Assignment 3

# Objective

This assignment demonstrates advanced Web API concepts in ASP.NET Core, including:  
- Custom model classes  
- Action methods with [FromBody]  
- Custom authorization filter using ActionFilterAttribute  
- Global exception handling using IExceptionFilter  
- Integration with Swagger for testing

# Project Structure

The following folders and files were added or modified in the existing FirstWebAPI project:

- Models/  
 - Employee.cs  
 - Department.cs  
 - Skill.cs  
- Filters/  
 - CustomAuthFilter.cs  
 - CustomExceptionFilter.cs  
- Controllers/  
 - EmployeeController.cs (Modified)  
- Program.cs (Modified)

# Model Classes

Model classes represent complex objects in the application. They are used in request and response bodies.

## 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 SkillName { get; set; }  
}

# Custom Authorization Filter

A custom filter was implemented to check if the request contains an 'Authorization' header with the value containing 'Bearer'. If not, a 400 BadRequest is returned.

## 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);  
 }  
}

# Custom Exception Filter

An exception filter was implemented to log exceptions to a file and return a 500 Internal Server Error.

## CustomExceptionFilter.cs

public class CustomExceptionFilter : IExceptionFilter  
{  
 public void OnException(ExceptionContext context)  
 {  
 var exception = context.Exception;  
 string filePath = Path.Combine(Directory.GetCurrentDirectory(), "error\_log.txt");  
 File.AppendAllText(filePath, $"[{DateTime.Now}] Exception: {exception.Message}\n");  
  
 context.Result = new ObjectResult("An unexpected error occurred.")  
 {  
 StatusCode = 500  
 };  
 }  
}

# EmployeeController.cs

This controller includes GET, POST, and PUT methods for Employee. It uses both custom filters.

# Program.cs

This file was updated to register filters and enable Swagger UI with the Authorize button.

# Testing in Swagger

The following scenarios were tested using Swagger UI:  
  
1. GET /api/Employee without Authorization → 400 Bad Request  
2. GET /api/Employee with Bearer token → 500 Internal Server Error, log created  
3. POST /api/Employee with Bearer token → 200 OK, employee added

Screenshots of Swagger responses and error\_log.txt can be added below.





