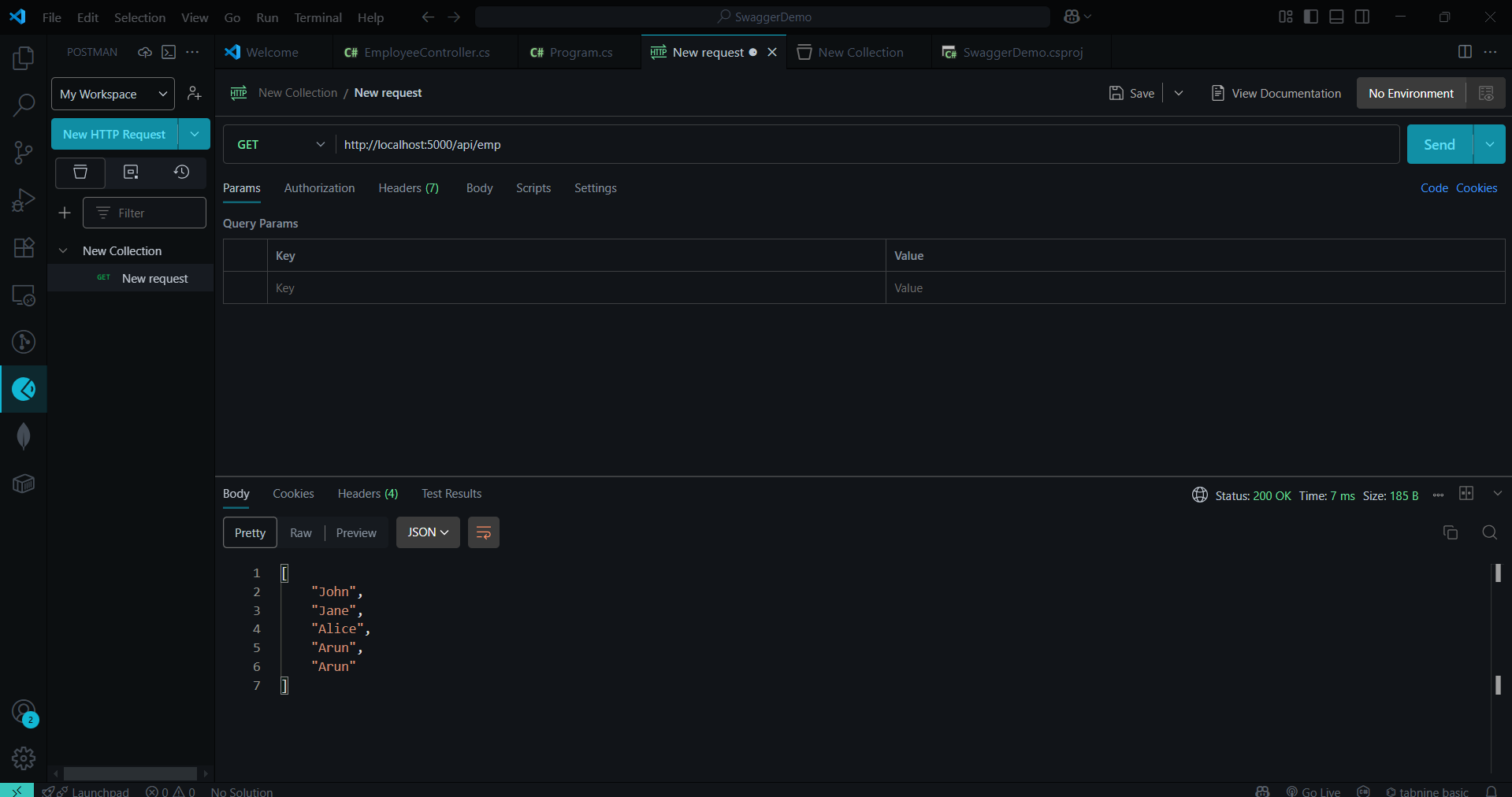
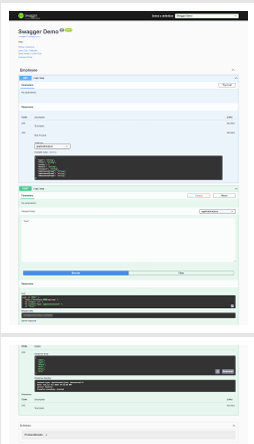
**<ItemGroup>**

**<PackageReference Include="Swashbuckle.AspNetCore" Version="6.5.0" />**

**</ItemGroup>**

**</Project>**

**OutPut:  
Postman**

**  
  
  
Site:**

**3.** **Web Api using custom model class**

**Create a Custom class ‘Employee’ of the below defined structure**

**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; }**

**}**

**Create a new controller - EmployeeController with Read Write actions**

**Constructor: Create few records, HTTPGet, HTTPPost/HTTPPut**

**Create a Private method GetStandardEmployeeList that returns a List of Employee class. Invoke this method in the Get action method of the EmployeeController that was created in the previous step.**

**Public ActionResult<Employee> GetStandrad()**

**Modify the return type of the Get action method(without parameter) to return List of Employee class object**

**Add ProducesResponseType to the GET action method for Status code 200**

**Check the Swagger description for the GET method for success status code**

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

**The requirement is to intercept incoming requests and check if there is a key ‘Authorization’ in the request header or not. If it is there, then to check if it contains a value ‘Bearer’ or not.**

**Create a folder ‘Filters’ in the application solution. Create a class ‘CustomAuthFilter’ to filter requests. Inherit ActionFilterAttribute. Override OnActionExecuting method to check if the request object has Header ‘Authorization’ or not. If not, throw BadRequestResult with the message**

**Invalid request - No Auth token**

**If the header is present, then check if the value contains the word ‘Bearer’. If not, throw BadRequestResult with the message**

**Invalid request - Token present but Bearer unavailable**

**Add an attribute CustomAuthFilter to the Employee controller to filter any request to check for the Authorization token in the request header.**

**3.** **Custom Exception filter**

**Create a class ‘CustomExceptionFilter’ to catch the exceptions occuring the application. Implement IExceptionFilter thru the OnException method  
  
 Use the exception context to fetch the exception detail. Capture that and write it to a File in the system.  
  
 Set the Result property of the exception context to ExceptionResult.  
  
 Throw an exception in GET action method.  
 Ensure that the GET action method has ProducesResponseType for 500 - Internal server error  
  
 Use Swagger to test the exception and message being thrown.  
  
 Note: This needs WebApiCompatShim NuGet package installation**

**Program:**

**EmployeeController.cs**

**using Microsoft.AspNetCore.Mvc;**

**using Microsoft.AspNetCore.Authorization;**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**[ApiController]**

**[Route("[controller]")]**

**[AllowAnonymous]**

**[CustomAuthFilter]**

**public class EmployeeController : ControllerBase {**

**private 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, 1, 1),**

**Department = new Department { Id=1, 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>> GetStandard() {**

**return Ok(\_employees);**

**}**

**[HttpPost]**

**[ProducesResponseType(StatusCodes.Status201Created)]**

**public ActionResult<Employee> Create([FromBody] Employee emp) {**

**emp.Id = \_employees.Max(e => e.Id) + 1;**

**\_employees.Add(emp);**

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

**}**

**[HttpPut("{id}")]**

**public ActionResult Update(int id, [FromBody] Employee emp) {**

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

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

**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 NoContent();**

**}**

**}**

**Filter:**

**CustomAuthFilter.cs**

**using Microsoft.AspNetCore.Mvc;**

**using Microsoft.AspNetCore.Mvc.Filters;**

**public class CustomAuthFilterAttribute : 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 auth = headers["Authorization"].ToString();**

**if (!auth.Contains("Bearer")) {**

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

**}**

**}**

**}**

**CustomExceptionFilter.cs  
  
using Microsoft.AspNetCore.Mvc;**

**using Microsoft.AspNetCore.Mvc.Filters;**

**using System;**

**using System.IO;**

**public class CustomExceptionFilter : IExceptionFilter {**

**public void OnException(ExceptionContext context) {**

**var ex = context.Exception;**

**var logFile = Path.Combine(Directory.GetCurrentDirectory(), "Logs", "exceptions.txt");**

**Directory.CreateDirectory(Path.GetDirectoryName(logFile));**

**File.AppendAllText(logFile, $"{DateTime.Now}: {ex.GetType().Name} – {ex.Message}{Environment.NewLine}");**

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

**StatusCode = 500**

**};**

**context.ExceptionHandled = true;**

**}**

**}**

**Model:**

**Department.cs**

**public class Department {**

**public int Id { get; set; }**

**public string Name { get; set; }**

**}**

**[Employee.cs](http://employee.cs)  
public class Department {**

**public int Id { get; set; }**

**public string Name { get; set; }**

**}**

**[Skill.cs](http://skill.cs)  
public class Skill {**

**public int Id { get; set; }**

**public string Name { get; set; }**

**}**

**[Program.cs](http://program.cs)**

**var builder = WebApplication.CreateBuilder(args);**

**builder.Services.AddControllers(options =>**

**{**

**options.Filters.Add<CustomExceptionFilter>();**

**}).AddWebApiConventions();**

**builder.Services.AddEndpointsApiExplorer();**

**builder.Services.AddSwaggerGen();**

**var app = builder.Build();**

***// Force Swagger in all environments***

**app.UseSwagger();**

**app.UseSwaggerUI();**

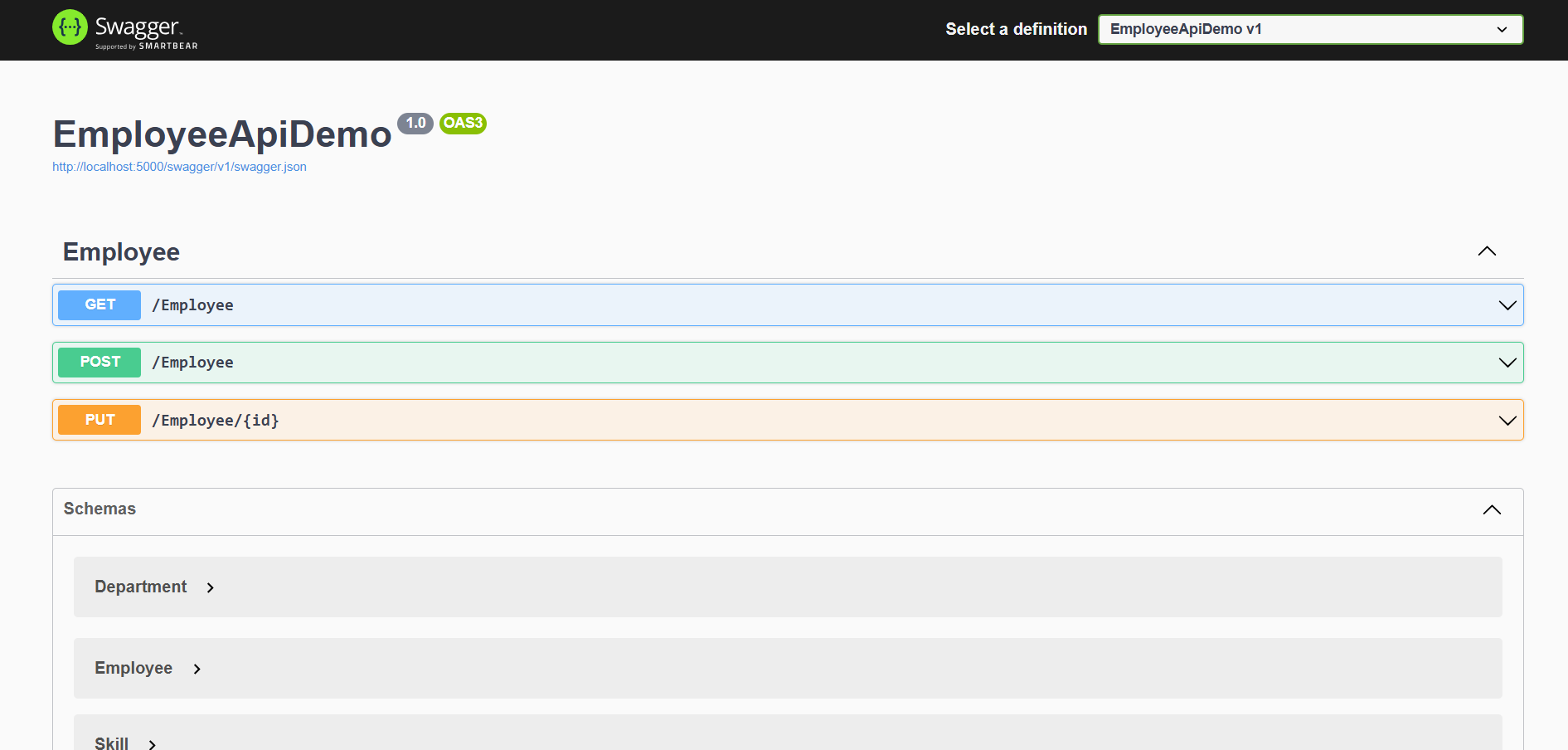
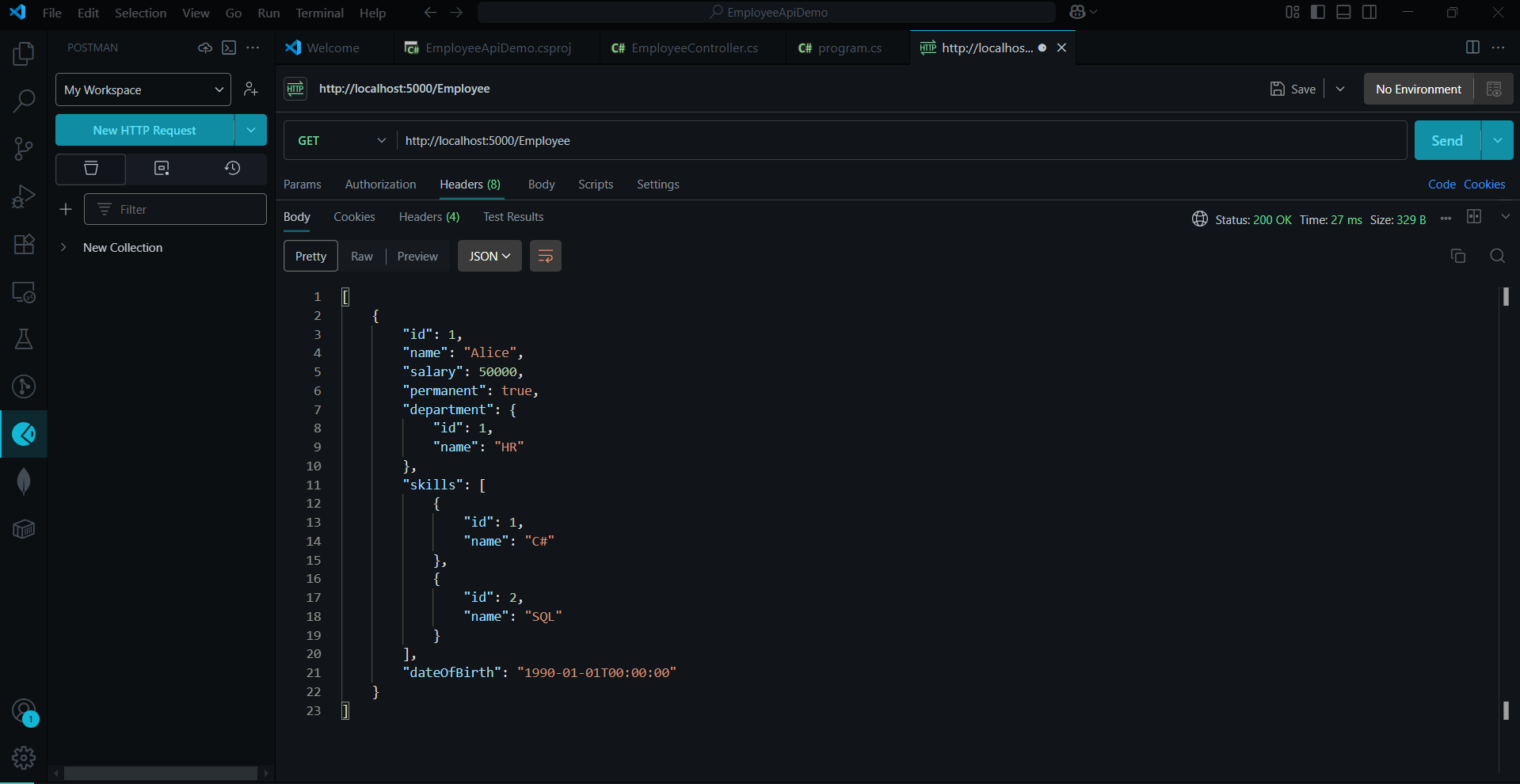
**app.UseHttpsRedirection(); *// 👉 Ensures HTTPS is handled***

**app.UseAuthorization();**

**app.MapControllers();**

**app.Run();**

**Output:  
Postman:**

**  
  
Objectives:**

**· Demonstrate creation of an Action method to perform data create, update & delete operation**

**o Use FromBody attribute, extract data to custom model class using FromBody attribute, use hardcoded data to update & delete data, Use Swagger and POSTMAN to test**

**4.** **Web Api CRUD operation**

**Update Employee data as per the input thru Web API PUT action method call**

**Employee information has to be updated based on the user input. Use Swagger tool to invoke the action method mapped with Http PUT action verb to update an employee data.**

**Modify the action method to return Employee data thru ActionResult.**

**Check if the id value is lesser than or equal to 0. If true, throw BadRequest action result with the message ‘Invalid employee id’**

**If the value is greater than 0 but not available in the list of employee ids that is there in the hardcoded list of employees, throw BadRequest action result with the same message as stated above.**

**If the id value is valid, use the JSON data from the input body and update the hardcoded list. Filter the employee list data for the input id and return that as the output.**

**Program:**

**Controller:**

**[EmployeeController.cs](http://employeecontroller.cs)**

**using Microsoft.AspNetCore.Mvc;**

**using Models;**

**using Filters;**

**using Microsoft.AspNetCore.Authorization;**

**namespace Controllers**

**{**

**[ApiController]**

**[Route("[controller]")]**

**[ServiceFilter(typeof(CustomAuthFilter))]**

**public class EmployeeController : ControllerBase**

**{**

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

**[HttpGet]**

**[AllowAnonymous]**

**[ProducesResponseType(StatusCodes.Status200OK)]**

**public ActionResult<List<Employee>> GetStandard()**

**{**

**return Ok(\_employees);**

**}**

**[HttpPost]**

**public ActionResult<Employee> Create([FromBody] Employee employee)**

**{**

**employee.Id = \_employees.Max(e => e.Id) + 1;**

**\_employees.Add(employee);**

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

**}**

**[HttpPut("{id}")]**

**public ActionResult<Employee> UpdateEmployee(int id, [FromBody] Employee emp)**

**{**

**if (id <= 0)**

**return BadRequest("Invalid employee id");**

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

**if (existingEmp == null)**

**return BadRequest("Invalid employee id");**

**existingEmp.Name = emp.Name;**

**existingEmp.Salary = emp.Salary;**

**existingEmp.Permanent = emp.Permanent;**

**existingEmp.Department = emp.Department;**

**existingEmp.Skills = emp.Skills;**

**existingEmp.DateOfBirth = emp.DateOfBirth;**

**return Ok(existingEmp);**

**}**

**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 = "C#" } },**

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

**}**

**};**

**}**

**}**

**}**

**Models:**

**namespace Models**

**{**

**public class Department**

**{**

**public int Id { get; set; }**

**public string? Name { get; set; }**

**}**

**}**

**using System;**

**using System.Collections.Generic;**

**namespace 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; }**

**}**

**}**

**namespace Models**

**{**

**public class Skill**

**{**

**public int Id { get; set; }**

**public string? Name { get; set; }**

**}**

**}**

**Demp api**

**<Project Sdk="Microsoft.NET.Sdk.Web">**

**<PropertyGroup>**

**<TargetFramework>net9.0</TargetFramework>**

**<Nullable>enable</Nullable>**

**<ImplicitUsings>enable</ImplicitUsings>**

**</PropertyGroup>**

**<ItemGroup>**

**<PackageReference Include="Swashbuckle.AspNetCore" Version="6.5.0" />**

**</ItemGroup>**

**</Project>**

**[program.cs](http://program.cs)**

**var builder = WebApplication.CreateBuilder(args);**

**builder.Services.AddControllers();**

**builder.Services.AddEndpointsApiExplorer();**

**builder.Services.AddSwaggerGen();**

**builder.Services.AddScoped<Filters.CustomAuthFilter>();**

**builder.Services.AddMvc(options => options.Filters.Add(typeof(Filters.CustomExceptionFilter)));**

**var app = builder.Build();**

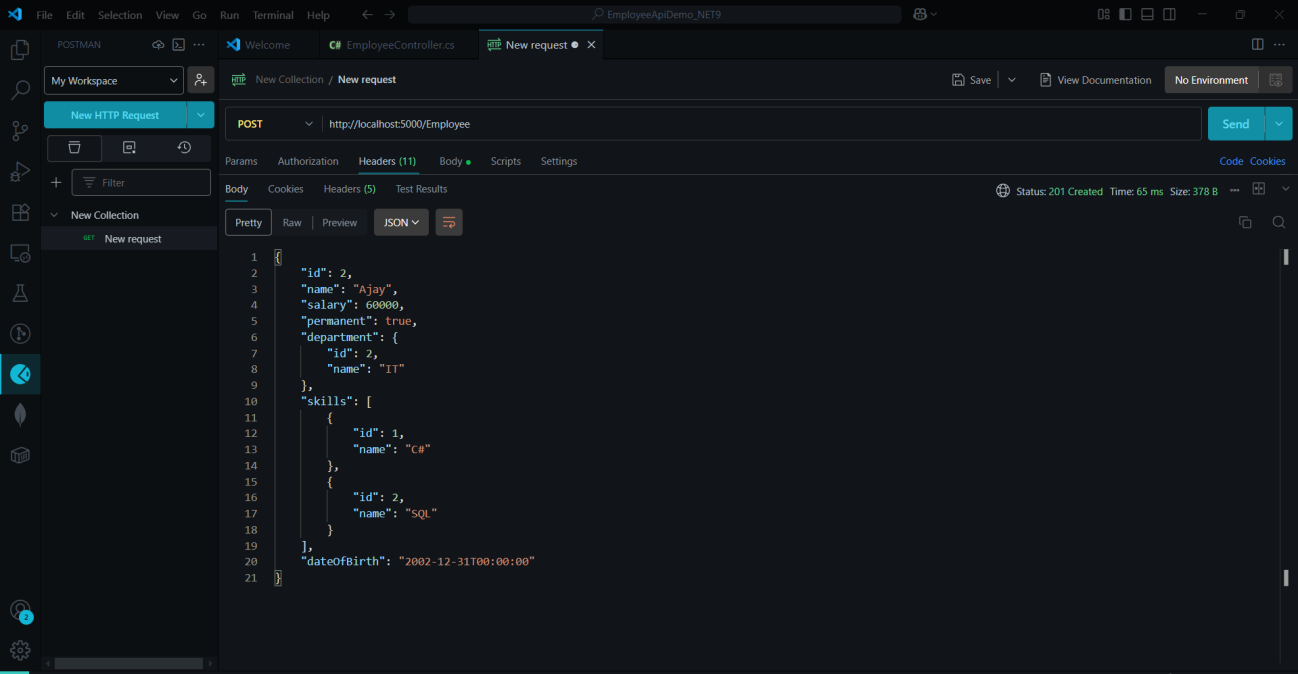
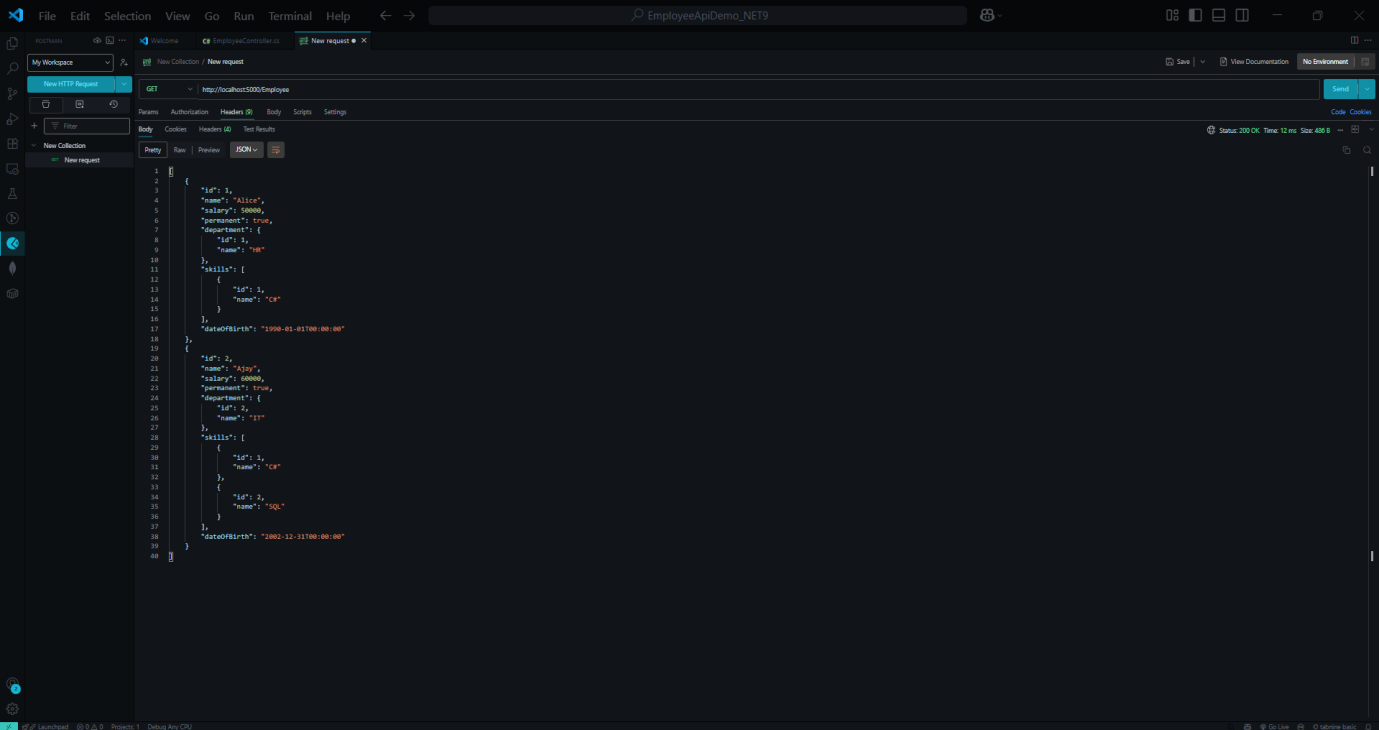
**app.UseSwagger();**

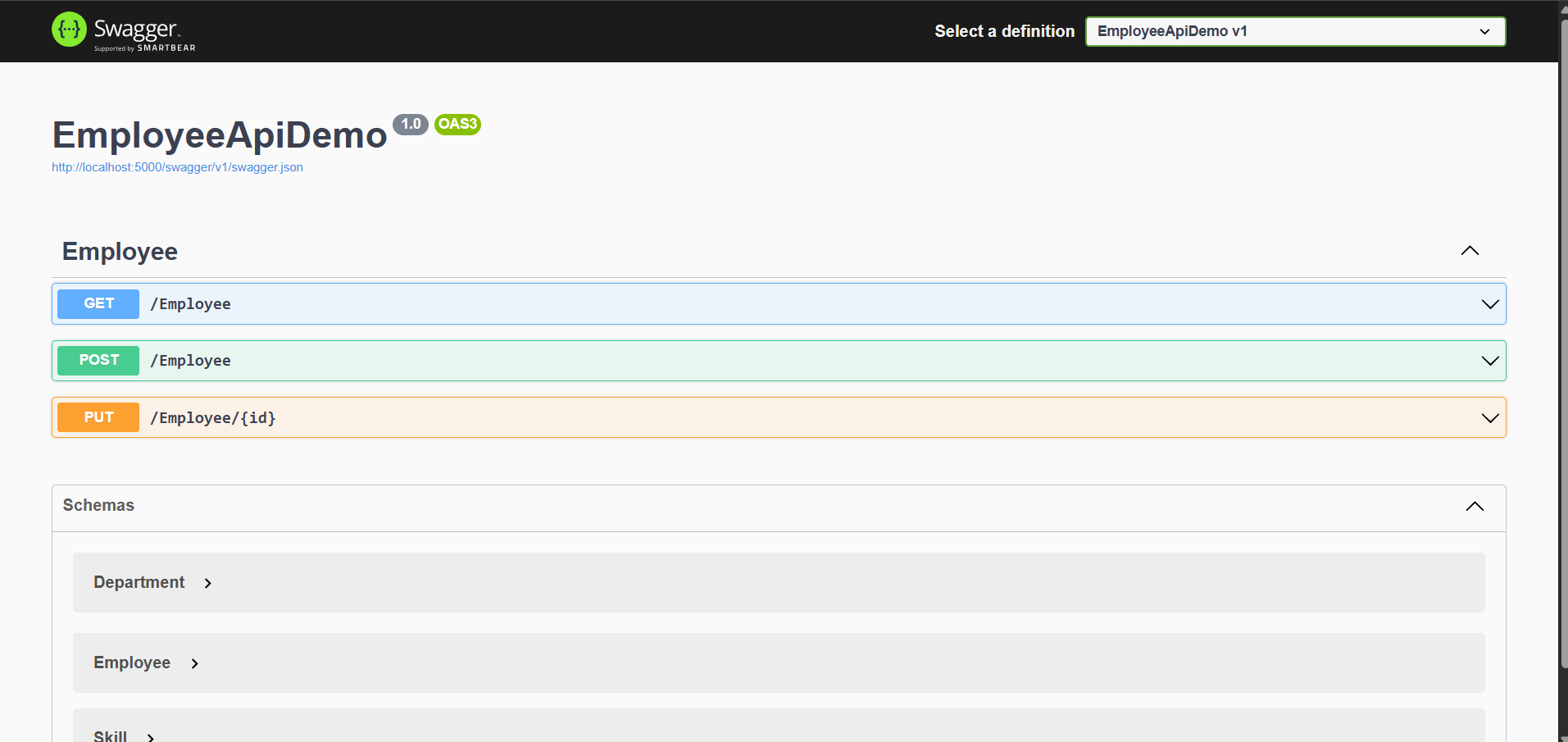
**app.UseSwaggerUI();**

**app.UseAuthorization();**

**app.MapControllers();**

**app.Run();**

**Output:  
**

**  
  
  
  
5Objectives:**

**· Demonstrate creation of an Action method to return list of custom class entity**

**o Model class creation, Use AllowAnonymous attribute, Use HttpGet action method**

**· Explain the usage of FromBody attribute**

**o Read the model object from request, other than the query string parameter**

**· Demonstrate Custom filter**

**o Usage of ActionFilterAttribute, OnActionExecuting method to intercept the request, Create filter for Custom exception - Need to install Microsoft.AspNetCore.Mvc.WebApiCompatShim package**

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

**Create a Custom class ‘Employee’ of the below defined structure**

**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; }**

**}**

**Create a new controller - EmployeeController with Read Write actions**

**Constructor: Create few records, HTTPGet, HTTPPost/HTTPPut**

**Create a Private method GetStandardEmployeeList that returns a List of Employee class. Invoke this method in the Get action method of the EmployeeController that was created in the previous step.**

**Public ActionResult<Employee> GetStandrad()**

**Modify the return type of the Get action method(without parameter) to return List of Employee class object**

**Add ProducesResponseType to the GET action method for Status code 200**

**Check the Swagger description for the GET method for success status code**

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

**The requirement is to intercept incoming requests and check if there is a key ‘Authorization’ in the request header or not. If it is there, then to check if it contains a value ‘Bearer’ or not.**

**Create a folder ‘Filters’ in the application solution. Create a class ‘CustomAuthFilter’ to filter requests. Inherit ActionFilterAttribute. Override OnActionExecuting method to check if the request object has Header ‘Authorization’ or not. If not, throw BadRequestResult with the message**

**Invalid request - No Auth token**

**If the header is present, then check if the value contains the word ‘Bearer’. If not, throw BadRequestResult with the message**

**Invalid request - Token present but Bearer unavailable**

**Add an attribute CustomAuthFilter to the Employee controller to filter any request to check for the Authorization token in the request header.**

**3.** **Custom Exception filter**

**Create a class ‘CustomExceptionFilter’ to catch the exceptions occuring the application. Implement IExceptionFilter thru the OnException method  
  
 Use the exception context to fetch the exception detail. Capture that and write it to a File in the system.  
  
 Set the Result property of the exception context to ExceptionResult.  
  
 Throw an exception in GET action method.  
 Ensure that the GET action method has ProducesResponseType for 500 - Internal server error  
  
 Use Swagger to test the exception and message being thrown.  
  
 Note: This needs WebApiCompatShim NuGet package installation**

**Programs**

**Controller:**

**[AuthController.cs](http://authcontroller.cs):**

**using Microsoft.AspNetCore.Authorization;**

**using Microsoft.AspNetCore.Mvc;**

**using Microsoft.IdentityModel.Tokens;**

**using System;**

**using System.Collections.Generic;**

**using System.IdentityModel.Tokens.Jwt;**

**using System.Security.Claims;**

**using System.Text;**

**namespace WebAPI\_JWT\_CORS\_Demo.Controllers**

**{**

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

**[ApiController]**

**[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("mysuperdupersecretkey@1234567890!"));**

**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);**

**}**

**}**

**}**

**[EmployeeController.cs](http://employeecontroller.cs):**

**using Microsoft.AspNetCore.Authorization;**

**using Microsoft.AspNetCore.Mvc;**

**namespace WebAPI\_JWT\_CORS\_Demo.Controllers**

**{**

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

**[ApiController]**

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

**public class EmployeeController : ControllerBase**

**{**

**[HttpGet]**

**public IActionResult GetData()**

**{**

**return Ok(new { message = "Authorized access!" });**

**}**

**}**

**}**

**[Program.cs](http://program.cs):**

**using Microsoft.AspNetCore.Hosting;**

**using Microsoft.Extensions.Hosting;**

**namespace WebAPI\_JWT\_CORS\_Demo**

**{**

**public class Program**

**{**

**public static void Main(string[] args)**

**{**

**CreateHostBuilder(args).Build().Run();**

**}**

**public static IHostBuilder CreateHostBuilder(string[] args) =>**

**Host.CreateDefaultBuilder(args)**

**.ConfigureWebHostDefaults(webBuilder =>**

**{**

**webBuilder.UseStartup<Startup>();**

**});**

**}**

**}**

**[Startup.cs](http://startup.cs)**

**using Microsoft.AspNetCore.Builder;**

**using Microsoft.AspNetCore.Hosting;**

**using Microsoft.AspNetCore.Authentication.JwtBearer;**

**using Microsoft.Extensions.DependencyInjection;**

**using Microsoft.Extensions.Hosting;**

**using Microsoft.IdentityModel.Tokens;**

**using System.Text;**

**namespace WebAPI\_JWT\_CORS\_Demo**

**{**

**public class Startup**

**{**

**public void ConfigureServices(IServiceCollection services)**

**{**

***// Enable CORS for local frontend testing***

**services.AddCors(options =>**

**{**

**options.AddPolicy("AllowAll", builder =>**

**builder.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader());**

**});**

***// JWT Authentication setup***

**string securityKey = "mysuperdupersecretkey@1234567890!";**

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

**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**

**};**

**});**

**services.AddControllers();**

**}**

**public void Configure(IApplicationBuilder app, IWebHostEnvironment env)**

**{**

**app.UseCors("AllowAll");**

**if (env.IsDevelopment())**

**{**

**app.UseDeveloperExceptionPage();**

**}**

**app.UseRouting();**

**app.UseAuthentication();**

**app.UseAuthorization();**

**app.UseEndpoints(endpoints =>**

**{**

**endpoints.MapControllers();**

**});**

**}**

**}**

**}**

**WebAPI\_JWT\_CORS\_Demo.csproj**

**<Project Sdk="Microsoft.NET.Sdk.Web">**

**<PropertyGroup>**

**<TargetFramework>net9.0</TargetFramework>**

**</PropertyGroup>**

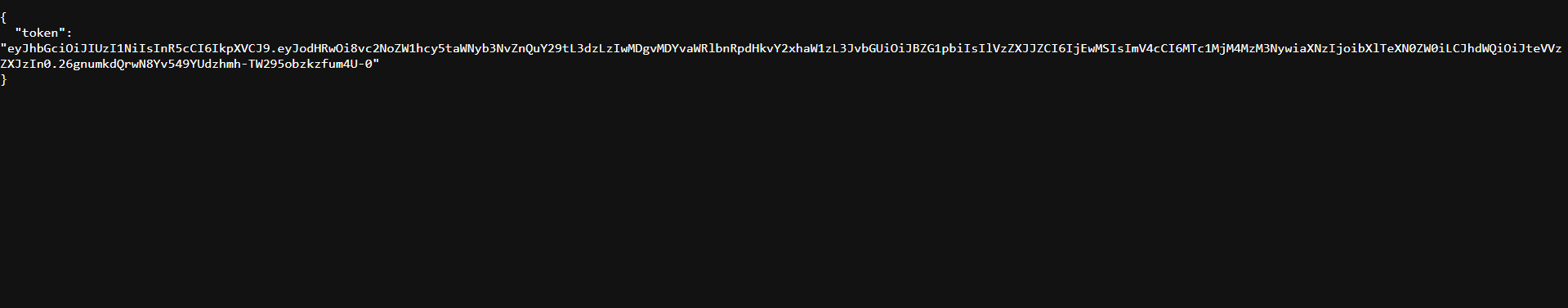
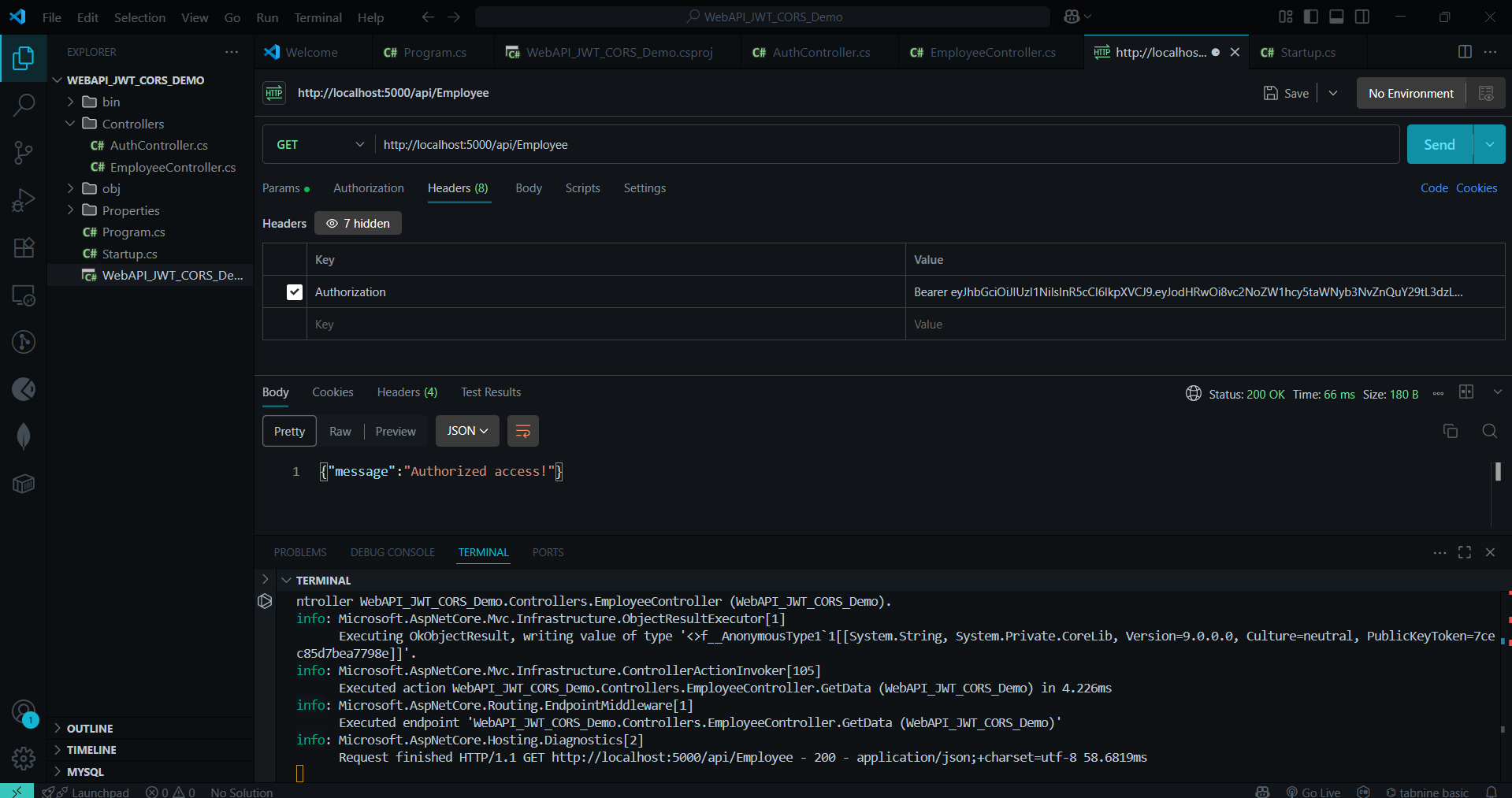
**<ItemGroup>**

**<PackageReference Include="Microsoft.AspNetCore.Authentication.JwtBearer" Version="9.0.0-preview.3.24172.9" />**

**</ItemGroup>**

**</Project>**

**Output:**

**PostMan:  
**