**Assignment**

**CartController.cs:**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using ShoppingApp.Interfaces;

using ShoppingApp.Models.DTOs;

namespace ShoppingApp.Controllers

{

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

[ApiController]

public class CartController : ControllerBase

{

private readonly ICartService \_cartService;

public CartController(ICartService cartService)

{

\_cartService = cartService;

}

/// <summary>

/// Adding items to the cart of particular user by authorizing the role as user.

/// </summary>

/// <param name="cartDTO">Cart items that should be added to the respective user</param>

/// <returns>This returns the added items or gives an error message</returns>

[Authorize(Roles ="User")]

[HttpPost("add")]

public IActionResult AddToCart(CartDTO cartDTO)

{

var result = \_cartService.AddToCart(cartDTO);

if (result)

return Ok(cartDTO);

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

}

/// <summary>

/// Removing items from the cart of particular user by authorizing the role as user.

/// </summary>

/// <param name="cartDTO">Cart items that should be removed to the respective user</param>

/// <returns>Returns either success message or error message</returns>

[Authorize(Roles = "User")]

[HttpPost("remove")]

public IActionResult RemoveFromCart(CartDTO cartDTO)

{

var result = \_cartService.RemoveFromCart(cartDTO);

if(result) return Ok("Cart Items Removed...");

return BadRequest("Could not remove item");

}

}

}

**CartService.cs:**

using Microsoft.CodeAnalysis;

using ShoppingApp.Interfaces;

using ShoppingApp.Models;

using ShoppingApp.Models.DTOs;

using ShoppingApp.Repositories;

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;

}

/// <summary>

/// Creates an cart item object which should be added in the cartitems.

/// </summary>

/// <param name="cartNumber">Cart number of the user.</param>

/// <param name="cartDTO">Items to be added in the cartItem</param>

/// <returns>Cart Item</returns>

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;

}

/// <summary>

/// Adds the product to the respective user's cart

/// </summary>

/// <param name="cartDTO">Item that needs to be added in the cart</param>

/// <returns>Returns boolean value</returns>

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;

}

/// <summary>

/// Increases the quantity in case of user adds same product.

/// </summary>

/// <param name="cartNumber">Cart number of the user</param>

/// <param name="cartDTO">Cart items that should be updated</param>

/// <returns>Returns boolean value</returns>

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;

}

/// <summary>

/// Checks if the product is already available in the user's cart.

/// </summary>

/// <param name="cartNumber">Cart number of the user</param>

/// <param name="productId">Product Id that should be added</param>

/// <returns>Returns boolean value</returns>

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;

}

/// <summary>

/// Removes items from the user's cart

/// </summary>

/// <param name="cartDTO">Item that should be removed</param>

/// <returns>Returns boolean value</returns>

public bool RemoveFromCart(CartDTO cartDTO)

{

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

int cartNumber = cartCheck.cartNumber;

var cartItem = \_cartItemRepository.GetById(cartNumber);

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

if (CartItemCheck)

{

var result = \_cartItemRepository.Delete(cartDTO.ProductId);

return true;

}

return false;

}

}

}

**Output:**

