“Savor Salmon Sensations: Fresh, Fast, and Flavorful Deliveries to Your Door”

I. Introduction: Business concept

Our recently launched recipe box service, specializing in five exquisite salmon-based menus, offers a seamless purchasing experience through a mobile app integrated with our backend XML database. Upon order confirmation, our team promptly prepares the package for delivery to the customer's specified address.

At this juncture, customers have the option to cancel their confirmed order, receiving a 50% refund of the total order value directly into their bank account. Once the package is prepared and ready for dispatch, our dedicated delivery personnel ensure the order reaches the customer's doorstep. To close the order cycle, we invite customers to provide ratings for the menus they've experienced. While this feedback step is voluntary, it is highly encouraged, as it serves as invaluable input for the continual enhancement of our offerings and service.

II. Method:

The recipe box company, Good Food, employs an XML-based data structure, utilizing XSD (XML Schema Definition) to delineate the structure and constraints of its XML data. This schema acts as a comprehensive guide, ensuring that the data is well-organized and adheres to predefined standards. The versatility and adaptability of the system are showcased through the implementation of various scenarios where XSLT (Extensible Stylesheet Language Transformations) is used. These transformations demonstrate the system's capability to not only present data in a user-friendly HTML format but also convert it into other formats such as XML and JSON, catering to diverse informational requirements.

The development of the Good Food database unfolds through a series of methodical steps:

1. Step 1: XML Structure Definition *(including data model)*
2. Step 2: XML Development
3. Step 3: XSD Creation
4. Step 4: 05 XSLT for HTML Transformation
5. Step 5: XSLT for XML Transformation
6. Step 6: XSLT for JSON Transformation

This structured approach not only streamlines the development process but also ensures that the Good Food database is robust, flexible, and capable of meeting the evolving needs of the business and its customers.

**Notes: The tasks assigned to each group member at Annex 1*

2.1. Step 1: XML Structure Definition

GOOD_FOOD

- a. MENUS
  - i. MENU (att: id)
    1. MENU_NAME
    2. DESCRIPTION
    3. INGREDIENTS
       - a. INGREDIENT
    4. METHODS
       - a. METHOD
    5. PRICE
- b. CUSTOMERS
  - i. CUSTOMER (att: id)
    1. CUSTOMER_FNAME
    2. CUSTOMER_LNAME
    3. CUSTOMER_PHONE
    4. CUSTOMER_ADDRESS
- c. ORDERS
  - i. ORDER (att: id)
    1. CUSTOMER_ORDERED (att: c)
       - a. ORDER_DATE
       - b. ORDER_STATUS
       - c. ITEMS
         - i. ITEM (att: m)

1. ORDER_QUANTITY

d. DELIVERY_PEOPLE
    i. DELIVERY_PERSON (att: id)
        1. DELIVERY_PERSON_FNAME
        2. DELIVERY_PERSON_LNAME
        3. DELIVERY_PERSON_PHONE
e. DELIVERY_ORDERS
    i. DELIVERY_ORDER (att: id)
        1. ORDER_DELIVERED (att: o)
            a. COURIR (att: dp)
                i. DELIVERY_STATUS
f. MENU_RATINGS
    i. MENU_RATING (att: id)
        1. DELIVERY (att: do)
            a. RATED_MENU (att: m)

Data Description

The XML data structure as shown in the picture above describe a simple placeholder for our recipe box company's data that can be populated inside six main nodes:

1. **MENUS**
   MENUS node contains information related to Good Food main menus such as menu name, description, ingredient, cooking method, and price. Attribute id is associated with the MENU node, representing a unique menu that can be referred to by using key=id.
2. **CUSTOMERS**
   CUSTOMERS node contains information related to the buyer of our menus such as name (first name and last name), phone number, and home address. Attribute id is associated with CUSTOMER node, representing a unique customer that can be referred to by using key=id.
3. **ORDERS**
   ORDERS node contains information related to the orders made by the customers such as order date, order status, ordered menu(s) and their quantity. Attribute id is associated with an ORDER node, representing a unique order that can be referred to by using key=id. Attribute c is associated with CUSTOMER_ORDERED node, referring to the attribute id in the CUSTOMER node that represents a unique customer who made the respected order. Attribute m is associated with the ITEM node, referring to the attribute id in the MENU node that represents a unique menu ordered by the customer.
4. **DELIVERY_PEOPLE**
   DELIVERY_PEOPLE node contains information about the person who is obliged to deliver the item to the customer's address such as name (first name and last name) and phone number. Attribute id is associated with DELIVERY_PERSON node, representing a unique delivery person that can be referred using key=id
5. **DELIVERY_ORDERS**
   DELIVERY_ORDERS node contains information about the orders that are ready to be delivered such as the information about the associated order, the delivery person assigned to deliver the food, and the status of the delivery. Attribute id is associated with DELIVERY_ORDER node, representing a unique delivery order that can be referred using key=id. Attribute o is associated with ORDER_DELIVERED node, referring to the attribute id in the ORDER node that represents a unique order made by the customer. Attribute dp is associated with COURIR node, referring to the attribute id in the DELIVERY_PERSON node that represents a unique delivery person who delivers the food to the customer's address.
6. **MENU_RATINGS**
   MENU_RATINGS order contains information about the ratings given by the customers with respect to their delivered orders. Attribute id is associated with MENU_RATING node, representing a unique customer that can be referred using key=id. Attribute do is associated with DELIVERY node, referring to the attribute id in the DELIVERY_ORDER node that represents a unique delivery order. Attribute m is associated with the RATED_MENU node, referring to the attribute id in the MENU node that represents a unique menu that has been delivered to the customer's address.

Comment about the data model

By looking at how our xml file is structured, it can be noticed that each main node is written in either nested or flat structure. Basically, the nodes that contain same level fields are written with flat structure, for example in the CUSTOMER,

```
…
<gf:CUSTOMER id="3">
```

```
  <gf:CUSTOMER_FNAME>Michael</gf:CUSTOMER_FNAME>
  <gf:CUSTOMER_LNAME>Johnson</gf:CUSTOMER_LNAME>
  <gf:CUSTOMER_PHONE>555-123-4567</gf:CUSTOMER_PHONE>
  <gf:CUSTOMER_ADDRESS>
        789 Pine St, Village, State, 98765
  </gf:CUSTOMER_ADDRESS>
</gf:CUSTOMER>
…
```

since we can infer that CUSTOMER_FNAME, CUSTOMER_LNAME, CUSTOMER_PHONE, and CUSTOMER_ADDRESS nodes are semantically non-hierarchical. However, the nodes with hierarchical fields, for example the ITEMS nodes,

```
…
<gf:ITEMS>
   <gf:ITEM m="1">
      <gf:ORDER_QUANTITY>10</gf:ORDER_QUANTITY>
   </gf:ITEM>
</gf:ITEMS>
…
```

contains information that is hierarchical by nature, e.g. ITEMS is a collection of several ITEM and for each item there is specified ORDER_QUANTITY associated with that ITEM.

However there are some nodes that the relationships between the child nodes are quite vague like DELIVERY_ORDER for example. This node can be written like this,

```
…
<gf:DELIVERY_ORDER id="44">
    <gf:ORDER_DELIVERED o="50" />
        <gf:COURIR dp="2" />
        <gf:DELIVERY_STATUS>
                        On the way
                <gf:DELIVERY_STATUS>
     </gf:ORDER_DELIVERED>
…
```

assuming that ORDER_DELIVERED, COURIR, and DELIVERY_STATUS are equal. But it can also be written using the nested structure like this,

```
<gf:DELIVERY_ORDER id="44">
  <gf:ORDER_DELIVERED o="50">
    <gf:COURIR dp="2">
      <gf:DELIVERY_STATUS>
        On the way
      </gf:DELIVERY_STATUS>
    </gf:COURIR>
  </gf:ORDER_DELIVERED>
…
```

To deal with the confusion, we ask Chat GPT using this following prompt:

I want to make xml file that represent delivery order of a customer, which one do you think would be a better structure, this

```
<gf:DELIVERY_ORDER id="44">
    <gf:ORDER_DELIVERED o="50" />
     <gf:COURIR dp="2" />
     <gf:DELIVERY_STATUS> On the way <gf:DELIVERY_STATUS>
 </gf:ORDER_DELIVERED>
```

or this ?

```
<gf:DELIVERY_ORDER id="44">
```

```
<gf:ORDER_DELIVERED o="50">
  <gf:COURIR dp="2">
    <gf:DELIVERY_STATUS>
      On the way
    </gf:DELIVERY_STATUS>
  </gf:COURIR>
</gf:ORDER_DELIVERED>
```

Chat GPT answer:

Both XML structures you provided are valid, but the choice between them depends on your specific use case, data modeling preferences, and the relationships you want to represent. Here are some considerations for each structure:

Flat Structure (First Example):

```
<gf:DELIVERY_ORDER id="44">
  <gf:ORDER_DELIVERED o="50" />
  <gf:COURIR dp="2" />
  <gf:DELIVERY_STATUS>On the way</gf:DELIVERY_STATUS>
</gf:DELIVERY_ORDER>
```

This structure is more flat and straightforward. Each element is at the same level, and the hierarchy is clear. It may be easier to read and understand, especially if the data is relatively simple and does not involve deeply nested relationships.

Nested Structure (Second Example):

```
<gf:DELIVERY_ORDER id="44">
  <gf:ORDER_DELIVERED o="50">
    <gf:COURIR dp="2">
      <gf:DELIVERY_STATUS>On the way</gf:DELIVERY_STATUS>
    </gf:COURIR>
  </gf:ORDER_DELIVERED>
</gf:DELIVERY_ORDER>
```

This structure introduces nesting, emphasizing the relationship between different components. The status is nested within the courier, which is nested within the delivered order. It may be more suitable if there are complex relationships between the elements, or if you want to represent a hierarchy.

Consider the following factors when making your decision:

- Readability: Choose a structure that makes the XML file easy to read and understand.
- Semantics: Choose a structure that accurately represents the relationships between different components. Consider whether the status is an inherent property of the courier, or if they are independent entities.
- Consistency: If you have other XML structures in your application, try to maintain consistency in how you organize and nest elements.
- If the relationships between the elements are straightforward and there's no need for deep nesting, the first example with a flat structure may be more suitable. If you want to emphasize a hierarchy or relationships between the elements, the second example with a nested structure could be a better choice.

Based on that response, I chose the nested structure because in our business model each delivery order is associated with the order, courier, and delivery status. By putting these nodes in nested structure, it can be assured the information about order, courier, and delivery status are also selected when we select the delivery order node.

2.2. Step 2: **XML Development:** Finalized & save under the filename: *good_food.xml*

2.3. Step 3: **XSD Creation:** Completed & save under the filename: good_food_schema.xml

2.4. Step 4: **05 XSLT for HTML Transformation:**

05 XSLT scenarios serve as visualization cases to address the following requirements:

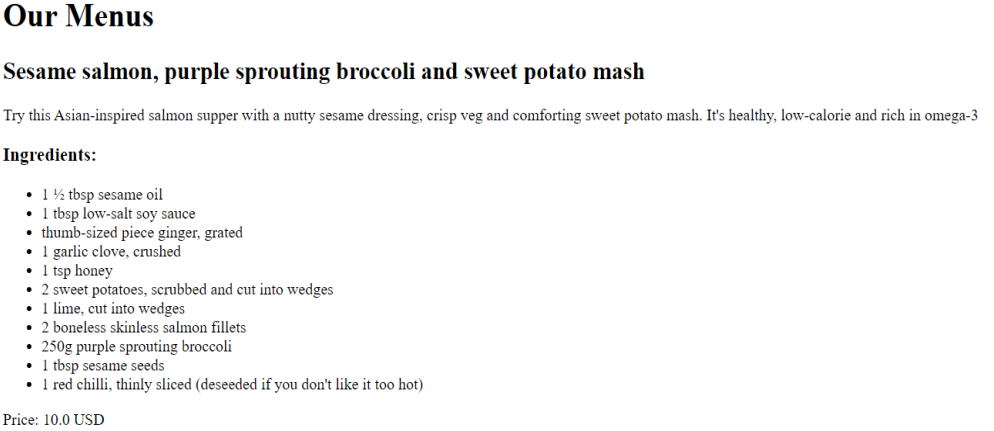
1. Catalogue each menu, specifying names, ingredients, and pricing details.
2. Display menus in a table layout, complete with descriptions, and order them based on price.
3. Determine which menu item is most frequently ordered.
4. Compute the mean rating assigned to each menu.
5. Evaluate delivery staff efficiency through metrics of successfully completed deliveries and overall transaction value.

Outcomes:

1. *Catalogue each menu, specifying names, ingredients, and pricing details.*

   Prior to delving into the specifics, a manager wishes to examine the menu and its ingredients to ensure alignment with the concepts.

   To accomplish this, we select MENU node and iterate their components as childs, namely Ingredients and prices.



Our Menus

Sesame salmon, purple sprouting broccoli and sweet potato mash

Try this Asian-inspired salmon supper with a nutty sesame dressing, crisp veg and comforting sweet potato mash. It's healthy, low-calorie and rich in omega-3

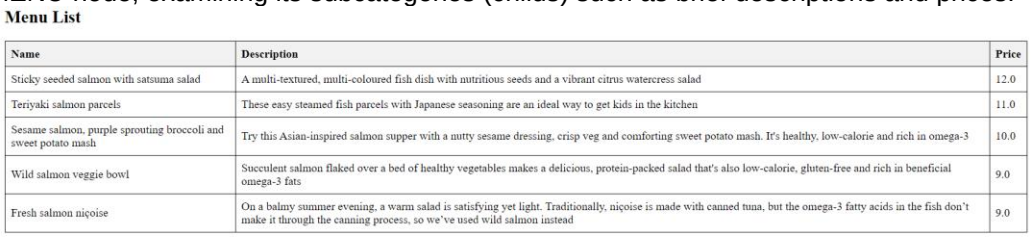
Ingredients:

- 1 ½ tbsp sesame oil
- 1 tbsp low-salt soy sauce
- thumb-sized piece ginger, grated
- 1 garlic clove, crushed
- 1 tsp honey
- 2 sweet potatoes, scrubbed and cut into wedges
- 1 lime, cut into wedges
- 2 boneless skinless salmon fillets
- 250g purple sprouting broccoli
- 1 tbsp sesame seeds
- 1 red chilli, thinly sliced (deseeded if you don't like it too hot)

Price: 10.0 USD

2. *Display menus in a table layout, complete with descriptions, and order them based on price.*

   Upon scrutinizing the detailed MENU information, the manager intends to view each menu item presented in a tabular format alongside their respective prices, aiming to predict which menu options may attract more customers.
   To achieve this, we structure the desired outcome into a table format and proceed to navigate through the MENU node, examining its subcategories (childs) such as brief descriptions and prices.



Menu List

Name	Description	Price
Sticky seeded salmon with satsuma salad	A multi-textured, multi-coloured fish dish with nutritious seeds and a vibrant citrus watercress salad	12.0
Teriyaki salmon parcels	These easy steamed fish parcels with Japanese seasoning are an ideal way to get kids in the kitchen	11.0
Sesame salmon, purple sprouting broccoli and sweet potato mash	Try this Asian-inspired salmon supper with a nutty sesame dressing, crisp veg and comforting sweet potato mash. It's healthy, low-calorie and rich in omega-3	10.0
Wild salmon veggie bowl	Succulent salmon flaked over a bed of healthy vegetables makes a delicious, protein-packed salad that's also low-calorie, gluten-free and rich in beneficial omega-3 fats	9.0
Fresh salmon niçoise	On a balmy summer evening, a warm salad is satisfying yet light. Traditionally, niçoise is made with canned tuna, but the omega-3 fatty acids in the fish don't make it through the canning process, so we've used wild salmon instead	9.0

3. *Determine which menu item is most frequently ordered.*
The CEO of our business wants to see which menu gets the most order, so we manage to write ordered_quantity_by_menu_html.xsl to see the sum of the order quantity per menu made by our customers.

Firstly, we select the MENU node and iterate through all the child nodes. Then we join the MENU node with the ITEM node by the id attribute in the MENU node and m attribute in the ITEM node.

Finally, the ORDER_QUANTITY (1 node below the ITEM) can be aggregated using the MENU_NAME as the group key. The result html file can be seen in ordered_quantity_by_menu.html

Based on our sales, this is the sum of ordered quantity by menu from the highest to lowest

- Menu name: Sesame salmon, purple sprouting broccoli and sweet potato mash
- Ordered Quantity: 103 pcs

- Menu name: Wild salmon veggie bowl
- Ordered Quantity: 99 pcs

- Menu name: Teriyaki salmon parcels
- Ordered Quantity: 68 pcs

- Menu name: Fresh salmon niçoise
- Ordered Quantity: 48 pcs

- Menu name: Sticky seeded salmon with satsuma salad
- Ordered Quantity: 33 pcs

4. *Compute the mean rating assigned to each menu.*

Business wants to understand which one menu is the most successful and receives the best feedback. To achieve this target we reached the MENU_RATINGS node, navigated through all child elements to RATED_MENU and joined this information to MENU->MENU_NAME node with an index m.

Based on our information about deliveries, average rating per menu is following

Menu name:	Average rating:
Wild salmon veggie bowl	4.277777777777778
Teriyaki salmon parcels	4
Sesame salmon, purple sprouting broccoli and sweet potato mash	3.875
Fresh salmon niçoise	3.714285714285714
Sticky seeded salmon with satsuma salad	3.666666666666667

5. *Evaluate delivery staff efficiency through metrics of successfully completed deliveries and overall transaction value.*

Management needs to understand how to evaluate and award or fine staff that is responsible for deliveries. To perform this estimating of couriers' performance we calculated the share of successfully delivered orders to the total amount of orders that have been assigned to each particular person.

In order to do that, we navigated to the node DELIVERY_ORDERS and reached the value under a child element DELIVERY_STATUS and calculated amount of statuses, where this value is equal to 1 (successfully completed) and divided by the total amount of DELIVERY_STATUS. All described above calculations are done with joining to DELIVERY_PERSON node via index dp.

Based on amount of succesfully delivered orders comparing to the total amount of orders to be delivered courirs' performance is following

Person name	Performance, %
John	100
David	100
Amy	100
Robert	100
Lisa	100
Mark	100
Emily	100
Jane	90.9090909090909
Susan	66.6666666666667
Michael	NaN

2.6. Step 5: XSLT for XML Transformation

XSLT transformation on orderinfo_tranform.xsl to produce the Output_xml_sorted.xml file from the order_info_transform.xml XML document. The output is an XML tree containing only the ORDER elements (not the GOOD_FOOD elements) which have been slightly transformed and sorted by Date of order.The goal is to transform the `order_info_transform.xml` XML document using the `orderinfo_tranform.xsl` stylesheet. The output should be an XML tree named `Output_xml_sorted.xml`.

-Transformed Document-

```
<Order_Overview xmlns:gf="http://mynamespace.fr/myvocabulary#">
    <gf:ORDERS>
        <gf:ORDER ondate="2023-01-01">
            <gf:CUSTOMER_ORDERED>3:
                        <gf:ORDER_STATUS/>
                <gf:ITEMS>
                    <gf:ITEM>2</gf:ITEM>
                </gf:ITEMS>
            </gf:CUSTOMER_ORDERED>
        </gf:ORDER>
    </gf:ORDERS>
    <gf:ORDERS>
        <gf:ORDER ondate="2023-01-01">
            <gf:CUSTOMER_ORDERED>5:
                        <gf:ORDER_STATUS/>
                <gf:ITEMS>
                    <gf:ITEM>4</gf:ITEM>
                </gf:ITEMS>
```

2.7. Step 6: XSLT for JSON Transformation

Created a JSON database for good_food, capturing the same information of the XML databases, slightly transformed for good_food.
Written an XSLT ordersdata_xsl_json.xsl stylesheet enabling the creation of a jSON Output_jsonTrans.json for “orders” database from an XML file good_food.xml GOOD_FOOD database.

Created a JSON schema for JSON file.

The task involves transforming data from an XML database (specifically, the `good_food.xml` file) into a JSON format. The goal is to create a `Output_jsonTrans.json` file containing information from the GOOD_FOOD database, with slight modifications.


```
{ "orders": [ { "id": "1", "customer_ordered": { "c": "3", "order_date": "2023-01-01", "status": "Ready for delivery", "items": [ { "m": "2", "quantity": "7" } , { "m": "5", "quantity": "3" } ] } } , { "id": "2",
"customer_ordered": { "c": "1", "order_date": "2023-01-02", "status": "Ready for delivery", "items": [ { "m": "1", "quantity": "8" } ] } } , { "id": "3", "customer_ordered": { "c": "5", "order_date": "2023-01-03", "status":
"Ready for delivery", "items": [ { "m": "3", "quantity": "5" } ] } } , { "id": "4", "customer_ordered": { "c": "2", "order_date": "2023-01-04", "status": "Ready for delivery", "items": [ { "m": "2", "quantity": "7" } ] } } , {
"id": "5", "customer_ordered": { "c": "4", "order_date": "2023-01-05", "status": "Ready for delivery", "items": [ { "m": "1", "quantity": "10" } ] } } , { "id": "6", "customer_ordered": { "c": "7", "order_date": "2023-01-06",
"status": "Ready for delivery", "items": [ { "m": "3", "quantity": "4" } ] } } , { "id": "7", "customer_ordered": { "c": "1", "order_date": "2023-01-07", "status": "Ready for delivery", "items": [ { "m": "4", "quantity": "3" } ]
} } , { "id": "8", "customer_ordered": { "c": "3", "order_date": "2023-01-08", "status": "Cancelled", "items": [ { "m": "2", "quantity": "8" } ] } } , { "id": "9", "customer_ordered": { "c": "6", "order_date": "2023-01-09",
"status": "Ready for delivery", "items": [ { "m": "5", "quantity": "6" } ] } } , { "id": "10", "customer_ordered": { "c": "2", "order_date": "2023-01-10", "status": "Ready for delivery", "items": [ { "m": "2", "quantity": "7" } ]
} } , { "id": "11", "customer_ordered": { "c": "5", "order_date": "2023-01-11", "status": "Ready for delivery", "items": [ { "m": "3", "quantity": "5" } ] } } , { "id": "12", "customer_ordered": { "c": "4", "order_date": "2023-
01-12", "status": "Ready for delivery", "items": [ { "m": "1", "quantity": "8" } ] } } , { "id": "13", "customer_ordered": { "c": "7", "order_date": "2023-01-13", "status": "Ready for delivery", "items": [ { "m": "3",
"quantity": "4" } ] } } , { "id": "14", "customer_ordered": { "c": "1", "order_date": "2023-01-14", "status": "Ready for delivery", "items": [ { "m": "4", "quantity": "3" } ] } } , { "id": "15", "customer_ordered": { "c": "3",
"order_date": "2023-01-15", "status": "Ready for delivery", "items": [ { "m": "2", "quantity": "7" } , { "m": "1", "quantity": "2" } , { "m": "4", "quantity": "9" } ] } } , { "id": "16", "customer_ordered": { "c": "6",
"order_date": "2023-01-16", "status": "Ready for delivery", "items": [ { "m": "1", "quantity": "10" } ] } } , { "id": "17", "customer_ordered": { "c": "2", "order_date": "2023-01-17", "status": "Cancelled", "items": [ { "m":
"2", "quantity": "7" } ] } } , { "id": "18", "customer_ordered": { "c": "5", "order_date": "2023-01-18", "status": "Ready for delivery", "items": [ { "m": "3", "quantity": "5" } ] } } , { "id": "19", "customer_ordered": { "c":
"4", "order_date": "2023-01-19", "status": "Ready for delivery", "items": [ { "m": "5", "quantity": "6" } ] } } , { "id": "20", "customer_ordered": { "c": "7", "order_date": "2023-01-20", "status": "Ready for delivery",
"items": [ { "m": "1", "quantity": "8" } ] } } , { "id": "21", "customer_ordered": { "c": "1", "order_date": "2023-01-21", "status": "Ready for delivery", "items": [ { "m": "3", "quantity": "5" } ] } } , { "id": "22",
"customer_ordered": { "c": "3", "order_date": "2023-01-22", "status": "Ready for delivery", "items": [ { "m": "4", "quantity": "3" } ] } } , { "id": "23", "customer_ordered": { "c": "6", "order_date": "2023-01-23", "status":
"Ready for delivery", "items": [ { "m": "2", "quantity": "7" } ] } } , { "id": "24", "customer_ordered": { "c": "2", "order_date": "2023-01-24", "status": "Ready for delivery", "items": [ { "m": "2", "quantity": "7" } ] } } , {
"id": "25", "customer_ordered": { "c": "5", "order_date": "2023-01-25", "status": "Ready for delivery", "items": [ { "m": "1", "quantity": "10" } ] } } , { "id": "26", "customer_ordered": { "c": "4", "order_date": "2023-01-
26", "status": "Ready for delivery", "items": [ { "m": "3", "quantity": "5" } ] } } , { "id": "27", "customer_ordered": { "c": "7", "order_date": "2023-01-27", "status": "Ready for delivery", "items": [ { "m": "2", "quantity":
"7" } ] } } , { "id": "28", "customer_ordered": { "c": "1", "order_date": "2023-01-28", "status": "Ready for delivery", "items": [ { "m": "4", "quantity": "3" } ] } } , { "id": "29", "customer_ordered": { "c": "3", "order_date":
"2023-01-29", "status": "Ready for delivery", "items": [ { "m": "1", "quantity": "10" } ] } } , { "id": "30", "customer_ordered": { "c": "6", "order_date": "2023-01-30", "status": "Paid", "items": [ { "m": "3", "quantity": "5"
} ] } } , { "id": "31", "customer_ordered": { "c": "2", "order_date": "2023-01-31", "status": "Paid", "items": [ { "m": "5", "quantity": "6" } ] } } , { "id": "32", "customer_ordered": { "c": "5", "order_date": "2023-01-01",
"status": "Ready for delivery", "items": [ { "m": "4", "quantity": "3" } ] } } , { "id": "33", "customer_ordered": { "c": "4", "order_date": "2023-01-02", "status": "Ready for delivery", "items": [ { "m": "3", "quantity": "5" } ]
} } , { "id": "34", "customer_ordered": { "c": "7", "order_date": "2023-01-03", "status": "Ready for delivery", "items": [ { "m": "2", "quantity": "7" } ] } } , { "id": "35", "customer_ordered": { "c": "1", "order_date": "2023-
01-04", "status": "Ready for delivery", "items": [ { "m": "1", "quantity": "8" } ] } } , { "id": "36", "customer_ordered": { "c": "3", "order_date": "2023-01-05", "status": "Ready for delivery", "items": [ { "m": "2",
"quantity": "7" } , { "m": "4", "quantity": "10" } ] } } , { "id": "37", "customer_ordered": { "c": "6", "order_date": "2023-01-06", "status": "Ready for delivery", "items": [ { "m": "3", "quantity": "5" } ] } } , { "id": "38",
"customer_ordered": { "c": "2", "order_date": "2023-01-07", "status": "Ready for delivery", "items": [ { "m": "4", "quantity": "3" } ] } } , { "id": "39", "customer_ordered": { "c": "5", "order_date": "2023-01-08", "status":
"Ready for delivery", "items": [ { "m": "5", "quantity": "6" } ] } } , { "id": "40", "customer_ordered": { "c": "4", "order_date": "2023-01-09", "status": "Ready for delivery", "items": [ { "m": "1", "quantity": "10" } ] } } , {
"id": "41", "customer_ordered": { "c": "7", "order_date": "2023-01-10", "status": "Ready for delivery", "items": [ { "m": "2", "quantity": "7" } ] } } , { "id": "42", "customer_ordered": { "c": "1", "order_date": "2023-01-
11", "status": "Ready for delivery", "items": [ { "m": "3", "quantity": "5" } ] } } , { "id": "43", "customer_ordered": { "c": "3", "order_date": "2023-01-12", "status": "Cancelled", "items": [ { "m": "4", "quantity": "3" } ] } }
, { "id": "44", "customer_ordered": { "c": "6", "order_date": "2023-01-13", "status": "Ready for delivery", "items": [ { "m": "5", "quantity": "6" } ] } } , { "id": "45", "customer_ordered": { "c": "2", "order_date": "2023-01-
14", "status": "Ready for delivery", "items": [ { "m": "1", "quantity": "8" } ] } } , { "id": "46", "customer_ordered": { "c": "5", "order_date": "2023-01-15", "status": "Ready for delivery", "items": [ { "m": "3", "quantity":
"5" } ] } } , { "id": "47", "customer_ordered": { "c": "4", "order_date": "2023-01-16", "status": "Ready for delivery", "items": [ { "m": "2", "quantity": "7" } , { "m": "1", "quantity": "1" } , { "m": "3", "quantity": "5" } , {
"m": "4", "quantity": "8" } ] } } , { "id": "48", "customer_ordered": { "c": "7", "order_date": "2023-01-17", "status": "Ready for delivery", "items": [ { "m": "2", "quantity": "7" } ] } } , { "id": "49", "customer_ordered": {
"c": "1", "order_date": "2023-01-18", "status": "Cancelled", "items": [ { "m": "1", "quantity": "10" } ] } } , { "id": "50", "customer_ordered": { "c": "3", "order_date": "2023-01-19", "status": "Ready for delivery", "items": [
{ "m": "3", "quantity": "5" } ] } } ] }
```


Annex 1: The contribution of each members

##	**Tasks**	**Responsible persons**
1	XML Structure Definition	All team members under leading of Advendio
2	XML Development	Advendio
3	XSD Creation	Advendio
4	05 XSLT for HTML Transformation	02 XSLT: Candy 02 XSLT: Anna 01 XSLT: Advendio
5	XSLT for XML Transformation	Vaishnavi
6	XSLT for JSON Transformation	Vaishnavi
7	Report	All