Add the Remove from cart functionality to the api

Cartservice:

using ShoppingApp.Interfaces;

using ShoppingApp.Models.DTO;

using ShoppingApp.Models;

using Microsoft.EntityFrameworkCore.Metadata.Internal;

namespace ShoppingApp.Services

{

public class CartService : ICartService

{

private readonly IRepository<int, Cart> \_cartRepository;

private readonly IRepository<int, CartItems> \_cartItemRepository;

private readonly IRepository<int, Product> \_productRepository;

public CartService(IRepository<int, Cart> cartRepository,

IRepository<int, CartItems> cartItemRepository,

IRepository<int, Product> productRepository)

{

\_cartRepository = cartRepository;

\_cartItemRepository = cartItemRepository;

\_productRepository = productRepository;

}

CartItems CreateCartItem(int cartNumber, CartDTO cartDTO)

{

var product = \_productRepository.GetById(cartDTO.ProductId);

if (product != null)

{

var myCartItem = new CartItems

{

CartNumber = cartNumber,

Product\_Id = cartDTO.ProductId,

Quantity = cartDTO.Quantity,

Price = product.Price

};

return myCartItem;

}

return null;

}

public bool AddToCart(CartDTO cartDTO)

{

var cartCheck = \_cartRepository.GetAll().FirstOrDefault(c => c.Username == cartDTO.Username);

int cartNumber = 0;

if (cartCheck == null)

{

var cart = \_cartRepository.Add(new Cart { Username = cartDTO.Username });

cartNumber = cart.cartNumber;

}

else

cartNumber = cartCheck.cartNumber;

bool CartItemCheck = CheckIfCartItemAlreadyPresent(cartNumber, cartDTO.ProductId);

if (CartItemCheck)

{

return IncrementQuantityInCart(cartNumber, cartDTO);

}

var myCartItem = CreateCartItem(cartNumber, cartDTO);

if (myCartItem != null)

{

var result = \_cartItemRepository.Add(myCartItem);

if (result != null)

return true;

}

return false;

}

private bool IncrementQuantityInCart(int cartNumber, CartDTO cartDTO)

{

var cartItem = \_cartItemRepository.GetAll()

.FirstOrDefault(ci => ci.CartNumber == cartNumber && ci.Product\_Id == cartDTO.ProductId);

cartItem.Quantity += cartDTO.Quantity;

var result = \_cartItemRepository.Update(cartItem);

if (result != null)

return true;

return false;

}

private bool CheckIfCartItemAlreadyPresent(int cartNumber, int productId)

{

var cartItem = \_cartItemRepository.GetAll()

.FirstOrDefault(ci => ci.CartNumber == cartNumber && ci.Product\_Id == productId);

return cartItem != null ? true : false;

}

public bool RemoveFromCart(CartDTO cartDTO)

{

int cartNumber = 0;

var cartCheck = \_cartRepository.GetAll().FirstOrDefault(c => c.Username == cartDTO.Username);

if (cartCheck == null)

{

return false;

}

else

{

cartNumber = cartCheck.cartNumber;

}

bool CartItemCheck = CheckIfCartItemAlreadyPresent(cartNumber, cartDTO.ProductId);

if (CartItemCheck)

{

return DecrementQuantityInCart(cartNumber, cartDTO);

}

return false;

}

private bool DecrementQuantityInCart(int cartNumber, CartDTO cartDTO)

{

var cartItem = \_cartItemRepository.GetAll()

.FirstOrDefault(ci => ci.CartNumber == cartNumber && ci.Product\_Id == cartDTO.ProductId);

cartItem.Quantity -= cartDTO.Quantity;

int ProductId = cartDTO.ProductId;

if (cartItem.Quantity == 0)

{

var result1 = \_cartItemRepository.Delete(ProductId);

if (result1 != null)

return true;

}

else if (cartItem.Quantity < 0)

{

return false;

}

var result = \_cartItemRepository.Update(cartItem);

if (result != null)

return true;

return false;

}

}

}

CartController:

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using ShoppingApp.Interfaces;

using ShoppingApp.Models.DTO;

namespace ShoppingApp.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("Remove")]

public IActionResult RemoveFromCart(CartDTO cartDTO)

{

var result = \_cartService.RemoveFromCart(cartDTO);

if (result)

return Ok(cartDTO);

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

}

}

}

Icartservice

using ShoppingApp.Models.DTO;

namespace ShoppingApp.Interfaces

{

public interface ICartService

{

bool AddToCart(CartDTO cartDTO);

bool RemoveFromCart(CartDTO cartDTO);

}

}

CartDTO

using System.ComponentModel.DataAnnotations;

namespace ShoppingApp.Models.DTO

{

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;

}

}



