HOL-#05

Q1. Your challenge is to create a nested component Angular application.

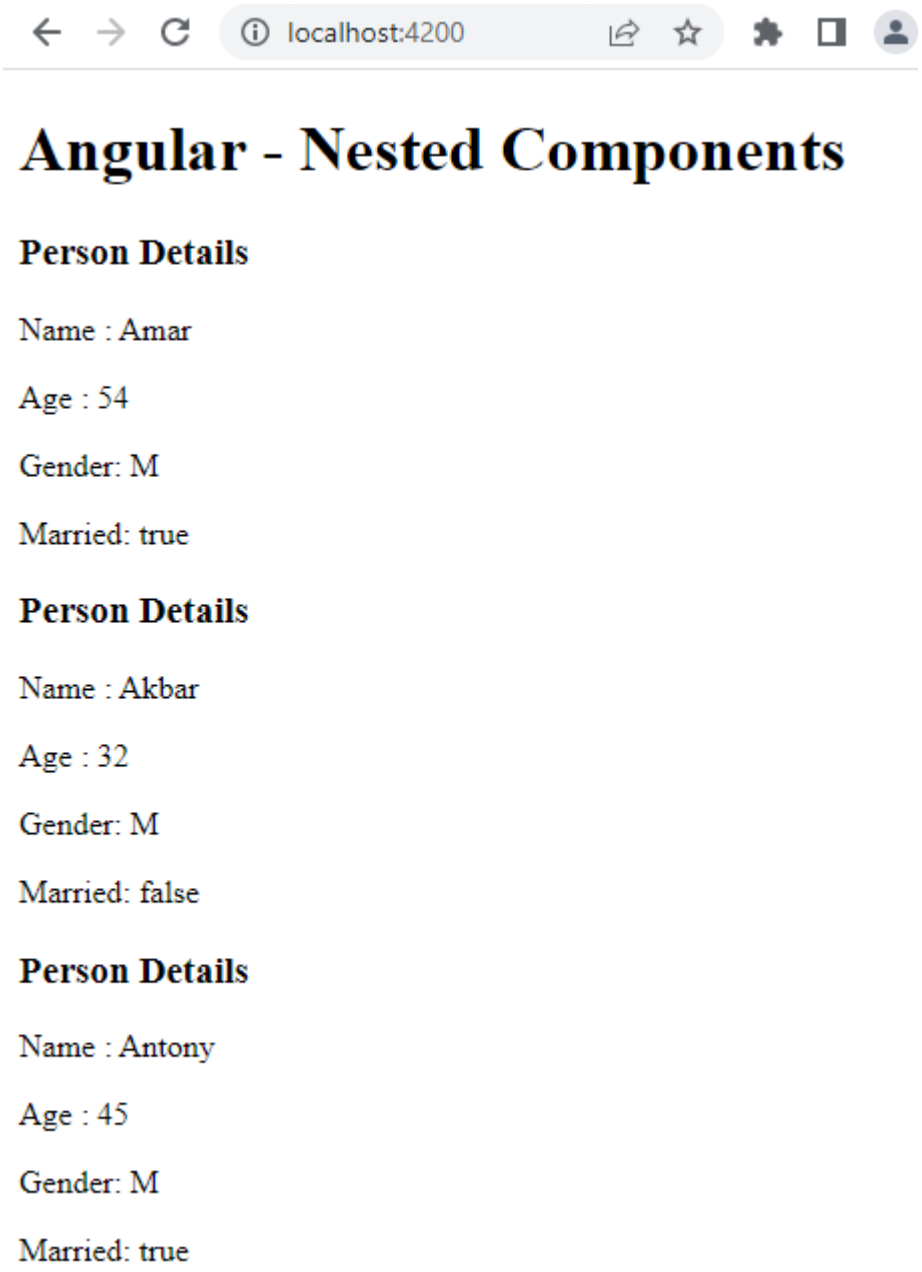
- The child component should be called from the parent component.

- The child component should display the user information which exits in the parent component.

- Note, that the parent component also get's the users data from the `**persons.ts**` file.

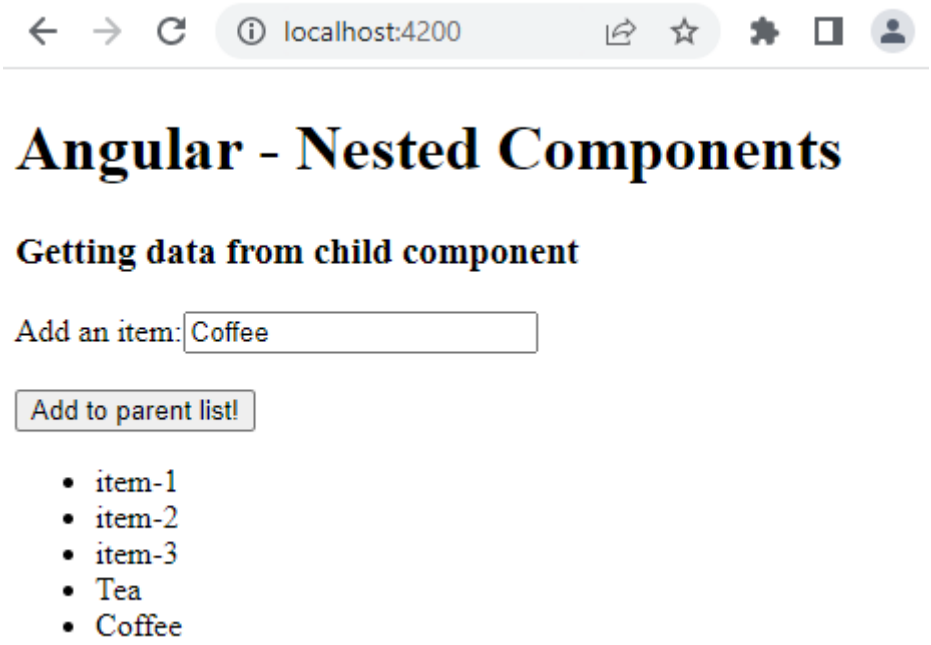
- Look into the `**persons.ts**` file and record your observation in terms of the `**Interface**` and the records it has.

When you execute your app, your web-browser screen should look as shown below:



Q2. Your next challenge is to ensure that the data which is entered in the child component is used to update the `items` array in the parent component.

You web-browser should look as shown below:

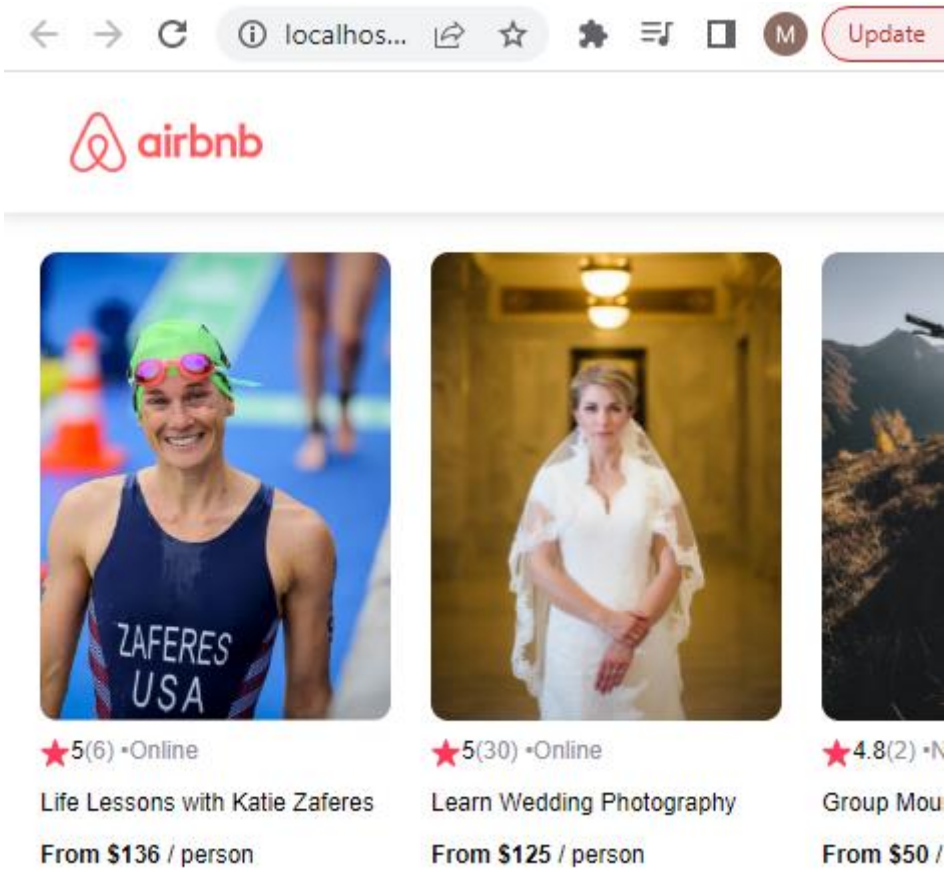


On entering the data in the textbox and clicking the button `**Add to parent list!**` will add the item to the `items` list and it gets rendered as shown above.

Q3. Now, since you are aware of how we can share data between the parent and the child components, you should be able to modify the `**airbnb**` application.

It should fetch the data from the `**cardData.ts**` file and pass it to the card component to display the cards.

On completing the `airbnb` application, you web-browser should look as shown below:



Q4. Your next challenge is to create the `**utility**` service which has methods as follows:

[a] A method `sysdate()` which return's the system date (only)

[b] A method `address()` which return's your company's address

Inject the `**utility**` service in your components and check if you are able to uses the methods.

Accordingly render them on the template to display their behaviour.

Q5. As Angular already provides us with the **HTTP** service, let's use this service and get the `posts` information from the following URL

https://jsonplaceholder.typicode.com/posts/1

On executing the app, you web-browser screen should look as shown below:

Q6. From the `**jsonplaceholder.typicode.com**` web-site we would like to get all the `**users**` data.

The URL is as follows:
https://jsonplaceholder.typicode.com/users

Using HTTP service, get the data and display the same in your web-browser as shown below:

Q7. Back to the `eComm` application.

- Execute a command to create a new child component by name 'product-alerts'

- Observe your VS Code project explorer should be displaying the following files:

    - product-alerts.component.ts
    - product-alerts.component.html
    - product-alerts.component.css

Q8. Your next challenge!

As the product alerts component needs product details, setup to receive the product data.

NOTE: It may not require the data, but the implementation of the data is what is required here.

- Now in the product alert component class, define a property name ‘product’ with an @Input decorator

NOTE: The @Input decorator indicates that the property value passes in from the component’s parent.

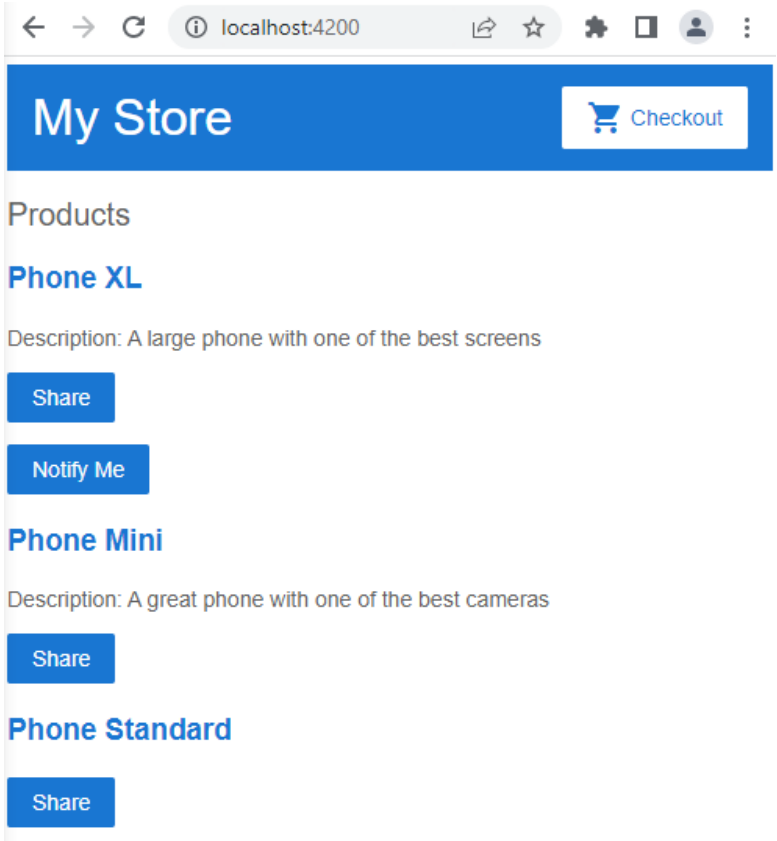
- Open the product alert component and have the ‘Notify Me’ button which appears if the product price is over $700

- Now do display product alerts component as a child of product list component add the <app-product-alerts> element in the product list component html file.

- Make sure the product alerts component takes a product as input from the product list.

- Thus the ‘Notify Me’ button will be rendered based on the product price

You web-browser should look as shown below:



NOTE: The 'Notify Me' button is not yet functional.

Q9. To make the Notify Me button work, the child component needs to notify and pass the data to the parent component.

The ProductAlertsComponent needs to emit an event when the user clicks Notify Me and
the ProductListComponent needs to respond to the event.

- In the product alerts component import 'Output' and 'EventEmitter' from @angular/core

- In the component class, define a property name 'notify' with the @Output() decorator and an instance of 'EventEmitter'

- Next in the product alerts HTML update the 'Notify Me' button with event binding to call the 'notify.emit()' method.

- Now, define the behaviour when the user click's the

'Notify Me' button in the product list component.

In other word, have a handle with deals with the notify emitted event.

Define the 'onNotify()' method which display the text:

'You will be NOTIFIED when the product goes on SALE!'

- Update the product list component HTML and bind the 'onNotify()' method to the 'notify' event which was emitted in the child component.

- Click the 'Notify Me' button now.
  Record your observation.

On click the `Notify Me` button the web-browser should look as shown below: