public class OnlineShoppingRevisedController {

public ApexPages.StandardSetController purchaseOrderController{get;set;}

public List<PurchaseOrder\_\_c> purchaseOrderList{get;set;} //list to contain purchase history

public Map<Id,Product2> productList2{get;set;} //will contain all products originally

public List<Product2> productList{get;set;} //will be used for pagination

public Map<Id,Product2> duplicateProductMap{get;set;} //will be clone of productList2

public final map<Id,Product2> selectedProductList{get;set;} //this will contain the product selected for cart

public Map<String , Decimal> quantityAvailableMap; //this will contain productcode to quantity map used for

public List<PurchaseOrderLineItem\_\_c> cartItems{get;set;}

public Map<String,PurchaseOrderLineItem\_\_c> cartItems2{get;set;}

public List<PurchaseOrderLineItem\_\_c> finalCartItemsToBeInserted{get;set;}

public String searchKeyword{get;set;}

public Boolean flag{get;set;}

Public Boolean cartFlag{get;set;}

Public Boolean finalCartFlag{get;set;}

Public Boolean checkoutButtonFlag{get;set;}

public Integer invoiceNumber{get;set;}

public Map<Id,Boolean> selectedIdMap{get;set;}

public Decimal totalOrderPrice{get;set;}

public String productToBeDeleted{

get;

set{

productToBeDeleted = value;

}

}

public String productCodeOfQuantityUpdated{

get;

set{

productCodeOfQuantityUpdated = value;

}

}

public Integer quantityValue{

get;

set{

quantityValue = value;

}

}

Public Integer ResultSize1{

get;

set;

}

public Integer PageSize1{

get;

set;

}

private String sortDirection = 'ASC';

private String sortExp = 'name';

Public Boolean hasPrevious1 {get;set;}

Public Boolean hasNext1 {get;set;}

Public Integer pageNumber1 {get;set;}

//Constructor initialize the all required variables.

public OnlineShoppingRevisedController(ApexPages.StandardSetController stdController){

purchaseOrderController = stdController;

flag = false;

cartFlag = false;

finalCartFlag = false;

checkoutButtonFlag = false;

totalOrderPrice = 0;

ResultSize1 = 1;

PageSize1 = 1;

selectedProductList = new map<Id,Product2>();

productList = new List<Product2>();

productList2 = new Map<Id,Product2>();

duplicateProductMap = new Map<Id,Product2>();

//Purchase Order List

purchaseOrderList = [SELECT Id , OrderPrice\_\_c , OrderStatus\_\_c FROM PurchaseOrder\_\_c];

cartItems = new List<PurchaseOrderLineItem\_\_c>();

cartItems2 = new Map<String , PurchaseOrderLineItem\_\_c>();

finalCartItemsToBeInserted = new List<PurchaseOrderLineItem\_\_c>();

quantityAvailableMap = new Map<String , Decimal>();

}

//show the product table.

public void getProductList(){

flag = true;

hasPrevious1 = false;

hasNext1 = true;

pageNumber1 = 1;

PageSize1 = 10;

productList.clear();

selectedIdMap = new Map<Id , Boolean>();

productList2 = new Map<Id,Product2>([SELECT Name , ProductCode ,PricePerUnit\_\_c,Description, Quantity\_\_c FROM Product2 Order by Name]); //WHERE QuantityAvailable1\_\_c > 0

duplicateProductMap.putAll(productList2);

ResultSize1 = productList2.size();

for(Product2 currentProduct : productList2.values()){

quantityAvailableMap.put(currentProduct.ProductCode , currentProduct.Quantity\_\_c);

}

for(integer i = 0 ; i < 10 ; i++){

productList.add(productList2.values()[i]);

}

for(Product2 singleProduct : productList2.values()){

selectedIdMap.put(singleProduct.Id , false);

}

}

//function to handle search functionality

public void searchProduct(){

PageNumber1 = 1;

productList.clear();

duplicateProductMap.clear();

if(searchKeyword != null && searchKeyword != ''){

for(Product2 currentProduct : productList2.values()){

if(currentProduct.Name.startsWithIgnoreCase(searchKeyword)){

duplicateProductMap.put(currentProduct.Id , currentProduct);

}

}

ResultSize1 = duplicateProductMap.size();

if(ResultSize1 < PageSize1){

PageSize1 = ResultSize1;

hasPrevious1 = false;

hasNext1 = false;

}

for(integer i = 0 ; i < PageSize1 ; i++){

productList.add(duplicateProductMap.values()[i]);

}

if(PageSize1 == 0){

PageSize1 = 1;

}

if(ResultSize1 == 0){

ResultSize1 = 1;

}

}

else if(searchKeyword == ''){

productList.clear();

hasNext1 = true;

pageSize1 = 10;

duplicateProductMap.putAll(productList2);

ResultSize1 = duplicateProductMap.size();

for(integer i = 0 ; i < PageSize1 ; i++){

productList.add(productList2.values()[i]);

}

}

}

//this function will add product to cart.

public void addToCart(){

selectedProductList.clear();

for(Product2 singleProduct : productList2.values()){

if(selectedIdMap.get(singleProduct.Id) == true){

if(singleProduct.Quantity\_\_c != 0){

singleProduct.Quantity\_\_c--;

selectedProductList.put(singleProduct.Id,singleProduct);

}

}

}

List<Product2> tempListOfProductForCart = selectedProductList.values();

if(tempListOfProductForCart.size() > 0){

cartFlag = true;

for(Product2 currentProduct : selectedProductList.values()){

if(cartItems2.containsKey(currentProduct.ProductCode)){

cartItems2.get(currentProduct.ProductCode).QuantityAvailable1\_\_c++;

}

else{

PurchaseOrderLineItem\_\_c orderItem = new PurchaseOrderLineItem\_\_c(ProductName\_\_c = currentProduct.Name , ProductCode\_\_c = currentProduct.ProductCode ,

PricePerUnit\_\_c = currentProduct.PricePerUnit\_\_c , QuantityAvailable1\_\_c = 1 , Description\_\_c = currentProduct.Description

);

cartItems2.put(currentProduct.ProductCode , orderItem);

}

}

cartItems = cartItems2.values();

}

}

//this will delete product from cart.

public PageReference deleteFromCart(){

for(Product2 currentProduct : productList2.values() ){

if(currentProduct.ProductCode == productToBeDeleted){

currentProduct.Quantity\_\_c = currentProduct.Quantity\_\_c + cartItems2.get(productToBeDeleted).QuantityAvailable1\_\_c;

}

}

cartItems2.keySet().remove(productToBeDeleted);

cartItems = cartItems2.values();

return null;

}

//this method will generate final purchase items

public PageReference checkout(){

invoiceNumber = ((Integer)(Math.random()\*100000));

finalCartFlag = true;

for(PurchaseOrderLineItem\_\_c item : cartItems2.values()){

item.Total\_\_c = item.QuantityAvailable1\_\_c \* item.PricePerUnit\_\_c;

totalOrderPrice += item.Total\_\_c;

}

finalCartItemsToBeInserted = cartItems2.values();

return null;

}

//handle quantity change in cartItems

public PageReference updateProductListOnQuantityChange(){

Map<String,Decimal> mapOfCodeByQuantity = new map<string,Decimal>();

system.debug(productCodeOfQuantityUpdated);

if(cartItems2.containsKey(productCodeOfQuantityUpdated)){

mapOfCodeByQuantity.put(productCodeOfQuantityUpdated , cartItems2.get(productCodeOfQuantityUpdated).QuantityAvailable1\_\_c);

}

for(Product2 singleProduct : ProductList2.values()){

if(singleProduct.ProductCode == productCodeOfQuantityUpdated){

if(mapOfCodeByQuantity.get(productCodeOfQuantityUpdated) < 0){

checkoutButtonFlag = true;

ApexPages.addmessage(new ApexPages.message(ApexPages.severity.WARNING,'Quantity Should not be negative'));

}

else if(mapOfCodeByQuantity.get(productCodeOfQuantityUpdated) == 0){

checkoutButtonFlag = true;

ApexPages.addmessage(new ApexPages.message(ApexPages.severity.WARNING,'Quantity Should not be Zero'));

}

else if(quantityAvailableMap.get(productCodeOfQuantityUpdated) - mapOfCodeByQuantity.get(productCodeOfQuantityUpdated) >= 0){

checkoutButtonFlag = false;

singleProduct.Quantity\_\_c = quantityAvailableMap.get(productCodeOfQuantityUpdated) - mapOfCodeByQuantity.get(productCodeOfQuantityUpdated);

}

else{

Decimal temp = quantityAvailableMap.get(productCodeOfQuantityUpdated);

checkoutButtonFlag = true;

ApexPages.addmessage(new ApexPages.message(ApexPages.severity.FATAL,'Only ' + temp + ' Products are left'));

}

}

}

return null;

}

//function to place order and perform query

public PageReference placeOrder(){

Map<String,Id> productCodeToIdMap = new Map<String,Id>();

for(Product2 currentProduct : productList2.values()){

productCodeToIdMap.put(currentProduct.ProductCode , currentProduct.Id);

}

PurchaseOrder\_\_c purchaseOrderRecord = new PurchaseOrder\_\_c(OrderPrice\_\_c = totalOrderPrice , OrderStatus\_\_c = 'Pending');

insert purchaseOrderRecord;

purchaseorderList.add(purchaseOrderRecord);

system.debug(purchaseOrderRecord);

for(PurchaseOrderLineItem\_\_c singleItem : cartItems2.values()){

singleItem.PurchaseOrderId\_\_c = purchaseOrderRecord.Id;

singleItem.ProductId\_\_c = productCodeToIdMap.get(singleItem.ProductCode\_\_c);

}

insert cartItems2.values();

update productList2.values();

cartFlag = false;

finalCartFlag = false;

flag = false;

productList2.clear();

productList.clear();

selectedProductList.clear();

duplicateProductMap.clear();

selectedIdMap.clear();

cartItems2.clear();

cartItems.clear();

return null;

}

//sort ProductTable

public PageReference sortProductTable(){

if(productList.size() > 0){

productList.sort();

}

return null;

}

//Pagination functions

Public void First1(){

hasNext1 = true;

productList.clear();

hasPrevious1 = false;

pageNumber1 =1;

for(integer i = 0 ; i < PageSize1 ; i++){

productList.add(duplicateProductMap.values()[i]);

}

}

Public void Previous1(){

hasNext1 = true;

productList.clear();

if(pageNumber1 != 1)

{

pageNumber1 = pageNumber1 - 1;

If(pageNumber1 == 1)

hasPrevious1 = false;

for(integer i = (pageNumber1 - 1) \* 10 ; i < (pageNumber1 \* 10) ; i++ )

{

productList.add(duplicateProductMap.values()[i]);

}

}

else{

hasPrevious1 = false;

for(integer i = (pageNumber1 - 1 ) \* 10 ; i < (pageNumber1 \* 10 ) ; i++ ){

productList.add(duplicateProductMap.values()[i]);

}

}

}

Public void Next1(){

pageNumber1 = pageNumber1 + 1;

integer pgno;

productList.clear();

if(math.mod(duplicateProductMap.size() , 10) == 0){

pgno = duplicateProductMap.size() / 10;

}

else

pgno = (duplicateProductMap.size() / 10) + 1;

if(pageNumber1 == pgno)

{

hasNext1 = false;

for(integer i = (pageNumber1 - 1) \* 10 ; i < duplicateProductMap.size() ; i++ ){

productList.add(duplicateProductMap.values()[i]);

}

}

else

for(integer i = (pageNumber1 - 1) \* 10 ; i < (pageNumber1 \* 10) ; i++ ){

productList.add(duplicateProductMap.values()[i]);

}

hasPrevious1 = true;

}

Public void Last1(){

hasPrevious1 = true;

productList.clear();

hasNext1 = false;

if( math.mod(duplicateProductMap.size(),10) == 0){

pageNumber1 = duplicateProductMap.size()/10;

}

else{

pageNumber1 = (duplicateProductMap.size() / 10) + 1;

}

for(integer i = (pageNumber1 - 1) \* 10 ; i < duplicateProductMap.size() ; i++){

if(i == duplicateProductMap.size()){

break;

}

productList.add(duplicateProductMap.values()[i]);

}

}

}