**Description**

**Objective:**  
To work with service to share data between components. Implement two-way binding using ngModel.

**Scenario:**  
HayDay Bakes is a newly started bakery in the city. They need an web application to extend their service. You are assigned with the task to develop an application using angular. You are provided with AppComponent, ProductsComponent, OrderComponent, Order model class and OrderService class. The requirement is to add the selected product items in ProductsComponent to orderList in Addservice class. Then in OrderComponent, you must display the selected products and calculate the corresponding cost. The Order model class is already provided as part of the code skeleton.

**Class : Order**

|  |  |
| --- | --- |
| **Attribute** | **Type** |
| productName | string |
| quantity | number |

**OrderService class Specification:**

|  |  |  |
| --- | --- | --- |
| **Class** | **Attributes** | **Methods** |
| OrderService | orderList : Array<any> = [] (To store the order objects) | addItem( productName : string, quantity : number )  This method should add the productName and quantity to the order class object, then add this order object into the orderList array. |
| calculateBill() This method should calculate the total bill by iterating the orderList and return the total bill.  Products price table   |  |  | | --- | --- | | **Product Name** | **Cost (in Rs)** | | Cream | 100 | | Butter | 60 | | Cheese | 60 | | Panner | 120 | | Milk powder | 150 | | Bread | 30 | | Cookies | 60 | | Muffin | 20 | | Rolls | 40 | | Pies | 50 |   **Formula** : Total = cost \* quantity |
| getOrder() This method is invoked from OrderComponent TS file . It should return the orderList which is used to store the selected products. |

**AppComponent**  
Refer the screenshot 1 and design the page with the below UI specification.

|  |  |  |
| --- | --- | --- |
| **Element Name and Id** | **Control Type** | **RouterLink** |
| products | Hyper link | products |
| order | Hyper link | order |

While there's click-on the products link, it should route to **ProductsComponent** and while there's click-on order link, it should route to **OrderComponent**.

**Note**: Routing path should be defined in app.module.ts file

**Screenshot :1**

A screenshot of a computer

AI-generated content may be incorrect.  
**Products Component**  
Refer the screenshot 2 and 3 to design the page with the below UI specification.  
Refer the screenshot and design the **ProductsComponent** UI with hyperlinks - **"Baking Products", "Dairy Products"**.  
While click-on the hyperlinks, ProductsComponent should dynamically render the UI for Dairy Products or Baking Products.

**Note:**Use QueryParams with a flag to dynamically render the UI.   
If the flag is 1, then ProductsComponent UI should display the Dairy Products Page.   
If the flag is 2, then display Baking Products page.

**Note:**Router Link path should be "products" and the queryParam key/values for each link is shown in the table below

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Label** | **Element Name and Id** | **Control Type** | **RuoterLink** | **Query Params : Key** | **Query Params: Value** |
| Dairy Products | dairy | Hyper link | products | flag | 1 |
| Baking Products | baking | Hyper link | products | flag | 2 |

**Design constraints**  
Inside Dairy Products menu and Baking Products menu

* Create the heading using h3 element (Refer the screenshot)
* Create a table.
* In table, each row should contain a **label, input, button** respectively
* Create button using button tag. Each button should invoke the **addItem** function in **ProductsComponent TS file** with arguments product name and quantity (which will be entered inside the input element. **For example :**addItem('Cream', dairy1.value)).
* Within the each button tag, add the symbol "**+**"**.**
* Refer the below table for label and input tag

**For Dairy Products - Refer screenshot 2**

|  |  |  |
| --- | --- | --- |
| **Label** | **Element Id / Name** | **Control Type** |
| Cream | dairy1 | number |
| Butter | dairy2 | number |
| Cheese | dairy3 | number |
| Panner | dairy4 | number |
| Milk powder | dairy5 | number |
|  | Each item should contain a button with ids "btn1", "btn2", "btn3", "btn4", "btn5" respectively | button |

**For Baking Products - Refer screenshot 3**

|  |  |  |
| --- | --- | --- |
| **Label** | **Element Id / Name** | **Control Type** |
| Bread | baking1 | number |
| Cookies | baking2 | number |
| Muffin | baking3 | number |
| Rolls | baking4 | number |
| Pies | baking5 | number |
|  | Each item should contain a button with ids "btn1", "btn2", "btn3", "btn4", "btn5" respectively | button |

**Component Specification:**

|  |  |  |
| --- | --- | --- |
| **Class** | **Attributes** | **Methods** |
| ProductsComponent |  | **addItem(productName : string, quantity : any)**  This method should be invoked by the Add (+) button click event. Invoke the OrderService class addItem() method by passing productName and quantity, to store in the orderList array. |

**Screenshot 2 : Products component UI to select Dairy Products**

A screenshot of a web page

AI-generated content may be incorrect.

**Screenshot 3 : Products component UI to select Baking Products**

A screenshot of a website

AI-generated content may be incorrect.

**OrderComponent - HTML file**  
**Design Constraints**  
-       Create **a table**  
-       In table, each row should contain a **label, input / button** respectively.

Design constraints for Order Details - Refer Screenshot4

|  |  |  |
| --- | --- | --- |
| **Label** | **Element Id / Name** | **Control Type** |
| Customer Name | name | text |
| Phone Number | phone | tel |
| Calculate Bill | calculate | button |

* The submit button should invoke the calculateBill function in OrderComponent TS file.
* Create a table using id "**displayTable**" with thead and tbody.
* In thead, create two data as Product Name, Quantity as per the screenshot.
* In tbody iterate the order array in TS file and display the product details.
* Create a p tag to display the status.
* For submitting empty inputs the error message should be displayed in the <p> tag with id as **"error"**

**Component Specification:**

|  |  |  |
| --- | --- | --- |
| **Class** | **Attributes** | **Methods** |
| OrderComponent | **status**  **error : string**  (To hold the error messages)  **customerName:string**  (To hold the customer name input value)  **phoneNumber:string**  (To hold the phone number input value)  **order:Array<any>**  (To hold the orderList returned from getOrder) | **calculateBill()**  This method should be invoked by the calculate button click in OrderComponent. Then invoke the validation function for input validation. If all inputs are valid, then invoke the OrderService class calculateBill() method and get the bill amount.  Then update the status attributes as **"Please pay Rs. <bill>"**.  Else catch the error and append it to the error variable. |
| **getOrder()**  This method should be invoked by ngOnInit(). Through this method, invoke the service class getOrder() method, Get the returned orderList and and assign it to the order array. |
| **validation()**  This method should check whether input elements have values or not. If any of the input field is empty throw an error with the message.  Kindly refer the below table for the error messages. Check the validation for input elements which are mentioned in the table. |

**Error message for invalid inputs :**

|  |  |
| --- | --- |
| **Input Field** | **Error Message** |
| Customer Name | Provide value for customer name |
| Phone Number | Provide value for phone number |

**Note:**

* The error message should be displayed as per the order.

**Screenshot 4: Input validation**

**A screenshot of a computer

AI-generated content may be incorrect.  
Screenshot 5 :**  
A screenshot of a computer

AI-generated content may be incorrect.