**CODE:**

**Department.cs**

namespace EmployeeWebApi.Models

{

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

}

}

**Employee.cs**

using System;

using System.Collections.Generic;

namespace EmployeeWebApi.Models

{

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; }

}

}

**Skill.cs**

namespace EmployeeWebApi.Models

{

public class Skill

{

public int Id { get; set; }

public string Name { get; set; }

}

}

**EmployeeController.cs**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using JwtEmployeeApi.Models;

namespace JwtEmployeeApi.Controllers

{

[ApiController]

[Route("[controller]")]

[Authorize(Roles = "Admin,POC")]

public class EmployeeController : ControllerBase

{

private static List<Employee> \_employees = new List<Employee>

{

new Employee

{

Id = 1,

Name = "John",

Salary = 50000,

Permanent = true,

Department = new Department { Id = 101, Name = "IT" },

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "C#" },

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

},

DateOfBirth = new DateTime(1990, 1, 1)

}

};

[HttpGet("standard")]

public ActionResult<List<Employee>> GetStandard()

{

return \_employees;

}

}

}

**AuthController.cs**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace JwtEmployeeApi.Controllers

{

[ApiController]

[Route("[controller]")]

[AllowAnonymous]

public class AuthController : ControllerBase

{

[HttpGet("token")]

public IActionResult GetToken()

{

var token = GenerateJSONWebToken(1, "Admin");

return Ok(token);

}

private string GenerateJSONWebToken(int userId, string userRole)

{

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperduperultrasecuresecretkey123!"));

var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);

var claims = new List<Claim>

{

new Claim(ClaimTypes.Role, userRole),

new Claim("UserId", userId.ToString())

};

var token = new JwtSecurityToken(

issuer: "mySystem",

audience: "myUsers",

claims: claims,

expires: DateTime.Now.AddMinutes(2),

signingCredentials: credentials);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

}

Program.cs

using System.Text;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

string securityKey = "mysuperduperultrasecuresecretkey123!";

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

builder.Services.AddAuthentication(x =>

{

x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(x =>

{

x.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = "mySystem",

ValidAudience = "myUsers",

IssuerSigningKey = symmetricSecurityKey

};

});

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthentication();

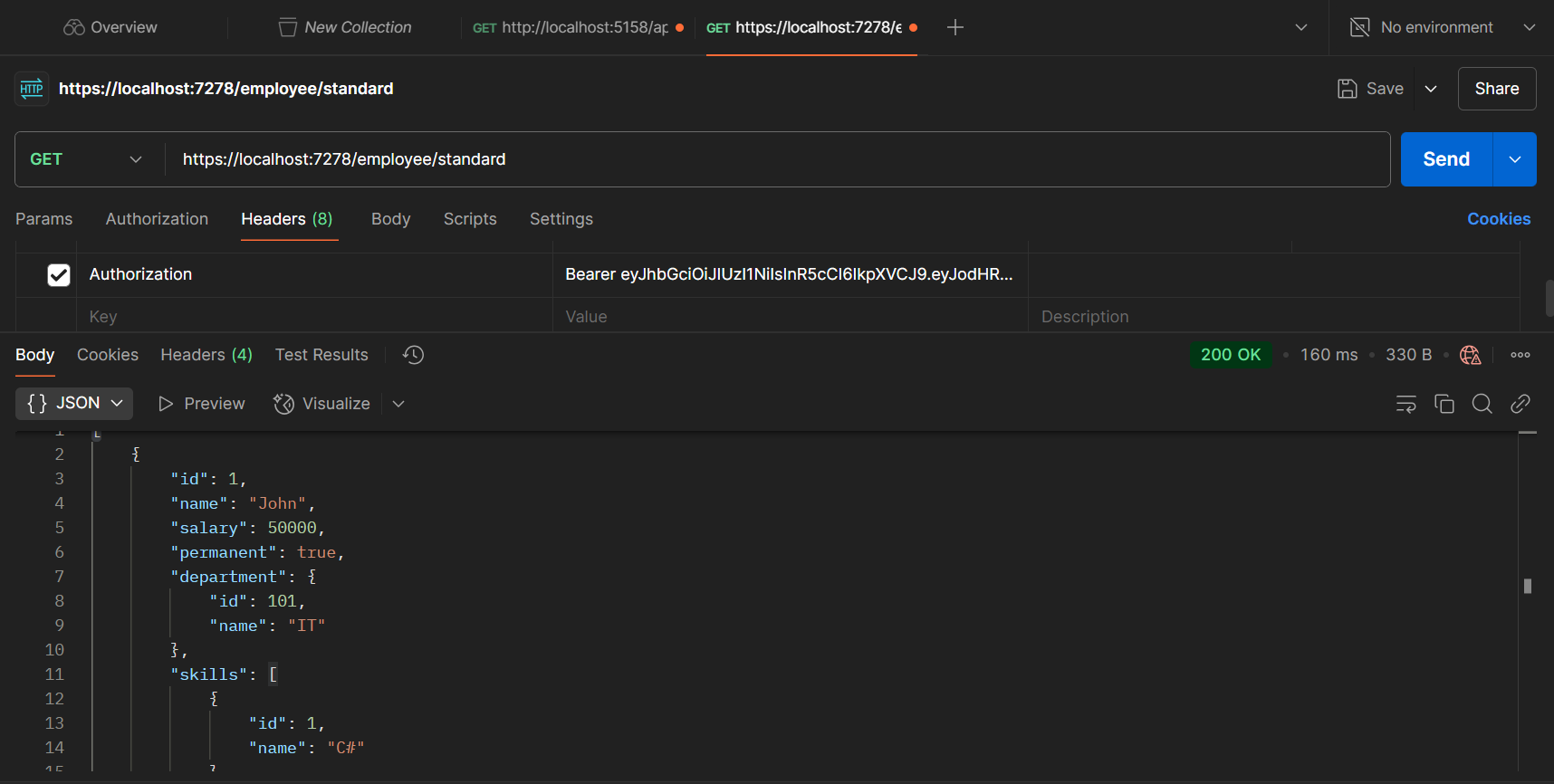
app.UseAuthorization();

app.MapControllers();

app.Run();

**OUTPUT**

**With JWT Token**



**Without JWT Token**

