CART.CS:

using System.ComponentModel.DataAnnotations.Schema;

namespace Shopping.Models

{

public class Cart

{

public int cartNumber { get; set; }

public string Username { get; set; }

[ForeignKey("Username")]

public User User { get; set; }

}

}

CARTDTO.CS:

using System.ComponentModel.DataAnnotations;

namespace Shopping.Models.DTOs

{

public class CartDTO

{

[Required(ErrorMessage = "Username is empty")]

public string Username { get; set; }

[Required(ErrorMessage = "Product Id is empty")]

public int ProductId { get; set; }

public int Quantity { get; set; } = 1;

}

}

CARTCONTROLLER:

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Shopping.Interfaces;

using Shopping.Models.DTOs;

namespace Shopping.Controllers

{

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

[ApiController]

public class CartController : ControllerBase

{

private readonly ICartService \_cartService;

public CartController(ICartService cartService)

{

\_cartService = cartService;

}

[HttpPost]

public IActionResult AddToCart(CartDTO cartDTO)

{

var result = \_cartService.AddToCart(cartDTO);

if (result)

return Ok(cartDTO);

return BadRequest("Could not add item to cart");

}

[HttpPost]

[Route("RemoveFromCart")]

public IActionResult RemoveFromCart(CartDTO cartDTO)

{

var result = \_cartService.RemoveFromCart(cartDTO);

if (result)

return Ok(cartDTO);

return BadRequest("Could not Remove item from cart");

}

}

}

ICARTSERVICE:

using Shopping.Models.DTOs;

namespace Shopping.Interfaces

{

public interface ICartService

{

bool AddToCart(CartDTO cartDTO);

bool RemoveFromCart(CartDTO cartDTO);

}

}

PROGRAM.CS:

using Microsoft.EntityFrameworkCore;

using Shopping.Contexts;

using Shopping.Interfaces;

using Shopping.Repositories;

using Shopping.Models;

using Shopping.Services;

using ShoppingApp.Reposittories;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

using Microsoft.OpenApi.Models;

namespace Shopping

{

public class Program

{

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

//builder.Services.AddControllersWithViews();

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(opt =>

{

opt.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

Name = "Authorization",

Type = SecuritySchemeType.Http,

Scheme = "Bearer",

BearerFormat = "JWT",

In = ParameterLocation.Header,

Description = "JWT Authorization header using the Bearer scheme."

});

opt.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 = false,

ValidateAudience = false,

IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Configuration["SecretKey"])),

ValidateIssuerSigningKey = true

};

});

builder.Services.AddDbContext<ShoppingContext>(opts =>

{

opts.UseSqlServer(builder.Configuration.GetConnectionString("conn"));

});

//builder.Logging.AddLog4Net();

builder.Services.AddScoped<IRepository<string, User>, UserRepository>();

builder.Services.AddScoped<IRepository<int, Product>, ProductRepository>();

builder.Services.AddScoped<IRepository<int, Cart>, CartRepository>();

builder.Services.AddScoped<IRepository<int, CartItems>, CartItemsRepository>();

builder.Services.AddScoped<IUserService, UserService>();

builder.Services.AddScoped<ITokenService, TokenService>();

builder.Services.AddScoped<IProductService, ProductService>();

builder.Services.AddScoped<ICartService, CartService>();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

//app.UseExceptionHandler("/Home/Error");

}

app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

//app.MapControllerRoute(

// name: "default",

// pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();

}

}

}

OUTPUT:









