**Name: Deepanshu Rathore**

**Superset ID: 6358199**  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Week 4  
**3. WebApi\_Handson**In this assignment I learned how to design a Web API endpoint that returns a collection of a custom data type. The core of this lies in defining a model class named Employee with properties such as Id, Name, Salary, Permanent, Department, Skills and DateOfBirth. By decorating the controller action with the attribute AllowAnonymous I enable access without requiring authentication. The HttpGet attribute indicates that this method will respond to GET requests. When the method signature is changed to return an ActionResult<List<Employee>> it provides a consistent way to return either the desired data or an error code. The ProducesResponseType attribute further documents that a 200 status code indicates successful retrieval of the list.

The FromBody attribute plays a crucial role when handling incoming POST or PUT requests. Without this attribute the ASP . NET Core framework binds parameters from the query string or route data. By specifying FromBody on a complex type parameter, the framework knows to deserialize the JSON payload of the request into the model object. This technique is helpful for scenarios where the client sends large or nested data structures rather than simple primitive values via the query string.

Custom filters allow interception of the request pipeline at various points. By creating a class CustomAuthFilter that derives from ActionFilterAttribute and overriding OnActionExecuting, I can inspect the HTTP request headers before the controller action runs. In this case the filter checks for an Authorization header and verifies it contains the word Bearer. If the header is missing or malformed a BadRequestResult is thrown. Applying this filter attribute to the controller ensures that every request to the EmployeeController passes through this validation logic.

Exception handling at a global level is facilitated by implementing IExceptionFilter. The CustomExceptionFilter class captures unhandled exceptions in its OnException method. Using the ExceptionContext it retrieves the exception details and logs them to a file. It then sets the Result property of the context to an ObjectResult or ExceptionResult which returns a 500 Internal Server Error. By adding ProducesResponseType for status code 500 to the action method, the API documentation clearly indicates that server errors may occur and what response format to expect.

EmployeeController.cs:  
using ApiWithSwagger.Filters;

using ApiWithSwagger.Models;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

namespace ApiWithSwagger.Controllers

{

[ApiController]

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

[ServiceFilter(typeof(CustomAuthFilter))]

public class EmployeeController : ControllerBase

{

private static List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "Alice",

Salary = 60000,

Permanent = true,

Department = new Department

{

Id = 1,

Name = "HR"

},

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "Communication" },

new Skill { Id = 2, Name = "Recruitment" }

},

DateOfBirth = new DateTime(1990, 5, 12)

}

};

}

[HttpGet]

[AllowAnonymous]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public ActionResult<List<Employee>> GetStandard()

{

try

{

var list = GetStandardEmployeeList();

return Ok(list);

}

catch

{

throw;

}

}

[HttpPost]

[ProducesResponseType(StatusCodes.Status201Created)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

public ActionResult<Employee> CreateEmployee([FromBody] Employee newEmployee)

{

if (newEmployee == null)

return BadRequest("Employee data is required");

var list = GetStandardEmployeeList();

list.Add(newEmployee);

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

}

[HttpPut("{id}")]

[ProducesResponseType(StatusCodes.Status204NoContent)]

[ProducesResponseType(StatusCodes.Status404NotFound)]

public IActionResult UpdateEmployee(int id, [FromBody] Employee updatedEmployee)

{

var list = GetStandardEmployeeList();

var existing = list.FirstOrDefault(e => e.Id == id);

if (existing == null)

return NotFound($"No employee found with Id = {id}");

existing.Name = updatedEmployee.Name;

existing.Salary = updatedEmployee.Salary;

existing.Permanent = updatedEmployee.Permanent;

existing.Department = updatedEmployee.Department;

existing.Skills = updatedEmployee.Skills;

existing.DateOfBirth = updatedEmployee.DateOfBirth;

return NoContent();

}

}

}

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.