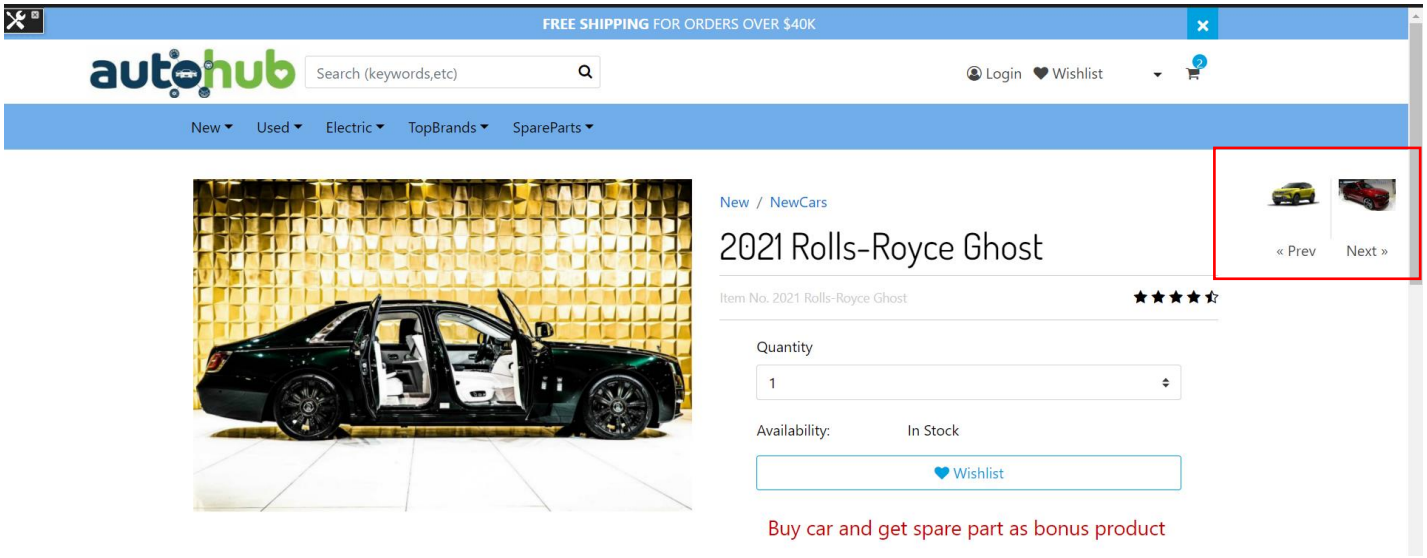
# Customization's Documentation

## 1.Previous and Next Products

In this customization I developed the custom feature which is used to navigate from one PDP to another PDP by using the Previous and Next product URLs as shown in the below picture.




## Product-ProductNavigation:-

1. **Retrieving Product Page Information**:
   - The code starts by calling the productHelper.getPageDesignerProductPage() function to retrieve information about the configured product page, invisible page, and aspect attributes based on the provided product.
   - It checks if the retrieved page has visibility rules or if there is an invisible page configured. If either condition is met, it sets the cache period to 0 to prevent caching.
2. **Fetching Product Details**:
   - It then requires the 'productModel.js' script to get details of a specific product using the Product.get() method with the product ID (params.pid.stringValue).
   - It checks if the product is visible using the product.isVisible() method.
3. **Setting Up Product Search**:
   - It initializes variables for paging and search using the Search.initializeProductSearchModel(params) method.

- It sets the product ID to null in the search model if no category ID is provided (!params.cgid.value).
- It determines the category for the product based on whether it has a primary category or a master variation category.

4. **Handling Category and Variation Models**:
   - It checks for the primary category of the product and sets the category variable accordingly.
   - If no primary category is found, it checks for the master variation category.
5. **Additional Information**:
   - This involves handling product pages, visibility rules, product visibility, and setting up product search models based on provided parameters.
6. The code snippet checks if the product has a primary category. If it does, it assigns the primary category to the 'category' variable. If not, it checks if the product has a variation model with a master product and assigns the primary category of the master product to the 'category' variable.
7. It then verifies if the 'category' variable exists and is online. If these conditions are met, it sets the 'categoryID' in the 'productSearchModel' to the ID of the category.
8. The 'productSearchModel' is then used to perform a search.
9. The code snippet requires the 'PagingModel' module and creates a new instance of 'PagingModel' using the product search hits and count.
10. It retrieves the 'start' parameter from the 'params' object and assigns it to 'valstart'.
11. The code sets the page size of the 'productPagingModel' to 3.
12. It calculates the 'URLStart' by converting the 'start' parameter to an integer.
13. Adjusts the start index of the 'productPagingModel' by subtracting 2 from the 'start' parameter.
14. Finally, it requires the 'productView.js' script.
15. **Require the productURLs.js Script**:
    - The code starts by requiring the 'productURLs.js' script, which likely contains functions related to generating URLs for products.

16.**Initialize Variables**:

- `ProductURLs` is initialized with the required script.
- `productURLs` is then called as a function with parameters like `productSearchModel`, `productPagingModel`, `URLStart`, and `proPage123`.
- The subsequent lines extract specific properties like `prevProd`, `prevProdUrls`, `nextProd`, and `nextProdUrls` from the `productURLs` object.

17.**Understanding the Variables**:

- `prevProd` and `nextProd` likely refer to the previous and next products in a sequence.
- `prevProdUrls` and `nextProdUrls` probably store the URLs corresponding to the previous and next products.

```
33  server.get('ProductNavigation',cache.applyPromotionSensitiveCache, consentTracking.consent,function (req, res, next) {
34      var productHelper = require('*/cartridge/scripts/helpers/productHelpers');
35      var showProductPageHelperResult = productHelper.showProductPage(req.querystring, req.pageMetaData);
36      var productType = showProductPageHelperResult.product.productType;
37      if (!showProductPageHelperResult.product.online && productType !== 'set' && productType !== 'bundle') {
38          res.setStatusCode(404);
39          res.render('error/notFound');
40      } else {
41          var pageLookupResult = productHelper.getPageDesignerProductPage(showProductPageHelperResult.product);
42
43          if ((pageLookupResult.page && pageLookupResult.page.hasVisibilityRules()) || pageLookupResult.invisiblePage) {
44              // the result may be different for another user, do not cache on this level
45              // the page itself is a remote include and can still be cached
46              res.cachePeriod = 0; // eslint-disable-line no-param-reassign
47          }
48
49          var Product = require('*/cartridge/scripts/productModel.js');
50          var product = Product.get(params.pid.stringValue);
51
52          if (product.isVisible()) {
53              var PagingModel;
54              var productPagingModel;
55
56              // Construct the search based on the HTTP params and set the categoryID.
57              var Search = require('*/cartridge/scripts/searchModel.js');
58              var productSearchModel = Search.initializeProductSearchModel(params);
59
60              // Reset pid in search.
61              productSearchModel.setProductID(null);
```

```
        // Special handling if no category ID is given in URL.
        if (!params.cgid.value) {
            var category = null;

            if (product.getPrimaryCategory()) {
                category = product.getPrimaryCategory();
            } else if (product.getVariationModel().getMaster()) {
                category = product.getVariationModel().getMaster().getPrimaryCategory();
            }

            if (category && category.isOnline()) {
                productSearchModel.setCategoryID(category.getID());
            }
        }

        // Execute the product searchs
        productSearchModel.search();

        // construct the paging model
        PagingModel = require('dw/web/PagingModel');
        productPagingModel = new PagingModel(productSearchModel.productSearchHits, productSearchModel.count);
        productPagingModel.setPageSize(3);
        productPagingModel.setStart(params.start.intValue - 2);
        var View = require('*/cartridge/scripts/productView.js');
        var ProductURLs = require('*/cartridge/scripts/productURLs.js');
        var productURLs = ProductURLs(productSearchModel,productPagingModel);
        var prevProd = productURLs.prevProd;
        var prevProdUrls = productURLs.prevProdUrls;
            var nextProd = productURLs.nextProd;
            var nextProdUrls = productURLs.nextProdUrls;
            // obj1 = new View(parameters || {});
            var obj={};
            obj.ProductPagingModel  = productPagingModel;
            obj.ProductSearchResult = productSearchModel;
            obj.prevProd = prevProd;
            obj.prevProdUrls = prevProdUrls;
            obj.nextProd = nextProd;
            obj.nextProdUrls = nextProdUrls;
            res.setViewData(obj);

        }
        if (pageLookupResult.page) {
            res.page(pageLookupResult.page.ID, {}, pageLookupResult.aspectAttributes);
        } else {
            res.render(showProductPageHelperResult.template, {
                product: showProductPageHelperResult.product,
                addToCartUrl: showProductPageHelperResult.addToCartUrl,
                resources: showProductPageHelperResult.resources,
                breadcrumbs: showProductPageHelperResult.breadcrumbs,
                canonicalUrl: showProductPageHelperResult.canonicalUrl,
                schemaData: showProductPageHelperResult.schemaData
            });
        }
    }
    next();
});
```

## productURLs.js file from script folder :-

- By using this script file we are generating the prevoius and next product urls.
- We have a variation product selection url first and grab its query string
- This is the right way but since ProductSearchResult.url takes either an action name or URL and urlSelectVariationValue return url in a String thus have to append variation attr value manually.
- Append the variation url query string with the search url (search url could have refinements)

```
'use strict';
function productURLs(ProductSearchResult,ProductPagingModel,URLStart,proPage123){
        // local function to create a product link along with any refinements and color variation value
        var URLUtils = require('dw/web/URLUtils');
        var removeParameter = function(queryString, parameter) {

            var prefix = encodeURIComponent(parameter)+'=';
            var parameters = queryString.split(/[&;]/g);
            for (var i= parameters.length; i-->0;) {
                if (parameters[i].lastIndexOf(prefix, 0)!==-1) {
                    parameters.splice(i, 1);
                }
            }
            return parameters.join('&');
        }
        var createLink = function(product, productHit, start, finalPrev, finalNext) {
            var varAttrColor = null;
            var selectableColor = null;
            if ( product != null && product.master ) {
                var varModel : dw.catalog.ProductVariationModel = product.variationModel;
                varAttrColor = varModel.getProductVariationAttribute("color");

                if( varAttrColor != null )
                {
                    var repVarVals = productHit.getRepresentedVariationValues( varAttrColor );
                    if (repVarVals && repVarVals.size() > 0) {
                        selectableColor = repVarVals.get(0);
                    }
```

```
31          var productUrl = "";
32          var imgUrl = "";
33          var imgFile;
34          var hashAppend = removeParameter(request.httpQueryString, 'start');
35          if(start==5){
36              start=start-1;
37          }
38          hashAppend = removeParameter(hashAppend, 'pid') + '&start=' + start;
39          if (!empty(product)) {
40              productUrl = ProductSearchResult.url(URLUtils.url('Product-Show', 'pid', product.ID));
41              if (!empty(product) && !empty(selectableColor) && !empty(varAttrColor)) {
42                  productUrl = URLUtils.url('Product-Show', 'pid', product.ID, product.variationModel.getHtmlName
                    (varAttrColor), selectableColor.value);
43                  imgFile = selectableColor.getImage('small');
44              }
45              else {
46                  imgFile = product.getImage('small');
47              }
48          }
49          else {
50              return {pUrl: "", imgUrl: URLUtils.staticURL('/images/noimagesmall.png'), imgAlt: "", imgTitle: ""};
51          }
52          if(finalNext === product.ID){
53                  var URLStart1 = URLStart+1;
54                  return {pUrl: productUrl + '&' + 'start=' + URLStart1
55                  , imgUrl: (imgFile != null) ? imgFile.getURL(): URLUtils.staticURL('/images/noimagesmall.png'), imgAlt:
                    (imgFile != null) ? imgFile.alt: product.name, imgTitle: (imgFile != null) ? imgFile.title: product.name};
```

```
57          else{
58              var URLStart2 = URLStart-1;
59              return {pUrl: productUrl + '&' + 'start=' + URLStart2
60              , imgUrl: (imgFile != null) ? imgFile.getURL(): URLUtils.staticURL('/images/noimagesmall.png'), imgAlt: (imgFile
                != null) ? imgFile.alt: product.name, imgTitle: (imgFile != null) ? imgFile.title: product.name};
61          }
62      };
63      var rList       = ProductPagingModel.pageElements.asList();
64      var listSize    = rList.size();
65      var start       = request.httpParameterMap.parameterCount;
66      var start123    = request.httpParameterMap.parameterNames[0];
67      var totalCount = ProductPagingModel.count;
68      var prevProd = null, prevProdHit = null;
69      var nextProd = null, nextProdHit = null;
70      var currentProd = null, currentProdHit = null;
71      var currentProdInd = 0;
72      var finalNext;
73      var finalPrev;
74      if (listSize > 0 && (start > 1) && (totalCount > 1)) {
75          if(URLStart > 1){
76              prevProdHit = rList.get(0);
77              prevProd = prevProdHit.product;
78              finalPrev = prevProd.ID;
79          currentProdInd = 1; // if we have prev then the current product is the prev+one
80          }
81      }
82      if (listSize > 0 && start < totalCount) {
```

```
82          if (listSize > 0 && start < totalCount) {
83              var i = 1;
84              if(URLStart != totalCount){
85                  if (start == 1 && listSize > 2) i = 2;
86                  nextProdHit = rList.get(listSize-i);
87                  nextProd = nextProdHit.product;
88                  finalNext = nextProd.ID;
89              currentProdInd = listSize-i-1; // if we have next then the current product is the next-one
90              }
91          }
92          start = parseInt(start);
93          currentProdHit = rList.get(currentProdInd);
94          currentProd = currentProdHit.product;
95          var prevProdUrls    = createLink(prevProd, prevProdHit, start-1, finalPrev, finalNext);
96          var nextProdUrls    = createLink(nextProd, nextProdHit, start+1, finalPrev, finalNext);
97          var objct = {};
98          objct.prevProd = prevProd;
99          objct.prevProdUrls = prevProdUrls;
100         objct.nextProd = nextProd;
101         objct.nextProdUrls = nextProdUrls;
102
103         return objct;
104     }
105     module.exports = productURLs;
```

## searchModel.js script file:-

1. **Initialization**:
   - The function initializes a ProductSearchModel object by requiring it and then initializing it with the httpParameterMap.
2. **Setting Recursive Category Search**:
   - It sets the recursiveCategorySearch property of the productSearchModel to true, enabling recursive category searches.
3. **Setting Product ID, Price Range, and Promotion ID**:
   - It checks if certain parameters like pid, pmin, pmax, and pmid are submitted in the httpParameterMap and sets the product ID, minimum price, maximum price, and promotion ID accordingly in the productSearchModel.
4. **Setting Sorting Rule**:
   - It checks if a sorting rule is submitted in the httpParameterMap and sets the sorting rule in the productSearchModel using CatalogMgr.getSortingRule.
5. **Category ID**:
   - If a category ID is submitted in the httpParameterMap, it retrieves the category and sets its ID in the productSearchModel.

6. **Return**:
    - Finally, the function returns the productSearchModel with all the set parameters.

```
/**
 * Creates and initializes a {dw.catalog.ProductSearchModel} based on the given HTTP parameters.
 *
 * @param {dw.web.HttpParameterMap} httpParameterMap HttpParameterMap to read product search parameters from.
 * @returns {dw.catalog.ProductSearchModel} Created and initialized product serach model.
 */
SearchModel.initializeProductSearchModel = function (httpParameterMap) {
    var ProductSearchModel = require('dw/catalog/ProductSearchModel');
    var productSearchModel = this.initializeSearchModel(new ProductSearchModel(), httpParameterMap);

    productSearchModel.setRecursiveCategorySearch(true);

    if (httpParameterMap.pid.submitted) {
        productSearchModel.setProductID(httpParameterMap.pid.value);
    }

    if (httpParameterMap.pmin.submitted) {
        productSearchModel.setPriceMin(httpParameterMap.pmin.doubleValue);
    }

    if (httpParameterMap.pmax.submitted) {
        productSearchModel.setPriceMax(httpParameterMap.pmax.doubleValue);
    }

    if (httpParameterMap.pmid.submitted) {
        productSearchModel.setPromotionID(httpParameterMap.pmid);
    }

    var sortingRule = httpParameterMap.srule.submitted ? CatalogMgr.getSortingRule(httpParameterMap.srule.value) : null;
    if (sortingRule) {
        productSearchModel.setSortingRule(sortingRule);
    }

    // only add category to search model if the category is online
    if (httpParameterMap.cgid.submitted) {
        var category = CatalogMgr.getCategory(httpParameterMap.cgid.value);
        if (category && category.isOnline() && productSearchModel) {
            productSearchModel.setCategoryID(category.getID());
        }

    }

    return productSearchModel;
};
```

## productNav.isml :-

By using the below isml code we are fetching the information through pdict and displaying the images and their corresponding urls by checking the condition previous and next products are not equals to null.

1. The <iscontent> tag:
    - The <iscontent> tag is used to set the content type and character encoding of the output stream in an ISML template.
    - In your code snippet, <iscontent type="text/html" charset="UTF-8" compact="true"/> is setting the content type to HTML with UTF-8 character encoding and compacting the output by removing unnecessary spaces.
    - This tag ensures that the output generated by the template is properly encoded and formatted for the web.
2. Conditional Logic:
    - The code contains conditional statements <isif> that check certain conditions before executing specific blocks of code.
    - For example, <isif condition="${pdict.ProductPagingModel != null && !pdict.ProductPagingModel.empty}"> checks if the ProductPagingModel is not null and not empty before proceeding.
    - Similarly, <isif condition="${!empty(pdict.ProductSearchResult) && pdict.ProductSearchResult.count > 0 }"> checks if the ProductSearchResult is not empty and has a count greater than 0.
3. Dynamic Content:
    - The code snippet also includes dynamic content generation using expressions like ${Resource.msgf(...)} to fetch and display localized messages.
    - It sets values dynamically based on conditions, like setting the backLinkText based on certain criteria.
4. HTML Output:
    - The code generates HTML output for product navigation, including links to previous and next products with corresponding images and text.

```
<iscontent type="text/html" charset="UTF-8" compact="true"/>
<div id="product-nav-container">

    <isif condition="${pdict.ProductPagingModel != null && !pdict.ProductPagingModel.empty}">
        <isif condition="${!empty(pdict.ProductSearchResult) && pdict.ProductSearchResult.count > 0 }">

            <isset name="backLinkText" value="${Resource.msgf('productnav.back','search',null, (!empty(pdict.ProductSearchResu
            categoryID) ? pdict.ProductSearchResult.category.displayName: pdict.CurrentHttpParameterMap.q.stringValue))}"
            scope="page"/>

            <isif condition="${pdict.prevProd != null}">
                <div class="product-previous <isif condition="${pdict.nextProd != null}">divided</isif>">
                    <a href="${pdict.prevProdUrls.pUrl}" title="${pdict.prevProdUrls.imgTitle}">
                        <span><img alt="${pdict.prevProdUrls.imgAlt}" title="${pdict.prevProdUrls.imgTitle}" src="${pdict.
                        prevProdUrls.imgUrl}"/></span>
                        &laquo; ${Resource.msgf('global.previous.abbr','locale',null)}
                    </a>
                </div>
            </isif>

            <isif condition="${pdict.nextProd != null}">
                <div class="product-next">
                    <a href="${pdict.nextProdUrls.pUrl}" title="${pdict.nextProdUrls.imgTitle}">
                        <span><img alt="${pdict.nextProdUrls.imgAlt}" title="${pdict.nextProdUrls.imgTitle}" src="${pdict.
                        nextProdUrls.imgUrl}" /></span>
                        ${Resource.msgf('global.next','locale',null)} &raquo;
                    </a>
                </div>
            </isif>
        </isif>
    </isif>
</div>
```

## Client Side SCSS :-

The below is the Client side scss which is used in the code while developing the customization.

```scss
//Prev - Next Products
#product-nav-container {
    overflow: hidden;
    position: absolute;
    right: 0;
    width: 160px;
    div {
        float: left;
        text-align: center;
        width: 77px;
    }
    img {
        max-height: 100%;
        max-width: 80%;
    }
    span {
        display: block;
        height: 65px;
    }
    .divided span {
        border-right: 1px solid #dcdcdc;
        padding-right: 5px;
    }
}
```

```
.pdp-main {
    font-family: sans-serif;
    margin: 0 1%;
    ul {
        list-style: none;
        margin: 0;
        padding: 0;
    }
    label {
        padding: 0;
        text-align: left;
        text-transform: uppercase;
        width: auto;
    }
    .product-col-1 {
        @media screen and (min-width: 480px) {
            float: left;
            width: 37%;
        }
    }
    .product-col-2 {
        @media screen and (min-width: 480px) {
            float: right;
            width: 59.5%;
        }
```

```
    .product-primary-image {
        max-width: 100%;
        text-align: center;
    }
    .product-info {
        clear: both;
        padding-top: 1em;
        // Height for PDP Tabs
        @media screen and (min-width: 768px) {
            height: 25em;
        }
        ul {
            font-size: .8rem;
            list-style: square;
            padding: 0 5em;
            @media screen and (min-width: 768px) {
                padding-bottom: 1em;
                padding-top: 1em;
            }
        }
    }
```

That’s all about the Previous and next Products Navigation customization.

# 2.Last Visited Products in Cart

In this customization I developed the custom feature which is used to show the last visited products in the Cart-Show Page.

## Product-IncludeLastVisited :-

- In the Product controller we are creating the new route i.e Product-IncludeLastVisited.
- In this we are requiring the script file that is recentlyViewedItemsModel.js which having the method getRecentlyViewedProducts() and this method takes input as the integer like how many products need to display in the Cart-Show page of storefront.

cart.isml 9+ | Product.js

app_custom_TrainingBajigam > cartridges > app_custom_project > cartridge > controllers > Product.js > ...

```
1  'use strict';
2
3  /**
4   * @namespace Product
5   */
6
7  var server = require('server');
8  var cache = require('*/cartridge/scripts/middleware/cache');
9  var consentTracking = require('*/cartridge/scripts/middleware/consentTracking');
10 var pageMetaData = require('*/cartridge/scripts/middleware/pageMetaData');
11 var params = request.httpParameterMap;
12
13 server.get("IncludeLastVisited",function(req,res,next){
14     var RecentlyViewedItems = require('*/cartridge/scripts/RecentlyViewedItemsModel.js');
15
16     var LastVisitedProducts = RecentlyViewedItems.getRecentlyViewedProducts(3);
17     res.render('product/lastvisited',{LastVisitedProducts:LastVisitedProducts});
18     next();
19 });
```

## Script file “recentlyViewedItemsModel.js”:-

In this script file we are having the getRecentlyViewedProducts() function, this function will returns an array of products visited in the current session.

- **Initialization**:
  - var numberOfProducts = maxLength || 5;: This line sets the variable numberOfProducts to either the value of maxLength or 5 if maxLength is not defined.
  - var clicks = session.getClickStream().getClicks();: Retrieves a list of click events from the session.
  - var products = new ArrayList();: Initializes an empty list to store products.
  - The array is retrieved from the live click stream recorded in the session.
- **Loop through Clicks**:
  - The code iterates through the click events in reverse order.
  - All clicks on a page rendered from a "Product-Show", "Link-Product", or "Link-CategoryProduct" URL are interpreted as a product visit.
  - Clicks on different variations of the same master product are merged and treated as one click.
  - For each click event:
    - Checks if the click event belongs to certain pipeline names.
    - Retrieves the SKU (product ID) from the click event.
    - Retrieves the product using the SKU.
    - Checks for duplicates in the products list.
    - If the product is not a duplicate, it is added to the products list.
- **Control the Number of Products**:
  - The loop continues until the number of unique products in the products list reaches numberOfProducts.
  - If the number of unique products reaches or exceeds numberOfProducts, the loop breaks.
  - Here the maxSize (Number) is the maximum number of visited products to include in the returned list.
- **Explanation**:

- This is part of a process that tracks user clicks, identifies products based on those clicks, and ensures a maximum number of unique products are captured.
- It avoids duplicates and stops processing once the desired number of unique products is reached.

- The function will returns the list of last visited products. If no products have been visited, the returned list is empty.

cart.isml 9+ | Product.js | RecentlyViewedItemsModel.js

app_custom_TrainingBajigam > cartridges > app_custom_project > cartridge > scripts > RecentlyViewedItemsModel.js > RecentlyViewedItemsModel

```
1  'use strict';
2
3  /**
4   * Model for manaaging information about last visited items based on the current session. Currently
    only products are
5   * supported.
6   * @module models/RecentlyViewedItemsModel */
7
8  /* API Includes */
9  var ArrayList = require('dw/util/ArrayList');
10 var ProductMgr = require('dw/catalog/ProductMgr');
11
12 /* Constants */
13 var PRODUCT_ID_PARAMETER_NAME = 'pid';
14 var PRODUCT_PIPELINE_NAMES = new ArrayList(
15     'Product-Show',
16     'Product-ShowInCategory',
17     'Link-Product',
18     'Link-CategoryProduct');
19
20 /**
21  * Model holding information about last visited items based on the current session. Currently only
    products are
22  * supported.
23  *
24  * @class module:models/RecentlyViewedItemsModel~RecentlyViewedItemsModel
25  */
26 var RecentlyViewedItemsModel = ({
27
28     getRecentlyViewedProducts: function (maxLength) {
29
30         var numberOfProducts = maxLength || 5;
31
32         // Gets the click stream.
33         var clicks = session.getClickStream().getClicks();
34
35         // build the last visted
36         var products = new ArrayList();
37
38         for (var i = clicks.size() - 1; i >= 0; i--) {
39             var click = clicks[i];
40
41             // Checks whether it was a product click.
42             if (PRODUCT_PIPELINE_NAMES.contains(click.getPipelineName())) {
```

```
            // Checks whether it was a product click.
            if (PRODUCT_PIPELINE_NAMES.contains(click.getPipelineName())) {

                var sku = click.getParameter(PRODUCT_ID_PARAMETER_NAME);

                if (sku && sku.length > 0) {
                    // Gets the product.
                    var product = ProductMgr.getProduct(sku);
                    if (product) {
                        // Checks for the product or a shared master.
                        var duplicate = false;
                        for (var j = 0; j < products.size(); j++) {
                            // Calculates a potential master.
                            var p1 = (product.isVariant() ? product.getVariationModel().getMaster() : product);
                            var p2 = (products[j].isVariant() ? products[j].getVariationModel().getMaster() : products[j]);

                            // Checks if it is the same product.
                            if (p1 === p2) {
                                duplicate = true;
                                break;
                            }
                        }

                        // Adds the product if it is not a duplicate.
                        if (!duplicate) {
                            products.add(product);
                        }
```

```
                        }
                    }
                }

                // Checks whether the maximum is reached.
                if (products.size() >= numberOfProducts) {
                    break;
                }
            }

            return products;

        }
    });

    /** The RecentlyViewedItems class */
    module.exports = RecentlyViewedItemsModel;
```

## lastVisited.isml:-

- This template dynamically generates a section on the webpage to display up to 3 last visited products.
- It ensures that the content is properly structured and styled for a user-friendly experience.
- Make sure to have the necessary product data available in the pdict.LastVisitedProducts object for this template to function correctly.

cart.isml 9+ | Product.js | lastvisited.isml 9+ | RecentlyViewedItemsModel.js

app_custom_TrainingBajigam > cartridges > app_custom_project > cartridge > templates > default > product > lastvisited.isml > iscontent

```
1  <iscontent type="text/html" charset="UTF-8" compact="true"/>
2
3
4
5  <isif condition="${!empty(pdict.LastVisitedProducts)}">
6      <div class="product-listing last-visited">
7          <h2 style="background-color:gainsboro;"><b>${Resource.msg('cart.lastvisited.header','checkout',
           null)}</b></h2>
8          <ul class="search-result-items tiles-container">
9              <isloop items="${pdict.LastVisitedProducts}" var="product" begin="0" end="2">
10                 <li class="grid-tile">
11                     <isinclude url="${URLUtils.url('Tile-Show','pid',product.ID,'showswatches','false',
                       'showpricing','false','showpromotion','true','showrating','false')}">
12                 </li>
13             </isloop>
14         </ul>
15     </div>
16 </isif>
```

## cart.isml:-

In the Cart.isml template include the Product-IncludeLasstVisited controller and route in order to show the Last visisted products in the Cart-Show page.

cart.isml 9+

app_custom_TrainingBajigam > cartridges > app_custom_project > cartridge > templates > default > cart > cart.isml > isdecorate > div.cart-recommendations > isinclude

```
1   <isdecorate template="common/layout/page">
94
95      <div class="cart-recommendations">
96          <isinclude url="${URLUtils.url('Product-IncludeLastVisited')}"/>
97          <isslot id="cart-footer" description="Footer for Cart page" context="global" />
98      </div>
99      <div class="container">
100         <isslot id="cart-recommendations-m" description="Recommended products" context="global" />
101     </div>
102 </isdecorate>
```

**Client side SCSS**:-

The below is the Client side scss which is used in the code while developing the customization.

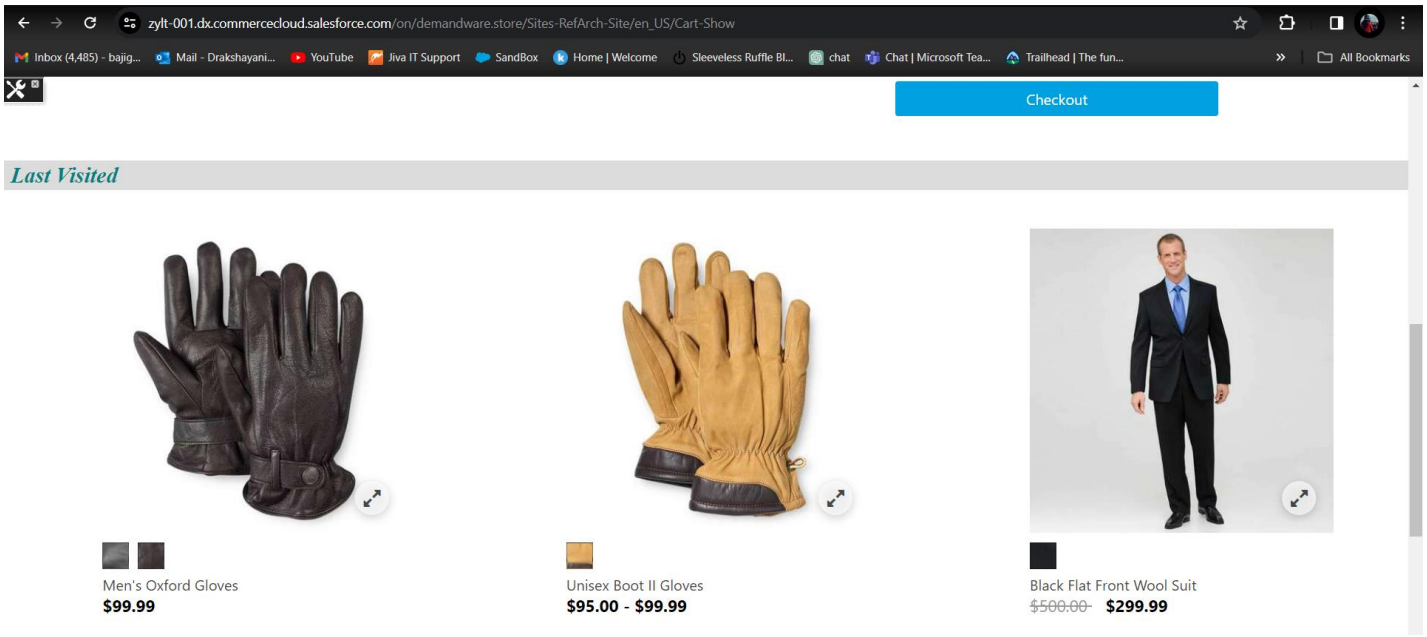
```scss
//Cart-Show Last Visited Produts
.cart-empty {
    margin-top: 4rem;
    h1 {
        text-align: center;
    }
    .cart-action-continue-shopping {
        float: none;
        text-align: center;
        margin-bottom: 4rem;
    }
    .product-listing {
        background-color: white-smoke;
        padding: 2rem 0;
        h2 {
            margin-left: 2.3rem;
        }
        .search-result-items {
            margin-top: 1em;
        }
    }
}

.cart-recommendations {
    margin-top: 1rem;
    .product-listing {
        background-color: white-smoke;
        padding: 1rem 0;
    }
}
```

```
141  .cart-recommendations,
142  .category-slot,
143  .product-slot {
144      .tiles-container {
145          display: flex;
146          flex-flow: row wrap;
147          justify-content: center;
148          .grid-tile {
149              box-shadow: none;
150              margin: 4rem;
151              padding: 2rem;
152              width: 25%;
153          }
154      }
155  }
```

```scss
.search-result-items {
    .grid-tile {
        box-shadow: none;
        margin: 0;
        padding: 0 .15%;
        width: 50%;
        @media screen and (min-width: 768px) {
            width: 33%;
        }
        .product-tile {
            margin: 1 .15%;
            padding: 2rem 0;
            width: 70%;
            margin-left: 5.5em;
            .product-name {
                font-family: sans-serif-alt;
                font-size: 1.2rem;
                font-weight: 100;
                height: auto;
                margin-top: .5em;
                overflow: hidden;
                text-overflow: ellipsis;
                white-space: nowrap;
                a {
                    color: cerulean;
                }
            }
```

```
252  .search-result-items {
253      margin: 0;
254      padding: 0;
255      li {
256          list-style: none outside none;
257      }
258      .new-row {
259          clear: both;
260      }
261      .grid-tile {
262          background: none repeat scroll 0 0 white;
263          box-shadow: 0 0 9px very-light-gray;
264          float: left;
265          list-style: none outside none;
266          margin: 0 0 2% 20px;
267          padding: 5%;
268          width: 16%;
269      }
270      .invisible {
271          display: none;
272      }
273  }
```

As below showed the last visited products will be displayed in the Cart-Show page based on the session information.




This is all about showing the last visited products in the Cart-Show page customization.

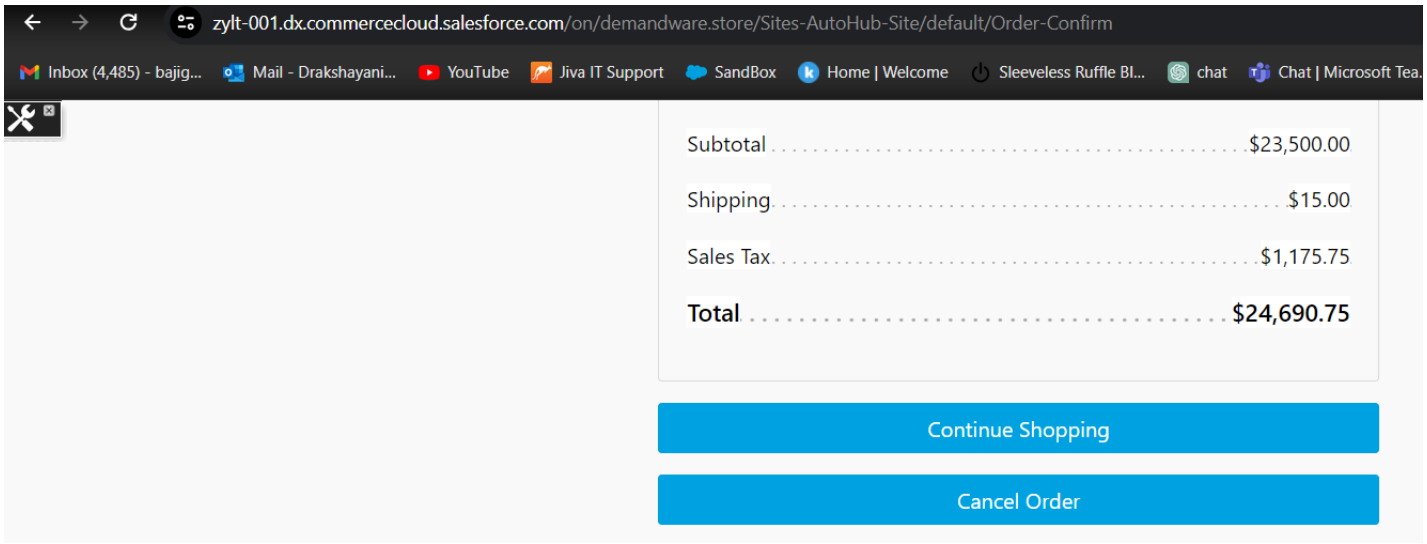
# 3.Cancel Order

In this customization I developed the custom feature which is used to cancel the order placed by the user.

- In Confirmation.isml we are including the cancel order button in-order to redirect to the OrderCancel1.js controller.
- In this file we are adding the querystring values as the current order number and current order shipping email address to the URL of OrderCancel1-Show.

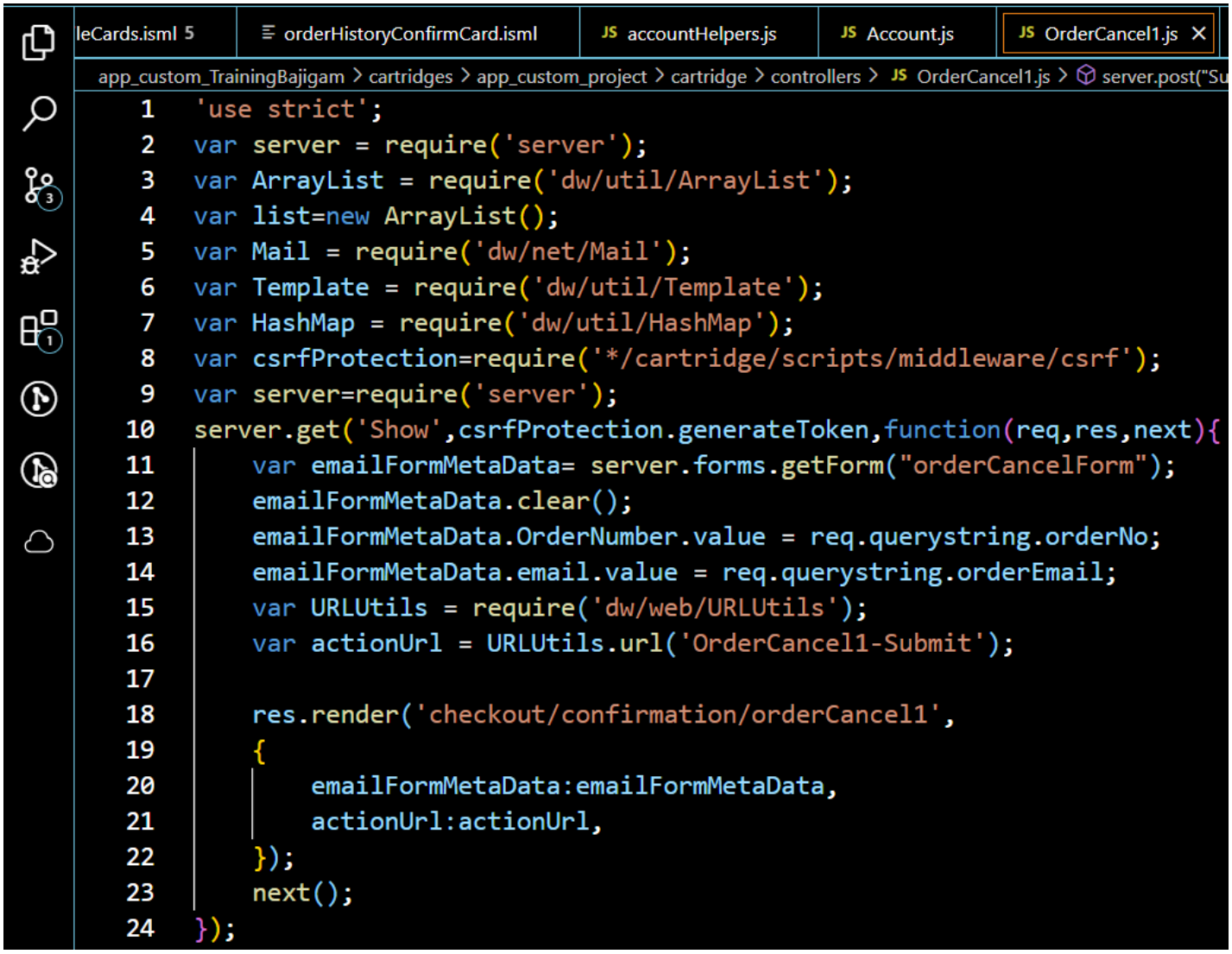
```
30          <div class="${pdict.returningCustomer ? 'col-sm-6 offset-sm-3' : 'col-sm-6 offset-sm-3 offset-md-0 pull-md-6' }">
31              <isinclude template="checkout/confirmation/confirmationDetails" />
32              <div class="mb-3">
33                  <a href="${URLUtils.url('Home-Show')}" class="btn btn-primary btn-block order-confirmation-continue-shopping"
                    role="button" aria-pressed="true">
34                      ${Resource.msg('button.continue.shopping','confirmation',null)}
35                  </a>
36              </div>
37              <div class="mb-3">
38                  <a href="${URLUtils.url('OrderCancel1-Show' ,"orderEmail",pdict.order.orderEmail,"orderNo",pdict.order.
                    orderNumber)}" class="btn btn-primary btn-block order-confirmation-continue-shopping" role="button"
                    aria-pressed="true">
39                      Cancel Order
40                  </a>
41              </div>
42          </div>
43      </div>
44  </div>
45  </isdecorate>
```

Like below showed the Cancel Order Button will be displayed in the storefront from the Order-Confirm Controller and route after we are placing the order.




- OrderCancel.js is the Controller for canceling the order that uses the “server.get” as a request method.

- In this we are requiring the MetaData form from the forms folder i.e “orderCancelForm”.
- And also we are requiring the current order number and current shipping email address from the query string.



## Form meta Data:-

- The below is the Form Meta Data Code that is used to create the form fields and their attributes as a constraints.

```xml
<?xml version="1.0"?>
<form xmlns="http://www.demandware.com/xml/form/2008-04-19">

    <field
        formid="email"
        label="label.input.email.profile"
        mandatory="true"
        max-length="50"
        missing-error="error.message.required"
        parse-error="error.message.parse.email.profile.form"
        range-error="error.message.50orless"
        regexp="^[\w.%+-]+@[\w.-]+\.[\w]{2,6}$"
        value-error="ValueErrorText"
        type="string"/>
        <!-- TODO value-error not comming back from the platform -->

    <field
        formid="category"
        label="Problem with site?"
        type="string"
        mandatory="false"
        binding="category"
        missing-error="address.category.missing"/>

    <field
        formid="OrderNumber"
        label="OrderNumber"
        type="string"
        mandatory="true"
```

```
        mandatory="true"
        missing-error="error.message.required"/>
    <field
        formid="unexpectedProducts"
        label="label.unexpectedProducts"
        type="boolean"/>
    <field
        formid="time"
        label="label.time"
        type="boolean"/>
    <field
        formid="reason"
        label="Reasons"
        type="string"
        mandatory="true"
        binding="countryCode"
        missing-error="address.country.missing">
        <options>
            <option optionid="" label="resource.select" value="select country"/>
            <option optionid="IND" label="Price is too high" value="Price is too high" />
            <option optionid="US" label="Didn't got any bonus products" value="Didn't got any bonus products" />
            <option optionid="UK" label="There is no COD option" value="There is no COD option" />
            <option optionid="PK" label="Reviews of this product is bad" value="Reviews of this product is bad" />
            <option optionid="JP" label="Order placed by mistake" value="Order placed by mistake" />
        </options>
    </field>
</form>
```

# ISML for displaying form:-

By using the form metadata which we are requiring from the controller and accessing through the pdict we are displaying the form in the storefront from orderCancel1.isml template.

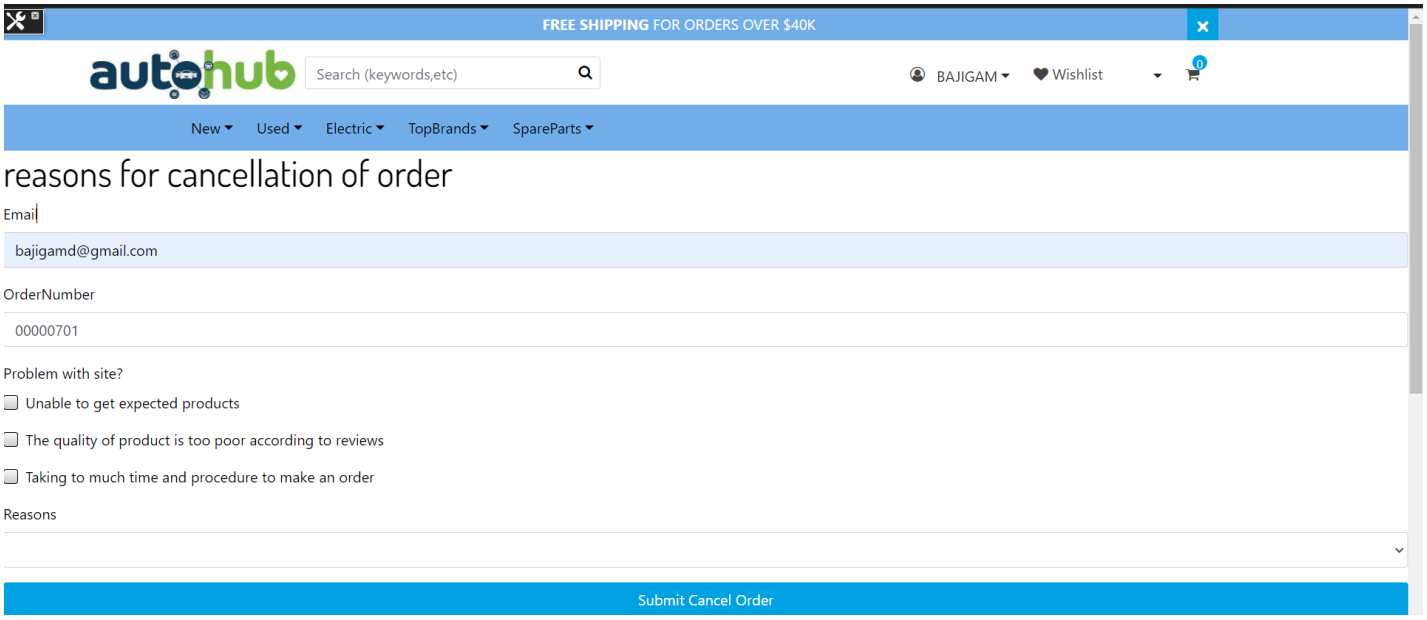
```
<isdecorate template="common/layout/page">
    <h1>reasons for cancellation of order</h1>

    <form action="${pdict.actionUrl}" method="POST">

        <div class="form-group
            <isif condition="${!!pdict.emailFormMetaData.email.mandatory === true}">required</isif>">
            <label class="form-control-label" for="emailFormMetaData-form-email">
                <isprint value="${pdict.emailFormMetaData.email.label}" encoding="htmlcontent" />
            </label>
            <input
                type="text"
                class="form-control"
                id="emailFormMetaData-form-email"
                data-missing-error="${Resource.msg('error.message.required','forms',null)}"
                data-pattern-mismatch="${Resource.msg('error.message.parse.email.profile.form','forms',null)}"
                data-range-error="${Resource.msg('error.message.50orless','forms',null)}"
                aria-describedby="form-email-error"
                <isprint value=${pdict.emailFormMetaData.email.attributes} encoding="off" />>
                <isif condition="${pdict.emailFormMetaData.email.valid == false}">
                    <h6 style="color:red;">The Email was already exists</h6>
                </isif>
            <div class="invalid-feedback" id="form-email-error"></div>
        </div>
        <div class="form-group
            <isif condition="${!!pdict.emailFormMetaData.OrderNumber.mandatory === true}">required</isif>">
            <label class="form-control-label" for="emailFormMetaData-form-email">
                <isprint value="${pdict.emailFormMetaData.OrderNumber.label}" encoding="htmlcontent" />
            </label>
```

```
30              <input
31                  type="text"
32                  class="form-control"
33                  id="emailFormMetaData-form-email"
34                  data-missing-error="${Resource.msg('error.message.required','forms',null)}"
35                  <isprint value=${pdict.emailFormMetaData.OrderNumber.attributes} encoding="off" />>
36              <div class="invalid-feedback" id="form-email-error"></div>
37          </div>
38
39          <label class="form-control-label" for="emailFormMetaData-form-fname">
40              <isprint value="${pdict.emailFormMetaData.category.label}" encoding="htmlcontent" />
41          </label>
42
43          <div class="form-group custom-control custom-checkbox">
44              <input
45                  type="checkbox"
46                  class="custom-control-input" id="unexpectedProducts" aria-required="false"
47                  <isprint value=${pdict.emailFormMetaData.unexpectedProducts.attributes} encoding="off" />
48              >
49              <label class="custom-control-label" for="unexpectedProducts" >
50                  Unable to get expected products
51              </label>
52          </div>
```

```
54          <div class="form-group custom-control custom-checkbox">
55              <input
56                  type="checkbox"
57                  class="custom-control-input" id="review" aria-required="false"
58                  <isprint value=${pdict.emailFormMetaData.women.attributes} encoding="off" />
59              >
60              <label class="custom-control-label" for="review" >
61                  The quality of product is too poor according to reviews
62              </label>
63
64          </div>
65
66          <div class="form-group custom-control custom-checkbox">
67              <input
68                  type="checkbox"
69                  class="custom-control-input" id="time" aria-required="false"
70                  <isprint value=${pdict.emailFormMetaData.time.attributes} encoding="off" />
71              >
72              <label class="custom-control-label" for="time" >
73                  Taking to much time and procedure to make an order
74              </label>
75
76          </div>
```

```
78          <div class="form-group">
79              <label class="form-control-label" for="reason">
80                  <isprint value="${pdict.emailFormMetaData.reason.label}" encoding="htmlcontent"/>
81              </label>
82              <select class="form-control" id="reason"
83                  <isprint value=${pdict.emailFormMetaData.reason.attributes} encoding="off" />>
84                  <isloop items="${pdict.emailFormMetaData.reason.options}" var="reason" status="loopstatus">
85                      <option id="${reason.id}" value="${reason.htmlValue}"  <isif condition="${reason.selected}">selected</isif>>
86                      ${reason.label || ' '}</option>
87                  </isloop>
88              </select>
89              <div class="invalid-feedback"></div>
90          </div>
91
92          <button type="submit" class="btn btn-block btn-primary">
93              Submit Cancel Order
94          </button>
95
96          <input type="hidden" name="${pdict.csrf.tokenName}" value="${pdict.csrf.token}"/>
97      </form>
98
99  </isdecorate>
```

- By using the above mentioned controller, form metadata and isml template we are displaying the order cancel form in the storefront .
- After we clicking on the Cancel Order Button in the OrderConfirm-Show page the user will redirect to the this particular page.




- After clicking the Submit Cancel Order button the control of execution comes to the OrderCancel1.js server.post “Submit” route.
- In the below shown “OrderCancel1- Submit” controller and route we are getting the submitted form and checking whether it is valid or not.
- If the submitted form is valid then we are cancelling the order by using the Order status from the OrderMgr class and sending the Email to the respective user.

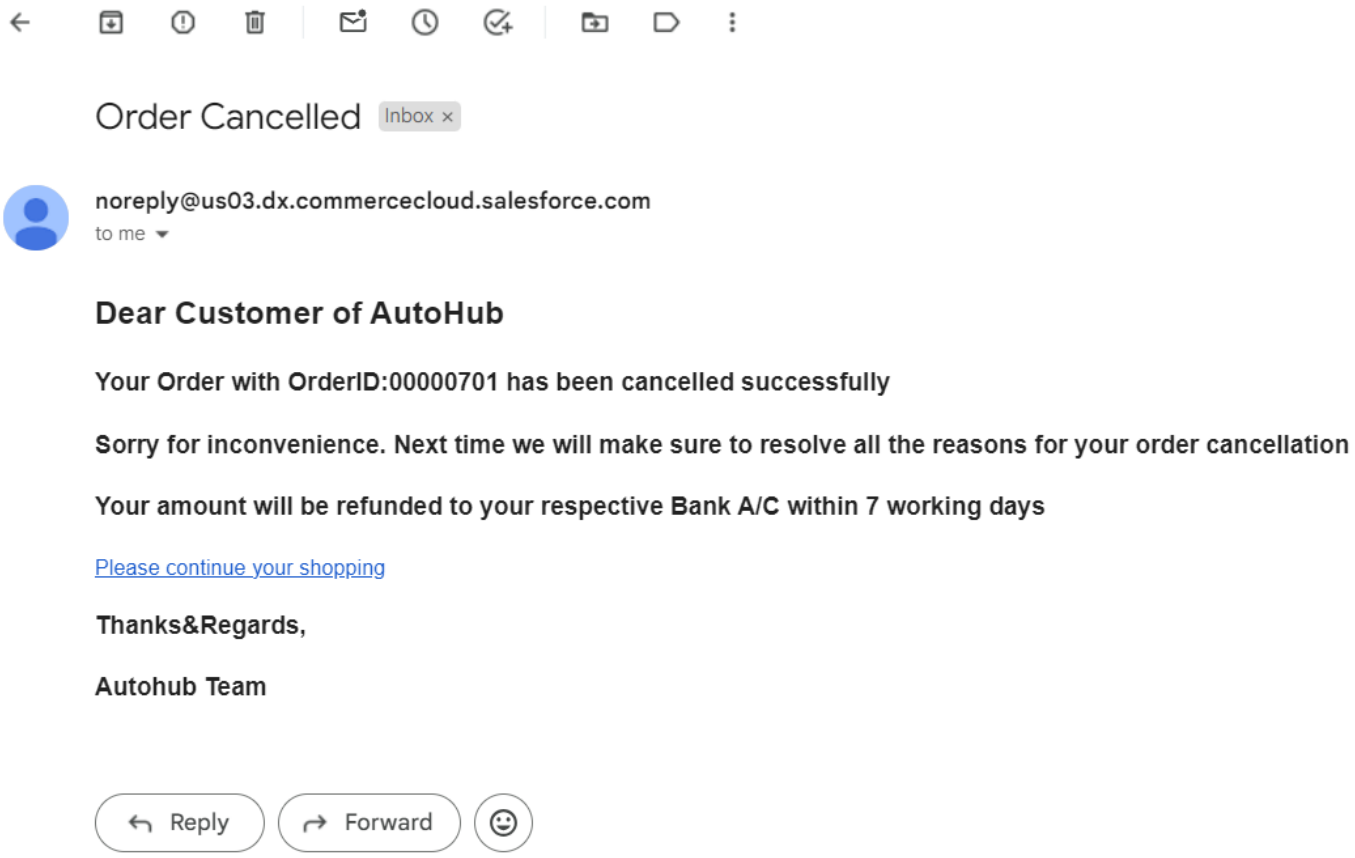
```
server.post("Submit",csrfProtection.validateRequest,function(req,res,next){
    var emailFormMetaData= server.forms.getForm("orderCancelForm");
    var CustomObjectMgr = require('dw/object/CustomObjectMgr');
    var Transaction = require('dw/system/Transaction');
    if(emailFormMetaData.valid){
            var currentOrderNo = emailFormMetaData.OrderNumber.value;
            var OrderMgr = require('dw/order/OrderMgr');
            var order = OrderMgr.getOrder(currentOrderNo);
            var Transaction = require('dw/system/Transaction');
            Transaction.wrap(function(){
                if (order) {
                    // Check if the order is in a deletable status
                        // Delete the order
                        OrderMgr.cancelOrder(order);
                }
            });
            var hashMap = new HashMap();
            hashMap.put("firstName","Customer");
            hashMap.put("lastName","of AutoHub");
            hashMap.put("OrderID",emailFormMetaData.OrderNumber.value);

            var template = new Template("orderCancelBody");
            var content = template.render(hashMap);

            var mail = new Mail();
            mail.addTo(emailFormMetaData.email.value); //receiver email address
            mail.setFrom("bajigcmd@gmail.com"); // sender email address
            mail.setSubject("Order Cancelled");
            mail.setContent(content);
            mail.send();
            if(customer.anonymous){
                res.redirect("Home-Show");
            }
            else{
                res.redirect("Order-History");
            }
    }
    else{
        res.render('checkout/confirmation/orderCancel1',{emailFormMetaData:emailFormMetaData});
    }
    next();
});
module.exports=server.exports();
```

## Email Body isml template:-

The email we sent to the user having the body as the below mentioned text from the orderCancelBody.isml template.

```
<h2>Dear ${pdict.firstName}  ${pdict.lastName} </h2>

<h3>Your Order with OrderID:<b>${pdict.OrderID}</b> has been cancelled successfully</h3>

<h3>Sorry for inconvenience. Next time we will make sure to resolve all the reasons for your order cancellation </h3>
<h3>Your amount will be refunded to your respective Bank A/C within 7 working days</h3>

<a href="${URLUtils.url('Home-Show')}">Please continue your shopping</a>

<h3><b>Thanks&Regards,</b></h3>
<h3><b>Autohub Team<b></h3>
```

- Like the Below shown the user will be notified through the Email as your order has been cancelled along with the respective Order number.




- After submitting the Order Cancel then we need to redirect to the Home-Show if he is unregistered customer.

- If the user is a registered customer then we need to redirect to the Order-History Page. Then in Order-History Page we need to display the Cancelled Orders and Confirmed Orders separately.
- For that I done the following modifications in the dashboardProfileCards.isml

```
<div class="row justify-content-center">
    <div class="col-sm-6">
        <!---Profile--->
        <isinclude template="account/profileCard"/>

        <isif condition="${!pdict.account.isExternallyAuthenticated}">
            <!---Password--->
            <isinclude template="account/passwordCard"/>
        </isif>

        <!---Address Book--->
        <isinclude template="account/addressBookCard"/>
    </div>
    <div class="col-sm-6">
        <!---Order History--->
        <isif condition="${pdict.account.orderHistory}">
            <isset name="order" value="${pdict.account.orderHistory}" scope="page"/>
                <isinclude template="account/order/orderHistoryCard"/><!---Order History for Cancelled Orders--->
                <isinclude template="account/orderHistoryConfirmCard"/><!---Order History for Confirmed Orders--->
        </isif>
        <!---Payment--->
        <isinclude template="account/paymentCard"/>
    </div>
</div>
```

## **orderHistoryCard.isml** :-

For displaying the Cancelled orders in the “Order-History” we are using the below isml code by checking the condition whether the orderStatus is CANCELLED or not.

```
<div class="card">
 <isif condition="${order.orderStatus.displayValue == "CANCELLED"}">
    <isif condition="${pdict.accountlanding}">
        <div class="card-header clearfix">
            <h2 class="pull-left">${Resource.msg('label.orderhistory','account',null)}</h2>
            <a href="${URLUtils.url('Order-History')}" class="pull-right"  aria-label="${Resource.
            msg('label.orderhistory.vieworderhistory','account',null)}">${Resource.msg('link.view',
            'account',null)}</a>
        </div>
    <iselse/>
        <div class="card-header clearfix">
            <h3 class="pull-left">${Resource.msg('label.orderhistory.orderno','account',null)} $
            {order.orderNumber}</h3>
            <a href="${URLUtils.url('Order-Details', 'orderID', order.orderNumber, 'orderFilter',
            pdict.orderFilter)}" class="pull-right" aria-label="${Resource.msgf('label.orderhistory.
            vieworderdetails', 'account', null, order.orderNumber)}">${Resource.msg('link.view',
            'account',null)}</a>
        </div>
    </isif>

    <div class="card-body card-info-group">
        <div class="row">
            <isif condition="${pdict.accountlanding}">
                <div class="col-12">
                    ${Resource.msg('label.orderhistory.mostrecentorder','account',null)}
                </div>
            </isif>
            <div class="col-4 hidden-xs-down">
```

```
23                  <img class="img-fluid" src="${order.firstLineItem.imageURL}" alt="${order.
                    firstLineItem.alt}" title="${order.firstLineItem.title}">
24              </div>
25              <div class="col-sm-8 col-12">
26                  <isif condition="${pdict.accountlanding}">
27                      <p>${Resource.msg('label.orderhistory.ordernumber','account',null)} <isprint
                        value="${order.orderNumber}"/></p>
28                  </isif>
29                  <p>${Resource.msg('label.orderhistory.dateordered','account',null)} <isprint
                    value="${order.creationDate}"/></p>
30                  <p>${Resource.msg('label.orderhistory.orderstatus','account',null)} <span
                    class="dashboard-order-card-status">${order.orderStatus}</span></p>
31                  <isinclude template="account/order/shippingTitle"/>
32              </div>
33          </div>
34      </div>
35      <div class="card-footer">
36          <div class="row">
37              <div class="col-7 dashboard-order-card-footer-columns card-info-group">
38                  <p>${Resource.msg('label.orderhistory.totalitems','account',null)}</p>
39                  <p class="dashboard-order-card-footer-value">
40                      <isprint value="${order.productQuantityTotal}" formatter="#"/>
41                  </p>
42              </div>
```

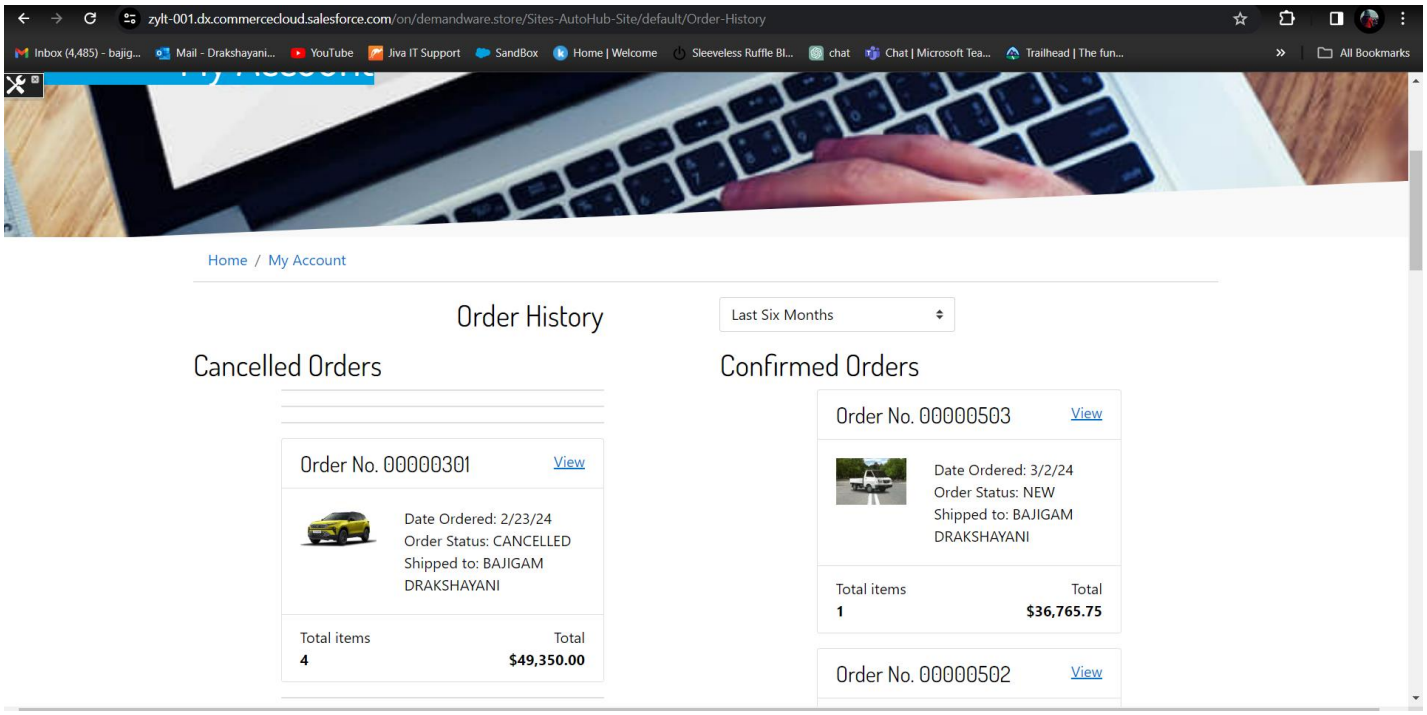
```
43              <div class="col-5 dashboard-order-card-footer-columns card-info-group">
44                  <p>${Resource.msg('label.orderhistory.ordertotal','account',null)} </p>
45                  <p class="dashboard-order-card-footer-value">
46                      ${order.priceTotal}
47                  </p>
48              </div>
49          </div>
50      </div>
51  </isif>
52  </div>
```

## orderHistoryConfirmCard.isml:-

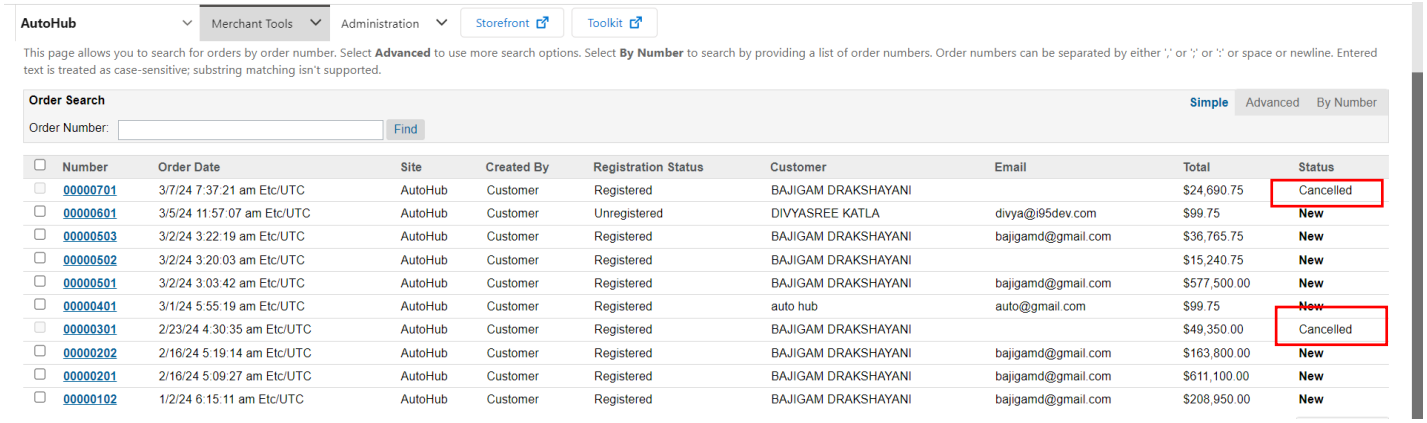
- For displaying the Confirmed orders in the “Order-History” we are using the below isml code by checking the condition whether the orderStatus is CANCELLED or not.

```
<div class="card">
<isif condition="${order.orderStatus.displayValue != "CANCELLED"}">
    <isif condition="${pdict.accountlanding}">
        <div class="card-header clearfix">
            <h2 class="pull-left">${Resource.msg('label.orderhistory','account',null)}</h2>
            <a href="${URLUtils.url('Order-History')}" class="pull-right"  aria-label="${Resource.msg('label.orderhistory.
            vieworderhistory','account',null)}">${Resource.msg('link.view','account',null)}</a>
        </div>
    <iselse/>
        <div class="card-header clearfix">
            <h3 class="pull-left">${Resource.msg('label.orderhistory.orderno','account',null)} ${order.orderNumber}</h3>
            <a href="${URLUtils.url('Order-Details', 'orderID', order.orderNumber, 'orderFilter', pdict.orderFilter)}"
            class="pull-right" aria-label="${Resource.msgf('label.orderhistory.vieworderdetails', 'account', null, order.
            orderNumber)}">${Resource.msg('link.view','account',null)}</a>
        </div>
    </isif>
    <div class="card-body card-info-group">
        <div class="row">
            <isif condition="${pdict.accountlanding}">
                <div class="col-12">
                    ${Resource.msg('label.orderhistory.mostrecentorder','account',null)}
                </div>
            </isif>
            <div class="col-4 hidden-xs-down">
                <img class="img-fluid" src="${order.firstLineItem.imageURL}" alt="${order.firstLineItem.alt}" title="${order.
                firstLineItem.title}">
            </div>
            <div class="col-sm-8 col-12">
                <isif condition="${pdict.accountlanding}">
                    <p>${Resource.msg('label.orderhistory.ordernumber','account',null)} <isprint value="${order.orderNumber}"/></
                    p>
                </isif>
                <p>${Resource.msg('label.orderhistory.dateordered','account',null)} <isprint value="${order.creationDate}"/></p>
                <p>${Resource.msg('label.orderhistory.orderstatus','account',null)} <span class="dashboard-order-card-status">$
                {order.orderStatus}</span></p>
                <isinclude template="account/order/shippingTitle"/>
            </div>
        </div>
    </div>
    <div class="card-footer">
        <div class="row">
            <div class="col-7 dashboard-order-card-footer-columns card-info-group">
                <p>${Resource.msg('label.orderhistory.totalitems','account',null)}</p>
                <p class="dashboard-order-card-footer-value">
                    <isprint value="${order.productQuantityTotal}" formatter="#"/>
                </p>
            </div>
            <div class="col-5 dashboard-order-card-footer-columns card-info-group">
                <p>${Resource.msg('label.orderhistory.ordertotal','account',null)} </p>
                <p class="dashboard-order-card-footer-value">
                    ${order.priceTotal}
                </p>
            </div>
```

As below showed we will get the Cancelled and Confirmed orders separately in the Order-History




# Business Manager cancelled orders status:-




That's all about the Cancel-Order Customization.

Thanks for reading 😊.