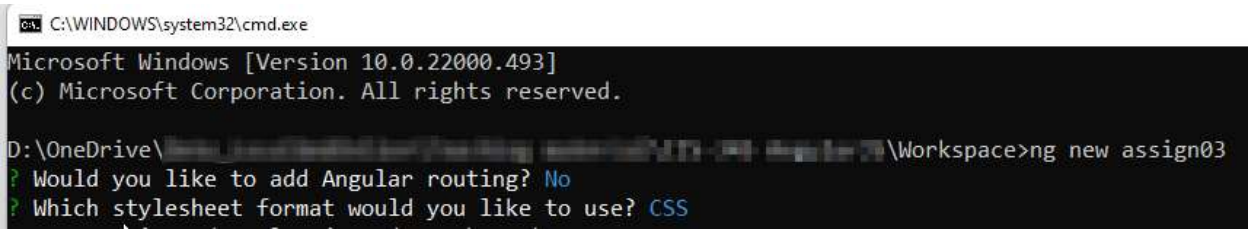
Assign 03

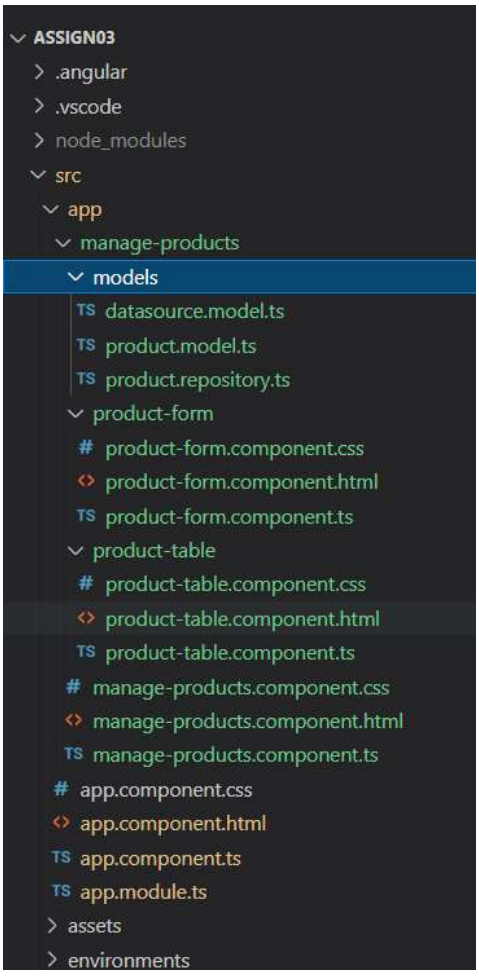
Due date and time:03/16/2022 11:59 pm Total points:7

In this assignment, you will use template driven approach to create and submit a form and then data binding using @input, @output decorators between parent and child components to pass data around.

1. Create an angular project “assign03” using angular-cli.



2. You will create folder structure as shown below:



3. Components that you need to create manage-products, product-form, product-table.
    a. product-form component – will create form part of the view.
    b. product-table component – will create table view.
    c. Both components will be added to the manage-products component.
    d. manage-products component will be added to the root component.
4. In the manage-products component, create a folder named “models” with 3 files in it as shown below.
5. datasource.model.ts – will provide a static data source which we will use just once to fetch data to populate product array in product.repository.ts class.

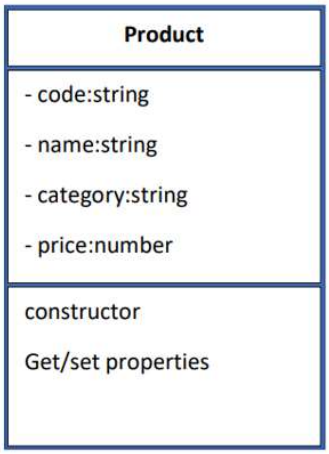
```
import { Product } from "./product.model";

export class SimpleDataSource {
    private _data: Product[];

    constructor() {
        this._data = new Array<Product>(
            new Product('100','Cap','Winter wear',200),
      new Product('200','Jacket','Winter wear',1000),
      new Product('300','Coat','Winter wear',2090),
      new Product('400','Gloves','Winter wear',100),
      new Product('500','Guitar','Musical instruments',350)
     )
    }

    getData(): Product[] {
        return this._data;
    }
}
```

6. product.model.ts uses following model



7. **product.repository.ts**

   This class is responsible for manipulation of the product model, basically CRUD. You will copy data to product [] by using static data source. You will need at least these two fields:

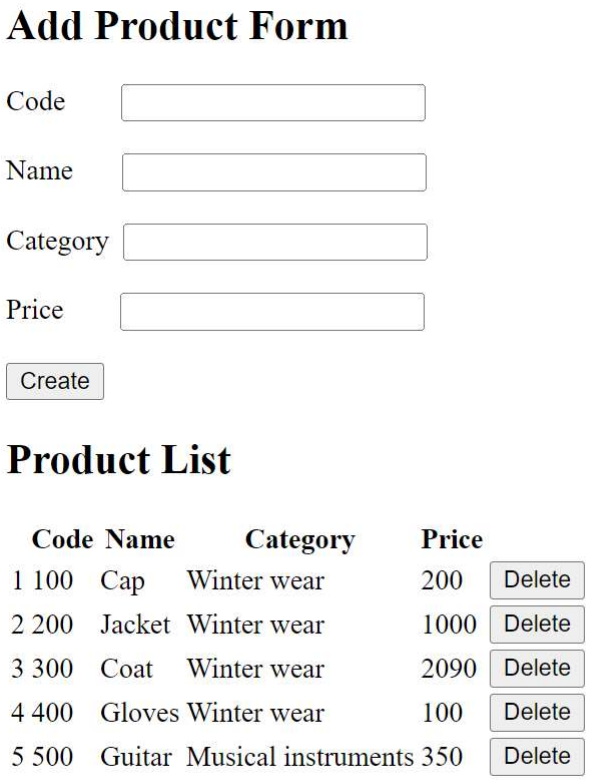
   private dataSource: SimpleDataSource;

   private products: Product[];

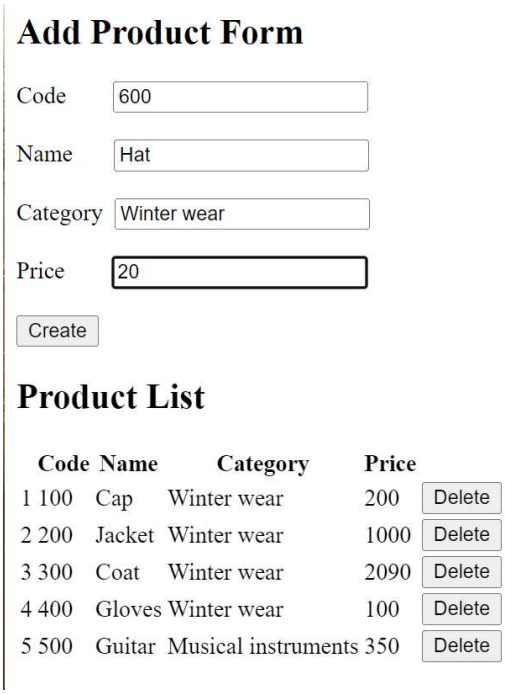
   In addition to other methods which you might have, make sure you have at least getProducts, deleteProduct, and addProduct methods in this class.

Expected output:

1. **Project, components, files, and folders are created and implemented as instructed. 1 point**
2. **When viewed in browser with proper nesting of components – 1 point**



3. If all four values are provided in the form, then only add that product; otherwise reset the form. Once a product is created, then update the table as shown below. Use ngForm on form control, ngModel directive on child controls of the form, use ngSubmit to submit the form as learned in the class. 2 points



Add Product Form

Code

Name

Category

Price

Create

Product List

Code	Name	Category	Price	
1	100	Cap	Winter wear	200	Delete
2	200	Jacket	Winter wear	1000	Delete
3	300	Coat	Winter wear	2090	Delete
4	400	Gloves	Winter wear	100	Delete
5	500	Guitar	Musical instruments	350	Delete
6	600	Hat	Winter wear	20	Delete

4. Delete functionality – below two products are deleted, and table is updated. **1 point**

Add Product Form

Code

Name

Category

Price

Create

Product List

Code	Name	Category	Price	
1	100	Cap	Winter wear	200	Delete
2	200	Jacket	Winter wear	1000	Delete
3	400	Gloves	Winter wear	100	Delete
4	500	Guitar	Musical instruments	350	Delete

You will be using @input and @output decorators we learned in class to make this parent child interaction happen. When a product is added in the product-form component, emit an event to inform parent component and then parent should pass down the updated repository model to child component product-table to update the table view. 2 points

Submission:

1. Submit zip file of “src” folder of your angular project.