**WEEK 4 WEB API HANSON**

**Exercise 4:**

**CustomAuthFilter.cs:** using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace EmployeeApi.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

if (!context.HttpContext.Request.Headers.TryGetValue("Authorization", out var token))

{

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");

return;

}

base.OnActionExecuting(context);

}

}

}

**Program.cs:**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

using EmployeeApi.Filters;

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

var key = "secretlooooongkeys1234567adgezedffe";

var symmetricSecurityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(key));

builder.Services.AddControllers(options =>

{

options.Filters.Add<CustomExceptionFilter>();

});

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(options =>

{

options.SwaggerDoc("v1", new OpenApiInfo { Title = "Employee API", Version = "v1" });

options.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

Name = "Authorization",

Type = SecuritySchemeType.ApiKey,

Scheme = "Bearer",

BearerFormat = "JWT",

In = ParameterLocation.Header,

Description = "Enter: Bearer <your JWT token>"

});

options.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme {

Reference = new OpenApiReference {

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

new string[] {}

}

});

});

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddJwtBearer(options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = "mySystem",

ValidAudience = "myUsers",

IssuerSigningKey = symmetricSecurityKey

};

});

builder.Services.AddScoped<CustomAuthFilter>();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

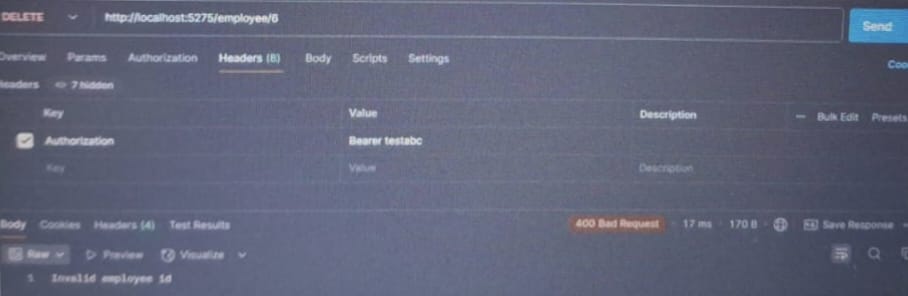
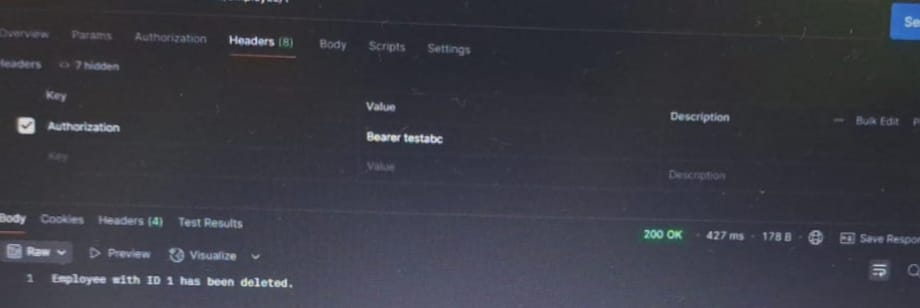
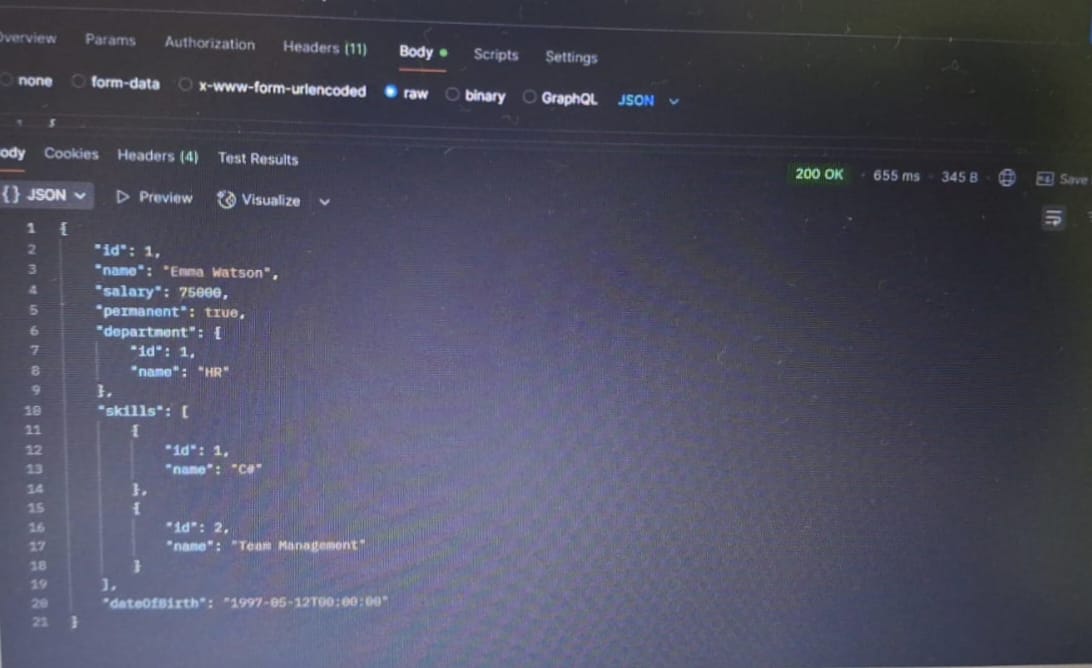
app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

**OUTPUT**

****