1. Components.

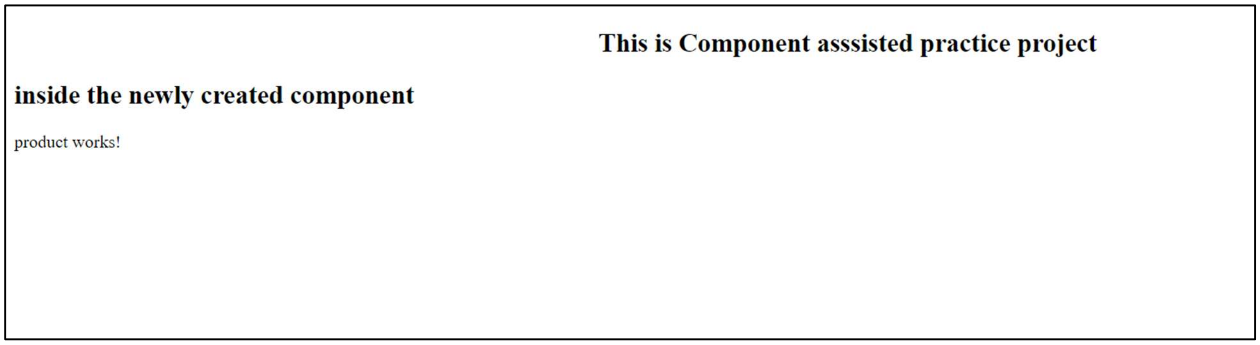
App.component.html

```html
<h2 style="text-align: center;">This is Component asssisted practice project</h2>
<product></product>
```

product.component.html

```html
<h2 >inside the newly created component</h2>
<p>product works!</p>
```

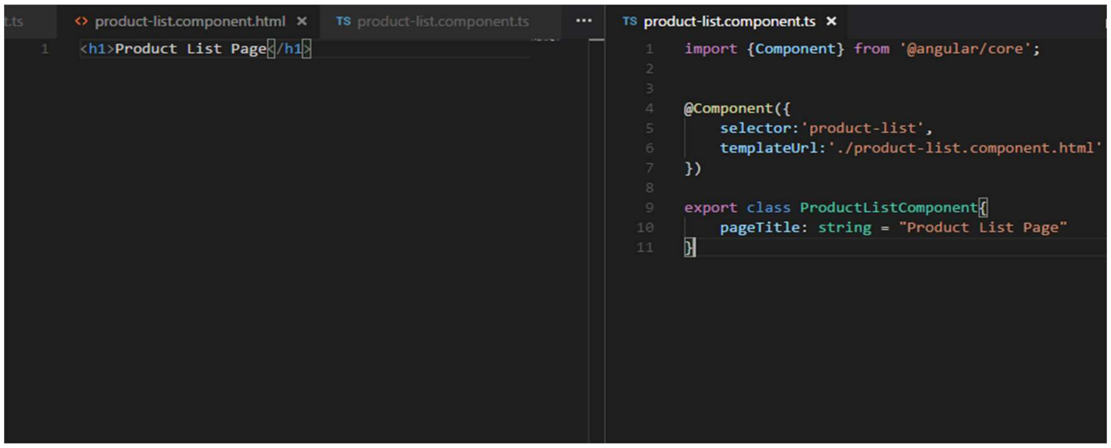
output



```
This is Component asssisted practice project
inside the newly created component
product works!
```

2. Property Binding.

- Create a folder called *products* inside the *src/app* folder.
- Create a *product-list.component.html* file in the *products* folder.
- Add the following code to *product-list.component.html*.
    - *<h1> Product List Page </h1>*
- Create a file called *product-list.component.ts*.

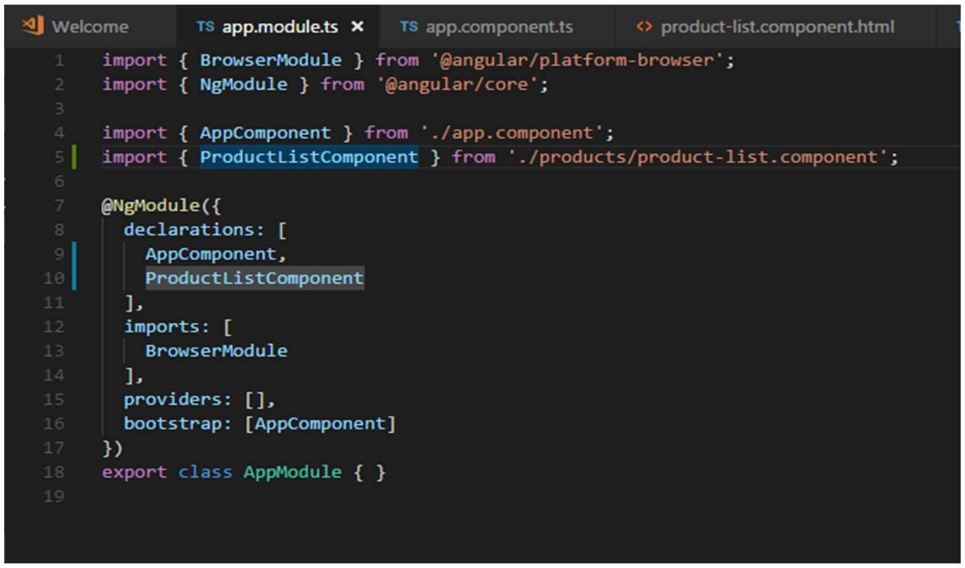


product-list.component.html
product-list.component.ts
product-list.component.ts

```
<h1>Product List Page</h1>
```

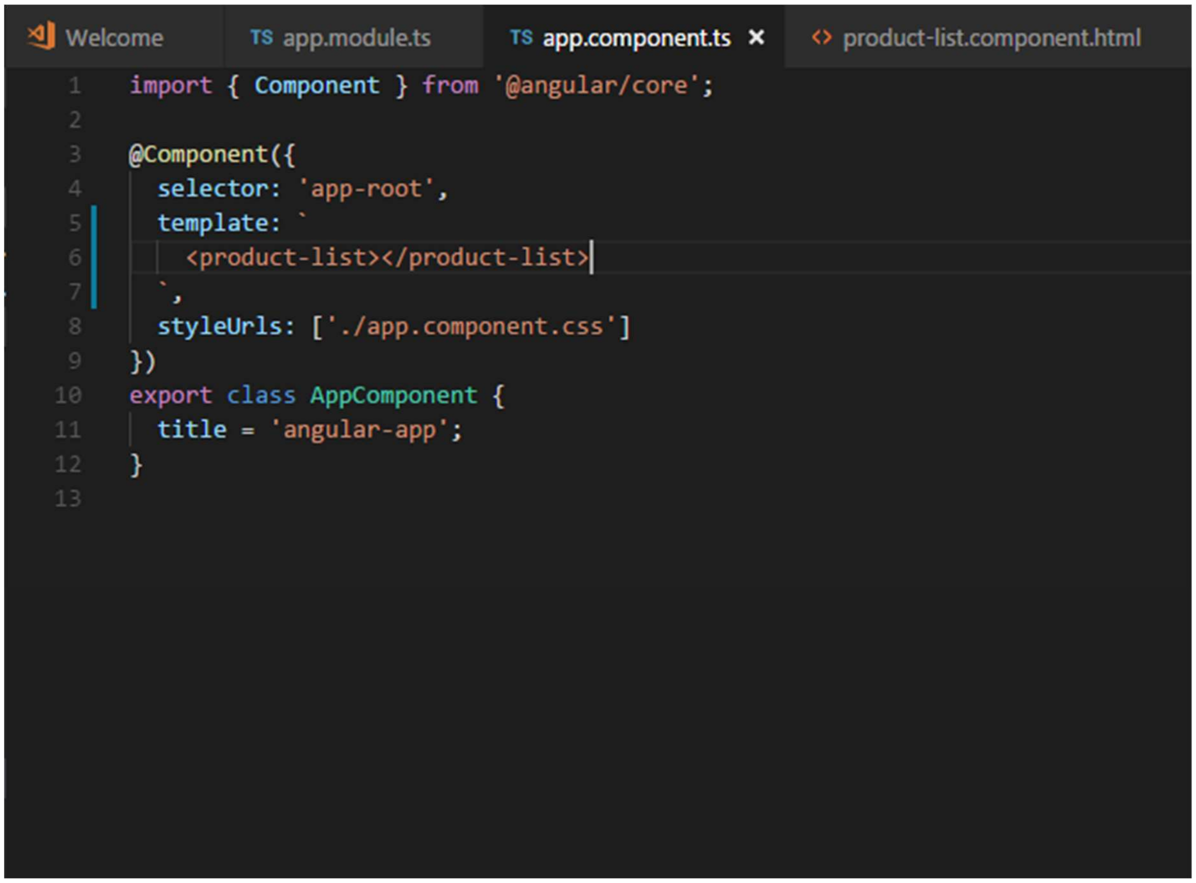
```
import {Component} from '@angular/core';


@Component({
    selector:'product-list',
    templateUrl:'./product-list.component.html'
})

export class ProductListComponent{
    pageTitle: string = "Product List Page"
}
```

- Add the following code in *app.module.ts*.



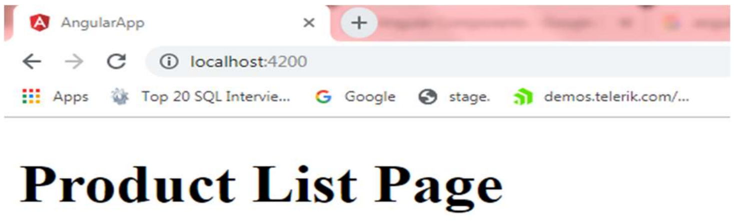
```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';
import { ProductListComponent } from './products/product-list.component';

@NgModule({
  declarations: [
    AppComponent,
    ProductListComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

- Add the following code in app.component.ts.



```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: `
    <product-list></product-list>
  `,
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'angular-app';
}

```

Output



Product List Page

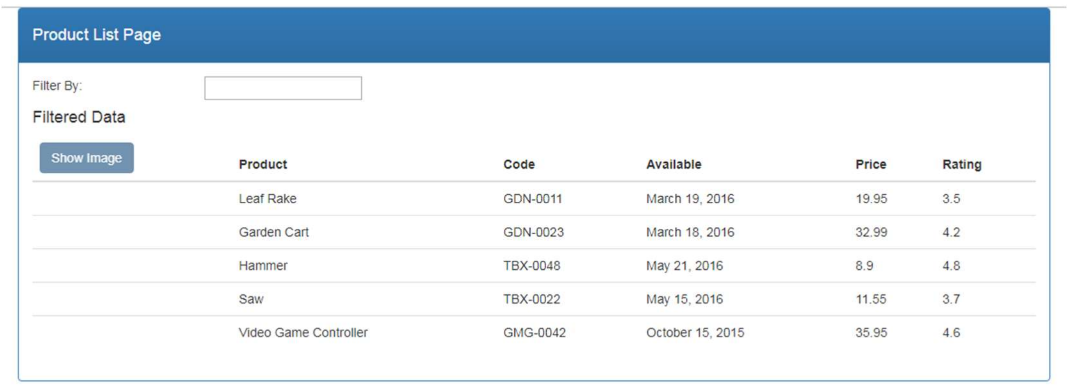
3. Class and Style Bindings.

Disabling a button using attribute binding

- Disable a button using *disabled* attribute of the element. Set the *disabled* to *false* by binding value to *attr.disabled* attribute property.
- Open Visual Studio Code
- Add the following code to *Product-list.component.html*.

```
  <button [attr.disabled]=”true” class="btn btn-primary"
(click)='toggleImage()'>

      Show Image

  </button>
```

Output:



Product	Code	Available	Price	Rating
Leaf Rake	GDN-0011	March 19, 2016	19.95	3.5
Garden Cart	GDN-0023	March 18, 2016	32.99	4.2
Hammer	TBX-0048	May 21, 2016	8.9	4.8
Saw	TBX-0022	May 15, 2016	11.55	3.7
Video Game Controller	GMG-0042	October 15, 2015	35.95	4.6

Implementing class binding

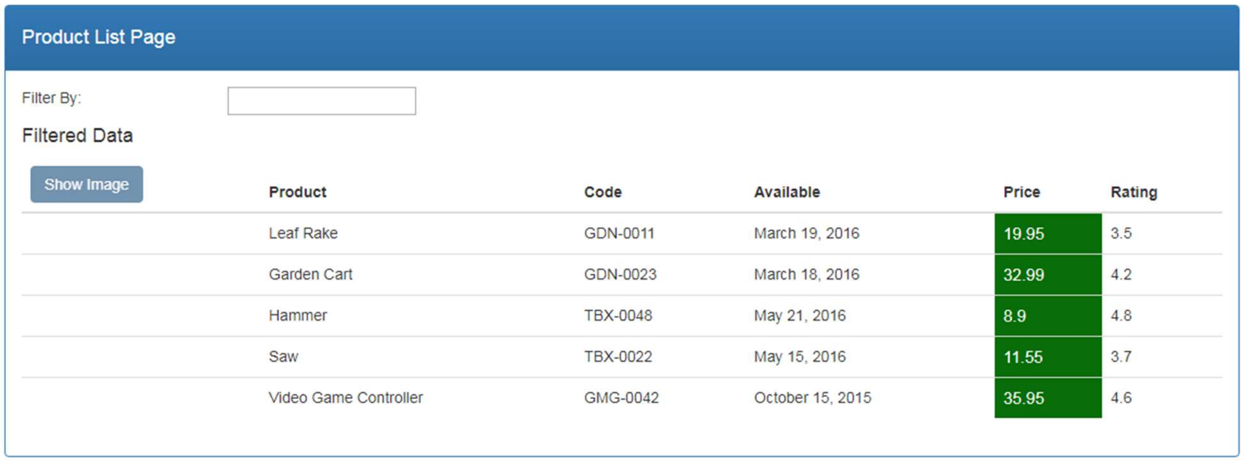
- Create an external CSS file *product-list.component.css* inside *products* folder.
- Create a CSS class name *price*.

```
.inStock{

   background: #096d09;

   color:#ffff;

   Font-size:15px;

}
```

- Apply the CSS class in *Product-list.component.html* file and set its value to *true*.

```
 <td [class.inStock]="true">
    {{ product.price}}
</td>
```

Output:



Product List Page

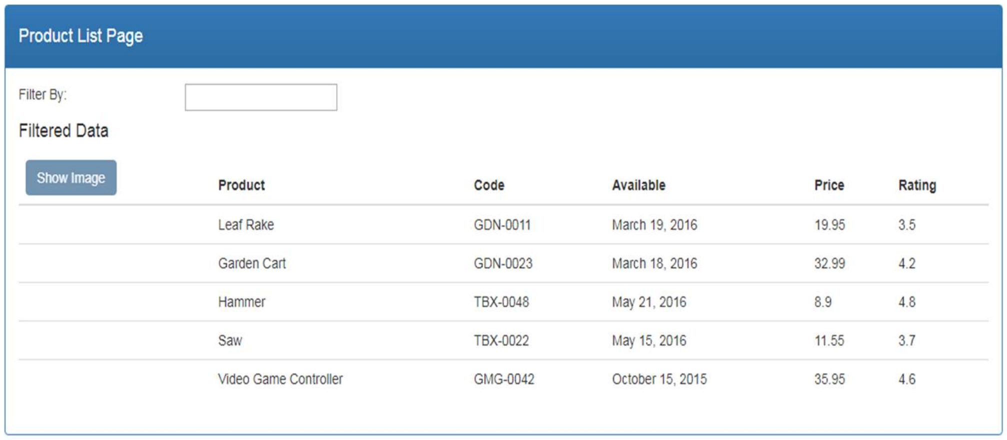
Filter By:

Filtered Data

Product	Code	Available	Price	Rating
Leaf Rake	GDN-0011	March 19, 2016	19.95	3.5
Garden Cart	GDN-0023	March 18, 2016	32.99	4.2
Hammer	TBX-0048	May 21, 2016	8.9	4.8
Saw	TBX-0022	May 15, 2016	11.55	3.7
Video Game Controller	GMG-0042	October 15, 2015	35.95	4.6

- Set class value to *false* in *Product-list.component.html*.

Output:



Product List Page

Filter By:

Filtered Data

Product	Code	Available	Price	Rating
Leaf Rake	GDN-0011	March 19, 2016	19.95	3.5
Garden Cart	GDN-0023	March 18, 2016	32.99	4.2
Hammer	TBX-0048	May 21, 2016	8.9	4.8
Saw	TBX-0022	May 15, 2016	11.55	3.7
Video Game Controller	GMG-0042	October 15, 2015	35.95	4.6

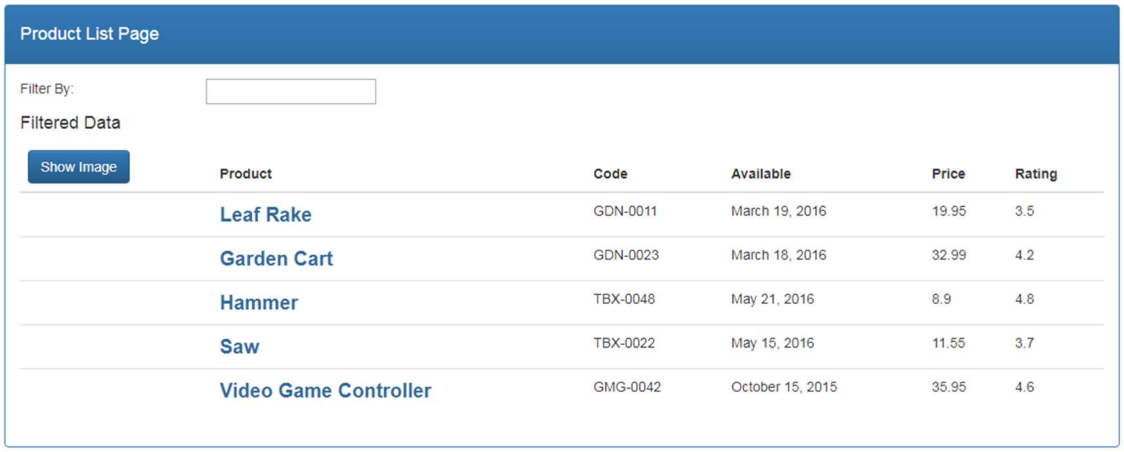
Implementing style binding

- Bind a color and font-weight style to the product element in *Product-list.component.html* file.

```
  <td [style.color]="'#306A9D'" [style.font-weight]="700">
```

```
      {{ product.productName }}

  </td>
```

Output:



Product List Page

Filter By:

Filtered Data

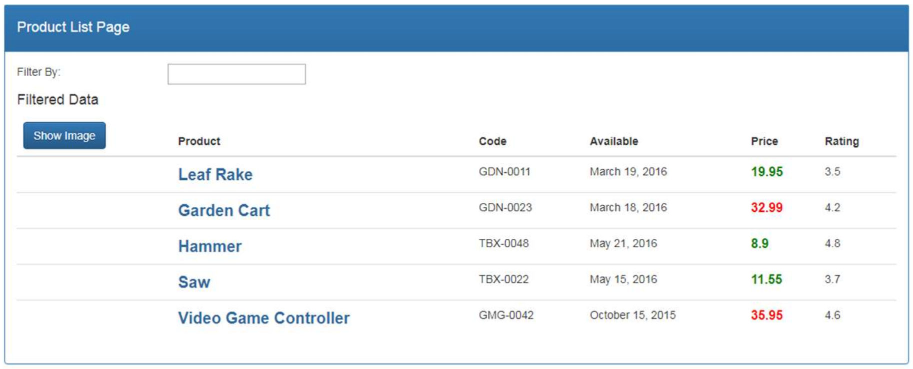
Show Image

	Product	Code	Available	Price	Rating
	Leaf Rake	GDN-0011	March 19, 2016	19.95	3.5
	Garden Cart	GDN-0023	March 18, 2016	32.99	4.2
	Hammer	TBX-0048	May 21, 2016	8.9	4.8
	Saw	TBX-0022	May 15, 2016	11.55	3.7
	Video Game Controller	GMG-0042	October 15, 2015	35.95	4.6

- You can also apply conditional CSS using style binding.

```
 <td [style.color]="product.price > 20 ? 'red': 'green'"
[style.font-size.px]="16"  [style.font-weight]="700">

       {{ product.price}}

 </td>
```

Output:



Product List Page

Filter By:

Filtered Data

Show Image

	Product	Code	Available	Price	Rating
	Leaf Rake	GDN-0011	March 19, 2016	19.95	3.5
	Garden Cart	GDN-0023	March 18, 2016	32.99	4.2
	Hammer	TBX-0048	May 21, 2016	8.9	4.8
	Saw	TBX-0022	May 15, 2016	11.55	3.7
	Video Game Controller	GMG-0042	October 15, 2015	35.95	4.6

4. Event Binding.

: Implementing event binding

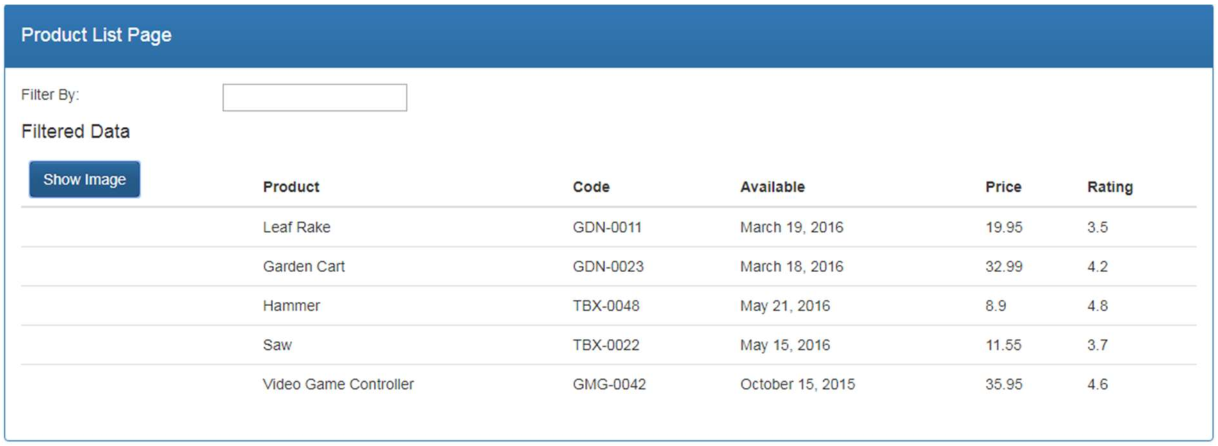
- Open Visual Studio Code
- Call *toggleImage()* when any button is clicked in *Product-list.component.html*.
- Add the following code in *Product-list.component.ts*.

```
export class ProductListComponent{

  pageTitle: string = "Product List Page";

  imageWidth:number = 80;

  imageMargin:number = 10;


  showImage:boolean = false;


  toggleImage() : void {

     this.showImage = !this.showImage;

     // (!false = true) // (!true == false)

         console.log('Value of ShowImage inside function ::',
this.showImage);

  }

}
```

- Add the following code in *Product-list.component.html.*

```
<button class="btn btn-primary" (click)='toggleImage()'>
   Show Image
</button>
```

Output:

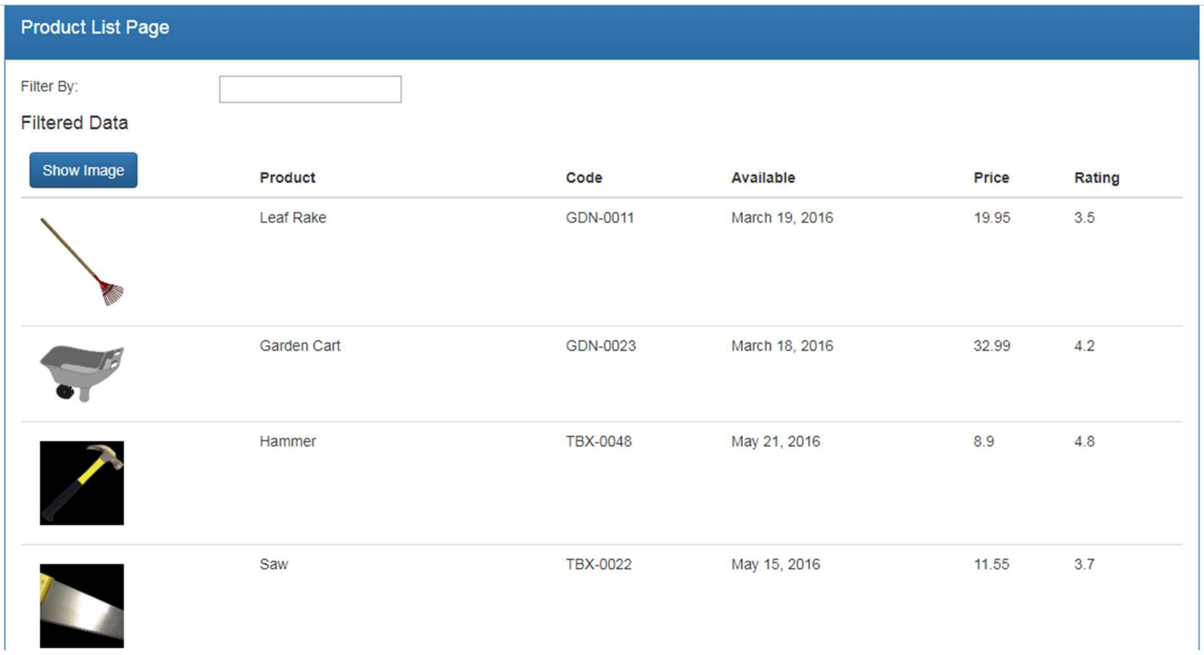


Product List Page

Filter By:

Filtered Data

Show Image	Product	Code	Available	Price	Rating
	Leaf Rake	GDN-0011	March 19, 2016	19.95	3.5
	Garden Cart	GDN-0023	March 18, 2016	32.99	4.2
	Hammer	TBX-0048	May 21, 2016	8.9	4.8
	Saw	TBX-0022	May 15, 2016	11.55	3.7
	Video Game Controller	GMG-0042	October 15, 2015	35.95	4.6



Product List Page

Filter By:

Filtered Data

Show Image	Product	Code	Available	Price	Rating
	Leaf Rake	GDN-0011	March 19, 2016	19.95	3.5
	Garden Cart	GDN-0023	March 18, 2016	32.99	4.2
	Hammer	TBX-0048	May 21, 2016	8.9	4.8
	Saw	TBX-0022	May 15, 2016	11.55	3.7

5. Two-way Binding.

Creating parent and child component

- Open Visual Studio Code
- Navigate to your project folder
- Run the below command to create a child component as your app component will be acting as a parent component
  ng g c child

Step 3.5.3: Transferring data from parent to child component and vice versa

- Add below code in app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { ChildComponent } from './child/child.component';

@NgModule({
 declarations: [
  AppComponent,
  ChildComponent
 ],
 imports: [
  BrowserModule,
  AppRoutingModule
 ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

- Add below code in **app.component.ts**

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
    public cdata: string;
}
```

- Add below code in **app.component.html**

```
<h2>Parent Component</h2>
This is Parent Component<br>
Enter Text:
<input type="text" #ptext (keyup)="0"/><br>
The value of Child component is: {{cdata}}
```

```
<app-child (cevent)="cdata=$event" [pdata]="ptext.value"></app-child>
```

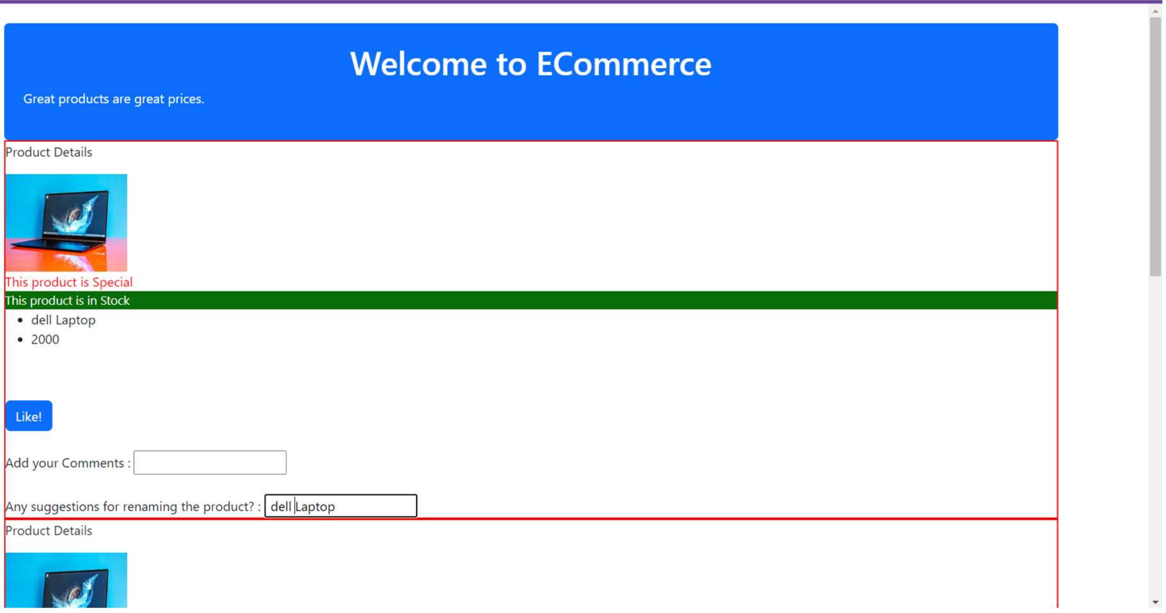
- Add below code in child.component.html

```
<h2>Child Component</h2>

This is Child Component<br>

Enter Text:

<input type="text" #cdata (keyup)="onChange(cdata.value)"/><br>

The value od Parent component is: {{pdata}}
```

- Add below code in child.component.ts

```
import { Component, OnInit, Input, EventEmitter } from '@angular/core';


@Component({
  selector: 'app-child',
  templateUrl: './child.component.html',
  styleUrls: ['./child.component.css'],
  inputs: [`pdata`],
  outputs: [`cevent`]
})
export class ChildComponent implements OnInit {


  constructor() { }


  ngOnInit() {
 }


 public pdata: string;
 cevent= new EventEmitter<string>();


 onChange(value:string){
   this.cevent.emit(value);
 }
}
```

Output



Welcome to ECommerce

Great products are great prices.

Product Details

This product is Special

This product is in Stock

- dell Laptop
- 2000

Add your Comments :

Any suggestions for renaming the product? : dell Laptop

Product Details

6. Form Validations.

App.component.html

```
<!DOCTYPE html>

<html>

<head>

  <meta charset="utf-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <title>Page Title</title>

</head>

<body style="padding:40px ">

<p style="text-align: center;font-size: 30px">  Hello <input type="text"
placeholder="Your name" (input)="ontyping($event)"/>, Welcome to the
<b>angCare!</b></p>

<p style="text-align: center;font-size: 20px">Hello {{name}}<br>I am Joe, your
personal assistant! I will guide you further...</p>

<div style="text-align:left;padding: 20px">

<p> Click on Sign up to create your account with angCare: <button (click)="signup()"
class="btn btn-primary">Sign up with {{title}}</button>   : {{status}}</p>

<p> Click on Sign up to create your account with angCare: <button>Sign In
</button></p>

</div>

<div >

<div class="jumbotron">

  <div class="container">

      <div class="row">

          <div class="col-md-6 offset-md-3">
```

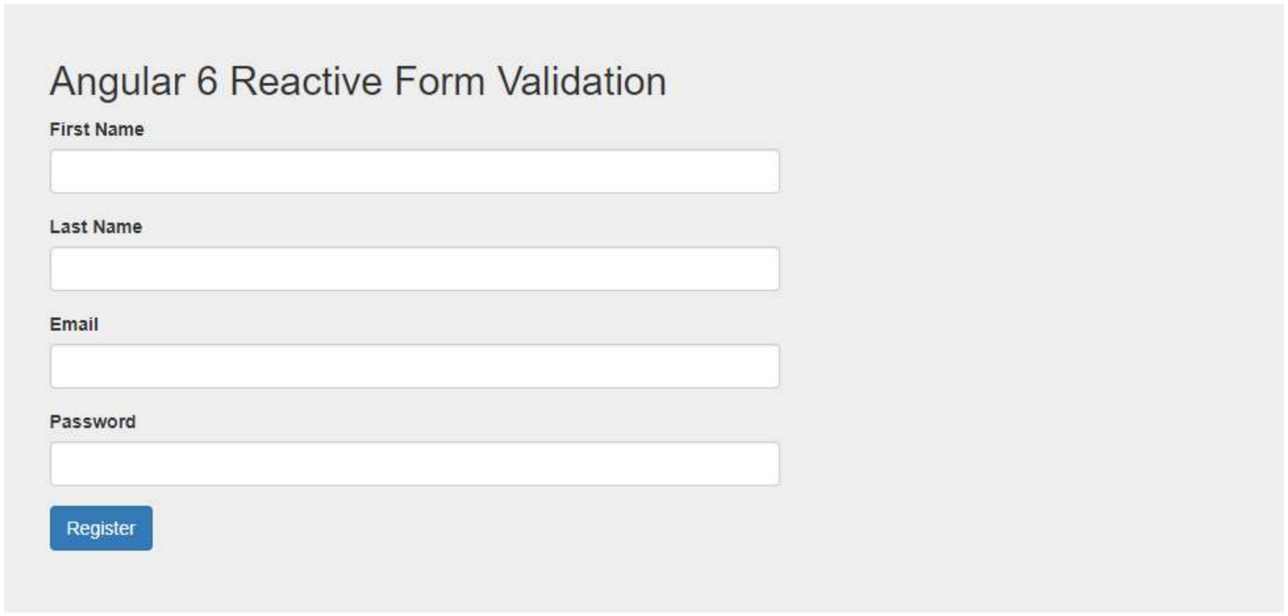
```
    <h2>Angular 6 Reactive Form Validation</h2>

    <form [formGroup]="registerForm" (ngSubmit)="onSubmit()">

      <div class="form-group">

        <label>First Name</label>

        <input type="text" formControlName="firstName" class="form-control"
[ngClass]="{ 'is-invalid': submitted && f.firstName.errors }" />

        <div *ngIf="submitted && f.firstName.errors" class="invalid-feedback">

          <div *ngIf="f.firstName.errors.required">First Name is
required</div>

        </div>

      </div>

      <div class="form-group">

        <label>Last Name</label>

        <input type="text" formControlName="lastName" class="form-control"
[ngClass]="{ 'is-invalid': submitted && f.lastName.errors }" />

        <div *ngIf="submitted && f.lastName.errors" class="invalid-feedback">

          <div *ngIf="f.lastName.errors.required">Last Name is required</div>

        </div>

      </div>

      <div class="form-group">

        <label>Email</label>

        <input type="text" formControlName="email" class="form-control"
[ngClass]="{ 'is-invalid': submitted && f.email.errors }" />

        <div *ngIf="submitted && f.email.errors" class="invalid-feedback">

          <div *ngIf="f.email.errors.required">Email is required</div>

          <div *ngIf="f.email.errors.email">Email must be a valid email
address</div>
```

```html
            </div>
          </div>
          <div class="form-group">
            <label>Password</label>
            <input type="password" formControlName="password" class="form-control" [ngClass]="{ 'is-invalid': submitted && f.password.errors }" />
            <div *ngIf="submitted && f.password.errors" class="invalid-feedback">
              <div *ngIf="f.password.errors.required">Password is required</div>
              <div *ngIf="f.password.errors.minlength">Password must be at least 6 characters</div>
            </div>
          </div>
          <div class="form-group">
            <button [disabled]="loading" class="btn btn-primary">Register</button>
          </div>
        </form>
      </div>
    </div>
  </div>
</div>
</div>
</body>
</html>
<router-outlet></router-outlet>
```

app.component.ts

```
import { Component,OnInit } from '@angular/core';
import {FormBuilder, FormGroup, Validators } from '@angular/forms';
import { Title } from '@angular/platform-browser';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {

  constructor(private formBuilder: FormBuilder) { }
  title = 'angCare';
  status = 'You haven\'t signed up yet';
  name = '';
  submitted = false;
  registerForm: FormGroup;
  ontyping(event:Event) {

    this.name = (<HTMLInputElement>event.target).value;
  }
  signup() {

    this.status = 'Oops! We are working on it!';

  }

  ngOnInit() {
    this.registerForm = this.formBuilder.group({
        firstName: ['', Validators.required],
        lastName: ['', Validators.required],
        email: ['', [Validators.required, Validators.email]],
        password: ['', [Validators.required, Validators.minLength(6)]]
    });
}

get f() { return this.registerForm.controls; }

onSubmit() {
    this.submitted = true;
    // stop here if form is invalid
    if (this.registerForm.invalid) {
        return;
    }
    alert('Your request has been submitted for approval')
}
}
```

Output


Angular 6 Reactive Form Validation
First Name
Last Name
Email
Password
Register

7.Directives

```
// Required services for custom directives
import { Directive, ElementRef, Renderer2 } from '@angular/core';

@Directive({
  selector: '[appChangeColor]' // Directive selector
})

export class ChangeColorDirective {

  constructor(elem: ElementRef, renderer: Renderer2) {
    renderer.setStyle(elem.nativeElement, 'color', 'olive');
  }

}
```

Step 3.7.2: Declaring the directive

- Declare the directive in declaration array.

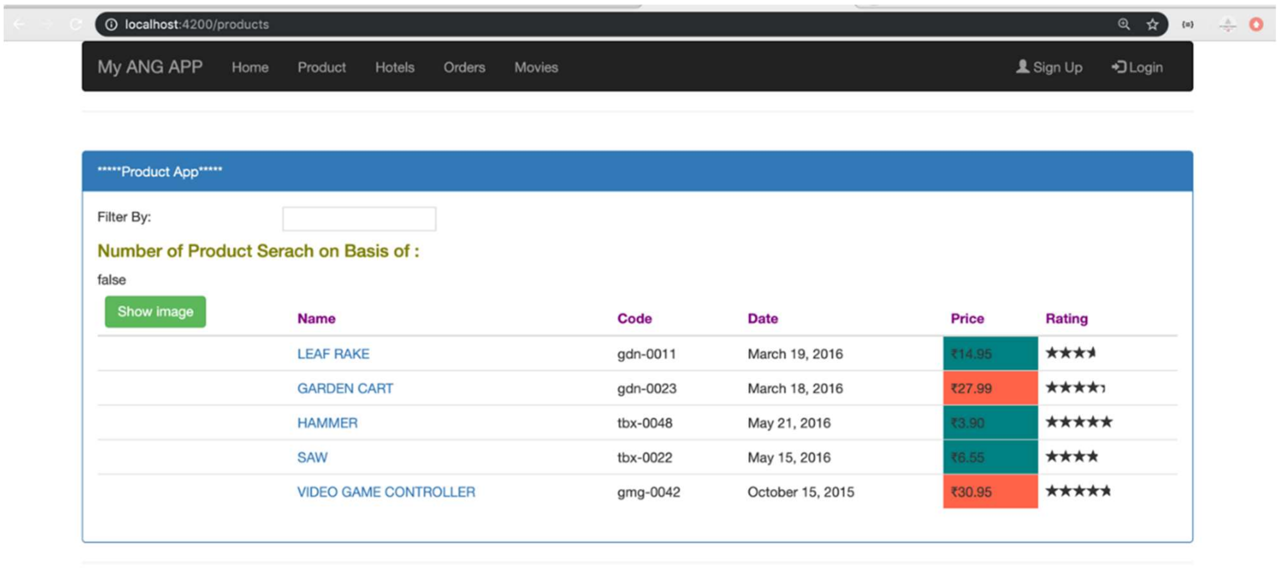
```
import { ChangeColorDirective } from './ChangeColor.directive';

@NgModule({
    imports: [
    SharedModule,
    AppRoutingModule
  ],
  declarations: [
    ChangeColorDirective,
    ProductComponent,
    MyUpperPipe,
    DiscountPipe,
    ProductSearch,
    ProductDetailComponent
  ],
```

Step 3.7.3: Adding the directive as a property

- Add the created directive as a property.

```
<h4 appChangeColor>Number of Product Serach on Basis of {{userInput}}:</h4>
```

Output:



localhost:4200/products

*****Product App*****

Filter By:

Number of Product Serach on Basis of :

false

	Name	Code	Date	Price	Rating
	LEAF RAKE	gdn-0011	March 19, 2016	₹14.95	★★★★
	GARDEN CART	gdn-0023	March 18, 2016	₹27.99	★★★★★
	HAMMER	tbx-0048	May 21, 2016	₹3.90	★★★★★
	SAW	tbx-0022	May 15, 2016	₹6.55	★★★★
	VIDEO GAME CONTROLLER	gmg-0042	October 15, 2015	₹30.95	★★★★★

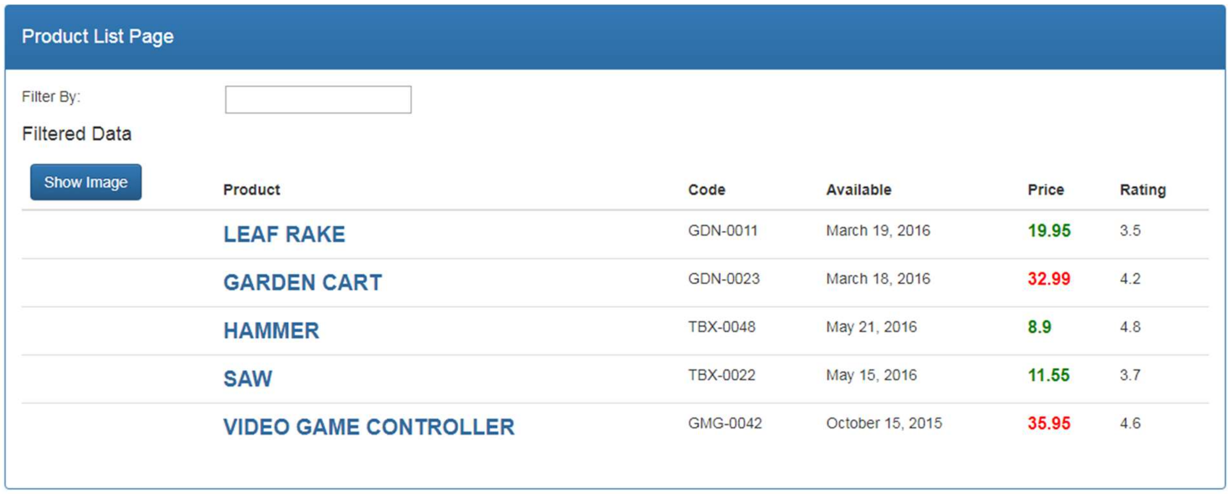
8. Pipes

Using built-in Angular pipes

- Open Visual Studio Code
- Set product name to uppercase using in-built pipe in *Product-list.component.html*.

```
<td [style.color]="'#306A9D'" [style.font-weight]="700" [style.font-size.px]="20">

  {{ product.productName | uppercase }}

</td>
```

Output:



Product	Code	Available	Price	Rating
LEAF RAKE	GDN-0011	March 19, 2016	19.95	3.5
GARDEN CART	GDN-0023	March 18, 2016	32.99	4.2
HAMMER	TBX-0048	May 21, 2016	8.9	4.8
SAW	TBX-0022	May 15, 2016	11.55	3.7
VIDEO GAME CONTROLLER	GMG-0042	October 15, 2015	35.95	4.6

Using custom Angular pipes

- Create a new file called *convert-to-spaces.pipe.ts*.

```
import { Pipe, PipeTransform } from "@angular/core";
```

```
@Pipe({

  name: 'convertToSpaces'

})

export class ConvertToSpacesPipe implements PipeTransform{

  transform(value:string, character:string, ) {

      return value.replace(character, '@');

  }

}
```

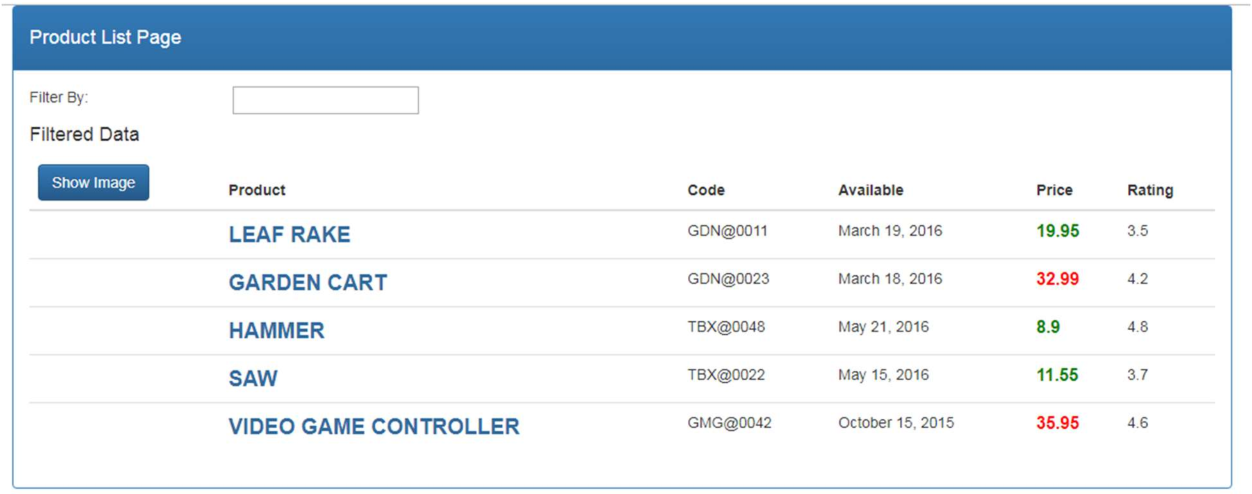
- Import *convert-to-spaces.pipe.ts* to *app.module.ts* and declare it inside declaration array.

```
import { ConvertToSpacesPipe } from 'src/app/products/convert-to-spaces.pipe';



declarations: [

  AppComponent,

  ProductListComponent,

  ConvertToSpacesPipe

 ]
```

- Add the following code in *Product-list.component.html*.

```

<td>{{ product.productCode | convertToSpaces:'-'}}</td>
```

Output:



Product List Page

Filter By:

Filtered Data

Product	Code	Available	Price	Rating
LEAF RAKE	GDN@0011	March 19, 2016	19.95	3.5
GARDEN CART	GDN@0023	March 18, 2016	32.99	4.2
HAMMER	TBX@0048	May 21, 2016	8.9	4.8
SAW	TBX@0022	May 15, 2016	11.55	3.7
VIDEO GAME CONTROLLER	GMG@0042	October 15, 2015	35.95	4.6

9. Routing Mechanisms.

- Open the app.component.html available in the app folder of the Angular application. Replace all the code available in the file to the source code mentioned below:

```
<!DOCTYPE html>

<html>

<head>

  <meta charset="utf-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <title>Page Title</title>

</head>

<body style="padding:40px ">

<p style="text-align: center;font-size: 30px">  Hello <input type="text"
placeholder="Your name" (input)="ontyping($event)"/>, Welcome to the
<b>angCare!</b></p>

<p style="text-align: center;font-size: 20px">Hello {{name}}<br>I am Joe, your
personal assistant! I will guide you further...</p>

<div style="text-align:left;padding: 20px">

<p> Click on Sign up to create your account with angCare: <button (click)="signup()"
class="btn btn-primary" [routerLink]="'/signup'">Sign up with {{title}}</button>   :
{{status}}</p>

<p> Click on Sign up to create your account with angCare: <button>Sign In
</button></p>

</div>



</body>

</html>
```

```
<router-outlet></router-outlet>
```

Completing the functionality by implementing validators

- Open the **app.component.ts** file and replace only the source code available in **AppComponent** class with the following source code:

```
import { Component,OnInit } from '@angular/core';

import {FormBuilder, FormGroup, Validators } from '@angular/forms';

import { Title } from '@angular/platform-browser';


@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})


export class AppComponent {


  constructor(private formBuilder: FormBuilder) { }

  title = 'angCare';

  status = 'You haven\'t signed up yet';

  name = '';


  ontyping(event:Event) {
    this.name = (<HTMLInputElement>event.target).value;

  }

  signup() {


     this.status = 'Oops! We are working on it!';


  }

}
```

- Open the **app.module.ts** file and add the source code mentioned below:

```
import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import {RouterModule, Routes} from '@angular/router'

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

import { ReactiveFormsModule} from '@angular/forms'

import { SignupComponent } from './signup/signup.component';

import { SigninComponent } from './signin/signin.component';


const routes: Routes = [{


  path:'',

  component:AppComponent

},
{
 path:'signup',
 component:SignupComponent
},
{
 path:'sigin',
 component:SigninComponent
}

]

@NgModule({
 declarations: [
  AppComponent,
  SignupComponent,
  SigninComponent
 ],
 imports: [
  BrowserModule,
  AppRoutingModule,
  ReactiveFormsModule,
  RouterModule.forRoot(routes)
 ],
 providers: [],
  bootstrap: [AppComponent]

})

export class AppModule { }
```

- Open the **signup.component.html** available in the **app/signup** folder of the Angular application. Replace all the code available in the file with the source code mentioned below:

```
<p>

  signup works!

</p>


<div class="jumbotron">

  <div class="container">

     <div class="row">

        <div class="col-md-6 offset-md-3">

           <h2>Angular 6 Reactive Form Validation</h2>

           <form [formGroup]="registerForm" (ngSubmit)="onSubmit()">

              <div class="form-group">

                 <label>First Name</label>

                 <input type="text" formControlName="firstName" class="form-control"
[ngClass]="{ 'is-invalid': submitted && f.firstName.errors }" />

                 <div *ngIf="submitted && f.firstName.errors" class="invalid-feedback">

                    <div *ngIf="f.firstName.errors.required">First Name is
required</div>

                 </div>

              </div>
```

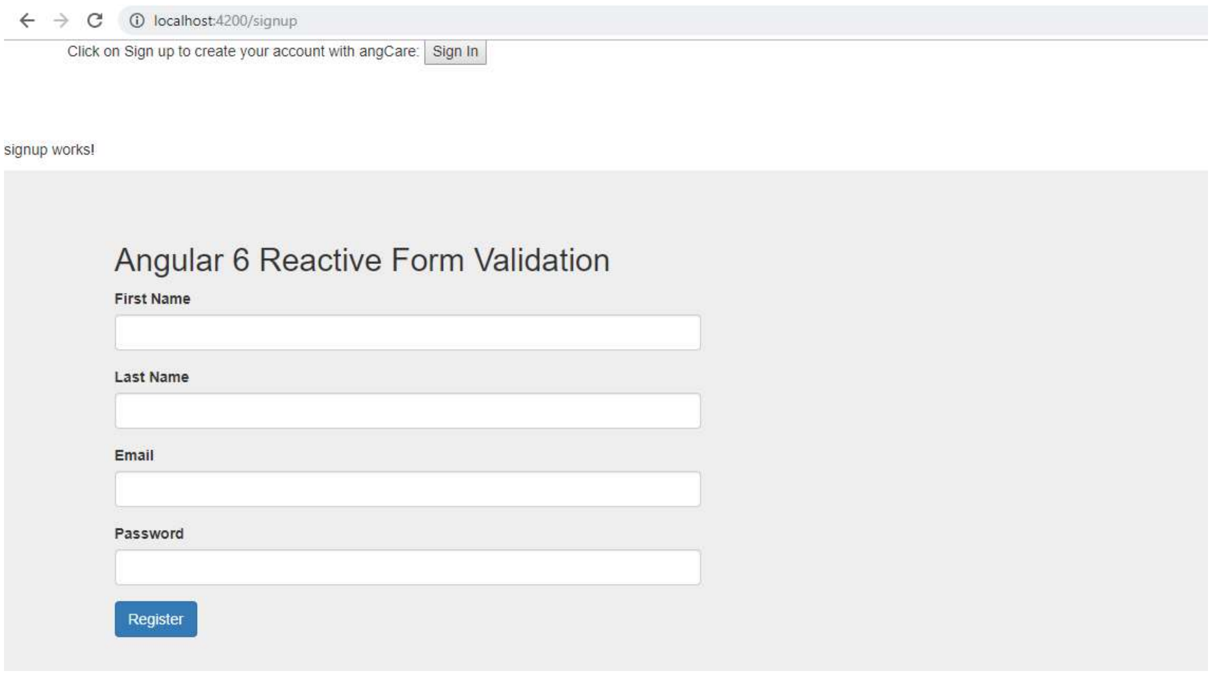
```
                <div class="form-group">
                    <label>Last Name</label>
                    <input type="text" formControlName="lastName" class="form-control" [ngClass]="{ 'is-invalid': submitted && f.lastName.errors }" />
                    <div *ngIf="submitted && f.lastName.errors" class="invalid-feedback">
                        <div *ngIf="f.lastName.errors.required">Last Name is required</div>
                    </div>
                </div>
                <div class="form-group">
                    <label>Email</label>
                    <input type="text" formControlName="email" class="form-control" [ngClass]="{ 'is-invalid': submitted && f.email.errors }" />
                    <div *ngIf="submitted && f.email.errors" class="invalid-feedback">
                        <div *ngIf="f.email.errors.required">Email is required</div>
                        <div *ngIf="f.email.errors.email">Email must be a valid email address</div>
                    </div>
                </div>
                <div class="form-group">
                    <label>Password</label>
                    <input type="password" formControlName="password" class="form-control" [ngClass]="{ 'is-invalid': submitted && f.password.errors }" />
                    <div *ngIf="submitted && f.password.errors" class="invalid-feedback">
                        <div *ngIf="f.password.errors.required">Password is required</div>
                        <div *ngIf="f.password.errors.minlength">Password must be at least 6 characters</div>
                </div>

              </div>

              <div class="form-group">

                <button [disabled]="loading" class="btn btn-
primary">Register</button>

              </div>

            </form>

          </div>

        </div>

      </div>

    </div>


<router-outlet></router-outlet>
```

- Open the **signup.component.ts** file and replace the source code available in **SignupComponent** class with the following source code:

```
import { Component, OnInit } from '@angular/core';

import {FormBuilder, FormGroup, Validators } from '@angular/forms';

@Component({

  selector: 'app-signup',

  templateUrl: './signup.component.html',

  styleUrls: ['./signup.component.css']

})

export class SignupComponent implements OnInit {

  submitted = false;

  registerForm: FormGroup;
```

```
 constructor(private formBuilder: FormBuilder) { }

 ngOnInit() {

  this.registerForm = this.formBuilder.group({

      firstName: ['', Validators.required],

      lastName: ['', Validators.required],

      email: ['', [Validators.required, Validators.email]],

      password: ['', [Validators.required, Validators.minLength(6)]]

  });

}

get f() { return this.registerForm.controls; }

onSubmit() {

 this.submitted = true;

 // stop here if form is invalid

 if (this.registerForm.invalid) {

     return;

 } alert('Your request has been submitted for approval')

}

}
```

Output


localhost:4200/signup
Click on Sign up to create your account with angCare:
Sign In
signup works!
Angular 6 Reactive Form Validation
First Name
Last Name
Email
Password
Register