**Use Case:**

Customer would like to be able to add products to the order while not leaving the order detail page.

Build 2 LWC components and put them on order record page:

1. “Available Products” - displays available products suitable for Order based on Order’s price book
2. “Order Products” – displays products added to the current order

**Acceptance Criteria:**

1. The solution is available as a repository on GitHub/Bitbucket etc
   1. Code should be deployable using SFDX
   2. Use git readme for feedback and notes
   3. Try to commit while developing
2. The “Available Products” component displays orderable products in a 2-column list displaying *Name* and *List Price*
   1. Products that are already added to the order should appear on top
   2. Each product can only appear once in the list
   3. **(Optional)** Sort by column Developer Comments: Sort by feature can be extended with in lightning-datatable (Example for reference http://amitsalesforce.blogspot.com/2020/07/lightning-datatable-sorting-in-lightning-web-components.html)
   4. **(Optional)** Search by product name
3. The “Available Products” component has to provide the ability for the user to add a product from the list to the order
   1. When the same product is not yet added to the order it will be added with a quantity of 1
   2. When the product already exists the quantity of the existing order product should be increased by 1
4. “Order Products” component has to display the order products in a table displaying the *Name*, *Unit Price, Quantity* and *Total Price*
   1. When the user adds a new product or updates an existing product on the order (see point 3) the list is refreshed to display the newly added
   2. **(Optional)** Sort the list by column
5. “Order Products” component has an “Activate” button that sets the status of the order and order items to “Activated”
   1. When activated the end user will not be able to add new order items or confirm the order for a second time.
6. A test coverage of at least 80% for both APEX components is required.
7. We would like to see LWC, but Aura/Vlocity is ok as well.
8. Please use apex for queries, DMLs.
9. Create a Salesforce Developer login for this assignment and build it as a SFDX project.

**Tips**

* We value how the solution is designed more than the number of requirements covered
* Try to use OOTB LWC components provided by Salesforce, see also the component library from Salesforce: <https://developer.salesforce.com/docs/component-library/overview/components>
* Make sure you commit the Order Flexi page to git
* Apply best practices, we will be reviewing both the functionality and implementation of your solution
* To avoid any minus points when your code is not deployable, include short video of your solution that covers the Acceptance Criteria and commit it to git
* Document your code to help us understand the technical choices you made when building the solution

**Extra Acceptance Criteria (sorted by priority):**

1. Components should be independent (we should be able to drag and drop them at any place in the layout)
   1. To ensure an optimal user experience the page should not be reloaded and only the changed or new items should be refreshed/added

Developer Comments:

1. Lightning Message Services
2. Platform Events

We have achieved the communication between two Independent LWC’s is via Lightning Message services. I see this as a very lightweight solution and easy in interpreting the logic. I preferred to implement this though it had drawbacks, as this is not supported in Mobile and Communities. Since I am beginner in LWC I did wanted to try something new.

The alternative solution this can be Achieved is via Platform Events. The platform events need to be Created as object and fields under it, also the records are Inserted and maintained via profile permissions.

1. Extend the logic of “Order Products” component’s “Activate” button to do the confirmation of the order in an external system.
   1. The request format expected by the external system should follow the following JSON structure:

{

   "accountNumber": "",

    "orderNumber": "",

    "type": "order type",

    "status": "order status",

    "orderProducts": [{

        "name": "product name",

        "code": "product code",

        "unitPrice": 10.00,

        "quantity": 1

    }]

}

Developer Comments: We need to instantiate the wrapper When activate button is clicked from orderProducts LWC, and populate the information retrieved from the Order details and serialize the wrapper -- JSON.serialize(wrapperInstance, true);.

We now have the JSON to send it to external System

public class OrderDetailsWrapper {

public String accountNumber;

public String orderNumber;

public String type;

public String status;

public List<OrderProducts> orderProducts;

public class OrderProducts {

public String name;

public String code;

public Double unitPrice;

public Integer quantity;

}

}

* 1. Request is sent as POST
  2. Order of the JSON fields in the above JSON structure is not relevant but the data type is.
  3. Errors and time-outs of the external system need to be handled
     1. All 200 responses are considered OK
     2. Any non-200 response is handled as ERROR
  4. For this use case generate a new endpoint URL at <https://requestcatcher.com/>
  5. We prefer to see this implemented using apex

1. The number of products can exceed 200; the solution needs to be able to handle this while providing a proper user experience.

Developer Comments: The pagination can be achieved via client side controller. I prefer to do it as referring to the below link:

https://vkambham.blogspot.com/2020/02/lwc-paginator.html