1. Create new folder as model
2. In that create classes employeeV2.cs
   1. EmployeeV2.cs

| namespace WebApplication1.Models  {  public class EmployeeV2  {  public int Id { get; set; }  public string Name { get; set; } = string.Empty;  public int Salary { get; set; }  public bool Permanent { get; set; }  public Department Department { get; set; } = default!;  public List<Skill> Skills { get; set; } = new();  public DateTime DateOfBirth { get; set; }  }  public class Department  {  public int Id { get; set; }  public string Name { get; set; } = string.Empty;  }  public class Skill  {  public int Id { get; set; }  public string Name { get; set; } = string.Empty;  }  } |
| --- |

1. Create new controller with Create ,Read ,Update And Delete
   1. EmployeeV2Controller.cs

| using Microsoft.AspNetCore.Mvc;  using WebApplication1.Filters;  using WebApplication1.Models;  using System.Linq;  namespace WebApplication1.Controllers  {  [ApiController]  [Route("api/emp2")]  [ServiceFilter(typeof(CustomAuthFilter))]  public class EmployeeV2Controller : ControllerBase  {  private static readonly List<EmployeeV2> \_data = new()  {  new EmployeeV2  {  Id = 1,  Name = "Furnia",  Salary = 45000,  Permanent = true,  DateOfBirth = new DateTime(1996, 4, 15),  Department = new WebApplication1.Models.Department { Id = 1, Name = "IT" }, Skills = new List<Skill> {  new Skill { Id = 1, Name = "C#" },  new Skill { Id = 2, Name = "SQL" }  }  }  };  [HttpGet(Name = "GetAllEmployeesV2")]  [ProducesResponseType(typeof(IEnumerable<EmployeeV2>), 200)]  [ProducesResponseType(500)]  public ActionResult<IEnumerable<EmployeeV2>> GetStandard()  {  return Ok(\_data);  }  [HttpPost]  [ProducesResponseType(typeof(EmployeeV2), 201)]  public ActionResult<EmployeeV2> Post([FromBody] EmployeeV2 emp)  {  emp.Id = \_data.Max(e => e.Id) + 1;  \_data.Add(emp);  return CreatedAtAction(nameof(GetStandard), new { id = emp.Id }, emp);  }  [HttpPut("{id}")]  [ProducesResponseType(typeof(EmployeeV2), 200)]  [ProducesResponseType(400)]  public ActionResult<EmployeeV2> Put(int id, [FromBody] EmployeeV2 updated)  {  if (id <= 0)  return BadRequest("Invalid employee id");  var emp = \_data.FirstOrDefault(e => e.Id == id);  if (emp is null)  return BadRequest("Invalid employee id");  emp.Name = updated.Name;  emp.Salary = updated.Salary;  emp.Permanent = updated.Permanent;  emp.DateOfBirth = updated.DateOfBirth;  emp.Department = updated.Department;  emp.Skills = updated.Skills;  return Ok(emp);  }  [HttpDelete("{id}")]  [ProducesResponseType(204)]  [ProducesResponseType(400)]  public IActionResult Delete(int id)  {  if (id <= 0)  return BadRequest("Invalid employee id");  var emp = \_data.FirstOrDefault(e => e.Id == id);  if (emp is null)  return BadRequest("Invalid employee id");  \_data.Remove(emp);  return NoContent();  }  }  } |
| --- |

1. Create program.cs
   1. program.cs

| using Microsoft.OpenApi.Models;  using WebApplication1.Filters;  var builder = WebApplication.CreateBuilder(args);  builder.WebHost.UseUrls("http://localhost:5050");  builder.Services.AddControllers(opts =>  {  opts.Filters.Add<CustomExceptionFilter>();  });  builder.Services.AddScoped<CustomAuthFilter>();  builder.Services.AddEndpointsApiExplorer();  builder.Services.AddSwaggerGen(c =>  {  c.SwaggerDoc("v1", new OpenApiInfo  {  Title = "Swagger Demo",  Version = "v1",  Description = "TBD",  TermsOfService = new Uri("https://none"),  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")  }  });  c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme  {  Name = "Authorization",  Type = SecuritySchemeType.ApiKey,  Scheme = "Bearer",  BearerFormat = "JWT",  In = ParameterLocation.Header,  Description = "Enter \*\*Bearer\*\* \_space\_ then the token.\nExample: `Bearer abc123`"  });  c.AddSecurityRequirement(new OpenApiSecurityRequirement  {  {  new OpenApiSecurityScheme  {  Reference = new OpenApiReference  {  Type = ReferenceType.SecurityScheme,  Id = "Bearer"  }  },  Array.Empty<string>()  }  });  });  var app = builder.Build();  app.UseSwagger();  app.UseSwaggerUI(c =>  {  c.SwaggerEndpoint("/swagger/v1/swagger.json", "Swagger Demo");  });  app.UseAuthorization();  app.MapControllers();  app.Run(); |
| --- |

1. Create Custom Authorization Filter
2. New folder as filter and add class
   1. customAuthFilter.cs

| using Microsoft.AspNetCore.Mvc;  using Microsoft.AspNetCore.Mvc.Filters;  namespace WebApplication1.Filters  {  public class CustomAuthFilter : ActionFilterAttribute  {  public override void OnActionExecuting(ActionExecutingContext context)  {  var hasAuth = context.HttpContext.Request.Headers.TryGetValue("Authorization", out var token);  if (!hasAuth)  {  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");  }  }  }  } |
| --- |

1. Create Custom Exception Filter
2. Add new class in filter folder
   1. CustomExceptionFilter.cs

| using Microsoft.AspNetCore.Mvc;  using Microsoft.AspNetCore.Mvc.Filters;  namespace WebApplication1.Filters  {  public class CustomExceptionFilter : IExceptionFilter  {  public void OnException(ExceptionContext context)  { var exception = context.Exception;  var logPath = "logs/errors.txt";  Directory.CreateDirectory("logs");  File.AppendAllText(logPath, $"{DateTime.Now}: {exception.Message}\n");  context.Result = new ObjectResult("An internal error occurred.")  {  StatusCode = 500  }; } }} |
| --- |

