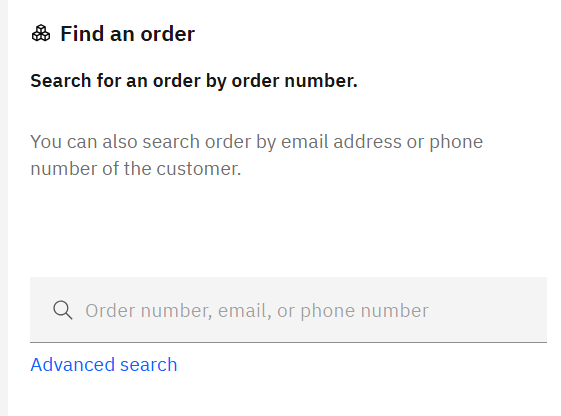
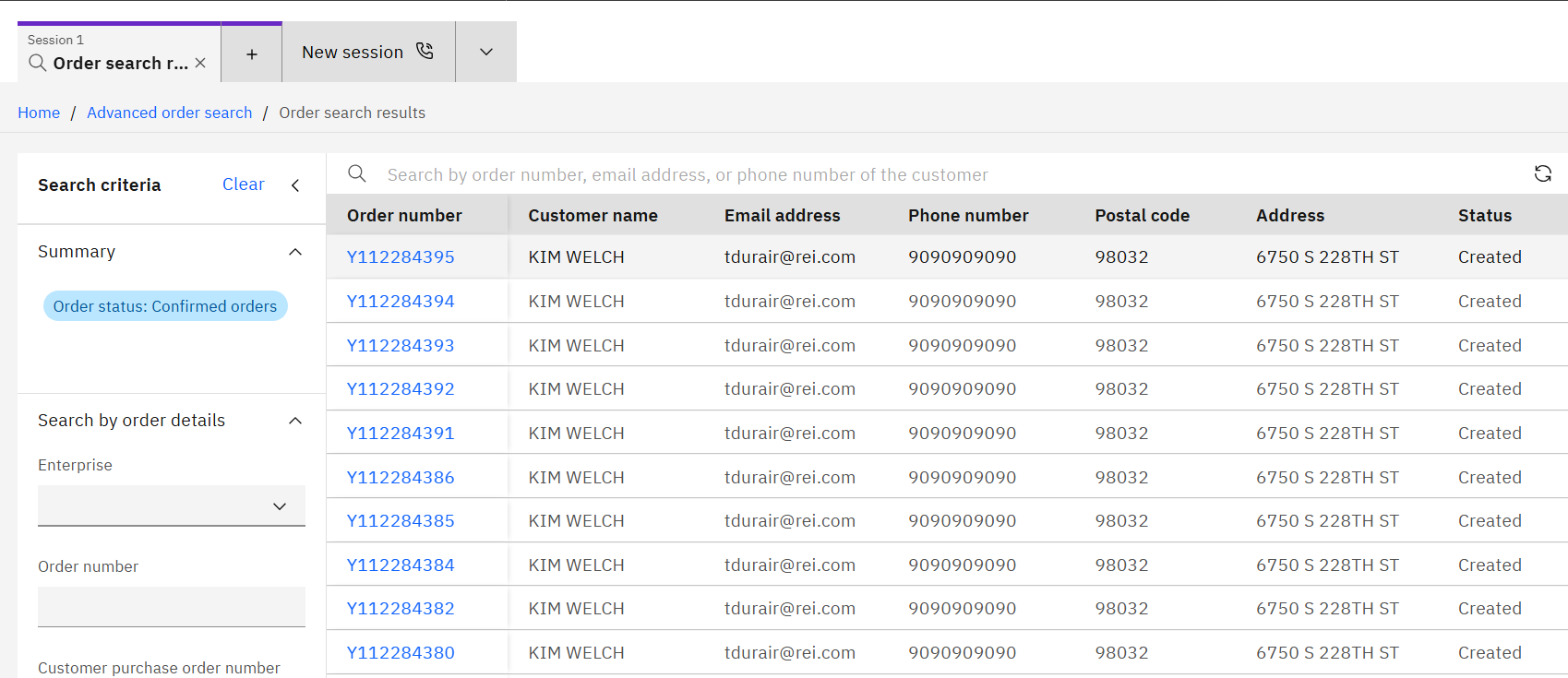
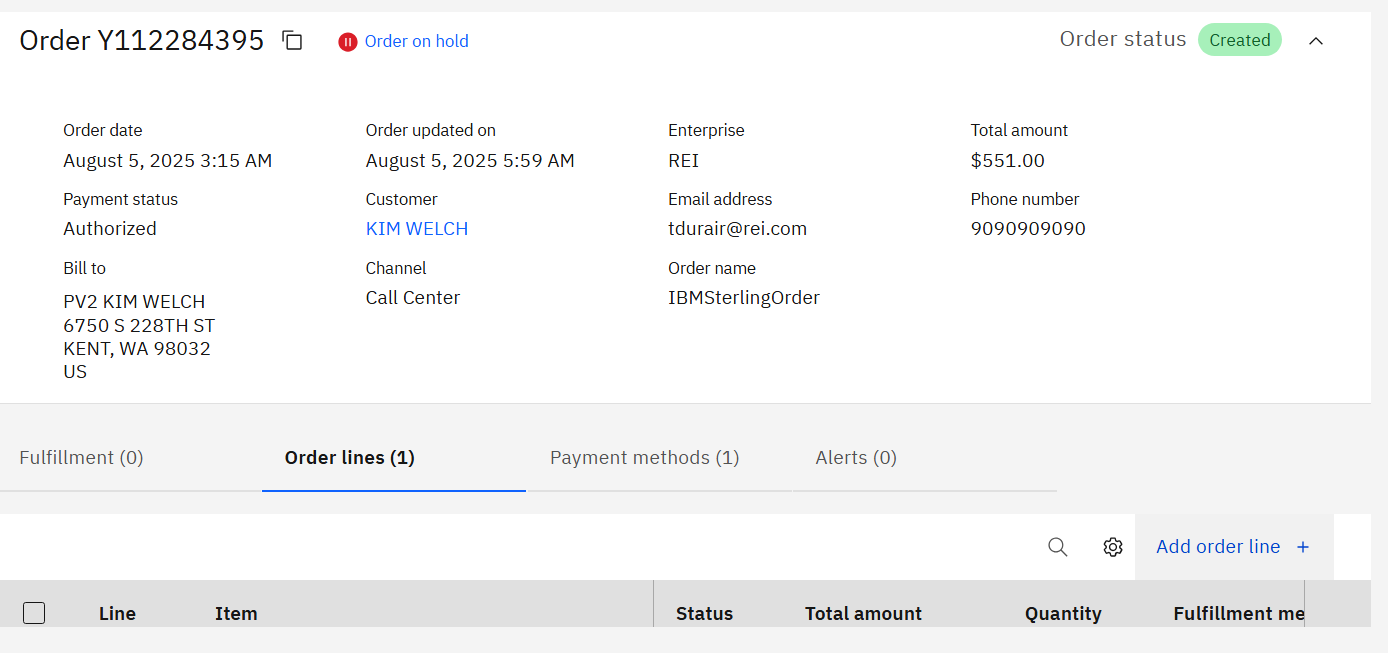
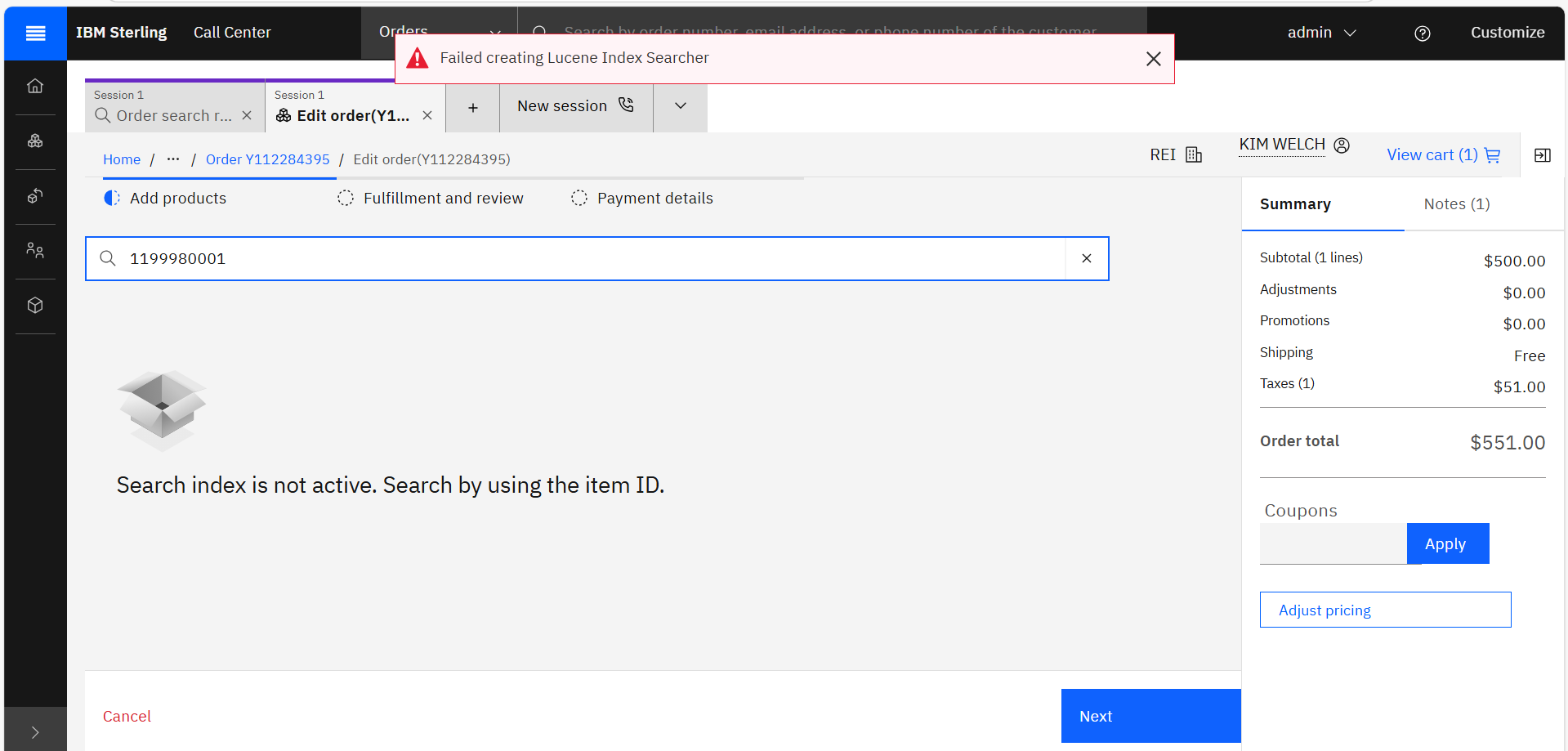
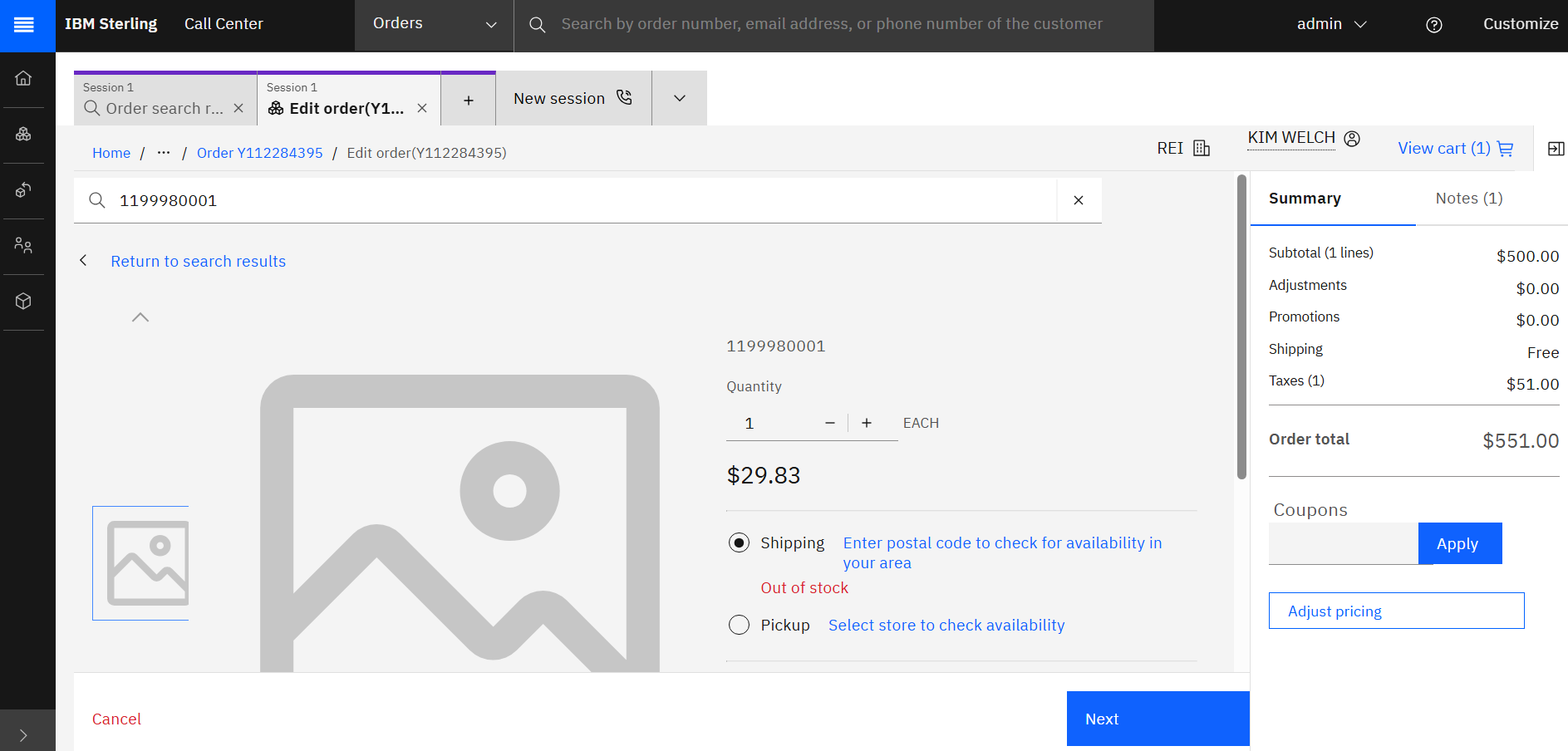
Icc - Customizing cc-components

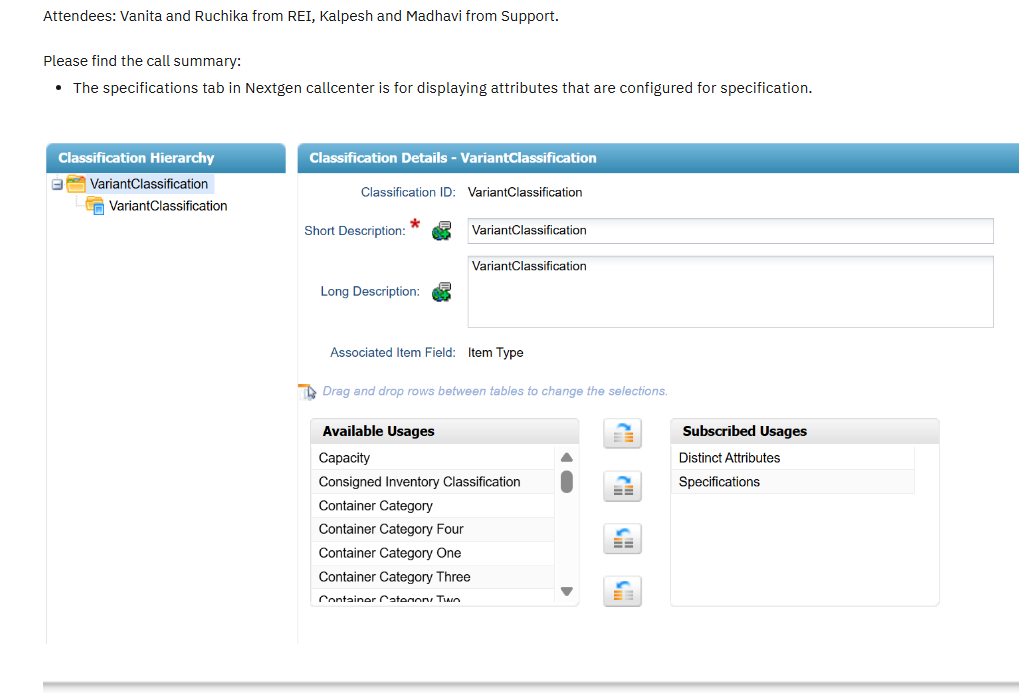
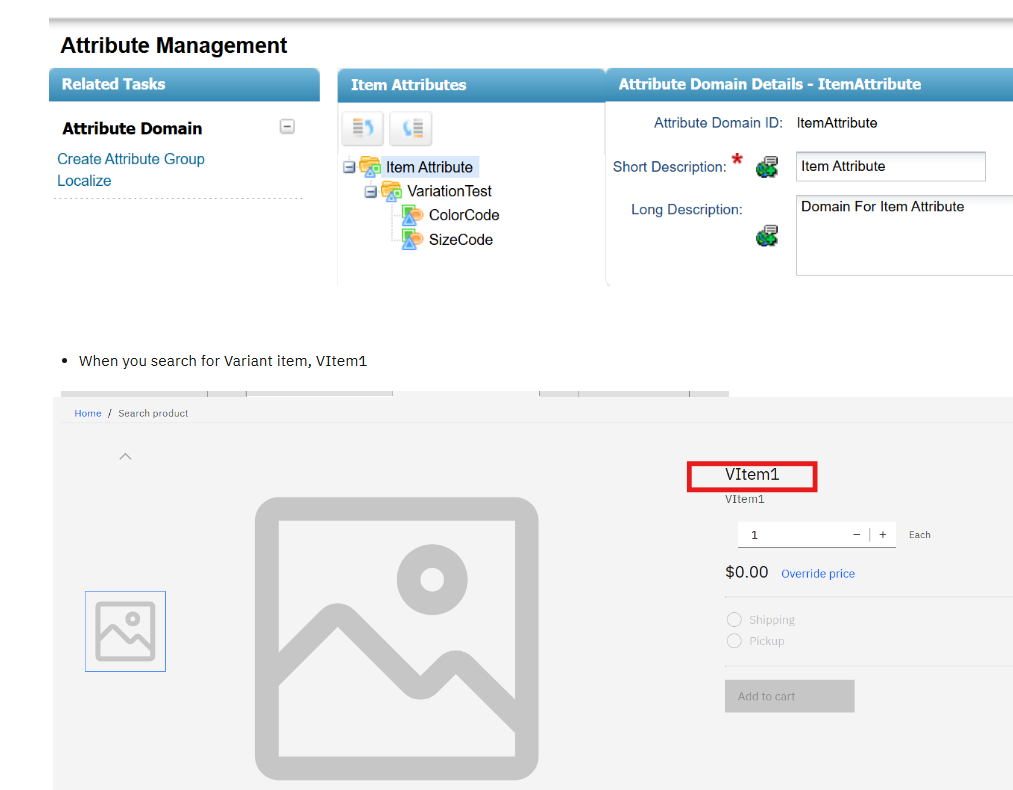
**Requirement – To Show Product variations for ModelItem = N (parent item sku) on UI in add product screen**

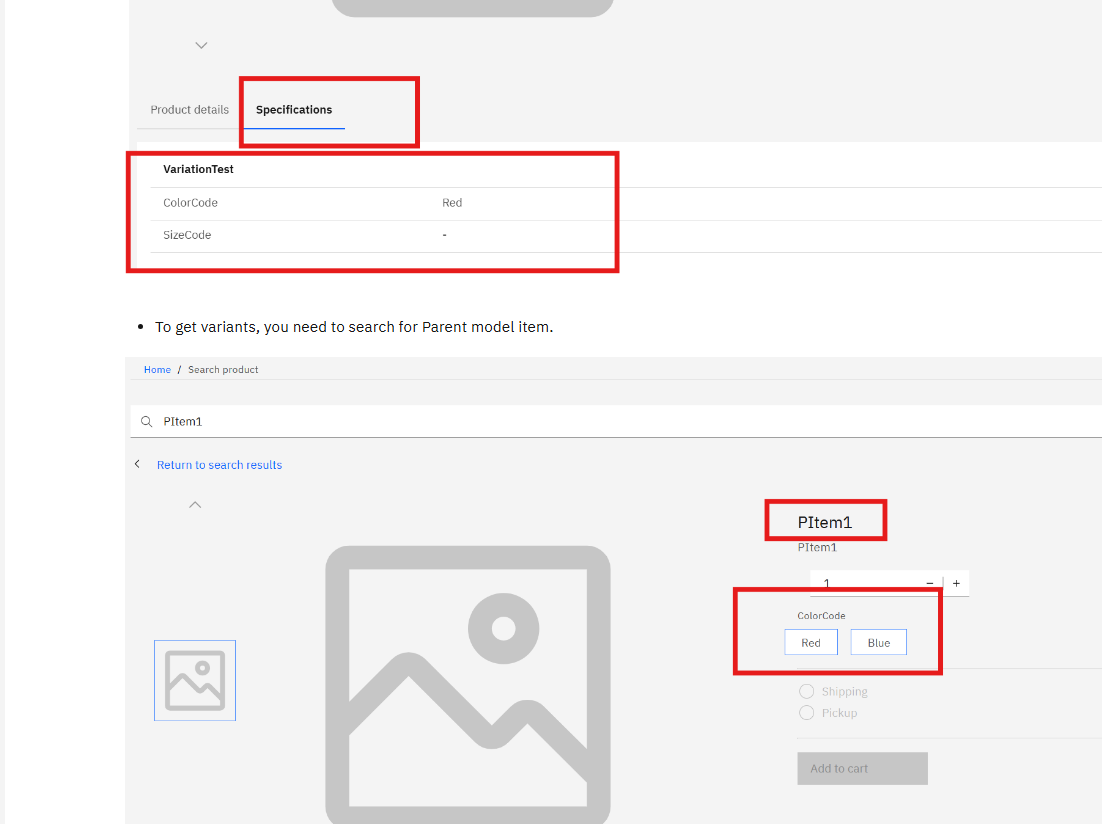
1. Search Order   
      
   Open created order

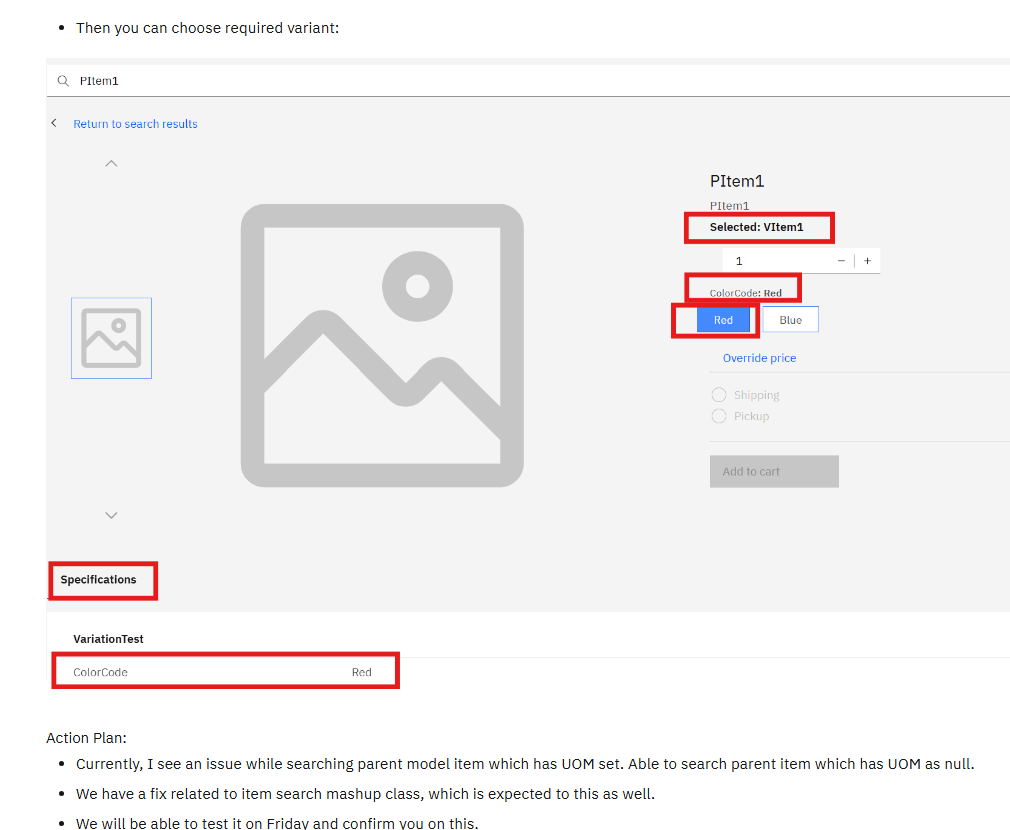
  
Add OrderLine  
  
Search for ItemId (Model Item = N) – catalog index is removed from the env

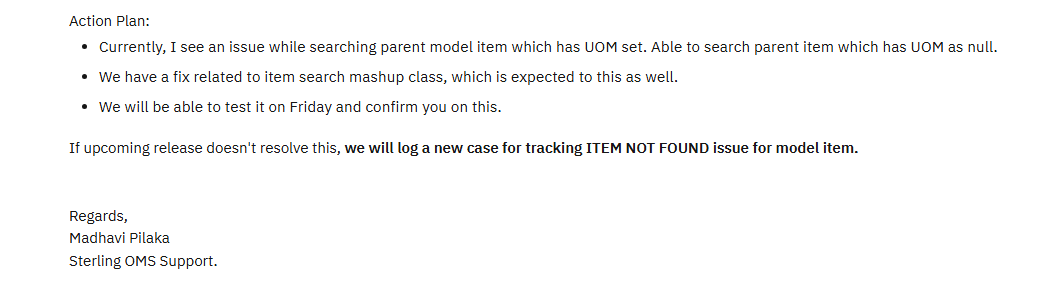
  
As we can see for Model Item = N product variations are not coming up 

In icc – IBM has not done changes to give product variations for ModelItem=N, Only for Model Item =Y, product variations come up . But for that also for item with UOM as null was giving product variations. This is a bug from IBM side, But here in this solution we will also resolve the bug from IBM until they resolve it in next fix pack.

IBM PMR Response -   
  






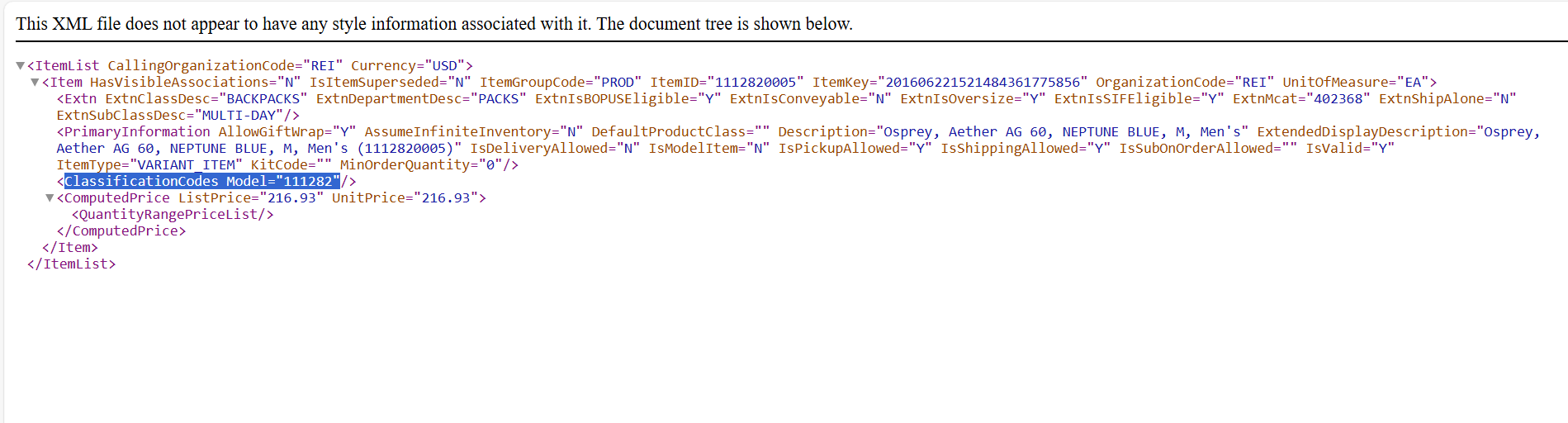


**Solution**

**How to get Model item ?**

Call getCompleteItemList Api with ItemId passed in input and use the below the template

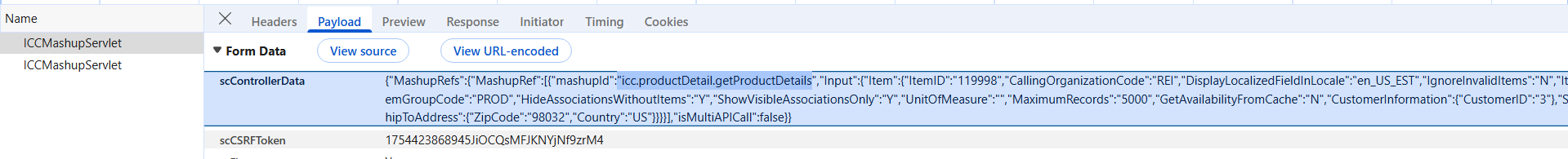
|  |
| --- |
| **Input-** <Item CallingOrganizationCode="REI"  DisplayLocalizedFieldInLocale="en\_US\_EST"  GetAvailabilityFromCache="N" HideAssociationsWithoutItems="Y"  IgnoreInvalidItems="N" IgnoreOrdering="Y" ItemGroupCode="PROD"  **ItemID="1112820005"** MaximumRecords="5000"  ShowVisibleAssociationsOnly="Y" UnitOfMeasure="">  <ShipToAddress Country="US" ZipCode="98032"/>  <CustomerInformation CustomerID="2"/>  </Item> |
| **Template –**  <ItemList CallingOrganizationCode="" Currency="">  <Item HasVisibleAssociations="" IsItemSuperseded=""  ItemGroupCode="" ItemID="" ItemKey="" OrganizationCode=""  UnitOfMeasure="" DisplayUnitOfMeasure="" ShowVariationPanel="">  <Extn/>  <PrimaryInformation AssumeInfiniteInventory=""  Description="" ExtendedDisplayDescription="" ImageID=""  ImageLabel="" ImageLocation="" ImageURL="" IsDeliveryAllowed=""  IsDeliveryAllowedOverridden="" IsModelItem="" IsPickupAllowed=""  IsPickupAllowedOverridden="" IsShippingAllowed="" AllowGiftWrap=""  IsShippingAllowedOverridden="" IsSubOnOrderAllowed="" IsValid=""  KitCode="" MinOrderQuantity="" DefaultProductClass="" ItemType=""/>  <ComputedPrice BundleTotal="" ListPrice=""  UnitPrice="">  <QuantityRangePriceList>  <QuantityRangePrice BreakQtyHigh=""  UnitPrice="" />  </QuantityRangePriceList>  </ComputedPrice>  <Availabilities>  <Availability DeliveryMethod="" ProductAvailDate="" Node="" Distance=""  HasAnyUnavailableQty="" ShipNode="" ShipNodeDescription="" EarliestShipDate=""/>  </Availabilities>  **<ClassificationCodes Model="" />**  </Item>  </ItemList> |
| Ouput- <ItemList CallingOrganizationCode="REI" Currency="USD">  <Item HasVisibleAssociations="N" IsItemSuperseded="N" ItemGroupCode="PROD" ItemID="**1112820005"** ItemKey="201606221521484361775856" OrganizationCode="REI" UnitOfMeasure="EA">  <Extn ExtnClassDesc="BACKPACKS" ExtnDepartmentDesc="PACKS" ExtnIsBOPUSEligible="Y" ExtnIsConveyable="N" ExtnIsOversize="Y" ExtnIsSIFEligible="Y" ExtnMcat="402368" ExtnShipAlone="N" ExtnSubClassDesc="MULTI-DAY"/>  <PrimaryInformation AllowGiftWrap="Y" AssumeInfiniteInventory="N" DefaultProductClass="" Description="Osprey, Aether AG 60, NEPTUNE BLUE, M, Men's" ExtendedDisplayDescription="Osprey, Aether AG 60, NEPTUNE BLUE, M, Men's (1112820005)" IsDeliveryAllowed="N" IsModelItem="N" IsPickupAllowed="Y" IsShippingAllowed="Y" IsSubOnOrderAllowed="" IsValid="Y" ItemType="VARIANT\_ITEM" KitCode="" MinOrderQuantity="0"/>  **<ClassificationCodes Model="111282"/>**  <ComputedPrice ListPrice="216.93" UnitPrice="216.93">  <QuantityRangePriceList/>  </ComputedPrice>  </Item>  </ItemList> |

  
From the above output we can see that in Classification Code Model Item (Parent item sku) came for ModelItem =N (Child item sku)  
So we will use this ModelItem value and pass it to the input of getCompleteItemList to get list of all the child items (variants) for the Model Item. And not the itemId

Below input and template to be used to get variants

|  |
| --- |
| **Input -**  <?xml version="1.0" encoding="UTF-8"?>  <Item CallingOrganizationCode="REI"  DisplayLocalizedFieldInLocale="en\_US\_EST"  GetAvailabilityFromCache="N" HideAssociationsWithoutItems="Y"  IgnoreInvalidItems="N" ItemGroupCode="PROD" ShowVisibleAssociationsOnly="Y">  <ItemAttributeGroupTypeList>  <ItemAttributeGroupType ItemAttributeGroupType="DISTINCT\_ATTRIBUTES"/>  </ItemAttributeGroupTypeList>  **<ClassificationCodes Model="119998"/>**  <PrimaryInformation ModelItemUnitOfMeasure="EA"/>  <ShipToAddress Country="US" ZipCode="98032"/>  <CustomerInformation CustomerID="2"/>  </Item> |
| **Template –**  <ItemList CallingOrganizationCode="" Currency="">  <Item HasVisibleAssociations="" ItemGroupCode="" ItemID=""  ItemKey="" OrganizationCode="" UnitOfMeasure="">  <AssetList TotalNumberOfRecords="">  <Asset ContentID="" ContentLocation=""  Description="" Label="" Type="" Value=""/>  </AssetList>  <AssociationTypeList>  <AssociationType CodeShortDescription=""  CodeValue="" Type=""/>  </AssociationTypeList>  <Availability CurrentAvailableQty=""  FutureAvailableDate="" FutureAvailableQuantity=""/>  <ClassificationCodes Model=""/>  <ComputedPrice BundleTotal="" ListPrice="" UnitPrice=""/>  <PrimaryInformation BundleFulfillmentMode=""  BundlePricingStrategy="" DefaultProductClass=""  Description="" ExtendedDescription=""  ExtendedDisplayDescription="" ImageID=""  ImageLabel="" ImageLocation="" IsModelItem=""  IsPickupAllowed="" IsShippingAllowed="" KitCode=""  ManufacturerItem="" ManufacturerName=""  MinOrderQuantity="" ShortDescription=""/>  </Item>  <AttributeList>  <Attribute AttributeID="" SequenceNo="" ShortDescription="">  <AssignedValueList>  <AssignedValue DisplayAttributeValue=""  DoubleValue="" ImageURL="" IntegerValue=""  SequenceNo="" ShortDescription="" Value=""/>  </AssignedValueList>  </Attribute>  </AttributeList>  <AllowedCombinationList>  <AllowedCombination ItemID="">  <ItemAttributeList>  <ItemAttribute DisplayAttributeValue="" ItemAttributeName=""/>  </ItemAttributeList>  </AllowedCombination>  </AllowedCombinationList>  </ItemList> |
| Output -    From the response you can see variants came for Model item |

We will update the mashup with the required template to achieve the above behaviour –

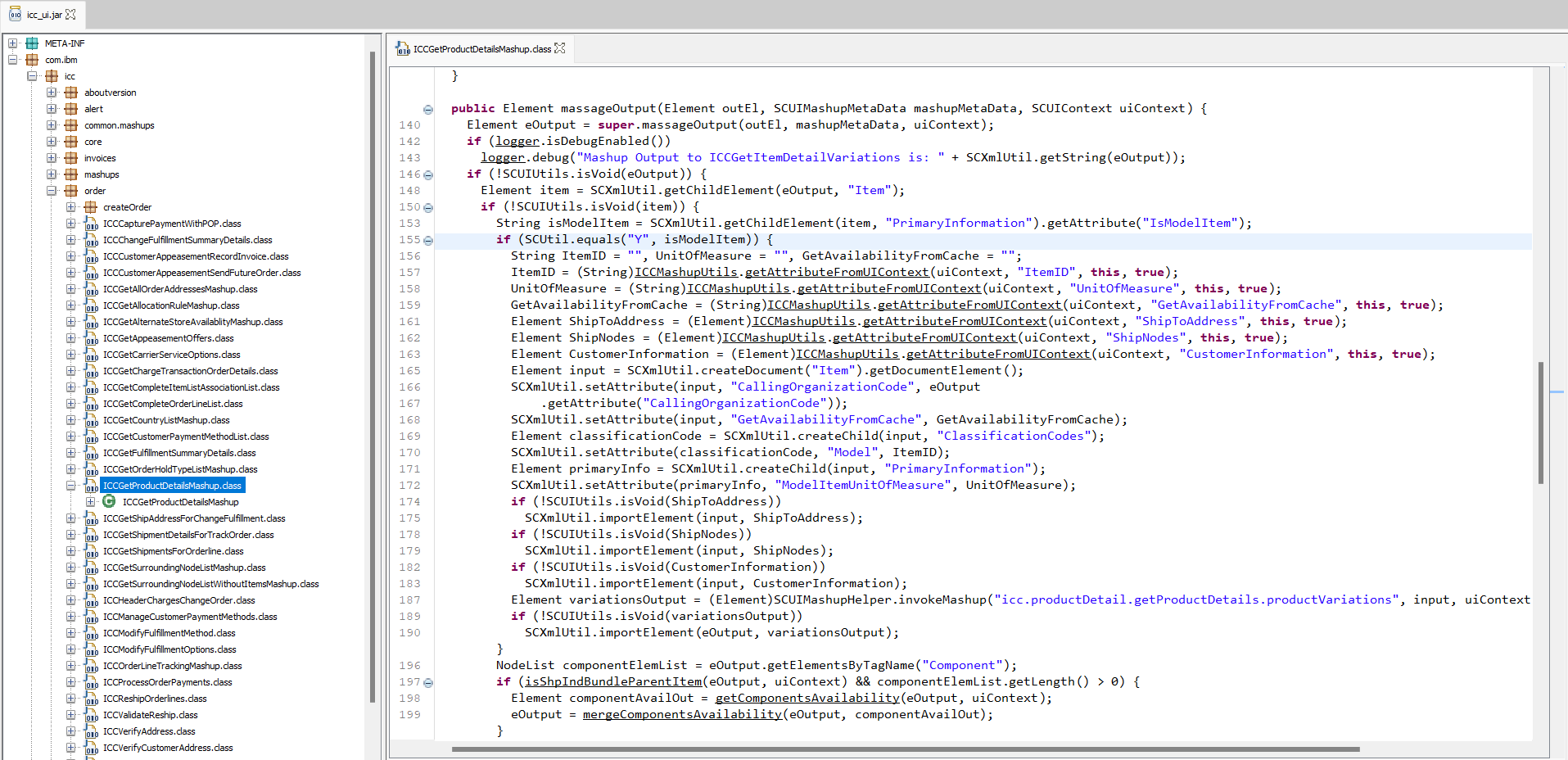


**Mashup to be Updated (Overridden) - icc.productDetail.getProductDetails - /mw/sterling/10/Foundation/extensions/icc/webpages/mashupxmls/order/product/ icc\_product\_mashups\_overridemashups.xml**Template is updated in below mashupand Class to be updated

|  |
| --- |
| <mashups>  <mashup description="Retrieves the details of product in the product details screen"  id="icc.productDetail.getProductDetails" mashuptype="XAPI" transactional="true" skipDataProvider="false">  <classInformation name="**com.ibm.custom.ExtnICCGetProductDetailsMashup**" />  <API Name="getCompleteItemList">  <Input>  <Item Currency="" DisplayLocalizedFieldInLocale="xml:CurrentUser:/User/@Localecode" GetAvailabilityFromCache="" IgnoreInvalidItems="N" ItemGroupCode="PROD" ProductClass=""  CallingOrganizationCode="" ItemID="" UnitOfMeasure="" MaximumRecords="5000" HideAssociationsWithoutItems="Y" ShowVisibleAssociationsOnly="Y">  <CustomerInformation CustomerContactID="" CustomerID="" />  <ShipNodes>  <ShipNode Node="" />  </ShipNodes>  <ShipToAddress Country="" City="" State="" ZipCode="" />  </Item>  </Input>  <Template>  <ItemList CallingOrganizationCode="" Currency="">  <Item HasVisibleAssociations="" IsItemSuperseded=""  ItemGroupCode="" ItemID="" ItemKey="" OrganizationCode=""  UnitOfMeasure="" DisplayUnitOfMeasure="" ShowVariationPanel="">  <Extn/>  <PrimaryInformation AssumeInfiniteInventory=""  Description="" ExtendedDisplayDescription="" ImageID=""  ImageLabel="" ImageLocation="" ImageURL="" IsDeliveryAllowed=""  IsDeliveryAllowedOverridden="" IsModelItem="" IsPickupAllowed=""  IsPickupAllowedOverridden="" IsShippingAllowed="" AllowGiftWrap=""  IsShippingAllowedOverridden="" IsSubOnOrderAllowed="" IsValid=""  KitCode="" MinOrderQuantity="" DefaultProductClass="" ItemType=""/>  <ComputedPrice BundleTotal="" ListPrice=""  UnitPrice="">  <QuantityRangePriceList>  <QuantityRangePrice BreakQtyHigh=""  UnitPrice="" />  </QuantityRangePriceList>  </ComputedPrice>  <Availabilities>  <Availability DeliveryMethod="" ProductAvailDate="" Node="" Distance=""  HasAnyUnavailableQty="" ShipNode="" ShipNodeDescription="" EarliestShipDate=""/>  </Availabilities>  <ClassificationCodes Model="" />  </Item>  </ItemList>  </Template>  </API>  <AdditionalDataProviders>  <AdditionalDataProvider class="com.ibm.icc.core.dataproviders.ICCUnitOfMeasureDisplayProvider">  <InterestedIn>  <Element path="/ItemList/Item" />  <Element path="/ItemList/Item/Components/Component" />  <Attribute path="Item" name="UnitOfMeasure" />  <Attribute path="Component" name="ComponentUnitOfMeasure" />  </InterestedIn>  <NewAttributes>  <Attribute name="DisplayUnitOfMeasure" />  </NewAttributes>  </AdditionalDataProvider>  </AdditionalDataProviders>  <AlternateResourceIds>  <AlternateResourceId altResourceId="ICC000060" />  <AlternateResourceId altResourceId="ICC000001" />  </AlternateResourceIds>  </mashup>  </mashups> |

**To update mashup in the env - do icc build and deployment in was – run the command in foundation/bin-**./buildear.sh -Dearfile=smcfs.ear -Dwarfiles=smcfs,sbc,sma,wscdev,icc -Dappserver=websphere -Dnowebservice=false -Ddevmode=true -Duidevmode=true -Dnodocear=true create-ear

**Why class is updated ?**

OOB code is doing call for product variants only for ModelItem=Y, so to make call happen for ModelItem=N we will modify the condition and also for ModelItem=Y product variations coming only for UOM = NULL (IBM ISSUE) that we will resolve there  


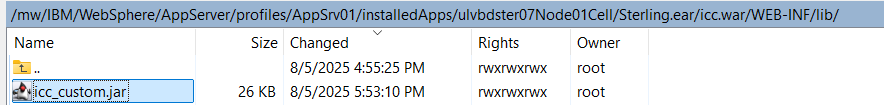
In /mw/IBM/WebSphere/AppServer/profiles/AppSrv01/installedApps/ulvbdster07Node01Cell/Sterling.ear/icc.war/WEB-INF/lib – icc\_custom.jar would be updated, we will take code from icc.jar and update it to icc\_custom.jar

Code changes added to ExtnICCGetProductDetailsMashup

|  |
| --- |
| package com.ibm.custom;  import com.ibm.icc.common.mashups.ICCBaseMashup;  import com.ibm.icc.mashups.utils.ICCMashupUtils;  import com.sterlingcommerce.baseutil.SCUtil;  import com.sterlingcommerce.baseutil.SCXmlUtil;  import com.sterlingcommerce.ui.web.framework.context.SCUIContext;  import com.sterlingcommerce.ui.web.framework.helpers.SCUIMashupHelper;  import com.sterlingcommerce.ui.web.framework.mashup.SCUIMashupMetaData;  import com.sterlingcommerce.ui.web.framework.utils.SCUIUtils;  import com.yantra.yfc.dom.YFCDocument;  import com.yantra.yfc.dom.YFCElement;  import com.yantra.yfc.dom.YFCNodeList;  import com.yantra.yfc.log.YFCLogCategory;  import java.util.ArrayList;  import java.util.HashMap;  import java.util.List;  import java.util.Map;  import org.w3c.dom.Element;  import org.w3c.dom.NodeList;  public class ExtnICCGetProductDetailsMashup extends ICCBaseMashup {  private static YFCLogCategory logger = YFCLogCategory.instance(ExtnICCGetProductDetailsMashup.class.getName());    public Element massageInput(Element inputEl, SCUIMashupMetaData mashupMetaData, SCUIContext uiContext) {  Element eInput = super.massageInput(inputEl, mashupMetaData, uiContext);  if (SCUIUtils.isVoid(eInput.getAttribute("GetAvailabilityFromCache"))) {  Element eRuleOutput = ICCMashupUtils.getRuleValue(uiContext, "YCD\_USE\_CACHE\_INVENTORY\_CHECK", inputEl.getAttribute("CallingOrganizationCode"), null);  if (!SCUtil.isVoid(eRuleOutput))  eInput.setAttribute("GetAvailabilityFromCache", eRuleOutput.getAttribute("RuleSetValue"));  }  ICCMashupUtils.addAttributeToUIContext(uiContext, "ItemID", this, eInput.getAttribute("ItemID"));  ICCMashupUtils.addAttributeToUIContext(uiContext, "UnitOfMeasure", this, eInput.getAttribute("UnitOfMeasure"));  ICCMashupUtils.addAttributeToUIContext(uiContext, "GetAvailabilityFromCache", this, eInput.getAttribute("GetAvailabilityFromCache"));  Element shipToAddress = SCXmlUtil.getChildElement(inputEl, "ShipToAddress");  Element shipNodes = SCXmlUtil.getChildElement(inputEl, "ShipNodes");  if (!SCUIUtils.isVoid(shipNodes))  ICCMashupUtils.addAttributeToUIContext(uiContext, "ShipNodes", this, shipNodes);  Element customerInfo = SCXmlUtil.getChildElement(inputEl, "CustomerInformation");  if (!SCUIUtils.isVoid(customerInfo))  ICCMashupUtils.addAttributeToUIContext(uiContext, "CustomerInformation", this, customerInfo);  if (!SCUIUtils.isVoid(shipToAddress)) {  String zipCode = SCXmlUtil.getAttribute(shipToAddress, "ZipCode");  if (!SCUtil.isVoid(zipCode)) {  String country = SCXmlUtil.getAttribute(shipToAddress, "Country");  if (!SCUtil.isVoid(country)) {  String fetchedCity = "";  String fetchedState = "";  Element getCityStateForZipInput = SCXmlUtil.createDocument("Region").getDocumentElement();  getCityStateForZipInput.setAttribute("DisplayLocalizedFieldInLocale", inputEl  .getAttribute("DisplayLocalizedFieldInLocale"));  getCityStateForZipInput.setAttribute("RegionSchemaKey", "ALL\_US");  getCityStateForZipInput.setAttribute("MaximumRecords", "");  Element personInfoEl = SCXmlUtil.createChild(getCityStateForZipInput, "PersonInfo");  personInfoEl.setAttribute("Country", country);  personInfoEl.setAttribute("ZipCode", zipCode);  if (logger.isDebugEnabled())  logger.debug("getCityStateForZip Mashup Input here: " +  SCXmlUtil.getString(getCityStateForZipInput));  Element getCityStateForZipMashupOutputElem = (Element)SCUIMashupHelper.invokeMashup("icc.common-components.getCityStateForZip", getCityStateForZipInput, uiContext);  if (logger.isDebugEnabled())  logger.debug("getCityStateForZip Mashup Output here: " +  SCXmlUtil.getString(getCityStateForZipMashupOutputElem));  if (!SCUIUtils.isVoid(getCityStateForZipMashupOutputElem)) {  Element regionShemaEl = SCXmlUtil.getChildElement(getCityStateForZipMashupOutputElem, "RegionSchema");  if (!SCUIUtils.isVoid(regionShemaEl)) {  ArrayList<Element> regionList = SCXmlUtil.getChildren(regionShemaEl, "Region");  if (!SCUtil.isVoid(regionList)) {  if (logger.isDebugEnabled())  logger.debug(regionList + "regionList output here");  for (int i = 0; i < regionList.size(); i++) {  Element currentRegion = regionList.get(i);  if (SCXmlUtil.getAttribute(currentRegion, "RegionLevelName")  .equals("City")) {  fetchedCity = SCXmlUtil.getAttribute(currentRegion, "RegionName");  shipToAddress.setAttribute("City", fetchedCity);  }  if (SCXmlUtil.getAttribute(currentRegion, "RegionLevelName")  .equals("State")) {  fetchedState = SCXmlUtil.getAttribute(currentRegion, "RegionName");  shipToAddress.setAttribute("State", fetchedState);  }  }  }  }  }  }  }  ICCMashupUtils.addAttributeToUIContext(uiContext, "ShipToAddress", this, shipToAddress);  }  if (logger.isDebugEnabled())  logger.debug("Mashup Input : " + SCXmlUtil.getString(eInput));  return eInput;  }    public Element massageOutput(Element outEl, SCUIMashupMetaData mashupMetaData, SCUIContext uiContext) {  Element eOutput = super.massageOutput(outEl, mashupMetaData, uiContext);  if (logger.isDebugEnabled())  logger.debug("Mashup Output to ICCGetItemDetailVariations is: " + SCXmlUtil.getString(eOutput));  if (!SCUIUtils.isVoid(eOutput)) {  Element item = SCXmlUtil.getChildElement(eOutput, "Item");  **//COAFEE-510 Getting UOM**  **String strUOM = item.getAttribute("UnitOfMeasure");**  if (!SCUIUtils.isVoid(item)) {  String isModelItem = SCXmlUtil.getChildElement(item, "PrimaryInformation").getAttribute("IsModelItem");  **//COAFEE-510 - Changes for Product Variations - || isModelItem != null**  if (SCUtil.equals("Y", isModelItem) **|| isModelItem != null** ) {  String ItemID = "", UnitOfMeasure = "", GetAvailabilityFromCache = "";  ItemID = (String)ICCMashupUtils.getAttributeFromUIContext(uiContext, "ItemID", this, true);  UnitOfMeasure = (String)ICCMashupUtils.getAttributeFromUIContext(uiContext, "UnitOfMeasure", this, true);  GetAvailabilityFromCache = (String)ICCMashupUtils.getAttributeFromUIContext(uiContext, "GetAvailabilityFromCache", this, true);  Element ShipToAddress = (Element)ICCMashupUtils.getAttributeFromUIContext(uiContext, "ShipToAddress", this, true);  Element ShipNodes = (Element)ICCMashupUtils.getAttributeFromUIContext(uiContext, "ShipNodes", this, true);  Element CustomerInformation = (Element)ICCMashupUtils.getAttributeFromUIContext(uiContext, "CustomerInformation", this, true);  Element input = SCXmlUtil.createDocument("Item").getDocumentElement();  SCXmlUtil.setAttribute(input, "CallingOrganizationCode", eOutput  .getAttribute("CallingOrganizationCode"));  SCXmlUtil.setAttribute(input, "GetAvailabilityFromCache", GetAvailabilityFromCache);  Element classificationCode = SCXmlUtil.createChild(input, "ClassificationCodes");  **// COAFEE-510 --START -- Product Variation**  **if (SCXmlUtil.getChildElement(item, "ClassificationCodes") != null) {**  **Element eleOutClassificationCode = SCXmlUtil.getChildElement(item, "ClassificationCodes");**  **if (eleOutClassificationCode.hasAttribute("Model")**  **&& eleOutClassificationCode.getAttribute("Model") != null) {**  **String strModelFromOutput = SCXmlUtil.getChildElement(item, "ClassificationCodes")**  **.getAttribute("Model");**  **SCXmlUtil.setAttribute(classificationCode, "Model", strModelFromOutput);**  **} else {**  **SCXmlUtil.setAttribute(classificationCode, "Model", ItemID);**  **}**  **} else {**  **SCXmlUtil.setAttribute(classificationCode, "Model", ItemID);**  **}**  **// COAFEE-510 -- ENDS - Product Variations**  // SCXmlUtil.setAttribute(classificationCode, "Model", ItemID);  Element primaryInfo = SCXmlUtil.createChild(input, "PrimaryInformation");  **//COAFEE-510 - if UOM is empty set it--start**  **if (UnitOfMeasure == null || UnitOfMeasure.isEmpty()) {**  **UnitOfMeasure = strUOM;**  **}**  **//ends**  SCXmlUtil.setAttribute(primaryInfo, "ModelItemUnitOfMeasure", UnitOfMeasure);  if (!SCUIUtils.isVoid(ShipToAddress))  SCXmlUtil.importElement(input, ShipToAddress);  if (!SCUIUtils.isVoid(ShipNodes))  SCXmlUtil.importElement(input, ShipNodes);  if (!SCUIUtils.isVoid(CustomerInformation))  SCXmlUtil.importElement(input, CustomerInformation);  Element variationsOutput = (Element)SCUIMashupHelper.invokeMashup("icc.productDetail.getProductDetails.productVariations", input, uiContext);  if (!SCUIUtils.isVoid(variationsOutput))  SCXmlUtil.importElement(eOutput, variationsOutput);  }  NodeList componentElemList = eOutput.getElementsByTagName("Component");  if (isShpIndBundleParentItem(eOutput, uiContext) && componentElemList.getLength() > 0) {  Element componentAvailOut = getComponentsAvailability(eOutput, uiContext);  eOutput = mergeComponentsAvailability(eOutput, componentAvailOut);  }  if (logger.isDebugEnabled())  logger.debug("output out of ICCGetItemDetailVariations is:" + SCXmlUtil.getString(eOutput));  }  }  return eOutput;  }    private Element mergeComponentsAvailability(Element outEl, Element componentAvailOut) {  Map<String, Element> tempAvailabilityElemMap = new HashMap<>();  NodeList componentItemElemList = componentAvailOut.getElementsByTagName("Item");  for (int i = 0; i < componentItemElemList.getLength(); i++) {  Element componentItem = (Element)componentItemElemList.item(i);  Element compAvailabilityElem = SCXmlUtil.getChildElement(componentItem, "Availability");  String itemID = componentItem.getAttribute("ItemID");  tempAvailabilityElemMap.put(itemID, compAvailabilityElem);  }  NodeList componentItemList = outEl.getElementsByTagName("Component");  for (int j = 0; j < componentItemList.getLength(); j++) {  Element componentItem = (Element)componentItemList.item(j);  String compItemID = componentItem.getAttribute("ComponentItemID");  Element availabilityEleFromMap = tempAvailabilityElemMap.get(compItemID);  SCXmlUtil.importElement(componentItem, availabilityEleFromMap);  }  return outEl;  }    private Element getComponentsAvailability(Element outEl, SCUIContext uiContext) {  List<String> compItemIDList = getComponentItemIDs(outEl, uiContext);  Element newInputElem = SCXmlUtil.createDocument("Item").getDocumentElement();  SCXmlUtil.setAttribute(newInputElem, "CallingOrganizationCode", outEl.getAttribute("CallingOrganizationCode"));  SCXmlUtil.setAttribute(newInputElem, "UnitOfMeasure", outEl.getAttribute("UnitOfMeasure"));  String GetAvailabilityFromCache = (String)ICCMashupUtils.getAttributeFromUIContext(uiContext, "GetAvailabilityFromCache", this, true);  SCXmlUtil.setAttribute(newInputElem, "GetAvailabilityFromCache", GetAvailabilityFromCache);  Element ShipToAddress = (Element)ICCMashupUtils.getAttributeFromUIContext(uiContext, "ShipToAddress", this, true);  Element ShipNodes = (Element)ICCMashupUtils.getAttributeFromUIContext(uiContext, "ShipNodes", this, true);  Element CustomerInformation = (Element)ICCMashupUtils.getAttributeFromUIContext(uiContext, "CustomerInformation", this, true);  if (!SCUIUtils.isVoid(ShipToAddress))  SCXmlUtil.importElement(newInputElem, ShipToAddress);  if (!SCUIUtils.isVoid(ShipNodes))  SCXmlUtil.importElement(newInputElem, ShipNodes);  if (!SCUIUtils.isVoid(CustomerInformation))  SCXmlUtil.importElement(newInputElem, CustomerInformation);  Element complexQryElem = SCXmlUtil.createChild(newInputElem, "ComplexQuery");  Element complexQryOrElem = SCXmlUtil.createChild(complexQryElem, "Or");  for (int i = 0; i < compItemIDList.size(); i++) {  String compItemID = compItemIDList.get(i);  Element complexQryExpElem = SCXmlUtil.createChild(complexQryOrElem, "Exp");  complexQryExpElem.setAttribute("Name", "ItemID");  complexQryExpElem.setAttribute("Value", compItemID);  }  Element compAvailOutput = (Element)SCUIMashupHelper.invokeMashup("icc.productDetail.getProductDetails.productAvailablity", newInputElem, uiContext);  return compAvailOutput;  }    private List<String> getComponentItemIDs(Element outEl, SCUIContext uiContext) {  List<String> compItemIDList = new ArrayList<>();  YFCElement outTempElem = YFCDocument.getDocumentFor(outEl.getOwnerDocument()).getDocumentElement();  YFCNodeList<YFCElement> componentElemList = outTempElem.getElementsByTagName("Component");  for (int i = 0; i < componentElemList.getLength(); i++) {  String compItemID = ((YFCElement)componentElemList.item(i)).getAttribute("ComponentItemID");  compItemIDList.add(compItemID);  }  return compItemIDList;  }    private boolean isShpIndBundleParentItem(Element outEl, SCUIContext uiContext) {  Element itemElem = SCXmlUtil.getChildElement(outEl, "Item");  if (!SCUIUtils.isVoid(itemElem)) {  Element primaryInfoElem = SCXmlUtil.getChildElement(itemElem, "PrimaryInformation");  if (!SCUIUtils.isVoid(primaryInfoElem)) {  String kitCode = primaryInfoElem.getAttribute("KitCode");  String fulfillmentMode = primaryInfoElem.getAttribute("BundleFulfillmentMode");  if ("BUNDLE".equals(kitCode) && "00".equalsIgnoreCase(fulfillmentMode))  return true;  }  }  return false;  }  } |



Update the code in icc\_custom.jar

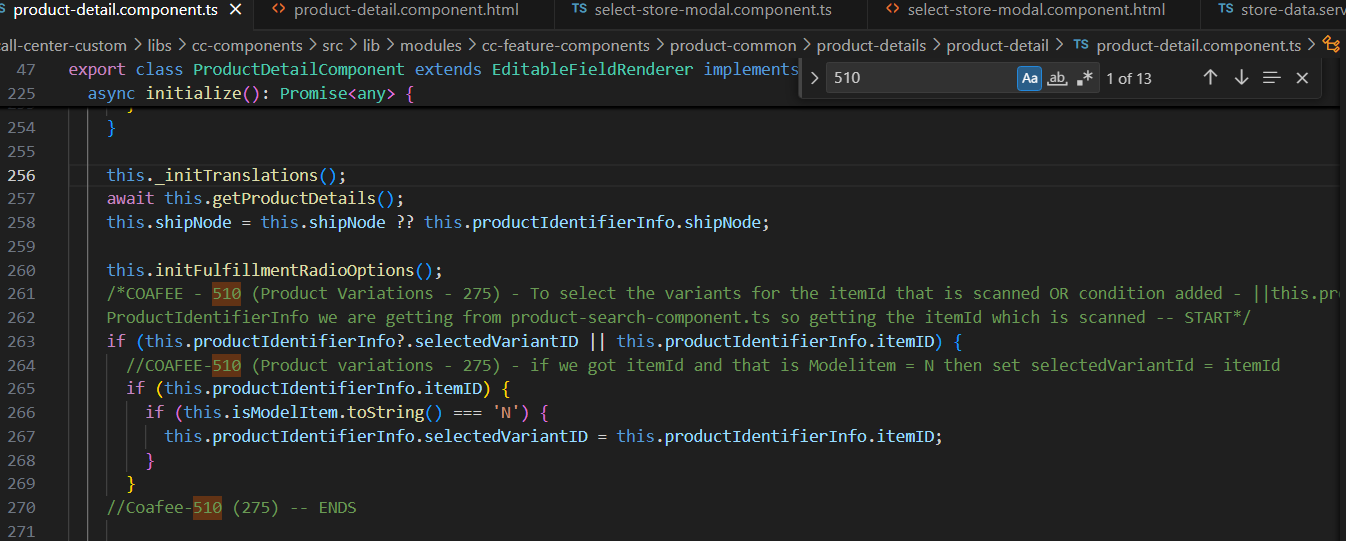


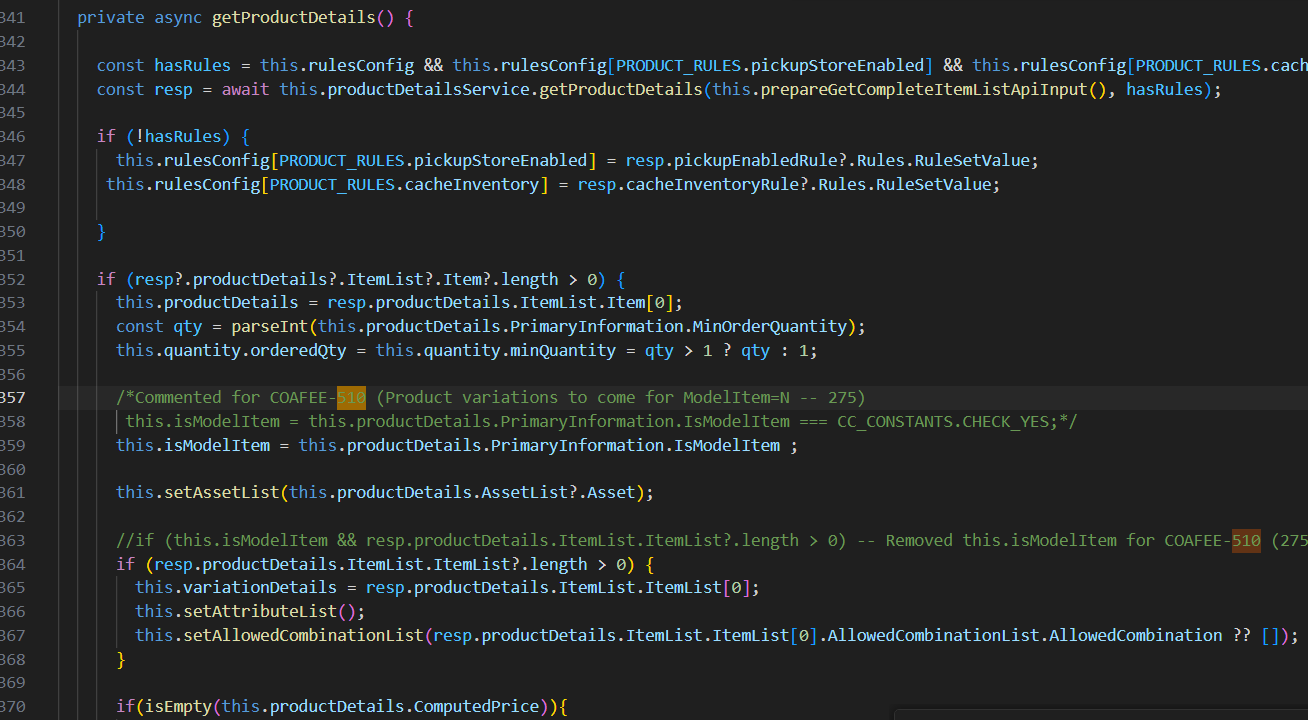
**Now we will check the UI code-**

In product-detail.component.ts -   
call-center-custom\libs\cc-components\src\lib\modules\cc-feature-components\product-common\product-details\product-detail\product-detail.component.ts , below are the code changes

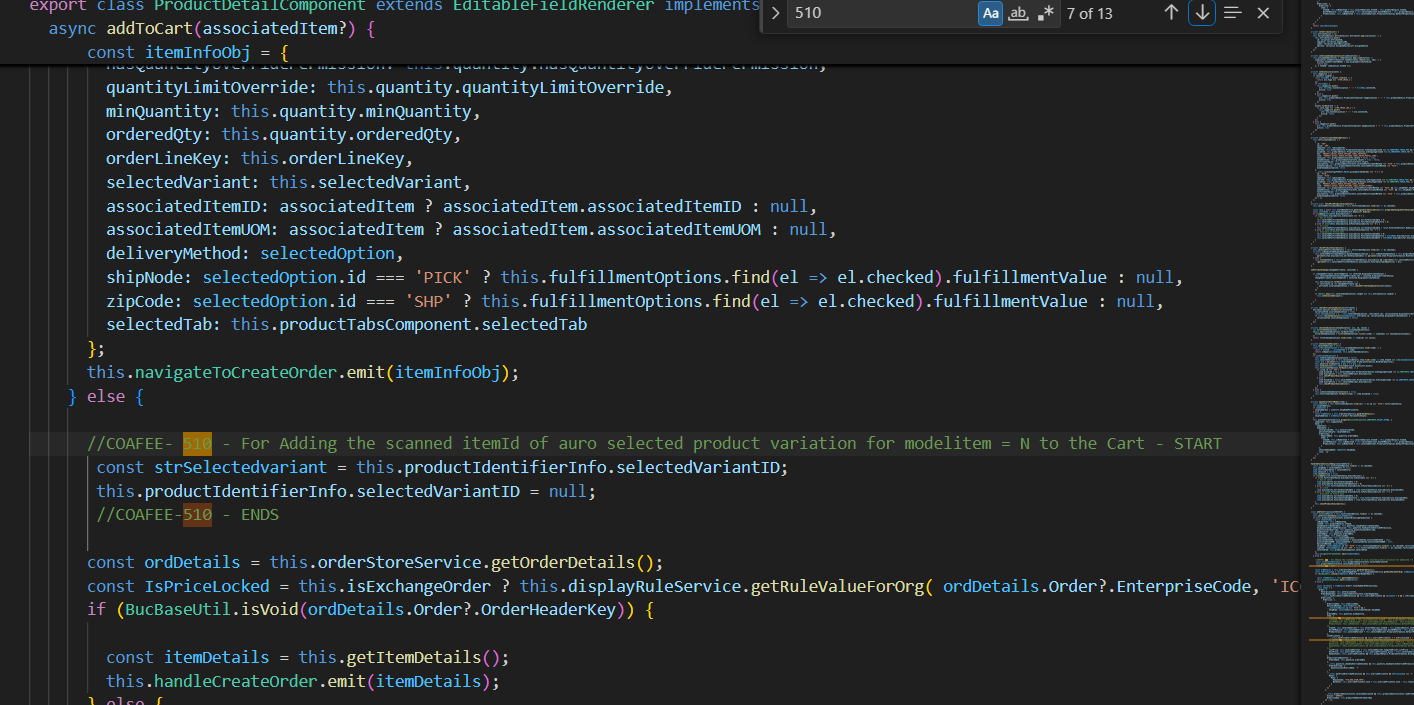
|  |
| --- |
| /\*   \* IBM Confidential   \* OCO Source Materials   \* 5737-D18, 5725-D10   \*   \* (C) Copyright International Business Machines Corp. 2019, 2025   \*   \* The source code for this program is not published or otherwise divested   \* of its trade secrets, irrespective of what has been deposited with the   \* U.S. Copyright Office.   \*/  import {    Component, ElementRef, EventEmitter,    Input, OnChanges, OnDestroy, OnInit,    Output, SimpleChanges, TemplateRef,    ViewChild, Injector    } from '@angular/core';  import { ActionProcessorService, Constants, DisplayRulesHelperService, localeBuc2Angular, TabMessageService } from '@buc/common-components';  import { BucCommonCurrencyFormatPipe } from '@buc/common-components';  import { BucNotificationModel } from '@buc/common-components';  import { BucNotificationService } from '@buc/common-components';  import { CCNotificationService } from '@buc/common-components';  import { TranslateService } from '@ngx-translate/core';  import { ProductDetailsService } from '../data-service/product-details.service';  import { isEmpty, isEqual } from 'lodash';  import { AddToCartErrorCodes, CC\_CONSTANTS, ExtensionConstants, PRODUCT\_RULES } from '../../../common/constants';  import { ActionParams } from '../../../common/types';  import { Subscription, Observable, of } from 'rxjs';  import { ActivatedRoute } from '@angular/router';  import { BucBaseUtil, BucSvcAngularStaticAppInfoFacadeUtil, getPostMessageDomain } from '@buc/svc-angular';  import { ProductTabsComponent } from '../product-tabs/product-tabs.component';  import { EditableFieldRenderer } from '@buc/common-components';  import { getArray } from '@buc/common-components';  import { StoreDataService } from '../../../store-search-result/data-service/store-data.service';  import { OrderStoreService } from '../../../common/services/order-store.service';  import { AddToOrderService } from '../../../add-to-order/add-to-order.service';  import { AddToOrderModalComponent } from '../../../add-to-order/add-to-order.component';  @Component({    selector: 'buc-product-detail',    templateUrl: './product-detail.component.html',    styleUrls: ['./product-detail.component.scss']  })  export class ProductDetailComponent extends EditableFieldRenderer implements OnInit, OnChanges, OnDestroy {    @Input() productIdentifierInfo = {      itemID: '',      unitOfMeasure: '',      defaultProductClass: '',      selectedFulfillmentMethod: 'SHP',      shipTo: '',      country: '',      countries: [],      shipNode: {} as any,      productQtyInCart: '',      orderLineKey: '',      selectedTab: '',      orderHeaderKey: '',      enterpriseCode: '',      isParentItemSearchResult: false,      productBrowsingStandalone: false,      readFromSession: false,      selectedVariantID: '',      isModelItem: false,      selectedVariant: '',      associatedItemID: '',      associatedItemUOM: '',      quantity: 0,      quantityLimitOverride: false,      hasQuantityOverridePermission: false,      showQtyOverrideCheckbox: false,      parentItemID: ''    }    @Input() currency = '';    @Input() rulesConfig = {      [PRODUCT\_RULES.cacheInventory]: '',      [PRODUCT\_RULES.pickupStoreEnabled]: ''    };    @Input() isReturnOrder = false;    @Input() isExchangeOrder = false;    @Input() customerInformation = {      customerId: '',      customerContactId: ''    };    @Input() isParentItemSearchResult = false;    @Input() handleAddToCartExternally = false;    // @Input() showGoToOrder = false;    @Output() returnToItemSearchResult: EventEmitter<any> = new EventEmitter();    @Output() navigateToCreateOrder: EventEmitter<any> = new EventEmitter();    @Output() setReadFromSessionFalse: EventEmitter<any> = new EventEmitter();    @Output() updateOrderSummary: EventEmitter<any> = new EventEmitter();    @Output() updateFulfillmentData: EventEmitter<any> = new EventEmitter();    @Output() handleCreateOrder: EventEmitter<any> = new EventEmitter();    FIELD\_ITEM\_ID = 'itemID';    FIELD\_DESC = 'itemDesc';    isScreenInitialized = false;    componentId = 'ProductDetailComponent'    fulfillmentOptions: any = [];    productDetails: any;    variationDetails: any;    attributeList = [];    allowedCombinations = [];    selectedCombination: any = {};    imageList = [];    selectedVariant: any;    isSelectedCombinationInvalid = false;    isModelItem = false;    isBundleItemShipIndependent = false;    isBundleCompInStock = false;    actionSub: Subscription;    enterpriseCode: string;    isItemInStock = false;    orderLineKey = '';    selectedFulfillmentDetails: any;    productToRemoveFromCartKey: string;    hasPriceOverridePermission = false;    overridePriceInfo;    EXTENSION = {      TOP: ExtensionConstants.PRODUCT\_DETAILS\_CC\_TOP,      BOTTOM: ExtensionConstants.PRODUCT\_DETAILS\_CC\_BOTTOM    };    quantity = {      orderedQty: 1,      minQuantity: 1,      isQuantityInvalid: false,      quantityInvalidText: '',      quantityLimitOverride: false,      hasQuantityOverridePermission: false,      showQtyOverrideCheckbox: false    }    showGoToOrder = false        @ViewChild('radioLabelTpl', { static: true }) radioLabelTpl: TemplateRef<any>;    @ViewChild('productTabs') productTabs: ElementRef;    @ViewChild('productItemID', { static: true }) productIDTpl: TemplateRef<any>;    @ViewChild('productDescription', { static: true }) productDescTpl: TemplateRef<any>;    @ViewChild(ProductTabsComponent, { static: false }) productTabsComponent: ProductTabsComponent;    readonly resourceIdAddToCart = "ICC000001";    readonly resourceIdAddToOrder = "ICC000061";    protected readonly nlsMap: any = {      'PRODUCT\_DETAIL.RADIO\_OPTIONS.LABEL\_SHIPPING': '',      'PRODUCT\_DETAIL.RADIO\_OPTIONS.LABEL\_PICKUP': '',      'PRODUCT\_DETAIL.NOTIFICATIONS.MSG\_ADDED\_TO\_CART': '',      'PRODUCT\_DETAIL.NOTIFICATIONS.MSG\_QTY\_FAILING\_VALIDATION': ''    };    selectedTab = '';    @Output() sendCartDetails: EventEmitter<any> = new EventEmitter();    @Output() sendCartErrorDetails: EventEmitter<any> = new EventEmitter();    productQtyInCart: any;    isCountryInvalid = false;    shipNode: any;    lcl = localeBuc2Angular(BucSvcAngularStaticAppInfoFacadeUtil.getUserLanguage());    public currPipe: BucCommonCurrencyFormatPipe = new BucCommonCurrencyFormatPipe(this.lcl)      constructor(      private translateService: TranslateService,      private productDetailsService: ProductDetailsService,      private actionProcessorService: ActionProcessorService,      private activatedRoute: ActivatedRoute,      private ccNotificationService: CCNotificationService,      private bucNS: BucNotificationService,      private orderStoreService: OrderStoreService,      private storeDataService: StoreDataService,      private displayRuleService: DisplayRulesHelperService,      private addToOrderService: AddToOrderService,      private route: ActivatedRoute,      private tabMessageService: TabMessageService,      inj: Injector    ) {      super(inj)    }    ngOnInit(): void {      this.initialize();      if(this.productIdentifierInfo.productBrowsingStandalone) {        const sessionId = BucSvcAngularStaticAppInfoFacadeUtil.getCurrentCCSessionId();        const uniqueId = this.route.snapshot.queryParams.uniqueId;        this.tabMessageService.registerTabForRefocusNotification(sessionId, uniqueId, this.onTabRefocus.bind(this, sessionId, uniqueId));      }      this.actionSub = this.actionProcessorService.selectUpdate<any>(this.componentId).subscribe(res => {        if (res.params.selectedStore) {          this.handleStoreSelectionResp(res.params.selectedStore);        }        if (res.params.overridePrice) {          this.overridePriceInfo = res.params.overridePrice;        }      })    }    onTabRefocus(sessionId, uniqueId) {      this.checkOpenDraftOrders();      window.postMessage({        action: Constants.NOTIFY\_TAB\_REFOCUS,        data: { sessionId, uniqueId }      }, getPostMessageDomain());    }    ngOnChanges(c: SimpleChanges): void {      if (c.productIdentifierInfo && !c.productIdentifierInfo.isFirstChange() && c.productIdentifierInfo.currentValue !== c.productIdentifierInfo.previousValue) {        this.initialize();      }    }    ngOnDestroy() {      if (this.actionSub) {        this.actionSub.unsubscribe();      }    }    async initialize(): Promise<any> {      this.resetValues();      const queryParams = this.activatedRoute.snapshot.queryParams;      this.enterpriseCode = queryParams?.sellerEnterpriseCode;      if (!this.currency) {        const userOrgsList = this.getUserOrgsList();        if (userOrgsList) {          const matchingOrg = userOrgsList.find(org => org.value === this.enterpriseCode);          this.currency = matchingOrg?.currency;        }      }      this.hasPriceOverridePermission = BucSvcAngularStaticAppInfoFacadeUtil.canUserAccessResource('ICC000021');      if (!this.productIdentifierInfo.orderHeaderKey) {        this.productIdentifierInfo.orderHeaderKey = queryParams?.orderHeaderKey;      }      if (this.productIdentifierInfo.enterpriseCode) {        this.enterpriseCode = this.productIdentifierInfo.enterpriseCode;      }      // Read item data from session to load product details if this page is redirected from product browsing      if (this.activatedRoute.snapshot.queryParams?.fromProductBrowsing === 'Y' && this.productIdentifierInfo.readFromSession) {        this.isModelItem = this.productIdentifierInfo.isModelItem;        this.selectedVariant = this.productIdentifierInfo.selectedVariant;        this.selectedTab = this.productIdentifierInfo.selectedTab;        if (this.quantity) {          this.quantity.orderedQty = this.productIdentifierInfo.quantity;          this.quantity.quantityLimitOverride = this.productIdentifierInfo.quantityLimitOverride;          this.quantity.hasQuantityOverridePermission = this.productIdentifierInfo.hasQuantityOverridePermission;          this.quantity.showQtyOverrideCheckbox = this.productIdentifierInfo.showQtyOverrideCheckbox;        }      }      this.\_initTranslations();      await this.getProductDetails();      this.shipNode = this.shipNode ?? this.productIdentifierInfo.shipNode;      this.initFulfillmentRadioOptions();      /\*COAFEE - 510 (Product Variations - 275) - To select the variants for the itemId that is scanned OR condition added - ||this.productIdentifierInfo.itemID      ProductIdentifierInfo we are getting from product-search-component.ts so getting the itemId which is scanned -- START\*/      if (this.productIdentifierInfo?.selectedVariantID || this.productIdentifierInfo.itemID) {        //COAFEE-510 (Product variations - 275) - if we got itemId and that is Modelitem = N then set selectedVariantId = itemId        if (this.productIdentifierInfo.itemID) {          if (this.isModelItem.toString() === 'N') {            this.productIdentifierInfo.selectedVariantID = this.productIdentifierInfo.itemID;          }        }      //Coafee-510 (275) -- ENDS        const combination = { ...this.allowedCombinations.find((item) => item.ItemID === this.productIdentifierInfo.selectedVariantID) };          this.attributeList.forEach((attribute) => {          const option = attribute.options.find((opt) => opt.DisplayAttributeValue === combination[attribute.id])          this.onAttributeChange(attribute, option);        })        this.productToRemoveFromCartKey = this.orderLineKey;        this.orderLineKey = '';      }        if (this.productIdentifierInfo.selectedFulfillmentMethod === 'PICK' && !this.shipNode) {        this.isItemInStock = false;      } else {        if (this.productIdentifierInfo.selectedFulfillmentMethod === 'PICK') {          await this.fetchPickProductAvailablity();        }        this.checkProductAvailablity();      }      if (this.productIdentifierInfo.quantity) {        this.quantity.orderedQty = this.productIdentifierInfo.quantity;      }      if (this.activatedRoute.snapshot.queryParams?.fromProductBrowsing === 'Y' && this.productIdentifierInfo.readFromSession) {        // Call add to cart on load if the current page is redirected to here from product browsing        if (!this.productIdentifierInfo.associatedItemID) {          this.addToCart();        }        this.setReadFromSessionFalse.emit();      }      this.checkOpenDraftOrders();    }    checkOpenDraftOrders() {      this.showGoToOrder = this.productIdentifierInfo.productBrowsingStandalone ? this.addToOrderService.hasOpenCreateOrderTabs(this.enterpriseCode) : false;    }    public getUserOrgsList(): any { // gets org list from sessionStorage      const keys = Object.keys(sessionStorage);      const key = keys.find((key) => key.includes("userOrgsList"));      const value = key ? JSON.parse(sessionStorage.getItem(key)) : null;      return value;    }    afterProductTabsLoaded() {      if (this.activatedRoute.snapshot.queryParams?.fromProductBrowsing === 'Y' && this.productIdentifierInfo.readFromSession) {        // Call add to cart on load if the current page is redirected to here from product browsing        if (this.productIdentifierInfo.associatedItemID && this.productIdentifierInfo.associatedItemUOM) {          const item = this.productTabsComponent.associatedProductsTabComponent.associatedItems.find(el => {            return (el.Item.ItemID === this.productIdentifierInfo.associatedItemID) && (el.Item.UnitOfMeasure === this.productIdentifierInfo.associatedItemUOM)          });          if (item && item.Item) {            this.productTabsComponent.associatedProductsTabComponent.addToCart(item.Item);          }        }        this.setReadFromSessionFalse.emit();      }    }    toLowerCase(str: string = null) {      return str ? str.toLowerCase() : str;    }    protected async \_initTranslations(): Promise<any> {      const keys = Object.keys(this.nlsMap);      const json = await this.translateService.get(keys).toPromise();      keys.forEach(k => this.nlsMap[k] = json[k]);    }    private async getProductDetails() {      const hasRules = this.rulesConfig && this.rulesConfig[PRODUCT\_RULES.pickupStoreEnabled] && this.rulesConfig[PRODUCT\_RULES.cacheInventory];      const resp = await this.productDetailsService.getProductDetails(this.prepareGetCompleteItemListApiInput(), hasRules);        if (!hasRules) {        this.rulesConfig[PRODUCT\_RULES.pickupStoreEnabled] = resp.pickupEnabledRule?.Rules.RuleSetValue;       this.rulesConfig[PRODUCT\_RULES.cacheInventory] = resp.cacheInventoryRule?.Rules.RuleSetValue;        }      if (resp?.productDetails?.ItemList?.Item?.length > 0) {        this.productDetails = resp.productDetails.ItemList.Item[0];        const qty = parseInt(this.productDetails.PrimaryInformation.MinOrderQuantity);        this.quantity.orderedQty = this.quantity.minQuantity = qty > 1 ? qty : 1;          /\*Commented for COAFEE-510 (Product variations to come for ModelItem=N -- 275)         this.isModelItem = this.productDetails.PrimaryInformation.IsModelItem === CC\_CONSTANTS.CHECK\_YES;\*/        this.isModelItem = this.productDetails.PrimaryInformation.IsModelItem ;        this.setAssetList(this.productDetails.AssetList?.Asset);        //if (this.isModelItem && resp.productDetails.ItemList.ItemList?.length > 0) -- Removed this.isModelItem for COAFEE-510 (275) - Product variation to show for ModelItem=N        if (resp.productDetails.ItemList.ItemList?.length > 0) {          this.variationDetails = resp.productDetails.ItemList.ItemList[0];          this.setAttributeList();          this.setAllowedCombinationList(resp.productDetails.ItemList.ItemList[0].AllowedCombinationList.AllowedCombination ?? []);        }        if(isEmpty(this.productDetails.ComputedPrice)){          this.productDetails.ComputedPrice.UnitPrice = '0';        }        this.isBundleItemShipIndependent = (this.productDetails.PrimaryInformation.KitCode === "BUNDLE") && (this.productDetails.PrimaryInformation.BundleFulfillmentMode === "00");      }      await this.initializeFieldDetailAttributes('product-details');      await this.loadPageAttributes().toPromise();      this.isScreenInitialized = true;      this.selectedTab = this.productIdentifierInfo.selectedTab;      this.productQtyInCart = this.productIdentifierInfo.productQtyInCart;      this.orderLineKey = this.productIdentifierInfo.orderLineKey;      if (this.selectedTab) {        setTimeout(() => {          this.productTabs.nativeElement.scrollIntoView({ behavior: "smooth", block: "start" });        }, 0);      }    }    protected fetchPageAttributeData(): Observable<Array<any>> {      return of(getArray(this.productDetails));    }    get isAddToCartDisabled() {      return this.isReturnOrder ? this.productDetails?.PrimaryInformation?.IsValid === 'N'      : (this.isModelItem && !this.selectedVariant) || (this.isBundleItemShipIndependent && !this.isBundleCompInStock) || (!this.isBundleItemShipIndependent && !this.isItemInStock)        || !this.quantity.orderedQty || this.quantity.isQuantityInvalid        || (!this.isBundleItemShipIndependent && this.getNumericValue(this.selectedFulfillmentDetails?.availablity?.CurrentAvailableQty) && this.quantity.orderedQty > this.getNumericValue(this.selectedFulfillmentDetails?.availablity?.CurrentAvailableQty))        || (!this.isBundleItemShipIndependent && this.getNumericValue(this.selectedFulfillmentDetails?.availablity?.FutureAvailableQuantity) && this.quantity.orderedQty > this.getNumericValue(this.selectedFulfillmentDetails?.availablity?.FutureAvailableQuantity))    }    protected getDataForAttribute(id: string, item: any): Promise<any> {      let attr: any = {};      let data: any = '';      switch (id) {        case this.FIELD\_ITEM\_ID:          data = { template: this.productIDTpl, templateData: { value: this.productIdentifierInfo } };          break;        case this.FIELD\_DESC:          data = { template: this.productDescTpl, templateData: item };          break;      }      attr = { ...attr, data };      return attr as any;    }    onCountrySelectionChange(item) {      this.isCountryInvalid = false;      this.productIdentifierInfo.country = item?.id;      if (!isEmpty(this.productIdentifierInfo.countries)) {        this.productIdentifierInfo.countries.forEach(el => el.selected = el.id === item.id);      }    }    onCountryValSearch(evt) {      this.isCountryInvalid = false;      if (evt) {        const searchedCountry = this.productIdentifierInfo.countries.find((item) => item.content.toLowerCase() === evt.toLowerCase());        this.isCountryInvalid = !searchedCountry;        if (searchedCountry) {          this.productIdentifierInfo.country = searchedCountry.id;        }      } else {        this.productIdentifierInfo.countries.forEach(el => el.selected = false);        this.isCountryInvalid = true;        this.productIdentifierInfo.country = '';      }    }      private prepareGetCompleteItemListApiInput() {      let fulfillmentMathodValue = this.productIdentifierInfo.selectedFulfillmentMethod === 'SHP' ? this.productIdentifierInfo.shipTo : this.productIdentifierInfo.shipNode?.ShipNode;      if (this.fulfillmentOptions?.length) {        const selectedMethod = this.fulfillmentOptions.find(el => el.checked);        this.productIdentifierInfo.selectedFulfillmentMethod = selectedMethod.id;        fulfillmentMathodValue = selectedMethod.id === 'SHP' ? selectedMethod.fulfillmentValue : selectedMethod.fulfillmentValue?.ShipNode;      }      //COAFEE-510 -- GetAvailabilityFromCache: 'N' - Out of stock issue resolution      const input = {        Item: {          ItemID: this.productIdentifierInfo.parentItemID ? this.productIdentifierInfo.parentItemID : this.productIdentifierInfo.itemID,          Currency: this.currency,          CallingOrganizationCode: this.enterpriseCode,          GetAvailabilityFromCache: 'N',          UnitOfMeasure: this.productIdentifierInfo.unitOfMeasure,          ProductClass: this.productIdentifierInfo.defaultProductClass,          ...(this.productIdentifierInfo.selectedFulfillmentMethod === 'SHP' && fulfillmentMathodValue) && {            ShipToAddress: {              ZipCode: fulfillmentMathodValue,              Country: this.productIdentifierInfo.country || '',              City: '',              State: ''            }          },          ...(this.productIdentifierInfo.selectedFulfillmentMethod === 'PICK' && fulfillmentMathodValue) && {            ShipNodes: {              ShipNode: {                Node: fulfillmentMathodValue              }            }          },          ...(this.customerInformation?.customerId || this.customerInformation?.customerContactId) && {            CustomerInformation: {              CustomerContactID: this.customerInformation?.customerContactId,              CustomerID: this.customerInformation?.customerId            }          }        }      };      return input;    }    private prepareGetSingleStoreAvailablity() {      const searchStoresInput = {        AlternateStore: {          Mode: '02',          OrganizationCode: this.enterpriseCode,          NodeList: {            Node: {              ShipNode: this.shipNode.ShipNode,            }          },          OrderLines: {            OrderLine: {              Item: {                ItemID: this.isModelItem ? this.selectedVariant.ItemID : this.productDetails.ItemID,                UnitOfMeasure: this.isModelItem ? this.selectedVariant.UnitOfMeasure : this.productDetails.UnitOfMeasure,                ProductClass: this.isModelItem ? this.selectedVariant.PrimaryInformation.DefaultProductClass : this.productDetails.PrimaryInformation.DefaultProductClass              }            }          }        }      };      return searchStoresInput;    }    private setAttributeList() {      this.attributeList = [];      this.variationDetails.AttributeList?.Attribute?.map((variation) => {        this.attributeList.push({          id: variation.AttributeID,          sequence: variation.SequenceNo,          label: variation.ShortDescription,          options: variation.AssignedValueList?.AssignedValue        })      })    }    private setAllowedCombinationList(combinations) {      this.allowedCombinations = combinations.map((combination) =>        combination.ItemAttributeList.ItemAttribute.reduce((acc, obj) => {          acc[obj.ItemAttributeName] = obj.DisplayAttributeValue;          return acc;        }, { ItemID: combination.ItemID }));    }    private setAssetList(assets) {      this.imageList = []      if (assets?.length) {        const firstObj = assets.find(ele => {          return ele.Type === 'ITEM\_IMAGE\_1';        });        if (firstObj) {          this.imageList.push({            url: firstObj.ContentLocation + '/' + firstObj.ContentID,            active: true          })        } else {          this.imageList.push({            url: this.productDetails.PrimaryInformation?.ImageLocation + '/' + this.productDetails.PrimaryInformation?.ImageID,            active: true          })        }        assets.forEach(ele => {          if (ele.Type === 'ITEM\_IMAGE\_LRG\_1') {            this.imageList.push({              url: ele.ContentLocation + '/' + ele.ContentID,              active: false            });          }        });      } else {        this.imageList.push({          url: this.productDetails.PrimaryInformation?.ImageLocation + '/' + this.productDetails.PrimaryInformation?.ImageID,          active: true        })      }    }    private initFulfillmentRadioOptions() {      this.fulfillmentOptions = [        {          id: 'SHP',          value: 'SHP',          template: this.radioLabelTpl,          checked: this.productDetails.PrimaryInformation.IsShippingAllowed === CC\_CONSTANTS.CHECK\_YES && this.productIdentifierInfo.selectedFulfillmentMethod !== 'PICK',          disabled: this.productDetails.PrimaryInformation.IsShippingAllowed !== CC\_CONSTANTS.CHECK\_YES || this.isModelItem,          key: 'PRODUCT\_DETAIL.RADIO\_OPTIONS.LABEL\_SHIPPING',          link: 'PRODUCT\_DETAIL.RADIO\_OPTIONS.LABEL\_ENTER\_POSTAL\_CODE',          showLink: this.productIdentifierInfo.shipTo ? false : true,          showEditIcon: this.productIdentifierInfo.shipTo ? true : false,          fulfillmentValue: this.productIdentifierInfo.shipTo,          availablity: this.productIdentifierInfo.selectedFulfillmentMethod !== 'PICK' ? this.productDetails.Availability : {},          showAvailablity: this.productIdentifierInfo.selectedFulfillmentMethod !== 'PICK',          bundleCompAvailablity: false        },        ...(this.rulesConfig[PRODUCT\_RULES.pickupStoreEnabled] === 'Y') ? [{          id: 'PICK',          value: 'PICK',          template: this.radioLabelTpl,          checked: (this.productDetails.PrimaryInformation.IsPickupAllowed === CC\_CONSTANTS.CHECK\_YES) && (this.productIdentifierInfo.selectedFulfillmentMethod === 'PICK'),          disabled: (this.productDetails.PrimaryInformation.IsPickupAllowed !== CC\_CONSTANTS.CHECK\_YES) || this.isModelItem,          key: 'PRODUCT\_DETAIL.RADIO\_OPTIONS.LABEL\_PICKUP',          link: 'PRODUCT\_DETAIL.RADIO\_OPTIONS.LABEL\_PICKUP\_STORES',          showLink: this.productIdentifierInfo.selectedFulfillmentMethod === 'PICK' && this.shipNode?.ShipNode ? false : true,          showEditIcon: this.productIdentifierInfo.selectedFulfillmentMethod === 'PICK' && this.shipNode?.ShipNode ? true : false,          fulfillmentValue: this.shipNode,          availablity: this.productIdentifierInfo.selectedFulfillmentMethod === 'PICK' ? this.productDetails.Availability : {},          bundleCompAvailablity: false        }] : []      ];    }    private async fetchPickProductAvailablity() {      this.selectedFulfillmentDetails = this.fulfillmentOptions.find((el) => el.checked);      const resp = await this.storeDataService.getSingleStoreAvailablity(this.prepareGetSingleStoreAvailablity());      const storeNode = resp.AlternateStores?.NodeList?.Node[0];      if (!isEmpty(storeNode.Availability)) {        if (storeNode.Availability.IsAvailable === 'N') {          // Out of stock          this.selectedFulfillmentDetails.availablity.CurrentAvailableQty = 0;          this.selectedFulfillmentDetails.availablity.FutureAvailableQuantity = 0;        } else if (storeNode.Availability.IsFutureAvailability === 'N') {          // In stock          this.selectedFulfillmentDetails.availablity.CurrentAvailableQty = resp?.AlternateStores?.NodeList?.Node[0]?.Availability?.AvailableQty ?? this.selectedFulfillmentDetails.fulfillmentValue.Availability.AvailableQty;        } else if (storeNode.Availability.IsFutureAvailability === 'Y') {          // Available on Future date          this.selectedFulfillmentDetails.availablity.CurrentAvailableQty = 0;          this.selectedFulfillmentDetails.availablity.FutureAvailableQuantity = storeNode.Availability?.AvailableQty ?? this.selectedFulfillmentDetails.fulfillmentValue.Availability.AvailableQty;          this.selectedFulfillmentDetails.availablity.FutureAvailableDate = storeNode.Availability?.AvailableDate ?? this.selectedFulfillmentDetails.fulfillmentValue.Availability.AvailableDate;        }      }    }    private checkProductAvailablity() {      this.selectedFulfillmentDetails = this.fulfillmentOptions.find((el) => el.checked);      if (this.isBundleItemShipIndependent) {        this.selectedFulfillmentDetails.bundleCompAvailablity = this.isBundleCompInStock = this.productDetails.Components?.Component?.every((item) => item.Availability &&          parseInt(item.Availability.CurrentAvailableQty) >= parseInt(item.Item.PrimaryInformation.MinOrderQuantity));      } else {        this.isItemInStock = this.selectedFulfillmentDetails.availablity && ((parseInt(this.selectedFulfillmentDetails.availablity.CurrentAvailableQty) >= this.quantity.minQuantity) ||          (parseInt(this.selectedFulfillmentDetails.availablity.FutureAvailableQuantity) > 0));      }    }    onAttributeChange(changedAttribute, selected) {        if (changedAttribute.selectedOption !== selected.DisplayAttributeValue) {        this.selectedCombination[changedAttribute.id] = selected.DisplayAttributeValue;        changedAttribute.selectedOption = selected.DisplayAttributeValue;        this.attributeList.forEach((attribute) => {          if (attribute.id !== changedAttribute.id) {            attribute.invalidCombination = this.checkAttributeCombination(attribute);          }        })        if (Object.keys(this.selectedCombination).length === this.attributeList.length) {          this.setSelectedVariant();          }      }    }    private checkAttributeCombination(attribute) {      attribute.options.forEach(variationItem => {        variationItem.invalidCombination = true;        const newCombination = { ...this.selectedCombination, [attribute.id]: variationItem.DisplayAttributeValue };        if (this.checkCombination(newCombination, attribute.id, variationItem.DisplayAttributeValue)) {          variationItem.invalidCombination = false;        }      });    }    private checkCombination(newCombination: any, id, value) {      let filteredCombinations = [...this.allowedCombinations];      Object.keys(newCombination).forEach((key) => {        filteredCombinations = filteredCombinations.filter((item) => item[key] === newCombination[key]);      })      return filteredCombinations.find((item) => item[id] === value);    }    private setSelectedVariant() {      this.selectedVariant = null;      const isValidCombination = this.allowedCombinations.find((comb) => {        const { ItemID, ...allowedComb } = comb;        return isEqual(allowedComb, this.selectedCombination);      });      if (isValidCombination) {        this.isSelectedCombinationInvalid = false;        this.selectedVariant = this.variationDetails.Item.find((item) => item.ItemID === isValidCombination.ItemID);        const qty = parseInt(this.selectedVariant.PrimaryInformation.MinOrderQuantity);        this.quantity.minQuantity = qty > 1 ? qty : 1;        this.setAssetList(this.selectedVariant?.AssetList?.Asset);        this.fulfillmentOptions.forEach((item) => {          if (item.id === 'SHP') {            item.disabled = this.selectedVariant.PrimaryInformation.IsShippingAllowed !== CC\_CONSTANTS.CHECK\_YES;            item.availablity = this.selectedVariant.Availability;            this.checkProductAvailablity();          } else {            item.disabled = (this.selectedVariant.PrimaryInformation.IsPickupAllowed !== CC\_CONSTANTS.CHECK\_YES) || (this.rulesConfig[PRODUCT\_RULES.pickupStoreEnabled] === 'N');            item.availablity = this.selectedVariant.Availability;            this.checkProductAvailablity();          }        });      } else {        this.isSelectedCombinationInvalid = true;        this.fulfillmentOptions.forEach((item) => item.disabled = true);      }    }    private openSelectStoreModal(item) {      const nodeInfo = this.fulfillmentOptions.find((el) => el.id === 'PICK').fulfillmentValue;      let shipToAddress;      if (nodeInfo) {        shipToAddress = nodeInfo.ShipNodePersonInfo;      } else {        const ordDetails = this.orderStoreService.getOrderDetails();        shipToAddress = ordDetails?.Order?.PersonInfoShipTo;      }      this.actionProcessorService.dispatch<ActionParams>(CC\_CONSTANTS.SELECT\_STORE, {        component: this.componentId,        data: {          modalText: '',          modalData: {            enterpriseCode: this.enterpriseCode,            personInfoShipTo: shipToAddress,            orderLines: {              RequiredQty: this.quantity.orderedQty,              Item: {                ItemID: this.isModelItem ? this.selectedVariant.ItemID : this.productDetails.ItemID,                UnitOfMeasure: this.isModelItem ? this.selectedVariant.UnitOfMeasure : this.productDetails.UnitOfMeasure,                ProductClass: this.isModelItem ? this.selectedVariant.PrimaryInformation.DefaultProductClass : this.productDetails.PrimaryInformation.DefaultProductClass              }            },            selectedShipNode: nodeInfo?.ShipNode,            size: 'lg'          },        }      });    }    handleStoreSelectionResp(selectedStore) {      const item = this.fulfillmentOptions.find(el => el.checked);      this.shipNode = selectedStore ?? null;      item.fulfillmentValue = selectedStore;      item.showLink = false;      item.showEditIcon = true;      if (!isEmpty(item.fulfillmentValue.Availability)) {        if (item.fulfillmentValue.Availability.IsAvailable === 'N') {          // Out of stock          item.availablity.CurrentAvailableQty = 0;          item.availablity.FutureAvailableQuantity = 0;        } else if (item.fulfillmentValue.Availability.IsFutureAvailability === 'N') {          // In stock          item.availablity.CurrentAvailableQty = item.fulfillmentValue.Availability.AvailableQty;        } else if (item.fulfillmentValue.Availability.IsFutureAvailability === 'Y') {          // Available on Future date          item.availablity.CurrentAvailableQty = 0;          item.availablity.FutureAvailableQuantity = item.fulfillmentValue.Availability.AvailableQty;          item.availablity.FutureAvailableDate = item.fulfillmentValue.Availability.AvailableDate;        }        this.checkProductAvailablity();      }    }    async addToCart(associatedItem?) {      const selectedOption = this.fulfillmentOptions.find(el => el.checked);      this.setFulfillmentData(selectedOption);      if (this.productIdentifierInfo.productBrowsingStandalone) {        const itemInfoObj = {          isModelItem: this.isModelItem,          itemID: this.productDetails.ItemID,          showQtyOverrideCheckbox: this.quantity.showQtyOverrideCheckbox,          hasQuantityOverridePermission: this.quantity.hasQuantityOverridePermission,          quantityLimitOverride: this.quantity.quantityLimitOverride,          minQuantity: this.quantity.minQuantity,          orderedQty: this.quantity.orderedQty,          orderLineKey: this.orderLineKey,          selectedVariant: this.selectedVariant,          associatedItemID: associatedItem ? associatedItem.associatedItemID : null,          associatedItemUOM: associatedItem ? associatedItem.associatedItemUOM : null,          deliveryMethod: selectedOption,          shipNode: selectedOption.id === 'PICK' ? this.fulfillmentOptions.find(el => el.checked).fulfillmentValue : null,          zipCode: selectedOption.id === 'SHP' ? this.fulfillmentOptions.find(el => el.checked).fulfillmentValue : null,          selectedTab: this.productTabsComponent.selectedTab        };        this.navigateToCreateOrder.emit(itemInfoObj);      } else {          //COAFEE- 510 - For Adding the scanned itemId of auro selected product variation for modelitem = N to the Cart - START         const strSelectedvariant = this.productIdentifierInfo.selectedVariantID;         this.productIdentifierInfo.selectedVariantID = null;         //COAFEE-510 - ENDS        const ordDetails = this.orderStoreService.getOrderDetails();        const IsPriceLocked = this.isExchangeOrder ? this.displayRuleService.getRuleValueForOrg( ordDetails.Order?.EnterpriseCode, 'ICC\_EXCHANGE\_PRICE\_LOCKED') : 'Y';        if (BucBaseUtil.isVoid(ordDetails.Order?.OrderHeaderKey)) {            const itemDetails = this.getItemDetails();          this.handleCreateOrder.emit(itemDetails);        } else {            const cartCount = ordDetails.Order?.TotalNumberOfParentLines;          const input = {            Order: {              EnterpriseCode: this.enterpriseCode,              OrderHeaderKey: this.productIdentifierInfo.orderHeaderKey,              ...(this.hasPriceOverridePermission && this.overridePriceInfo && cartCount > 0 && { isPriceOverride: 'Y' }),              OrderLines: {                OrderLine: [                  {                    OrderLineKey: this.orderLineKey,                    DeliveryMethod: selectedOption.id,                    ...(selectedOption.id === 'PICK') && {                      ShipNode: selectedOption.fulfillmentValue?.ShipNode                    },                    OrderedQty: this.quantity.minQuantity,                    Item: {                      //COAFEE-510 For updating ItemId for ModelItem=N on select varaint ON UI Description -this.isModelItem Is replaced with this.selectedVariant                      /\*ItemID: this.isModelItem ? this.selectedVariant.ItemID : this.productDetails.ItemID,                      UnitOfMeasure: this.isModelItem ? this.selectedVariant.UnitOfMeasure : this.productDetails.UnitOfMeasure,                      ProductClass: this.isModelItem ? this.selectedVariant.PrimaryInformation.DefaultProductClass : this.productDetails.PrimaryInformation.DefaultProductClass                    }\*/                      ItemID: this.selectedVariant ? this.selectedVariant.ItemID : this.productDetails.ItemID,                      UnitOfMeasure: this.selectedVariant ? this.selectedVariant.UnitOfMeasure : this.productDetails.UnitOfMeasure,                      ProductClass: this.selectedVariant ? this.selectedVariant.PrimaryInformation.DefaultProductClass : this.productDetails.PrimaryInformation.DefaultProductClass                    },                    LinePriceInfo: {                      ...(this.hasPriceOverridePermission && this.overridePriceInfo) ? { IsPriceLocked } : {},                      /\* COAFEE-510 - this.isModelItem Is replaced with this.selectedVariant                      ListPrice: this.isModelItem ? this.selectedVariant.ComputedPrice?.ListPrice : this.productDetails.ComputedPrice?.ListPrice,                      UnitPrice: this.overridePriceInfo ? this.overridePriceInfo.price : (this.isModelItem ? this.selectedVariant.ComputedPrice?.UnitPrice : this.productDetails.ComputedPrice?.UnitPrice),                      BundleTotal: (this.overridePriceInfo && this.productDetails.PrimaryInformation.KitCode === "BUNDLE") ? this.overridePriceInfo.price : (this.isModelItem ? this.selectedVariant.ComputedPrice?.BundleTotal : this.productDetails.ComputedPrice?.BundleTotal)                    },\*/                      ListPrice: this.selectedVariant ? this.selectedVariant.ComputedPrice?.ListPrice : this.productDetails.ComputedPrice?.ListPrice,                      UnitPrice: this.overridePriceInfo ? this.overridePriceInfo.price : (this.selectedVariant ? this.selectedVariant.ComputedPrice?.UnitPrice : this.productDetails.ComputedPrice?.UnitPrice),                      BundleTotal: (this.overridePriceInfo && this.productDetails.PrimaryInformation.KitCode === "BUNDLE") ? this.overridePriceInfo.price : (this.selectedVariant ? this.selectedVariant.ComputedPrice?.BundleTotal : this.productDetails.ComputedPrice?.BundleTotal)                    },                    OrderLineTranQuantity: {                      OrderedQty: this.quantity.orderedQty                    },                    ...(this.quantity.showQtyOverrideCheckbox && this.quantity.hasQuantityOverridePermission && this.quantity.quantityLimitOverride) && {                      OrderOverride: {                        QuantityLimitOverridden: 'Y'                      }                    },                    ...(this.hasPriceOverridePermission && this.overridePriceInfo && IsPriceLocked === 'Y' && {                      Notes: {                        Note: {                          ReasonCode: "YCD\_NEW\_ITEM\_INFO",                          NoteText: this.overridePriceInfo.note ? this.overridePriceInfo.note : this.translateService.instant('OVERRIDE\_PRICE.GENERAL.MSG\_PRICE\_OVERRIDDEN\_NOTE', { reason: this.overridePriceInfo.reason }),                        }                      }                    })                  },                    ...(this.productIdentifierInfo.selectedVariantID && !this.productIdentifierInfo.readFromSession) ? [{                    Action: "REMOVE",                    OrderLineKey: this.productToRemoveFromCartKey                  }] : [],                ]              }            }          }            //COAFEE-510 - STARTS          this.productIdentifierInfo.selectedVariantID  =strSelectedvariant ;          //COAFEE-510 - ENDS          const resp = await this.productDetailsService.addProductToCart(input);          this.handleAddToCartResponse(resp);        }      }    }    getItemDetails() {      const selectedOption = this.fulfillmentOptions.find(el => el.checked);      const ordDetails = this.orderStoreService.getOrderDetails();      const IsPriceLocked = this.isExchangeOrder ? this.displayRuleService.getRuleValueForOrg( ordDetails.Order?.EnterpriseCode, 'ICC\_EXCHANGE\_PRICE\_LOCKED') : 'Y';      const itemDetails = {        ...this.productDetails,        /\* COAFEE-510 - this.isModelItem is replaced with this.selectedVariant          ItemID: this.isModelItem ? this.selectedVariant.ItemID : this.productDetails.ItemID,        UnitOfMeasure: this.isModelItem ? this.selectedVariant.UnitOfMeasure : this.productDetails.UnitOfMeasure,        ProductClass: this.isModelItem ? this.selectedVariant.PrimaryInformation.DefaultProductClass : this.productDetails.PrimaryInformation.DefaultProductClass,        ComputedPrice: {          ListPrice: this.isModelItem ? this.selectedVariant.ComputedPrice?.ListPrice : this.productDetails.ComputedPrice?.ListPrice,          UnitPrice: this.overridePriceInfo ? this.overridePriceInfo.price : (this.isModelItem ? this.selectedVariant.ComputedPrice?.UnitPrice : this.productDetails.ComputedPrice?.UnitPrice),          BundleTotal: (this.overridePriceInfo && this.productDetails.PrimaryInformation.KitCode === "BUNDLE") ? this.overridePriceInfo.price : (this.isModelItem ? this.selectedVariant.ComputedPrice?.BundleTotal : this.productDetails.ComputedPrice?.BundleTotal)        }\*/        ItemID: this.selectedVariant ? this.selectedVariant.ItemID : this.productDetails.ItemID,        UnitOfMeasure: this.selectedVariant ? this.selectedVariant.UnitOfMeasure : this.productDetails.UnitOfMeasure,        ProductClass: this.selectedVariant ? this.selectedVariant.PrimaryInformation.DefaultProductClass : this.productDetails.PrimaryInformation.DefaultProductClass,        ComputedPrice: {          ListPrice: this.selectedVariant ? this.selectedVariant.ComputedPrice?.ListPrice : this.productDetails.ComputedPrice?.ListPrice,          UnitPrice: this.overridePriceInfo ? this.overridePriceInfo.price : (this.selectedVariant ? this.selectedVariant.ComputedPrice?.UnitPrice : this.productDetails.ComputedPrice?.UnitPrice),          BundleTotal: (this.overridePriceInfo && this.productDetails.PrimaryInformation.KitCode === "BUNDLE") ? this.overridePriceInfo.price : (this.selectedVariant ? this.selectedVariant.ComputedPrice?.BundleTotal : this.productDetails.ComputedPrice?.BundleTotal)        },        DeliveryMethod: selectedOption.id,        ...(selectedOption.id === 'PICK') && {          ShipNode: selectedOption.fulfillmentValue?.ShipNode        },        OrderLineTranQuantity: {          OrderedQty: this.quantity.orderedQty        },        ...(this.quantity.showQtyOverrideCheckbox && this.quantity.hasQuantityOverridePermission && this.quantity.quantityLimitOverride) && {          OrderOverride: {            QuantityLimitOverridden: 'Y'          }        },        ...(this.hasPriceOverridePermission && this.overridePriceInfo && IsPriceLocked === 'Y' && {          Notes: {            Note: {              ReasonCode: "YCD\_NEW\_ITEM\_INFO",              NoteText: this.overridePriceInfo.note ? this.overridePriceInfo.note : this.translateService.instant('OVERRIDE\_PRICE.GENERAL.MSG\_PRICE\_OVERRIDDEN\_NOTE', { reason: this.overridePriceInfo.reason }),            }          }        })      };     return itemDetails;    }    onAddToOrder(associatedItem?) {      const selectedOption = this.fulfillmentOptions.find(el => el.checked);      this.setFulfillmentData(selectedOption);      const itemDetails = this.getItemDetails();      this.modalService.destroy();      this.modalService.create({        component: AddToOrderModalComponent,        inputs: {          modalData: {            size: 'md',            enterpriseCode: this.enterpriseCode,            itemDetails: associatedItem ?? itemDetails          }        }      });    }    setFulfillmentData(selectedOption) {      const shipNode = selectedOption.id === 'PICK' ? this.fulfillmentOptions.find(el => el.checked).fulfillmentValue : null;      const zipCode = selectedOption.id === 'SHP' ? this.fulfillmentOptions.find(el => el.checked).fulfillmentValue : null;      const fulfillmentItem = {        selectedFulfillMethodItem: selectedOption.id,        isZipCodeAvailableItem: zipCode ? true : false,        showZipCodeInputFieldItem: zipCode ? false : true,        pickupStoreAddrItem: shipNode,        isPickupStoreAddrAvailableItem: shipNode ? true : false,        zipCodeItem: zipCode,      };      this.updateFulfillmentData.emit(fulfillmentItem);    }    handleAddToCartResponse(resp) {      if (resp?.Order) {        if (resp.Order.OrderLines?.OrderLine?.length) {          this.orderStoreService.setOrderDetails(resp);          this.updateOrderSummary.emit();          const orderLines = resp.Order.OrderLines.OrderLine;          this.quantity.hasQuantityOverridePermission = BucSvcAngularStaticAppInfoFacadeUtil.canUserAccessResource('ICC000020');          const errorArr = [];          if (resp.Order.HasValidationErrors === CC\_CONSTANTS.CHECK\_YES) {            resp.Order.ModifiedOrderLines?.OrderLine?.forEach((line) => {              if (line.Errors?.Error?.length) {                const error = line.Errors?.Error[0];                this.orderLineKey = line.OrderLineKey;                if (error.ErrorCode === AddToCartErrorCodes.MinQuantity) {                  if (line.Item.ItemID === (this.isModelItem ? this.selectedVariant.ItemID : this.productDetails.ItemID)) {                    this.quantity.isQuantityInvalid = true;                    const qty = error.Attribute.find((item) => item.Name === 'MinOrderQuantity').Value;                    this.quantity.quantityInvalidText = this.translateService.instant('PRODUCT\_DETAIL.GENERAL.ERR\_MIN\_QUANTITY\_ALLOWED', { quantity: parseInt(qty) });                    this.handleAddToCartError(error, errorArr);                  } else {                    errorArr.push({ error, errorMsg: this.nlsMap['PRODUCT\_DETAIL.NOTIFICATIONS.MSG\_QTY\_FAILING\_VALIDATION'] });                  }                } else if (error.ErrorCode === AddToCartErrorCodes.MaxQuantity) {                  if (line.Item.ItemID === (this.isModelItem ? this.selectedVariant.ItemID : this.productDetails.ItemID)) {                    this.quantity.isQuantityInvalid = true;                    const qty = error.Attribute.find((item) => item.Name === 'MaxOrderQuantity').Value;                    this.quantity.quantityInvalidText = this.translateService.instant('PRODUCT\_DETAIL.GENERAL.ERR\_MAX\_QUANTITY\_ALLOWED', { quantity: parseInt(qty) });                    this.handleAddToCartError(error, errorArr);                  } else {                    errorArr.push({ error, errorMsg: this.nlsMap['PRODUCT\_DETAIL.NOTIFICATIONS.MSG\_QTY\_FAILING\_VALIDATION'] });                  }                } else {                  this.handleAddToCartError(error, errorArr);                }                this.sendCartErrorDetails.emit({ itemID: (this.isModelItem ? this.selectedVariant.ItemID : this.productDetails.ItemID), orderLineKey: this.orderLineKey });              }            });            this.bucNS.send(errorArr.map((err) => {              return new BucNotificationModel({                statusType: 'error',                statusContent: err.errorMsg,              });            }));            if (errorArr.length === 1) {              if (errorArr[0].error.ErrorCode === AddToCartErrorCodes.MinQuantity || errorArr[0].error.ErrorCode === AddToCartErrorCodes.MaxQuantity) {                this.quantity.showQtyOverrideCheckbox = true;              }            }          } else {            this.quantity.isQuantityInvalid = false;            this.quantity.showQtyOverrideCheckbox = false;            // Reset quantity and send cart details to parent on success            this.quantity.quantityInvalidText = '';            this.orderLineKey = '';            this.quantity.orderedQty = this.quantity.minQuantity;            this.sendCartDetails.emit({ itemID: (this.isModelItem ? this.selectedVariant.ItemID : this.productDetails.ItemID), orderLinesObj: resp.Order.OrderLines, resp });            // Update product quantity in cart            this.updateProductQtyInCart(orderLines);            this.ccNotificationService.notify({              type: 'success',              title: this.translateService.instant(                (this.isReturnOrder ? 'PRODUCT\_DETAIL.NOTIFICATIONS.MSG\_ADDED\_TO\_RETURN\_CART' : 'PRODUCT\_DETAIL.NOTIFICATIONS.MSG\_ADDED\_TO\_CART'),                { item: this.productDetails.PrimaryInformation.ShortDescription }              )            });          }        }      }    }    updateProductQtyInCart(orderLines){        // Update product quantity in cart        const orderLinesWithCurrentItem = orderLines.filter(line => (line.Item.ItemID === (this.isModelItem ? this.selectedVariant.ItemID : this.productDetails.ItemID)) && line.Item.UnitOfMeasure === this.productDetails.UnitOfMeasure);        const productInCartCount = orderLinesWithCurrentItem.reduce((acc, el) => {          return acc + Number(el.OrderedQty)        }, 0);        this.productQtyInCart = productInCartCount;    }    handleAddToCartError(error, errorArr) {      let errorMsg = '';      if (error) {        errorMsg = error.ErrorDescription;        const errorCode = error.ErrorCode;        const bundleKey = 'APIERROR.' + errorCode;        if (this.translateService.instant(bundleKey) !== bundleKey) {          errorMsg = this.translateService.instant(bundleKey);        }      }      errorArr.push({ error, errorMsg });    }    getNumericValue(val) {      if (val) {        return parseInt(val);      }    }    onOverrideCheckboxChange(value) {      this.quantity.quantityLimitOverride = value;      this.quantity.isQuantityInvalid = false;    }    onFulfillmentRadioChange(item) {      if (item?.source?.id) {        this.fulfillmentOptions.forEach(el => el.checked = el.id === item.source.id);        const selected = this.fulfillmentOptions.find(el => el.id === item.source.id);        if (this.isBundleItemShipIndependent && (!selected.fulfillmentValue || !selected.bundleCompAvailablity)) {          this.isBundleCompInStock = false;        } else {          this.checkProductAvailablity();        }      }    }    dispactFullfillmentAction(item) {      this.fulfillmentOptions.forEach(el => el.checked = el.id === item.id);      switch (item.id) {        case 'SHP':          item.showLink = false;          item.showEditIcon = false;          break;        case 'PICK':          this.openSelectStoreModal(item);          break;      }    }    fulfillmentOptionSaveOrEditClick(item) {      this.fulfillmentOptions.forEach(el => el.checked = el.id === item.id);      switch (item.id) {        case 'SHP':          if (item.showEditIcon) {            item.showEditIcon = false;          } else {            if (item.fulfillmentValue) {              this.getAvailablity(item);            }          }          break;        case 'PICK':          this.openSelectStoreModal(item);          break;      }    }    private async getAvailablity(item) {      const hasRules = this.rulesConfig && this.rulesConfig[PRODUCT\_RULES.pickupStoreEnabled] && this.rulesConfig[PRODUCT\_RULES.cacheInventory];        const resp = await this.productDetailsService.getProductDetails(this.prepareGetCompleteItemListApiInput(), hasRules);      if (resp?.productDetails?.ItemList?.Item?.length > 0) {        const details = resp.productDetails.ItemList.Item[0];        item.availablity = details.Availability;        if (this.isModelItem && resp.productDetails.ItemList.ItemList?.length > 0) {          this.variationDetails = resp.productDetails.ItemList.ItemList[0];          item.availablity = this.variationDetails.Item.find((item) => item.ItemID === this.selectedVariant?.ItemID).Availability;        }        this.checkProductAvailablity();      }      item.showLink = false;      item.showEditIcon = true;    }    private resetValues() {      this.isScreenInitialized = false;      this.imageList = [];      this.selectedVariant = null;      this.selectedCombination = {};      this.isSelectedCombinationInvalid = false;      this.overridePriceInfo = null;      this.quantity = {        orderedQty: 1,        minQuantity: 1,        isQuantityInvalid: false,        quantityInvalidText: '',        quantityLimitOverride: false,        hasQuantityOverridePermission: false,        showQtyOverrideCheckbox: false      }    }    openOverridePriceModal() {      const price = !this.isModelItem ?        (this.productDetails.ComputedPrice.BundleTotal || this.productDetails.ComputedPrice.UnitPrice) :        (this.selectedVariant.ComputedPrice?.BundleTotal || this.selectedVariant.ComputedPrice?.UnitPrice);      this.actionProcessorService.dispatch<ActionParams>(CC\_CONSTANTS.OVERRIDE\_PRICE, {        component: this.componentId,        data: {          modalText: '',          modalData: {            enterpriseCode: this.enterpriseCode,            price: this.currPipe.transform(isNaN(price) ? 0 : price, this.productDetails.Currency ?? this.currency, 'symbol'),            currency: this.productDetails.Currency ?? this.currency,            orderLine: {              ItemDetails: {                PrimaryInformation: this.productDetails.PrimaryInformation,                ItemID: this.productIdentifierInfo.itemID              }            }          }        }      });    }  } |

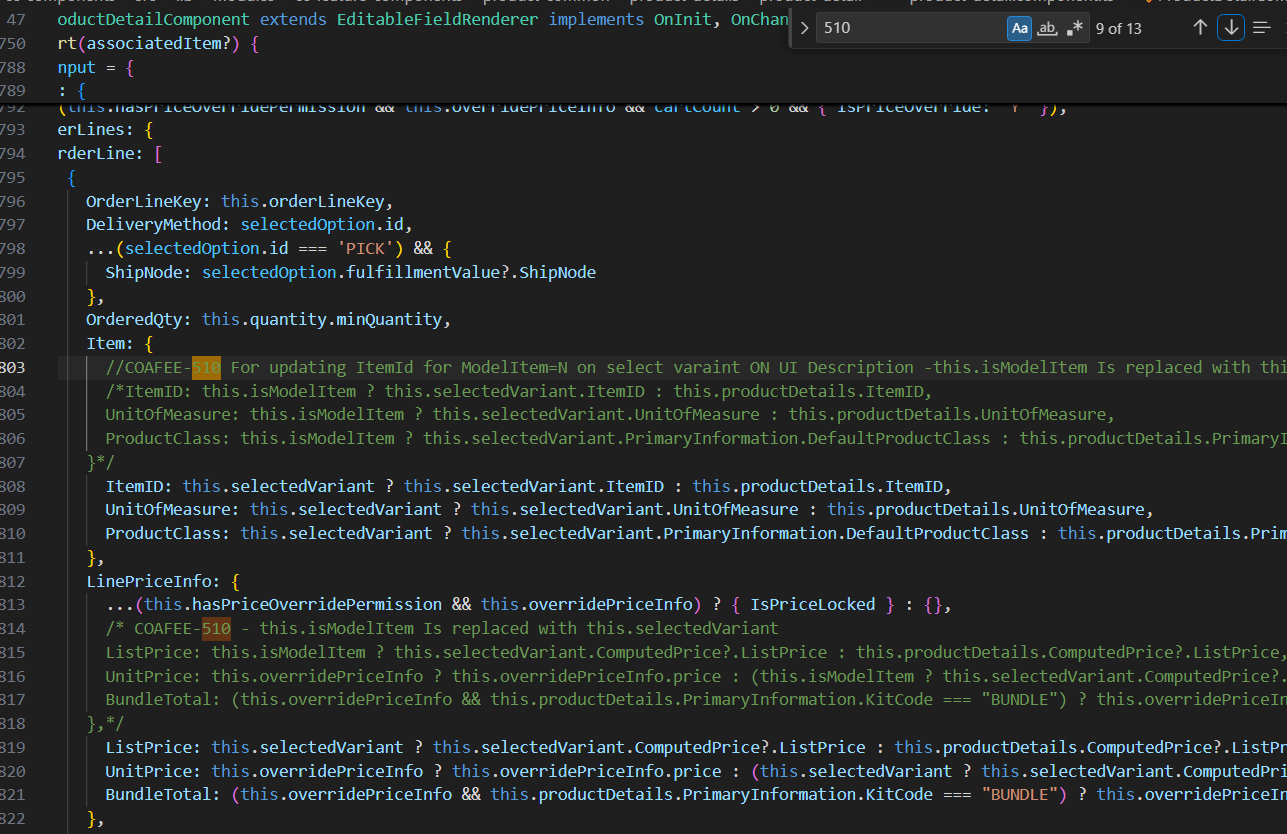
Please refer here for the changes done in code

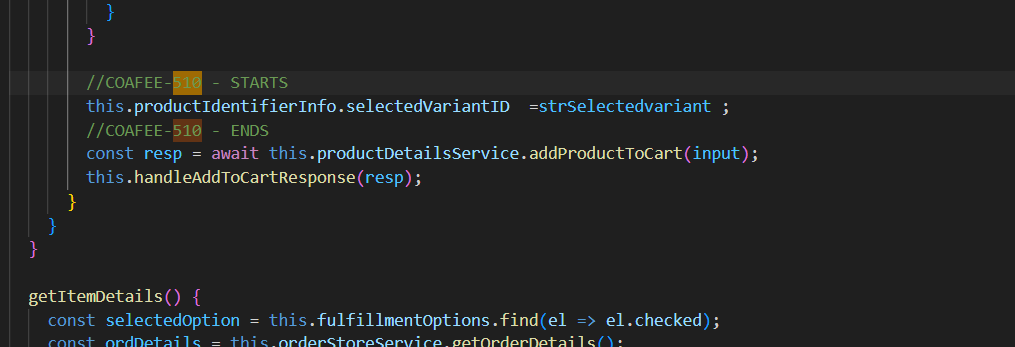


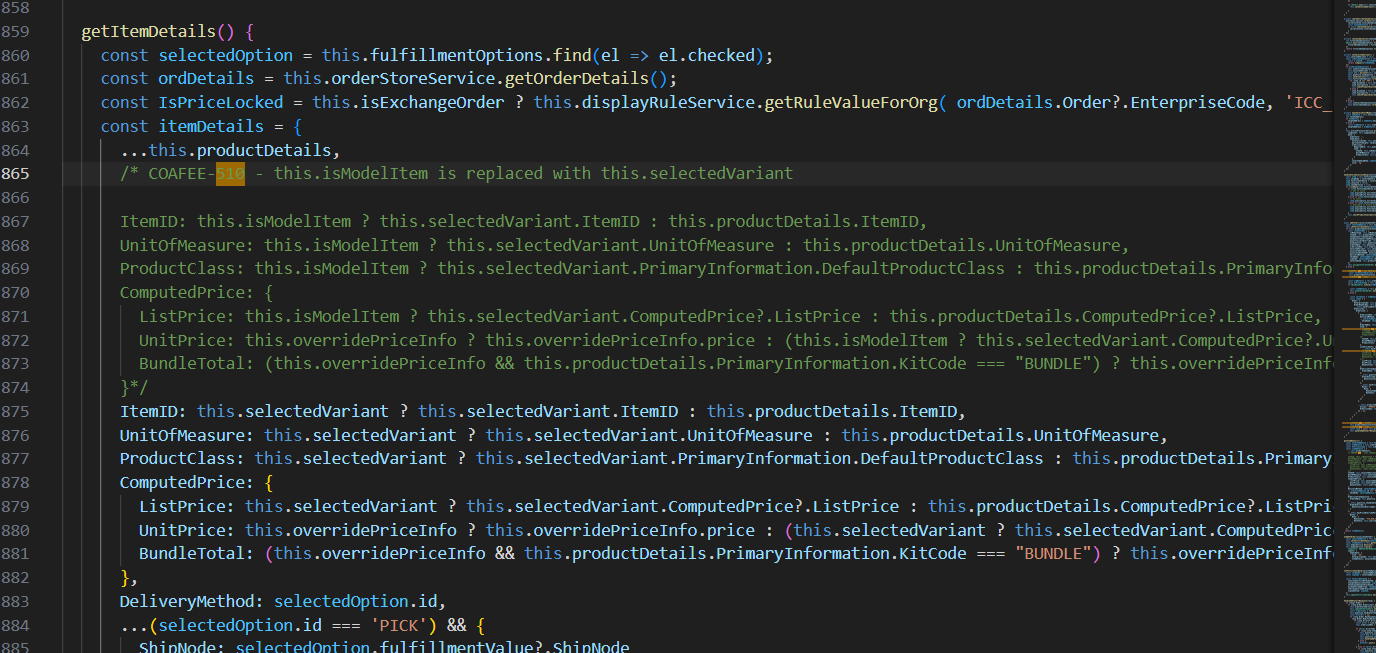












C:\cc-UpgradeCallCenter\call-center-custom\libs\cc-components\src\lib\modules\cc-feature-components\product-common\product-details\product-detail\product-detail.component.html

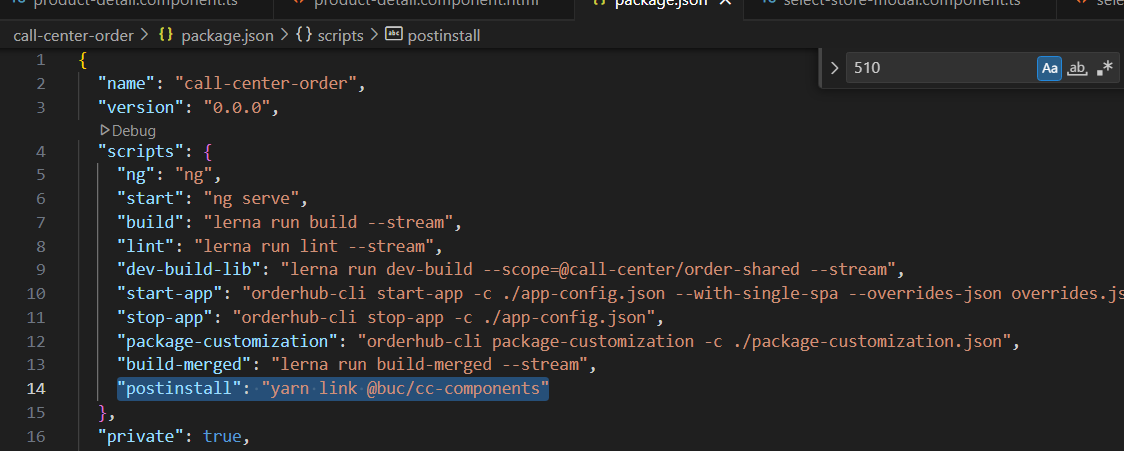
|  |
| --- |
| <!--  \* IBM Confidential  \* OCO Source Materials  \* 5737-D18, 5725-D10  \*  \* (C) Copyright International Business Machines Corp. 2019, 2025  \*  \* The source code for this program is not published or otherwise divested  \* of its trade secrets, irrespective of what has been deposited with the  \* U.S. Copyright Office.  -->  <div    \*ngIf="productIdentifierInfo.isParentItemSearchResult"    class="return-link d--flex"    (click)="returnToItemSearchResult.emit()">    <svg ibmIcon="chevron--left" size="16" class="cds--btn\_\_icon margin-right--1rem"></svg>    <buc-link [attr.tid]="componentId + '-return-to-search-results'">      <span>{{ 'PRODUCT\_DETAIL.GENERAL.LABEL\_RETURN\_TO\_SEARCH\_RESULT' | translate }}</span>    </buc-link>  </div>  <div class="screen" \*ngIf="!isScreenInitialized">    <buc-loading [isActive]="!isScreenInitialized"></buc-loading>  </div>  <div \*ngIf="isScreenInitialized && productDetails" [attr.tid]="componentId + 'details'" class="item-summary">    <ng-template bucExtension [componentName]="componentId" [placement]="'top'" [id]="EXTENSION.TOP" [parent]="this">    </ng-template>    <div>      <div class="cds--row top-section">        <div class="cds--col-lg-8 cds--col-md-12 cds--col-sm-12">          <div>            <div id="carousel-container" \*ngIf="!isModelItem || (isModelItem && !isSelectedCombinationInvalid)">              <buc-product-carousel [carouselImages]="imageList"></buc-product-carousel>            </div>            <div id="default-image-container" \*ngIf="isSelectedCombinationInvalid">              <img                [src]="productDetails.PrimaryInformation?.ImageLocation + '/' + productDetails.PrimaryInformation?.ImageID"                onerror="this.src='/call-center-order/assets/call-center-order/images/default-image.svg'"                title="invalid-variant" />            </div>          </div>        </div>        <div class="cds--col-lg-1 padding-right--0 padding-left--0"></div>        <div class="cds--col-lg-7 cds--col-md-9 cds--col-sm-12 item-details" [attr.tid]="componentId + '-item-information'">          <buc-summary-panel [isSummaryComponent]="false" [showConfigIcon]="false" [attributes]="attributes"            [context]="this" [options]="{ turnOffLargeBreakpoint: true }" [attr.tid]="componentId + '-'">          </buc-summary-panel>           <!--COAFEE-510 (275) To Show selected vairant itemId on Selecting variants on UI -- START-->         <div class="margin-top--0\_25rem d--flex-jc-flex-start d--flex-ai-center" \*ngIf="selectedVariant">            <h6 class=""> Selected Variant Item Id - {{this.selectedVariant.ItemID}}  </h6>          </div>          <!--COAFEE - 510 ENDS-->         <!--COAFEE-510 - Removed isModeItem from ngIf, so product description gets updated for ModelItem = N-->          <!--<div class="margin-top--0\_25rem d--flex-jc-flex-start d--flex-ai-center" \*ngIf="isModelItem && selectedVariant">-->          <div class="margin-top--0\_25rem d--flex-jc-flex-start d--flex-ai-center" \*ngIf="selectedVariant">            <h6 class="">{{ 'PRODUCT\_DETAIL.GENERAL.LABEL\_SELECTED' | translate }}</h6>            <h6 class="">: {{selectedVariant.PrimaryInformation?.ShortDescription}}</h6>          </div>          <div \*ngIf="isSelectedCombinationInvalid">            <buc-notification (click)="isSelectedCombinationInvalid=false"              [attr.tid]="componentId + '-invalid-variant-notification'" [notificationObj]="{                          type: 'error',                          title: 'PRODUCT\_DETAIL.GENERAL.ERR\_INVALID\_COMBINATION' | translate,                          message: 'PRODUCT\_DETAIL.GENERAL.MSG\_CHOOSE\_DIFF\_OPTION' | translate,                          showClose: false,                          lowContrast: true }"></buc-notification>          </div>          <div class="d--flex-jc-flex-start d--flex-ai-flex-end margin-bottom--1rem margin-top--1rem quantityInput">            <ibm-number class="cds--col-lg-6 cds--col-md-4  padding-left--0"              [label]="'PRODUCT\_DETAIL.GENERAL.LABEL\_QUANTITY' | translate" theme="light" [min]="1" [step]="1" size="sm"              [(ngModel)]="quantity.orderedQty" [attr.tid]="componentId + '-quantity-dropdown'"              [invalid]="quantity.isQuantityInvalid" [invalidText]="quantity.quantityInvalidText"              (ngModelChange)="quantity.isQuantityInvalid = false">            </ibm-number>            <div class="cds--label" [attr.tid]="componentId + '-unit-of-measure'"              \*ngIf="productDetails.DisplayUnitOfMeasure">{{productDetails.DisplayUnitOfMeasure}}</div>          </div>          <div class="margin-bottom--1rem"            \*ngIf="quantity.showQtyOverrideCheckbox && quantity.hasQuantityOverridePermission">            <buc-checkbox [checked]="quantity.quantityLimitOverride" (change)="onOverrideCheckboxChange($event.checked)"              [attr.tid]="componentId + '-override-quantity'">              {{'PRODUCT\_DETAIL.GENERAL.LABEL\_OVERRIDE\_VALIDATION' | translate}}            </buc-checkbox>          </div>            <!--<ng-container \*ngIf="isModelItem"> Commented for COAFEE-510 --Product Varition to be shown for ModelItem ='N' -->            <ng-container \*ngIf="attributeList?.length; else noVariant">              <ng-container \*ngFor="let attribute of attributeList; let i = index">                <div \*ngIf="attribute.options" class="product-variants margin-bottom--0\_5rem">                  <div>                    <span class="cds--label" [attr.tid]="componentId + 'VariantType' + i">{{ attribute.label                      }}</span>                    <span \*ngIf="attribute.selectedOption" class="cds--label bold-text"                      [attr.tid]="componentId + 'VariantTypeSelected' + i">:                      {{attribute.selectedOption}}</span>                  </div>                  <div class="cds--row margin-left--0">                    <div \*ngFor="let option of attribute.options; let j = index"                      class="display--inline-block margin-right--1rem margin-bottom--0\_5rem" [ngClass]="{                                                    'text-style': !option.ImageURL,                                                    'image-style': option.ImageURL,                                                    'unavailable': option.invalidCombination}">                      <button [attr.tid]="componentId + 'VariantBtn' + i + j" size="sm" iconOnly="true"                        (click)="onAttributeChange(attribute, option)"                        [class]="attribute.selectedOption === option.DisplayAttributeValue? 'btn btn-primary': 'btn button-secondary'">                        <ng-container \*ngIf="option.ImageURL">                          <img [src]="option.ImageURL">                        </ng-container>                          <ng-container \*ngIf="!option.ImageURL">                            <span \*ngIf="option.DisplayAttributeValue"                              [attr.tid]="componentId + 'VariantValue' + i + j">{{option.DisplayAttributeValue}}</span>                            <span \*ngIf="!option.DisplayAttributeValue" [attr.tid]="componentId + 'VariantValueNone' + i + j">{{                              'PRODUCT\_DETAIL.GENERAL.LABEL\_NONE' | translate }}</span>                          </ng-container>                        </button>                      </div>                    </div>                  </div>              </ng-container>            </ng-container>            <ng-template #noVariant>              <div class="margin-bottom--1rem">                <buc-notification [notificationObj]="{                          type: 'info',                          message: 'PRODUCT\_DETAIL.GENERAL.LABEL\_NO\_MODEL\_VARIATION' | translate,                          showClose: false,                          lowContrast: true }"></buc-notification>              </div>            </ng-template>          <!--</ng-container> commented for COAFEE-510-->          <ng-container \*ngIf="!isModelItem">            <div class="d--flex-jc-flex-start"              \*ngIf="(productDetails.ComputedPrice?.BundleTotal || productDetails.ComputedPrice?.UnitPrice)">              <h4 class="margin-bottom--1rem margin-right--1rem" [attr.tid]="componentId + '-unit-price-notModel'">                {{ (overridePriceInfo?.price || productDetails.ComputedPrice.BundleTotal ||                productDetails.ComputedPrice.UnitPrice)                | bucCurrencyFmt : currency: 'symbol'}}</h4>              <div [attr.tid]="componentId + '-list-price'"                \*ngIf="productDetails.ComputedPrice.ListPrice && ((productDetails.ComputedPrice.BundleTotal || productDetails.ComputedPrice.UnitPrice) !== productDetails.ComputedPrice.ListPrice)"                class="cds--label list-price margin-right--1rem">                <s>{{productDetails.ComputedPrice.ListPrice | bucCurrencyFmt : currency: 'symbol'}}</s>              </div>              <buc-link class="cds--label list-price"                \*ngIf="!isReturnOrder && hasPriceOverridePermission && (productDetails.PrimaryInformation?.BundlePricingStrategy === 'ALL' || productDetails.PrimaryInformation?.BundlePricingStrategy === 'PARENT')"                (click)="openOverridePriceModal()"                [attr.tid]="componentId + '-override-price'">{{'PRODUCT\_DETAIL.GENERAL.LABEL\_OVERRIDE\_PRICE' | translate                }}</buc-link>            </div>          </ng-container>          <ng-container \*ngIf="isModelItem && selectedVariant">            <div class="d--flex-jc-flex-start"              \*ngIf="(selectedVariant?.ComputedPrice?.BundleTotal || selectedVariant.ComputedPrice?.UnitPrice || overridePriceInfo?.price || productDetails.ComputedPrice.BundleTotal || productDetails.ComputedPrice.UnitPrice)">              <h4 class="margin-bottom--1rem margin-right--1rem" [attr.tid]="componentId + '-variant-unit-price'">                {{(overridePriceInfo?.price || selectedVariant.ComputedPrice?.BundleTotal ||                selectedVariant.ComputedPrice?.UnitPrice) |                bucCurrencyFmt : currency: 'symbol'}}</h4>              <div [attr.tid]="componentId + '-variant-list-price'"                \*ngIf="(selectedVariant.ComputedPrice?.BundleTotal || selectedVariant.ComputedPrice?.UnitPrice) !== selectedVariant.ComputedPrice.ListPrice"                class="cds--label list-price margin-right--1rem">                <s>{{selectedVariant.ComputedPrice?.ListPrice | bucCurrencyFmt : currency: 'symbol'}}</s>              </div>              <buc-link class="cds--label list-price"                \*ngIf="!isReturnOrder && hasPriceOverridePermission && (selectedVariant.PrimaryInformation?.BundlePricingStrategy === 'ALL' || selectedVariant.PrimaryInformation?.BundlePricingStrategy === 'PARENT')"                (click)="openOverridePriceModal()">{{'PRODUCT\_DETAIL.GENERAL.LABEL\_OVERRIDE\_PRICE' | translate                }}</buc-link>            </div>          </ng-container>          <ng-container \*ngIf="!isReturnOrder">            <hr>            <div class="margin-top--1rem" [ngClass]="{'hide-radio': fulfillmentOptions?.length === 1}">              <buc-radio-group orientation="vertical" [attr.tid]="componentId + '-fulfillment-radio'"                [radioList]="fulfillmentOptions" (bucRadioChange)="onFulfillmentRadioChange($event)">              </buc-radio-group>              <hr>            </div>          </ng-container>          <ng-container \*ngIf="!handleAddToCartExternally">            <div class="margin-top--1rem d--flex-jc-flex-start d--flex-ai-center"              \*ngIf="productIdentifierInfo.orderHeaderKey || productIdentifierInfo.productBrowsingStandalone || isExchangeOrder">              <buc-button [resourceId]="resourceIdAddToCart" btnSize="sm" [type]="isReturnOrder ? 'secondary' : 'primary'"                [attr.tid]="componentId + '-add-to-cart'" class="add-cart-btn" [isDisabled]="isAddToCartDisabled"                (click)="addToCart()">                {{ (productIdentifierInfo.orderLineKey || orderLineKey) ? ('PRODUCT\_DETAIL.BUTTONS.UPDATE\_CART' |                translate) : (isExchangeOrder) ? ('PRODUCT\_SEARCH.ITEM\_SEARCH.LABEL\_ADD\_TO\_EXCHANGE' |                translate) : ('PRODUCT\_DETAIL.BUTTONS.ADD\_TO\_CART' | translate) }}              </buc-button>              <!-- Product qty in cart starts -->              <div class="margin-left--1rem" \*ngIf="productQtyInCart">                <span>                  {{'PRODUCT\_DETAIL.GENERAL.LABEL\_PRODUCT\_IN\_CART' | translate: {num: productQtyInCart} }}                </span>              </div>              <!-- Product qty in cart ends -->            </div>            <!-- Add to order button starts-->            <div class="add-to-order-btn-container margin-top--1rem" \*ngIf="showGoToOrder">              <buc-button [resourceId]="resourceIdAddToOrder" type="secondary" class="add-to-order-btn" btnSize="sm" [attr.tid]="componentId+ '-add-to-order'"                (click)="onAddToOrder()" [isDisabled]="isAddToCartDisabled">                {{ 'ADD\_TO\_ORDER.BUTTON\_GO\_TO\_ORDER' | translate }}              </buc-button>            </div>            <!-- Add to order button ends -->          </ng-container>        </div>      </div>      <div class="margin-top--1rem" #productTabs>        <!--COAFEE-510 Product details - description to get updated for ModelItem = N, isModelItem condition removed        [productDetails]="isModelItem && selectedVariant ? selectedVariant : productDetails"-->        <buc-product-tabs          [productDetails]="selectedVariant ? selectedVariant : productDetails"          [selectedTab]="selectedTab"          [isExchangeOrder]="isExchangeOrder"          [isReturnOrder]="isReturnOrder"          [showGoToOrder]="showGoToOrder"          [orderHeaderKey]="productIdentifierInfo.orderHeaderKey"          [productBrowsingStandalone]="productIdentifierInfo.productBrowsingStandalone"          [enterpriseCode]="enterpriseCode"          [selectedFulfillmentDetails]="selectedFulfillmentDetails"          (productTabsLoaded)="afterProductTabsLoaded()"          (callAddToCart)="addToCart($event)"          (updateOrderSummary)="updateOrderSummary.emit($event)"          (addToOrder)="onAddToOrder($event)"          (sendCartDetails)="sendCartDetails.emit($event)" (sendCartErrorDetails)="sendCartErrorDetails.emit($event)">        </buc-product-tabs>      </div>    </div>    <ng-template bucExtension [componentName]="componentId" [placement]="'bottom'" [id]="EXTENSION.BOTTOM"      [parent]="this">    </ng-template>  </div>  <ng-template #productDescription let-data="data">    <div class="flex-align-center margin-bottom--0\_25rem">      <h4 class="margin-right--1rem">{{data.PrimaryInformation?.ShortDescription}}      </h4>      <div \*ngIf="(data.PrimaryInformation.KitCode === 'BUNDLE')" class="bundle-tag-container">        <ibm-tag type="purple" class="cds--tag cds--tag--purple cds--tag--sm">          <span class="padding-right--0\_5rem">{{ 'PRODUCT\_DETAIL.GENERAL.LABEL\_BUNDLE' | translate }}</span>          <!-- Bundle icon -->          <svg width="16" height="16" viewBox="0 0 16 16" fill="none" xmlns="http://www.w3.org/2000/svg">            <rect width="16" height="16" fill="#e8daff" style="mix-blend-mode:multiply" />            <path fill-rule="evenodd" clip-rule="evenodd"              d="M14.75 3.9957L11.25 2.0457C11.1 1.9457 10.9 1.9457 10.75 2.0457L7.25 3.9957C7.1 4.0957 7 4.2957 7 4.4457V6.3607L5.25 5.3857C5.1 5.2857 4.9 5.2857 4.75 5.3857L1.25 7.3357C1.1 7.4357 1 7.6357 1 7.7857V11.6357C1 11.8357 1.1 11.9857 1.25 12.0857L4.75 13.9857C4.8 14.0357 4.9 14.0357 5 14.0357C5.05 14.0357 5.0875 14.0232 5.125 14.0107C5.1625 13.9982 5.2 13.9857 5.25 13.9857L8.75 12.0357C8.9 11.9357 9 11.7857 9 11.5857V9.6957L10.75 10.6457C10.8 10.6957 10.9 10.6957 11 10.6957C11.05 10.6957 11.0875 10.6832 11.125 10.6707C11.1625 10.6582 11.2 10.6457 11.25 10.6457L14.75 8.6957C14.9 8.5957 15 8.4457 15 8.2457V4.4457C15 4.2957 14.9 4.0957 14.75 3.9957ZM9 8.6157L10.5 9.3957V6.6957L8 5.2957V6.91784L8.75 7.3357C8.9 7.4357 9 7.6357 9 7.7857V8.6157ZM11 5.7457L8.5 4.3457L11 2.9457L13.55 4.3457L11 5.7457ZM14 8.0957L11.5 9.3957V6.6457L14 5.2457V8.0957ZM2 11.4357L4.5 12.7357V10.0357L2 8.6357V11.4357ZM2.5 7.6857L5 9.0857L7.55 7.6857L5 6.2857L2.5 7.6857ZM5.5 12.7357L8 11.4357V8.5857L5.5 9.9857V12.7357Z"              fill="#8A3FFC" />          </svg>        </ibm-tag>      </div>    </div>  </ng-template>  <ng-template #productItemID let-data="data">    <div class="item-id-container fulfillment-item-description-start">      <span class="item-id item-description-container">        <span class="cds--label item-id">{{data.value?.itemID}}</span>        <ibm-code-snippet>{{data.value?.itemID}}</ibm-code-snippet>      </span>    </div>  </ng-template>  <!-- Radio label template -->  <ng-template #radioLabelTpl let-data="data">    <div class="radio-template fulfillment-radios" \*ngIf="!data.disabled; else showOnlyRadioKeys;">      <div class=" d--flex-jc-flex-start" [attr.tid]="componentId + '-radio-option'">        <div [attr.tid]="componentId + '-radio-value-' + data.id">{{ data.key | translate }}</div>        <!-- Show country dropdown if standalone product browsing is launched -->        <buc-combobox          \*ngIf="productIdentifierInfo?.productBrowsingStandalone && data.id === 'SHP' && productIdentifierInfo?.countries"          type="single" [attr.tid]="componentId + '-country-dropdown'" [label]="''"          [items]="productIdentifierInfo.countries" class="country-dropdown" [invalid]="isCountryInvalid"          [invalidText]="'PRODUCT\_SEARCH.GENERAL.MSG\_INVALID\_SEARCH\_TEXT' | translate" placeholder=""          (selectItemAction)="onCountrySelectionChange($event.item)" (searchHandler)="onCountryValSearch($event)">        </buc-combobox>        <a [attr.tid]="componentId + '-radio-link-' + data.id" \*ngIf="data.showLink" class="margin-left--1rem"          (click)="dispactFullfillmentAction(data)">{{data.link          | translate}}</a>        <ng-container \*ngIf="data.id === 'SHP' && !data.showEditIcon && !data.showLink">          <div class="d--flex zip-input">            <div class="margin-left--1rem cds--label align-self--flex-end"              [attr.tid]="componentId + '-postal-label-' + data.id">              {{'PRODUCT\_DETAIL.RADIO\_OPTIONS.LABEL\_POSTAL\_CODE' | translate}}            </div>            <buc-label class="margin-left--1rem margin-right--1rem" [label]="" placeholder=""              [(inputValue)]="data.fulfillmentValue" [attr.tid]="componentId + '-postal-input-' + data.id ">            </buc-label>            <svg ibmIcon="save" size="16" class="cds--btn\_\_icon align-self--flex-end"              [class.isSaveIconDisabled]="isCountryInvalid" [attr.tid]="componentId + '-postal-save-' + data.id"              (click)="fulfillmentOptionSaveOrEditClick(data)">            </svg>          </div>        </ng-container>        <ng-container \*ngIf="data.id === 'SHP' && data.showEditIcon && !data.showLink">          <span class="d--flex">            <a class="margin-left--1rem margin-right--1rem align-self--flex-end"              [attr.tid]="componentId + '-postal-value-' + data.id">{{ data.fulfillmentValue }}</a>            <svg ibmIcon="edit" size="16" role="button" class="align-self--flex-end"              (click)="fulfillmentOptionSaveOrEditClick(data)" [attr.tid]="componentId + '-postal-edit-' + data.id">            </svg>          </span>        </ng-container>        <ng-container \*ngIf="data.id === 'PICK' && data.showEditIcon && !data.showLink">          <span class="d--flex" (click)="fulfillmentOptionSaveOrEditClick(data)">            <a class="margin-left--1rem margin-right--1rem align-self--flex-end"              [attr.tid]="componentId + '-node-value-' + data.id">{{ data.fulfillmentValue?.Description }}</a>            <svg ibmIcon="edit" size="16" role="button" class="align-self--flex-end"              [attr.tid]="componentId + '-node-edit-' + data.id">            </svg>          </span>        </ng-container>      </div>      <div class="stock-info" \*ngIf="(data.showAvailablity) || (data.showEditIcon && !data.showLink)">        <ng-container \*ngIf="!isBundleItemShipIndependent; else bundleItemAvailablity">          <p            \*ngIf="(data.availablity && getNumericValue(data.availablity.CurrentAvailableQty) && getNumericValue(data.availablity.CurrentAvailableQty) >= quantity.minQuantity) && getNumericValue(data.availablity.CurrentAvailableQty) >= quantity.orderedQty; else futureAvailable;"            class="in-stock cds--label"            [attr.tid]="componentId + '-in-stock-' + data.id">            {{ productDetails.DisplayUnitOfMeasure ? ('PRODUCT\_DETAIL.GENERAL.UOM\_WITH\_QUANTITY' | translate : { quantity:            getNumericValue(data.availablity.CurrentAvailableQty),            uom: toLowerCase(productDetails.DisplayUnitOfMeasure) }) : ('PRODUCT\_DETAIL.GENERAL.AVAILABLITY\_IN\_STOCK' |            translate : { quantity: getNumericValue(data.availablity.CurrentAvailableQty) }) }}          </p>          <ng-template #futureAvailable>            <p \*ngIf="data.availablity && (getNumericValue(data.availablity.FutureAvailableQuantity) > 0); else outOfStock;"              class="available-on cds--label" [attr.tid]="componentId + '-future-date-' + data.id">{{              'PRODUCT\_DETAIL.GENERAL.AVAILABLITY\_ON' | translate : {date:              data.availablity.FutureAvailableDate | bucCommonDateFmt:'MM/D/YYYY'} }}</p>          </ng-template>        </ng-container>        <ng-template #outOfStock>          <p class="out-of-stock cds--label" [attr.tid]="componentId + '-out-stock-' + data.id">{{            'PRODUCT\_DETAIL.GENERAL.AVAILABLITY\_OUT\_OF\_STOCK' | translate}}</p>        </ng-template>        <ng-template #bundleItemAvailablity>          <p \*ngIf="data.bundleCompAvailablity; else outOfStock;" class="in-stock cds--label"            [attr.tid]="componentId + 'in-stock' + data.id">{{ 'PRODUCT\_DETAIL.GENERAL.AVAILABLITY\_IN\_STOCK' |            translate : {quantity: getNumericValue(data.availablity[0].CurrentAvailableQty) } }}          </p>        </ng-template>      </div>    </div>    <ng-template #showOnlyRadioKeys>      <div [attr.tid]="componentId + '-radio-value-' + data.id">{{ data.key | translate }}</div>    </ng-template>  </ng-template>  <ng-template #noPrice>    <h4  \*ngIf="!(isModelItem && selectedVariant === null)"  class="margin-bottom--1rem margin-right--1rem"        [attr.tid]="componentId + '-unit-price-noPrice'">        {{ 0 | bucCurrencyFmt : currency: 'symbol'}}</h4>      </ng-template> |

Product-detail.component.ts and Product.detail.component.html files are coming from common components –

Steps to customize that

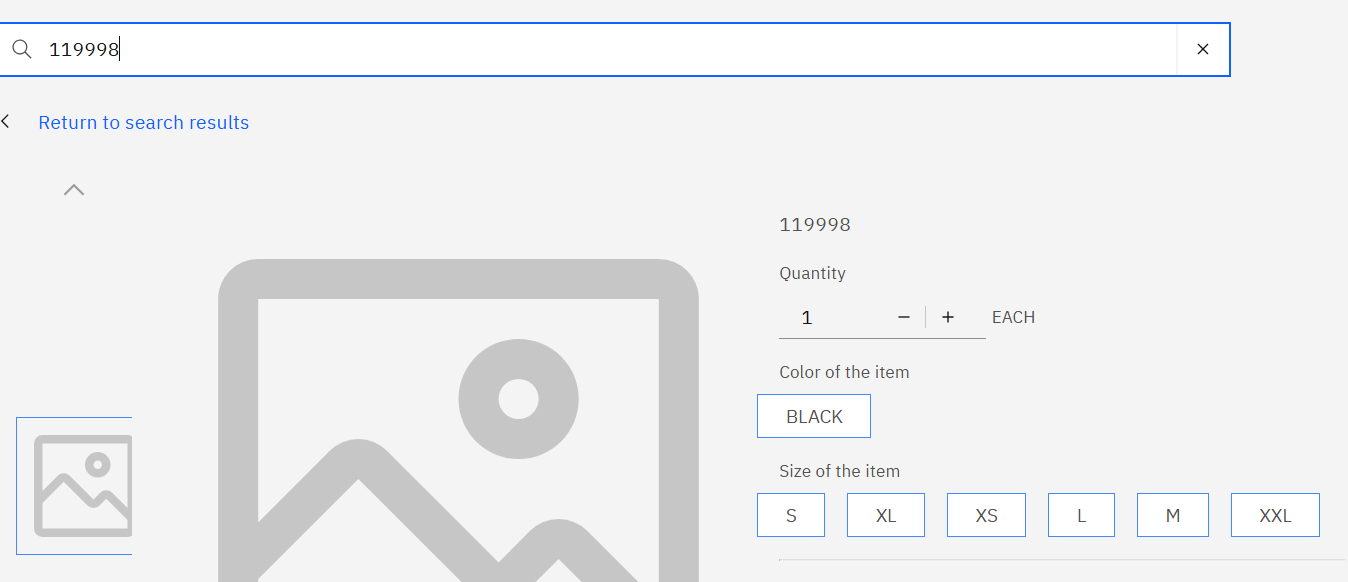
[Setting up the development environment - IBM Documentation](https://www.ibm.com/docs/en/call-center/10.0.0?topic=components-setting-up-development-environment)

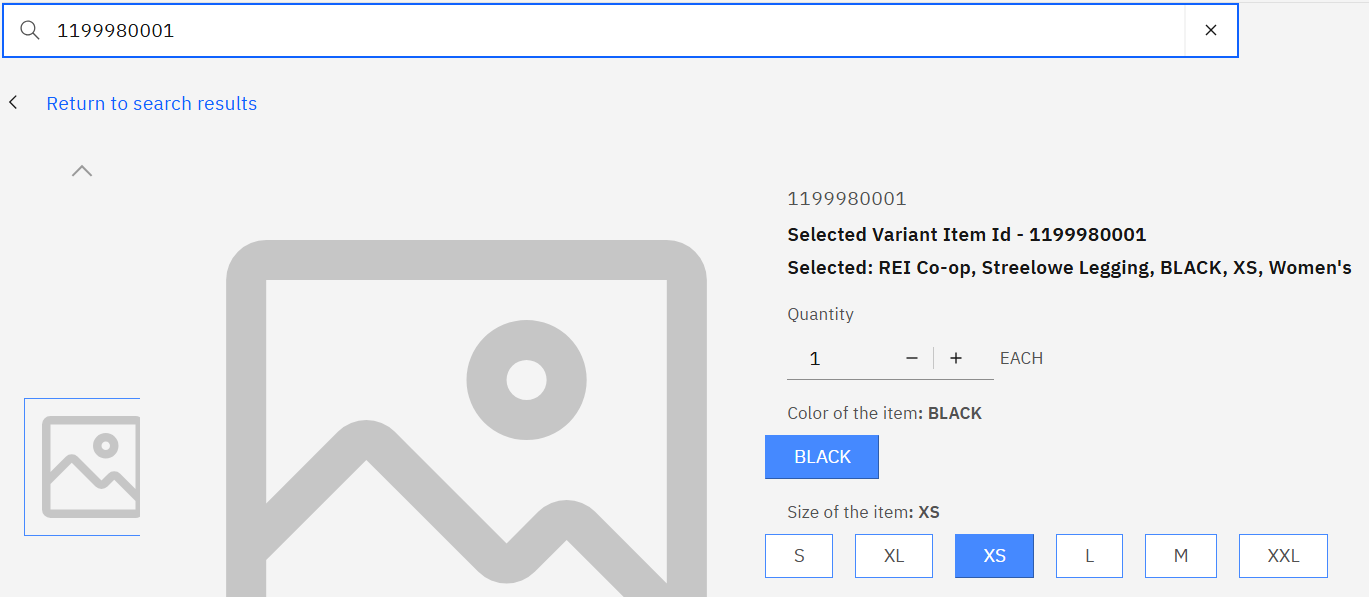
1. In the call-center-custom directory, run the following command to install the dependencies for the call-center-custom directory.
2. yarn install
3. In the module which you are customizing , in its package.json file add "postinstall": "yarn link @buc/cc-components" under scripts.



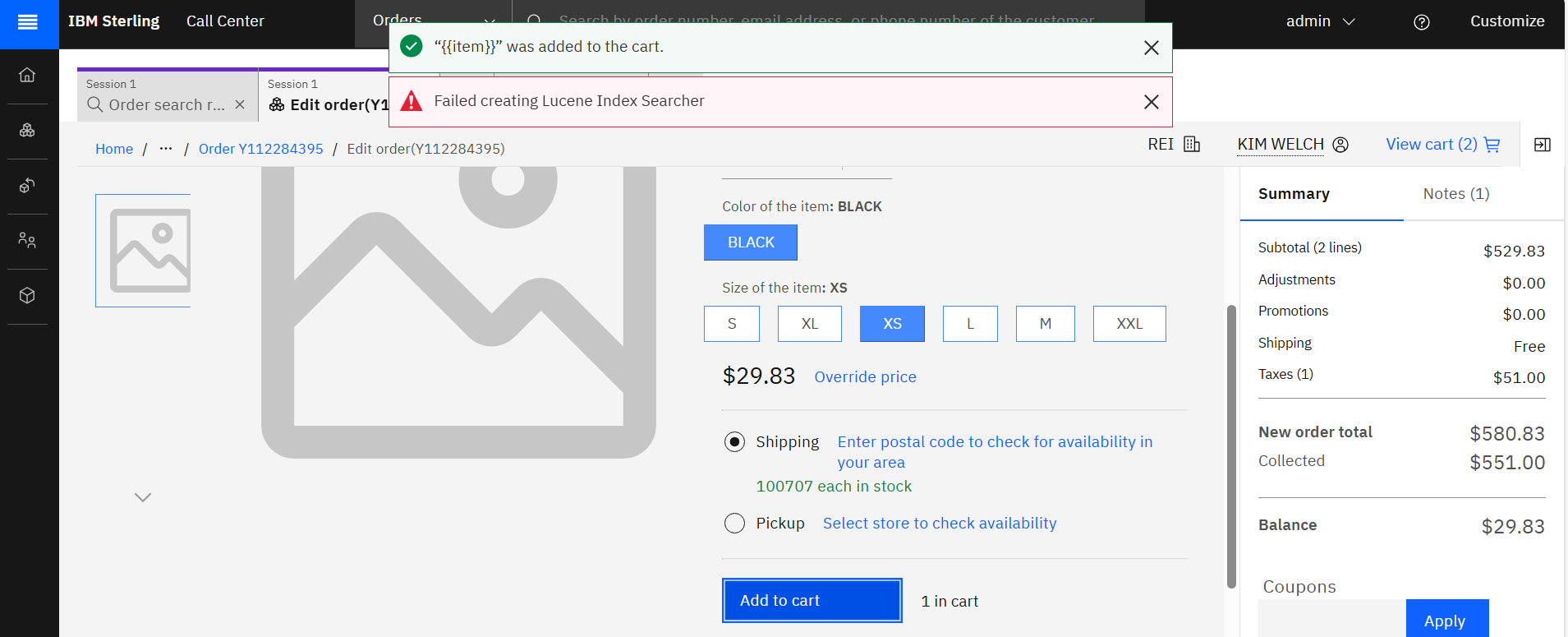
1. In call-center-custom folder run the command - yarn cc-components:dev
2. In the call-center-custom directory, change the directory to dist/libs/cc-components and run the following command
3. Again run - In call-center-custom folder run the command - yarn cc-components:dev
4. Now in call-center-order – run - yarn start-app, it will reflect your changes

For ModelItem = Y issue resolved, product variants coming up for UOM not null



For ModelItem=N, Products variants coming up and for the itemId which is searched variant is auto selected  


Select different variants, it will change the item id on UI and update the description for the selected variant. Try adding to cart as well



For Build process of cc-components

[Building customization for common components - IBM Documentation](https://www.ibm.com/docs/en/call-center/10.0.0?topic=components-building-customization-common)

* In call-center-order/package-customization.json

{

"repositoryName": "call-center-order",

"customizationContextRoot": "/call-center-customization",

"routes": {

"order-search": { "type": "code" }

}

}

* In call-center-custom  - yarn cc-components:prod
* In call-center-custom/dist/libs/cc-components - **yarn link**
* In container-build -

./build-customization.sh build-ui call-center-order

./build-customization.sh package-jar call-center-order

**Replace extension to var/www/html path like normal build flow restart nginx and see the changes , do remove devMode= true from app-bootstrap-config.json**