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

1. **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.

1. **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

CODE:

EmployeeController.cs (Updated)

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using MyWebApi.Filters; // Ensure this is present

using MyWebApi.Models;

namespace MyWebApi.Controllers

{

[Route("api/Emp")]

[ApiController]

[CustomAuthFilter]

[CustomExceptionFilter]

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 = "John Doe", Salary = 50000, Permanent = true, Department = new Department { Id = 1, Name = "IT" }, Skills = new List<Skill> { new Skill { Id = 1, Name = "C#" } }, DateOfBirth = new System.DateTime(1990, 1, 1) },

new Employee { Id = 2, Name = "Jane Smith", Salary = 60000, Permanent = false, Department = new Department { Id = 2, Name = "HR" }, Skills = new List<Skill> { new Skill { Id = 2, Name = "HRM" } }, DateOfBirth = new System.DateTime(1985, 5, 10) }

};

}

[HttpGet]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public ActionResult<List<Employee>> GetStandard()

{

throw new Exception("Test exception for CustomExceptionFilter");

//return Ok(\_employees);

}

[HttpGet("ByName")]

[ActionName("GetByName")]

[ProducesResponseType(StatusCodes.Status200OK)]

public ActionResult<List<Employee>> GetByName()

{

return Ok(GetStandardEmployeeList());

}

}

}

CustomAuthFilter.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace MyWebApi.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

if (!context.HttpContext.Request.Headers.ContainsKey("Authorization"))

{

context.Result = new BadRequestObjectResult("Invalid request - No Auth token");

return;

}

var authHeader = context.HttpContext.Request.Headers["Authorization"].ToString();

if (!authHeader.Contains("Bearer"))

{

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

}

base.OnActionExecuting(context);

}

}

}

CustomExceptionFilter.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System.IO;

namespace MyWebApi.Filters

{

public class CustomExceptionFilter : ExceptionFilterAttribute

{

public override void OnException(ExceptionContext context)

{

var exceptionMessage = context.Exception.Message;

File.WriteAllText("error.log", exceptionMessage);

context.Result = new ObjectResult("Internal Server Error")

{

StatusCode = 500

};

context.ExceptionHandled = true;

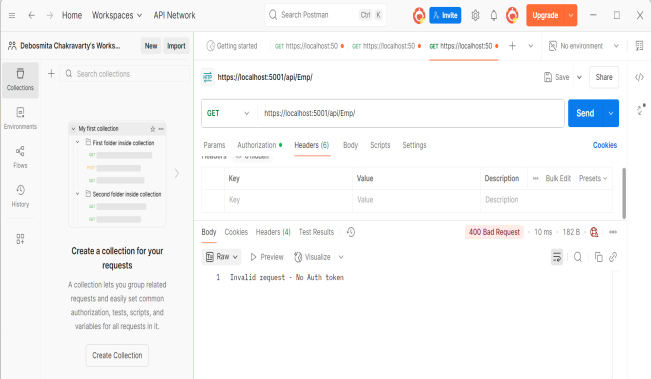
}

}

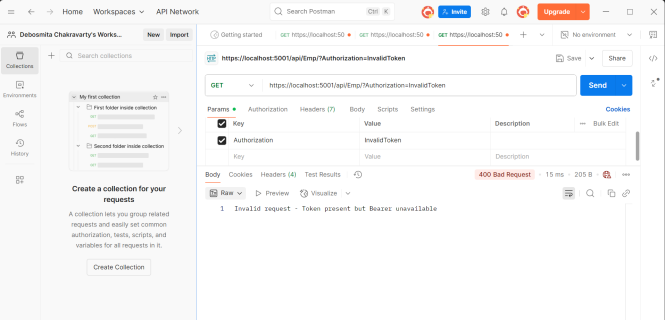
}

OUTPUT:

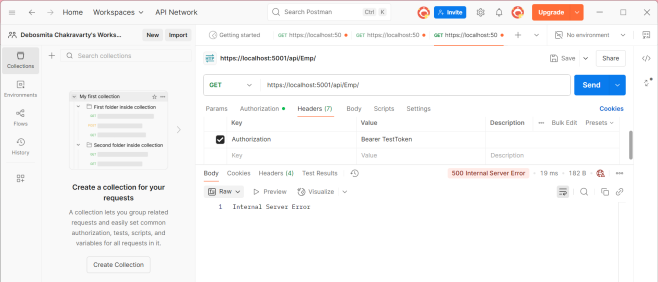
1. POSTMAN without Headers.



1. POSTMAN with Invalid Authorization Header.



1. POSTMAN with Valid Authorization Header.



1. Using Swagger

