**3.WebAPI Handson**

Web API – Employee CRUD with Custom Filters

**1. Objective**

• Demonstrate creation of an Action method to return list of custom class entity.  
• Use AllowAnonymous attribute, HttpGet, and FromBody attribute.  
• Demonstrate Custom Filters using ActionFilterAttribute and Exception Filter.

**2. Tools Used**

• Microsoft Visual Studio 2022  
• ASP.NET Core Web API (.NET 6 or 7)  
• Swagger UI for API Testing

**3. Model Classes**

Employee.cs

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; }  
}

Department.cs

public class Department  
{  
 public int Id { get; set; }  
 public string? Name { get; set; }  
}

Skill.cs

public class Skill  
{  
 public int Id { get; set; }  
 public string? Name { get; set; }  
}

**4. EmployeeController.cs**

[ApiController]  
[Route("api/[controller]")]  
[TypeFilter(typeof(CustomAuthFilter))]  
public class EmployeeController : ControllerBase  
{  
 private static List<Employee> employees = new List<Employee>();  
  
 public EmployeeController()  
 {  
 if (!employees.Any())  
 {  
 employees = GetStandardEmployeeList();  
 }  
 }  
  
 private List<Employee> GetStandardEmployeeList()  
 {  
 return new List<Employee>  
 {  
 new Employee  
 {  
 Id = 1,  
 Name = "Alice",  
 Salary = 60000,  
 Permanent = true,  
 Department = new Department { Id = 1, Name = "IT" },  
 Skills = new List<Skill> { new Skill { Id = 1, Name = "C#" } },  
 DateOfBirth = new DateTime(1990, 5, 23)  
 }  
 };  
 }  
  
 [HttpGet("standard")]  
 [ProducesResponseType(typeof(List<Employee>), 200)]  
 [ProducesResponseType(500)]  
 public ActionResult<List<Employee>> GetStandard()  
 {  
 throw new Exception("Sample exception for testing");  
 }  
}

**5. CustomAuthFilter.cs**

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");  
 return;  
 }  
  
 base.OnActionExecuting(context);  
 }  
}

**6. CustomExceptionFilter.cs**

public class CustomExceptionFilter : IExceptionFilter  
{  
 public void OnException(ExceptionContext context)  
 {  
 var exception = context.Exception;  
 var path = Path.Combine(Directory.GetCurrentDirectory(), "logs.txt");  
 File.AppendAllText(path, $"[{DateTime.Now}] {exception.Message}{Environment.NewLine}");  
  
 context.Result = new ObjectResult("An error occurred. Please try again later.")  
 {  
 StatusCode = 500  
 };  
 }  
}

**7. Program.cs Changes**

builder.Services.AddControllers(options =>  
{  
 options.Filters.Add(typeof(CustomExceptionFilter));  
});  
  
app.UseSwagger();  
app.UseSwaggerUI();  
app.UseHttpsRedirection();  
app.UseAuthorization();  
app.MapControllers();

**8. Testing in Swagger**

• Run the project. Swagger UI opens at https://localhost:xxxx/swagger.  
• Try GET /api/Employee/standard → Throws exception and logs to logs.txt.  
• Try POST /api/Employee without Authorization header → Returns "Invalid request - No Auth token".  
• Click "Authorize" and enter "Bearer token" → Then POST will work.

**OUTPUT :**



