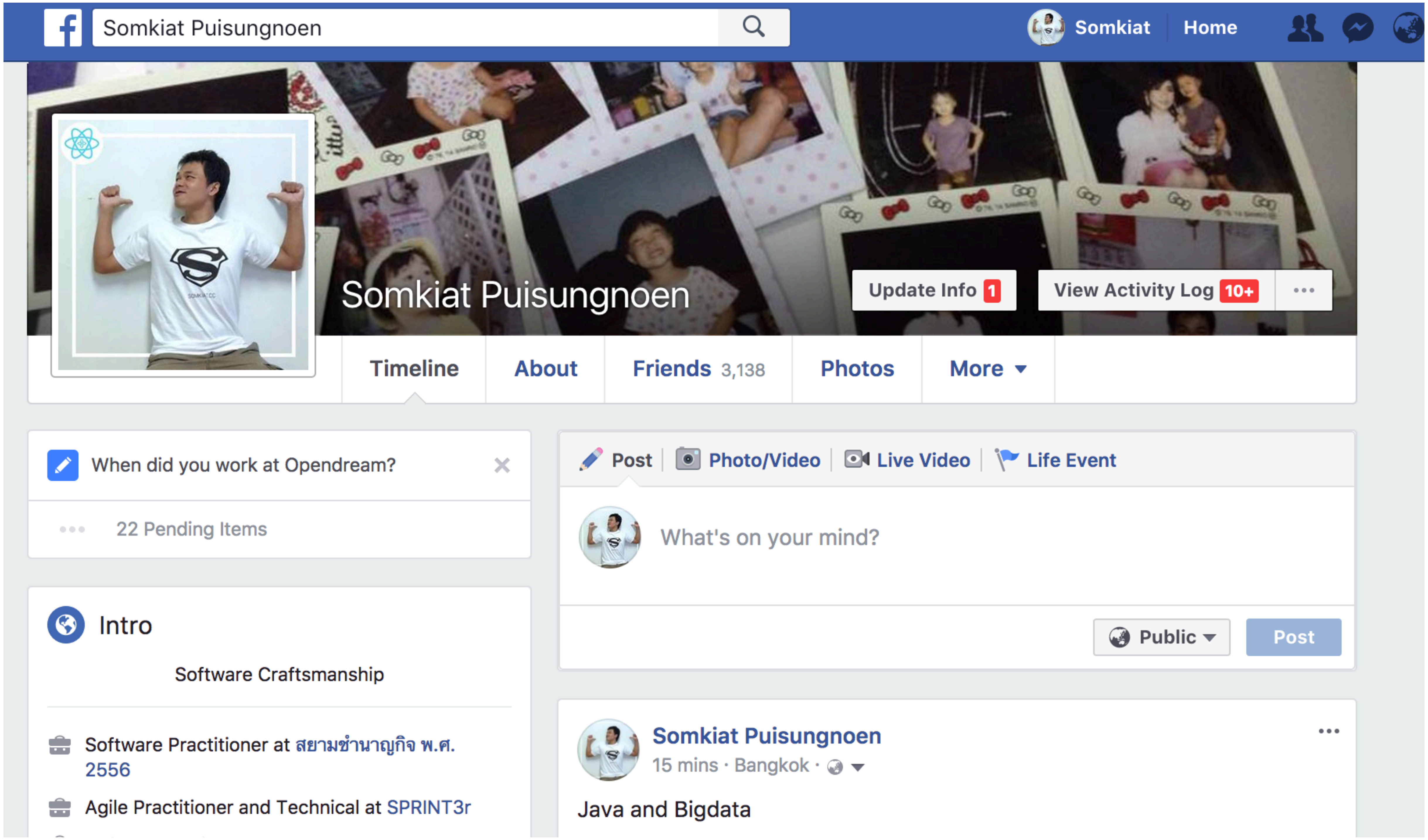
Workshop



When did you work at Opendream?

22 Pending Items

Intro

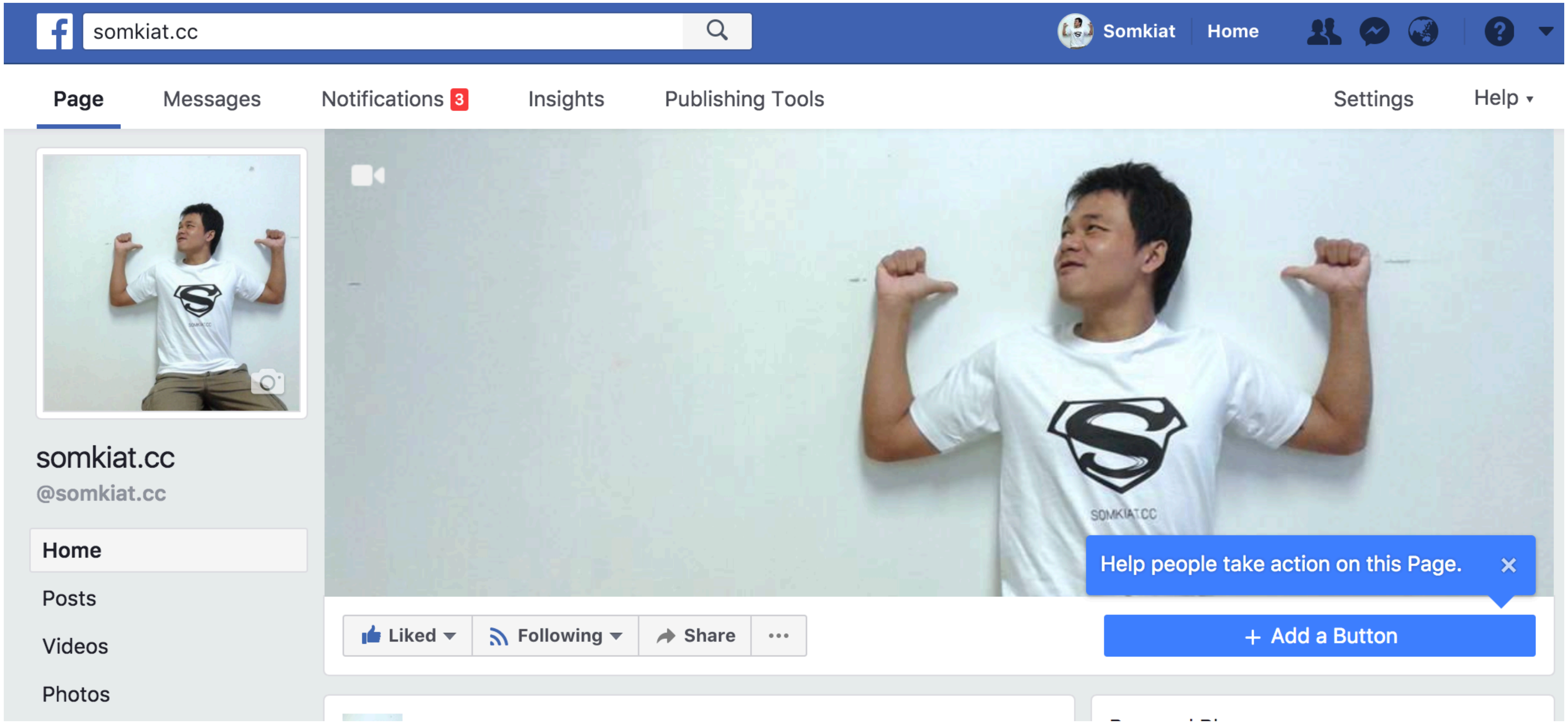
Software Craftsmanship

Software Practitioner at สยามชำนาญกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Somkiat Puisungnoen
15 mins · Bangkok ·

Java and Bigdata


somkiat.cc
Somkiat
Home
Page
Messages
Notifications 3
Insights
Publishing Tools
Settings
Help
somkiat.cc
@somkiat.cc
Home
Posts
Videos
Photos
Help people take action on this Page.
Liked
Following
Share
+ Add a Button

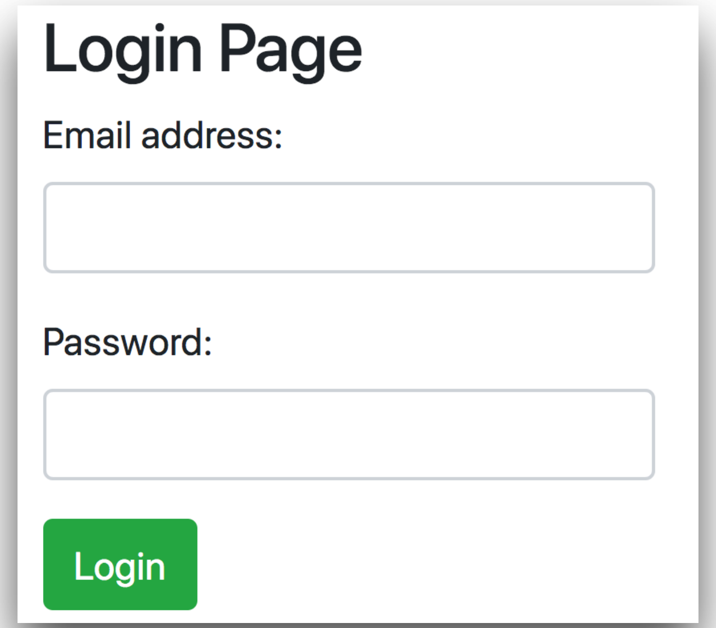
Workshop 02

https://github.com/up1/angular-workshop-02

Feature 1

Working with routing

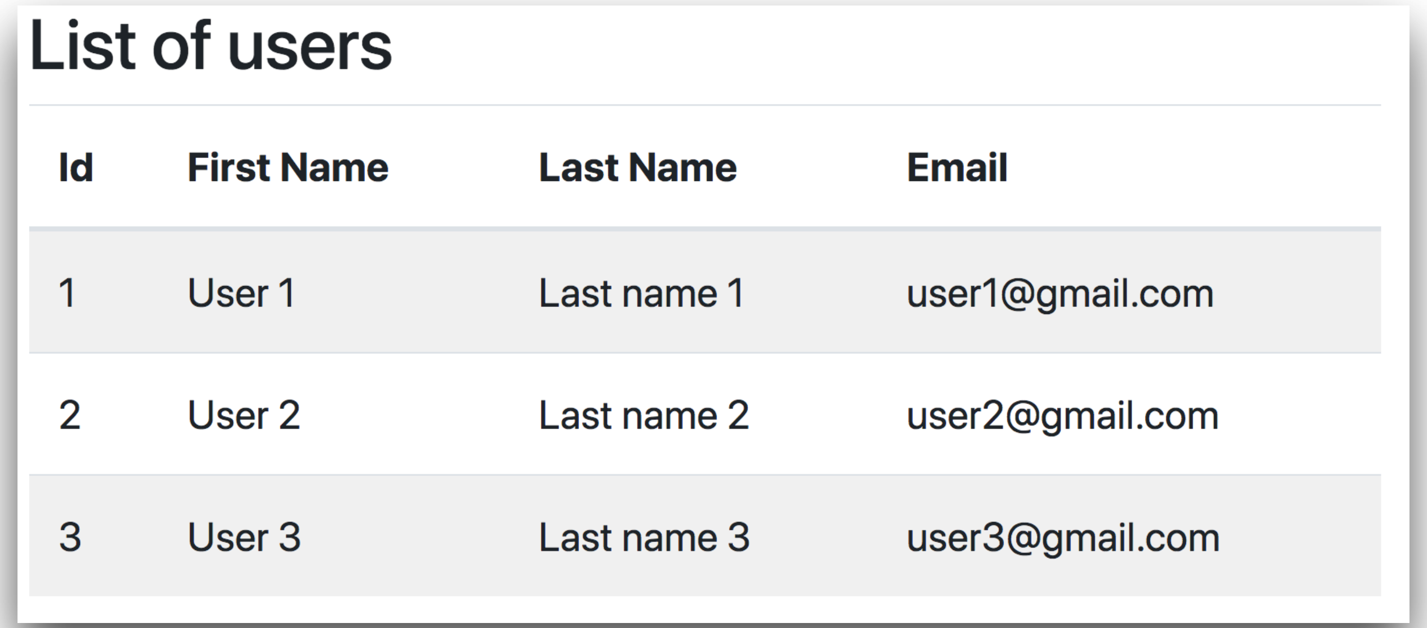
/login



Login Page

Email address:
Password:

/list



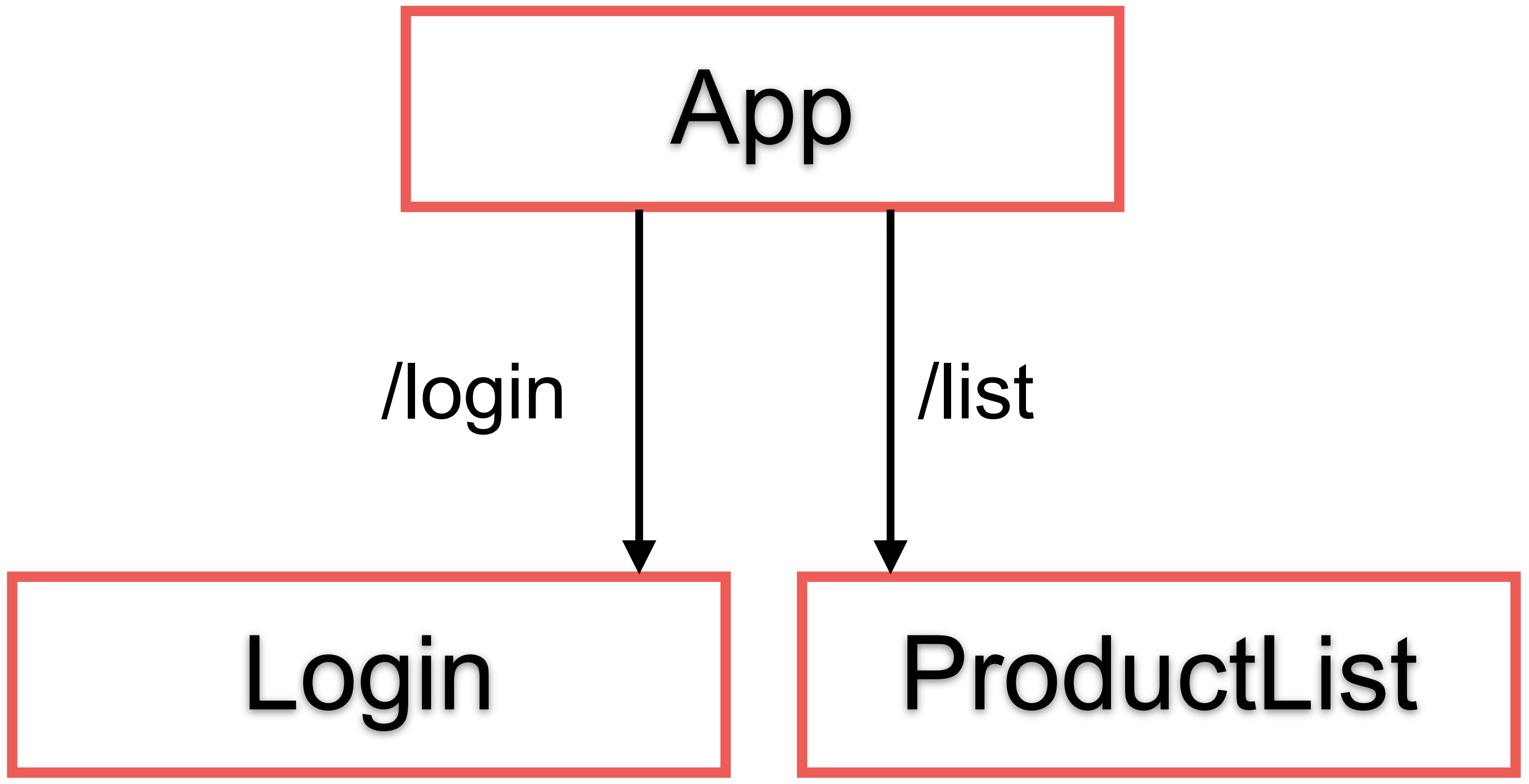
List of users

Id	First Name	Last Name	Email
1	User 1	Last name 1	user1@gmail.com
2	User 2	Last name 2	user2@gmail.com
3	User 3	Last name 3	user3@gmail.com

Login component

ProductList component

Component



app-routing.module.ts

```
import { ProductListComponent } from './product-list/product-
list.component';
import { LoginComponent } from './login/login.component';
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
```

```
const routes: Routes = [
  { path: 'login', component: LoginComponent },
  { path: 'list', component: ProductListComponent },
];
```

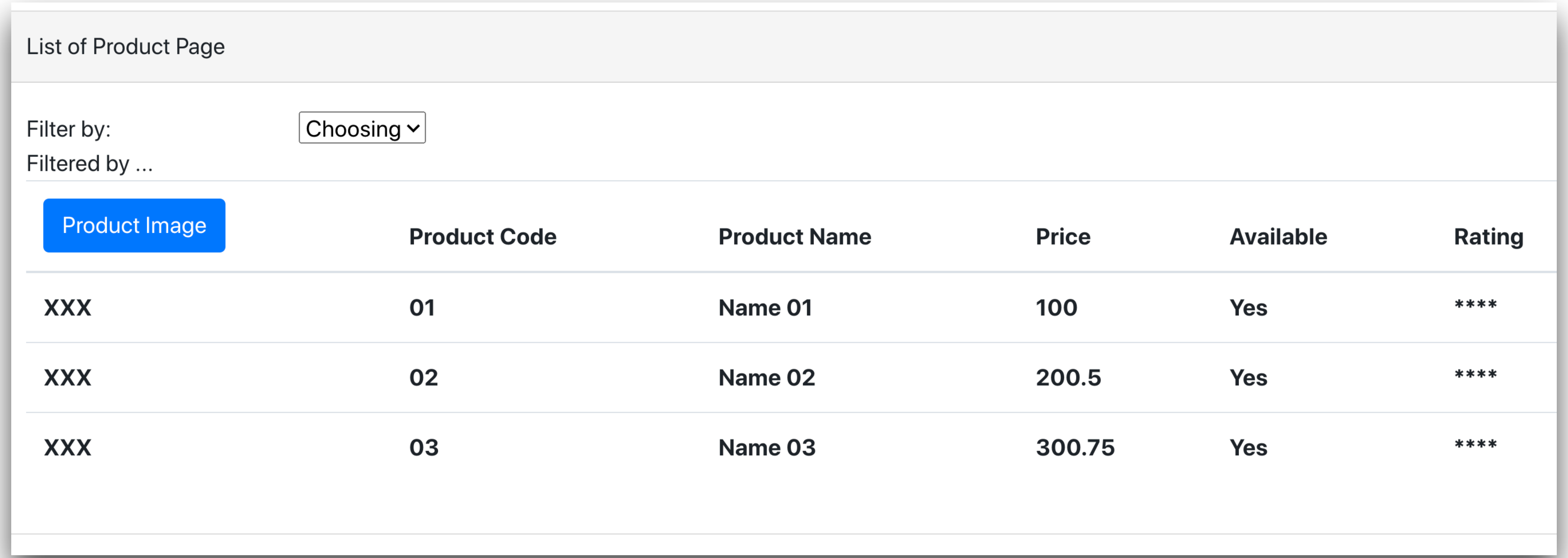
app.component.html

```html
<nav>
  <ul>
    <li>
      <a [routerLink]="['/login']">Login</a>
    </li>
    <li>
      <a [routerLink]="['/list']">Product list</a>
    </li>
  </ul>
</nav>
```

```html
<div>
  <router-outlet></router-outlet>
</div>
```

Feature 2

List of product



List of Product Page

Filter by: Choosing

Filtered by ...

Product Image	Product Code	Product Name	Price	Available	Rating
XXX	01	Name 01	100	Yes	****
XXX	02	Name 02	200.5	Yes	****
XXX	03	Name 03	300.75	Yes	****

Create data class

/models/user

```
export class Product {
  public rating: number;
  public available: boolean;
  public imageUrl: string;

  constructor(public code: string,
              public name: string,
              public price: number) {}
}
```

productList.component.ts

Store data in memory (Product[])

```
export class ProductListComponent implements OnInit {

  products: Product[] = [];

  constructor() {}

  ngOnInit(): void {
    const p1 = new Product('01', 'Name 01', 100.0);
    const p2 = new Product('02', 'Name 02', 200.5);
    const p3 = new Product('03', 'Name 03', 300.75);
    this.products.push(p1, p2, p3);
  }

}
```

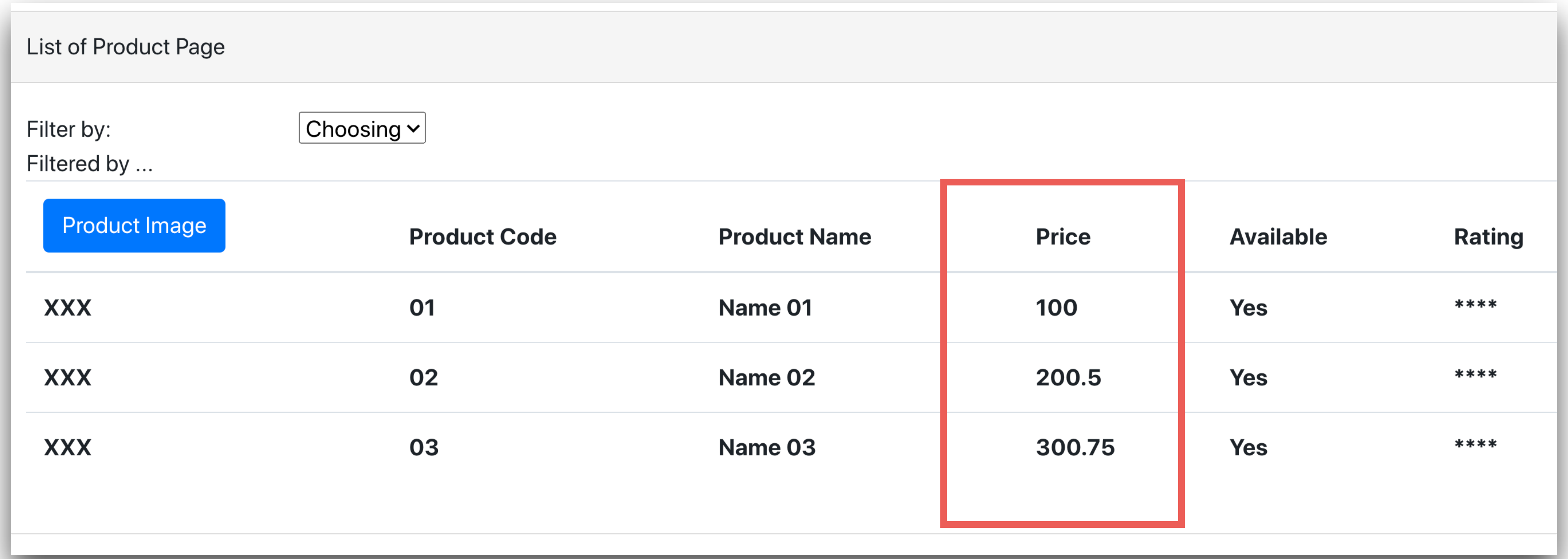
productList.component.html

Show all products (*ngIf, *ngFor)

```html
<table class="table">

   <tbody *ngIf="products.length > 0">
     <tr *ngFor="let product of products">
       <th>
         XXX
       </th>
       <th>{{ product.code }}</th>
       <th>{{ product.name }}</th>
       <th>{{ product.price }}</th>
       <th>Yes</th>
       <th>****</th>
     </tr>
   </tbody>

 </table>
```

Feature 3

Formatting a product's price (x,xxx.xx)


List of Product Page

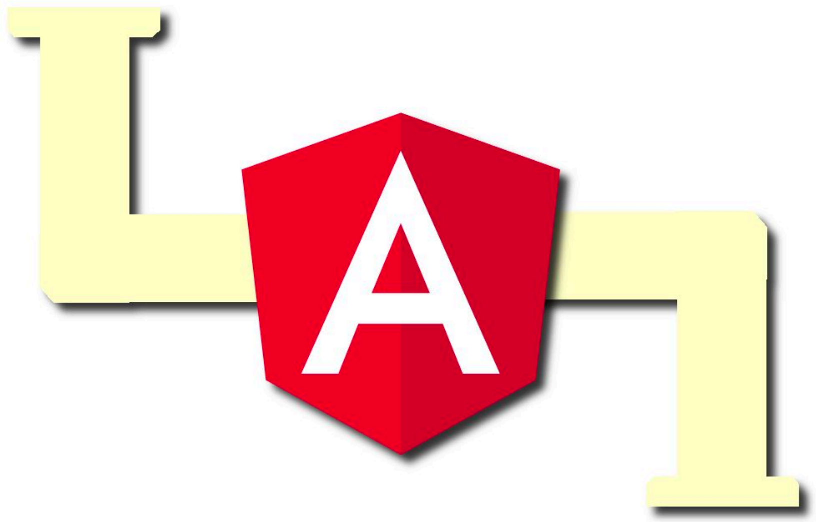
Filter by: Choosing

Filtered by ...

Product Image	Product Code	Product Name	Price	Available	Rating
XXX	01	Name 01	100	Yes	****
XXX	02	Name 02	200.5	Yes	****
XXX	03	Name 03	300.75	Yes	****

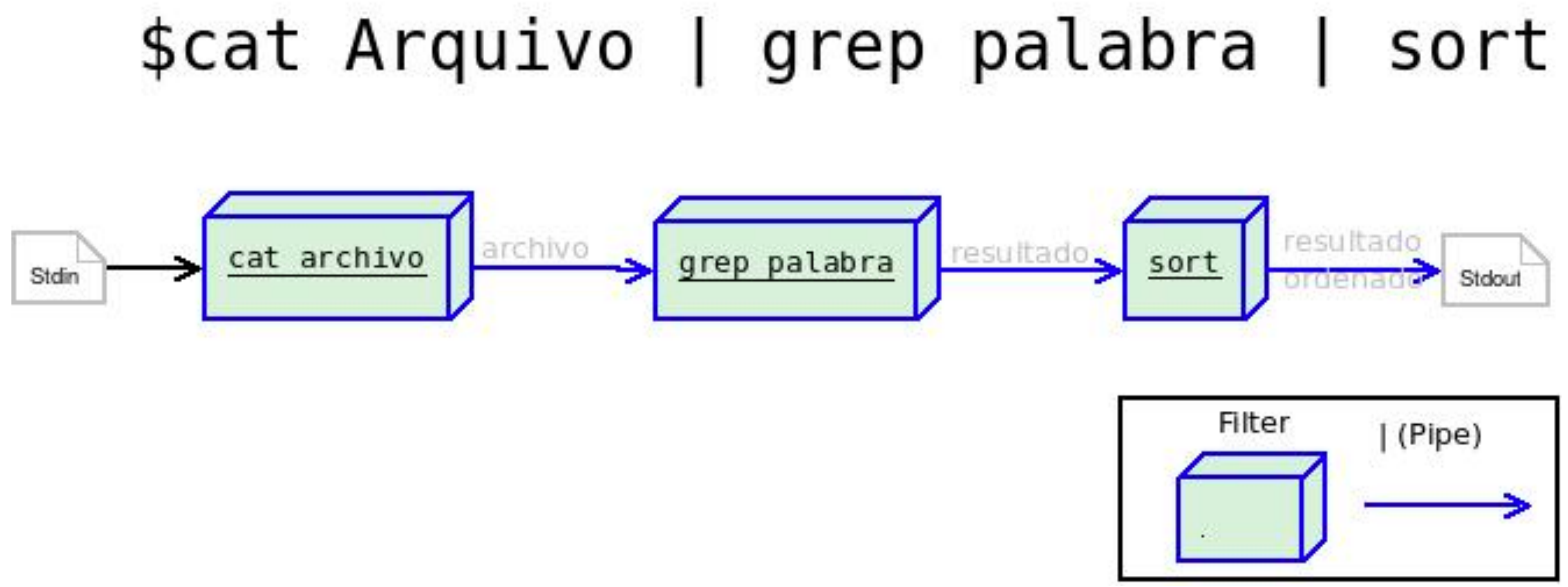
Using pipe

Transform data in a template



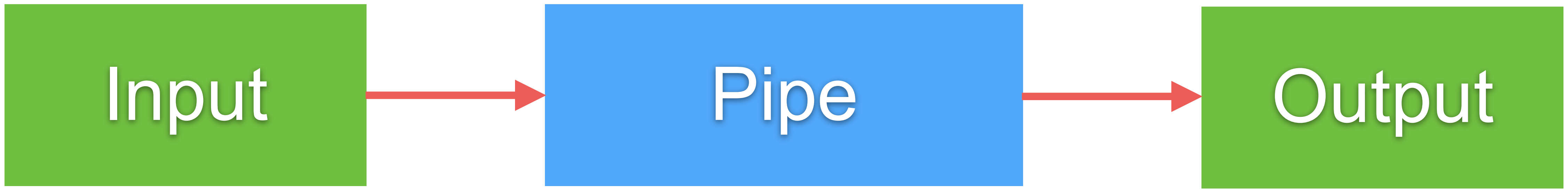
Using pipe

Transform data in a template



Angular Pipe

Pipes allow us to change the way to show data and transform data in our template



https://angular.io/api?type=pipe

Build-in pipes

Date
Lowercase
Uppercase
Currency
Percent

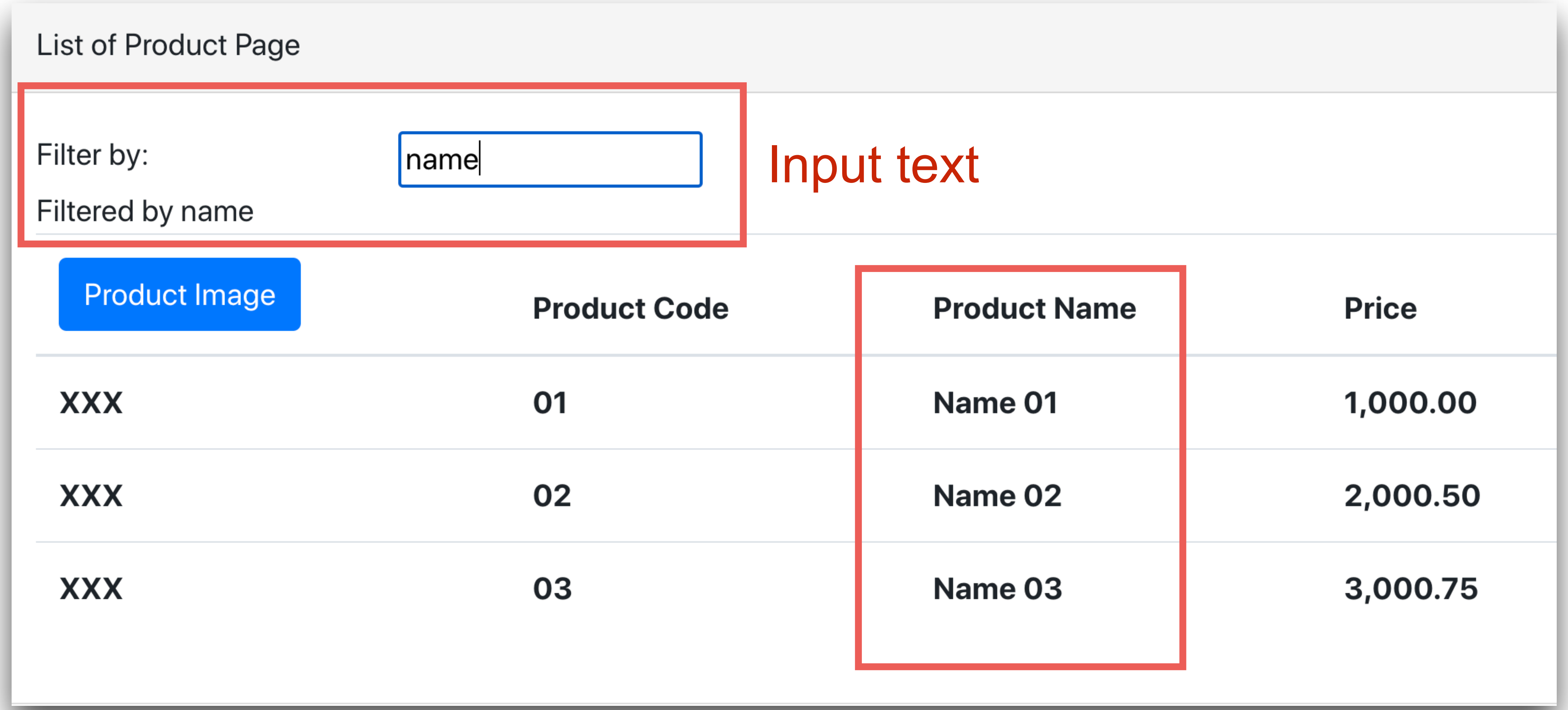
https://angular.io/api?type=pipe

productList.component.html

```
<tbody *ngIf="products.length > 0">
  <tr *ngFor="let product of products">
    <th>
      XXX
    </th>
    <th>{{ product.code }}</th>
    <th>{{ product.name }}</th>
    <th>{{ product.price | number: "1.2" }}</th>
    <th>Yes</th>
    <th>****</th>
  </tr>
</tbody>
```

Feature 4

Filter products by **name**



List of Product Page

Filter by: name

Filtered by name

Product Image	Product Code	Product Name	Price
XXX	01	Name 01	1,000.00
XXX	02	Name 02	2,000.50
XXX	03	Name 03	3,000.75

Input text

productList.component.html

Two-way binding

```html
<div class="row">
  <div class="col-md-2">Filter by:</div>
  <div class="col-md-4">
    <input type="text" [(ngModel)]="filterData" />
  </div>
</div>
<div class="row">
  <div class="col-md-6">Filtered by {{ filterData }}</div>
</div>
```

App.module.ts

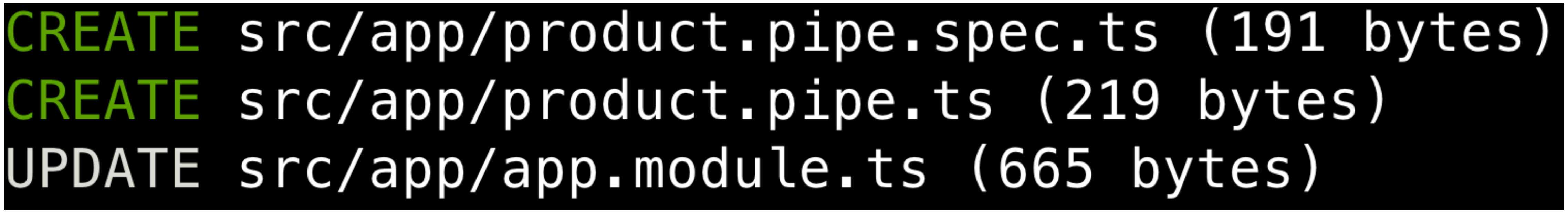
Enabled module Angular Forms

```
import { FormsModule } from '@angular/forms';
@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule, AppRoutingModule, FormsModule],
  providers: [],
  bootstrap: [AppComponent],
})
```

Filter by product name

Using pipe to filter data

$ng generate pipe product



```
CREATE src/app/product.pipe.spec.ts (191 bytes)
CREATE src/app/product.pipe.ts (219 bytes)
UPDATE src/app/app.module.ts (665 bytes)
```

product.pipe.ts

Using pipe to filter data

```typescript
@Pipe({
  name: 'product',
})
export class ProductPipe implements PipeTransform {
  transform(products: Product[], name: string): Product[] {
    return products.filter((p) => p.name.indexOf(name) != -1);
  }
}
```

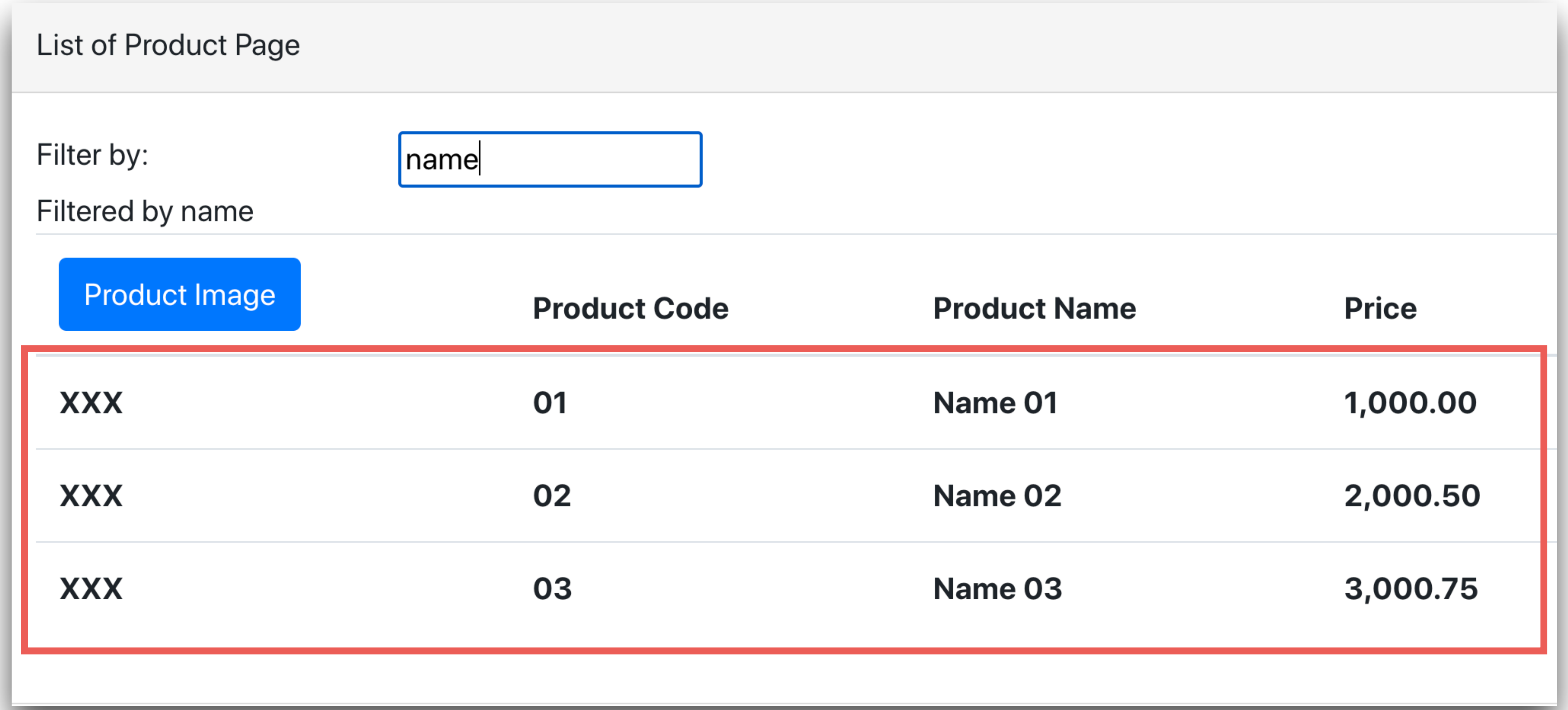
productList.component.html

Using product filter

```
<tbody *ngIf="products.length > 0">
  <tr *ngFor="let product of products | product: filterData">
    <th>
      XXX
    </th>
    <th>{{ product.code }}</th>
    <th>{{ product.name }}</th>
    <th>{{ product.price | number: "1.2" }}</th>
    <th>Yes</th>
    <th>****</th>
  </tr>
</tbody>
```

Feature 5

List all products from **service**



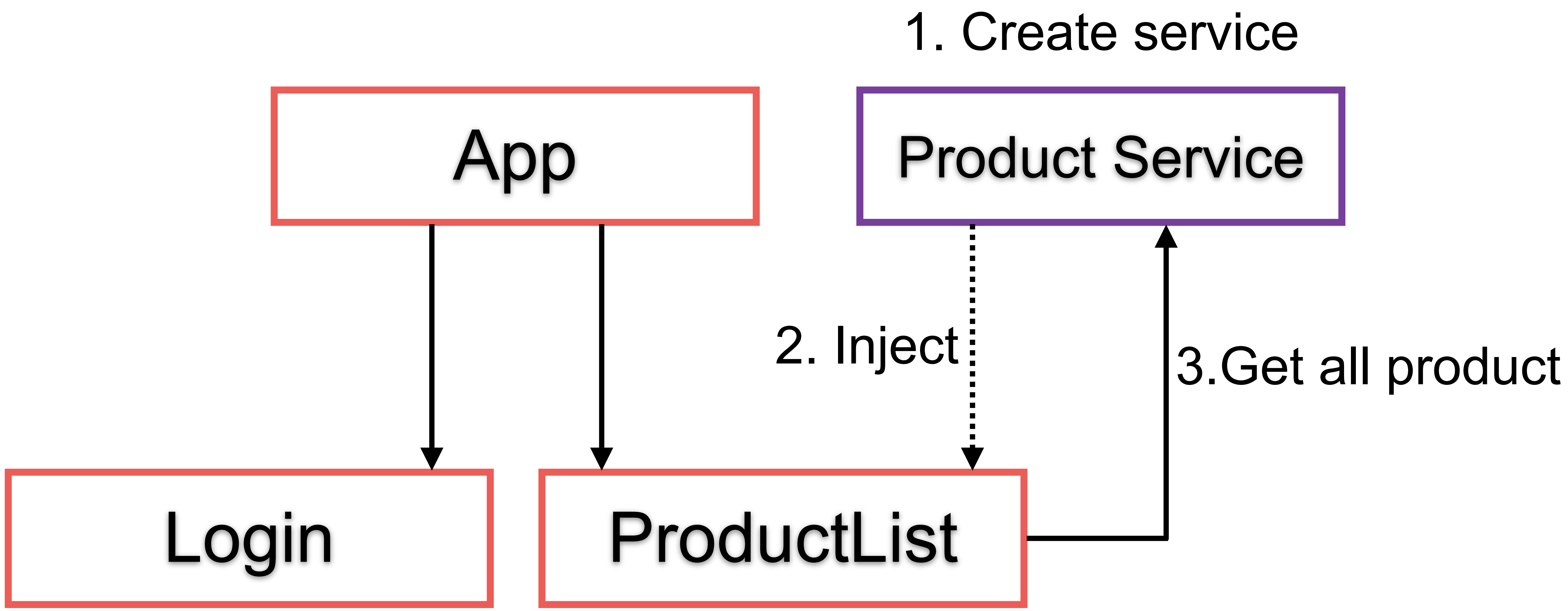
List of Product Page

Filter by: name

Filtered by name

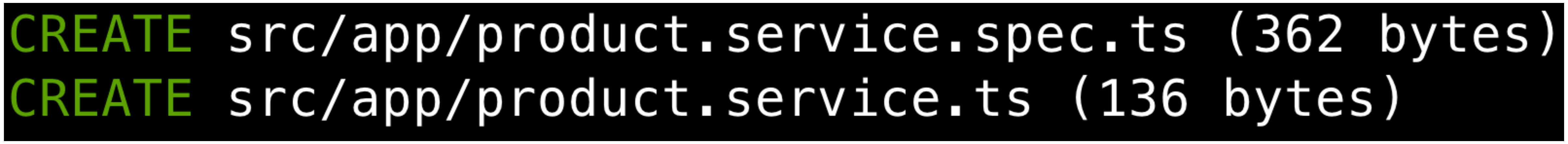
Product Image	Product Code	Product Name	Price
XXX	01	Name 01	1,000.00
XXX	02	Name 02	2,000.50
XXX	03	Name 03	3,000.75

Get all products from service



1. Create product service

$ng generate service product



```
CREATE src/app/product.service.spec.ts (362 bytes)
CREATE src/app/product.service.ts (136 bytes)
```

Product.service.ts

```
import { Injectable } from '@angular/core';
import { Product } from './models/product';

@Injectable({
  providedIn: 'root',
})
export class ProductService {

  getAllProoduct(): Product[] {
    const products: Product[] = [];
    const p1 = new Product('01', 'Name 01', 1000.0);
    const p2 = new Product('02', 'Name 02', 2000.5);
    const p3 = new Product('03', 'Name 03', 3000.75);
    products.push(p1, p2, p3);
    return products;
  }

}
```

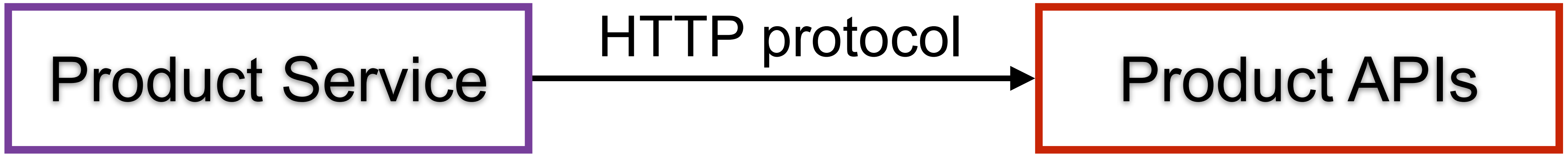
2. Inject service to component

Edit file productList.component.ts

```
export class ProductListComponent implements OnInit {
  products: Product[] = [];
  filterData = '';

  constructor(public service: ProductService) {}

  ngOnInit(): void {
    this.products = this.service.getAllProoduct();
  }
}
```

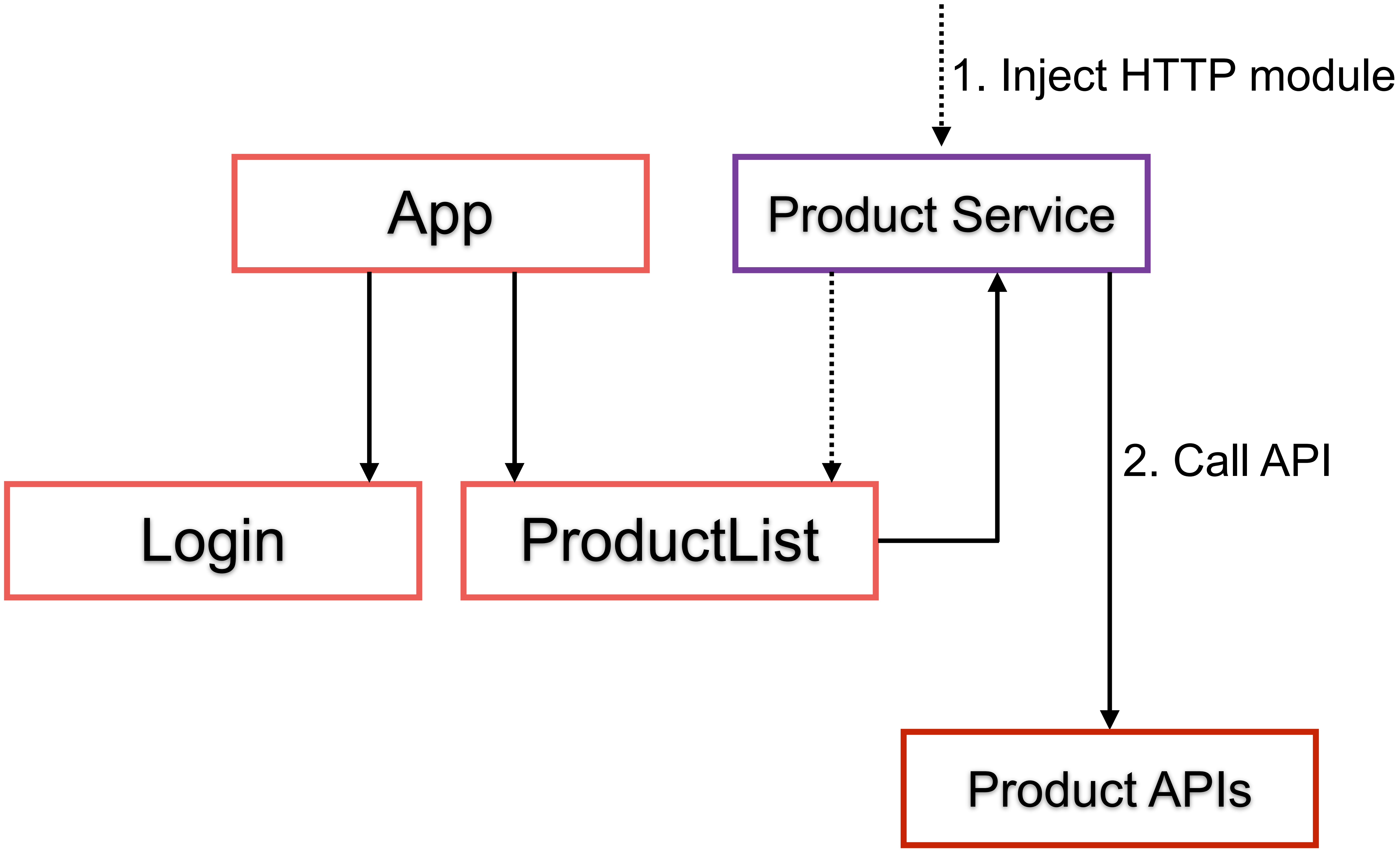
Feature 6

Get all products from Product APIs (HTTP)



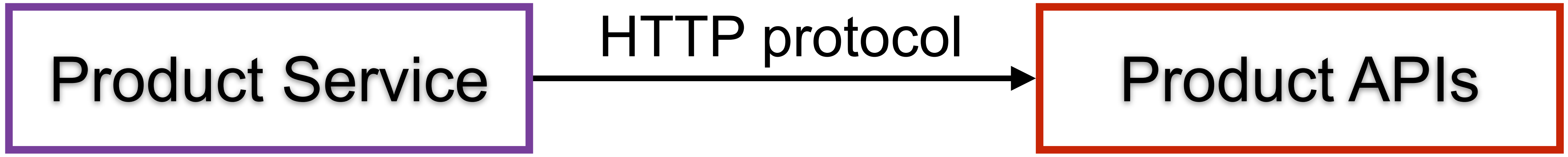
https://angular.io/guide/http

Get all products from service



Product API

https://product.free.beeceptor.com/products



1. Enable HTTP client module

Edit file app.module.ts

```
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  declarations: [

  ],
  imports: [HttpClientModule],
  providers: [],
  bootstrap: [],
})
```

2. Inject HTTP Client to service

Edit file product.service.ts

```
import { HttpClient, HttpHeaders } from '@angular/common/http';

@Injectable({
  providedIn: 'root',
})
export class ProductService {

  constructor(private http: HttpClient) {}

}
```

3. Using RxJS with asynchronous process

Edit file product.service.ts

```
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root',
})
export class ProductService {

  constructor(private http: HttpClient) {}

  getAllProoduct(): Observable<Product[]> {
    return this.http.get<Product[]>(
      'https://product.free.beeceptor.com/products'
    );
  }
}
```

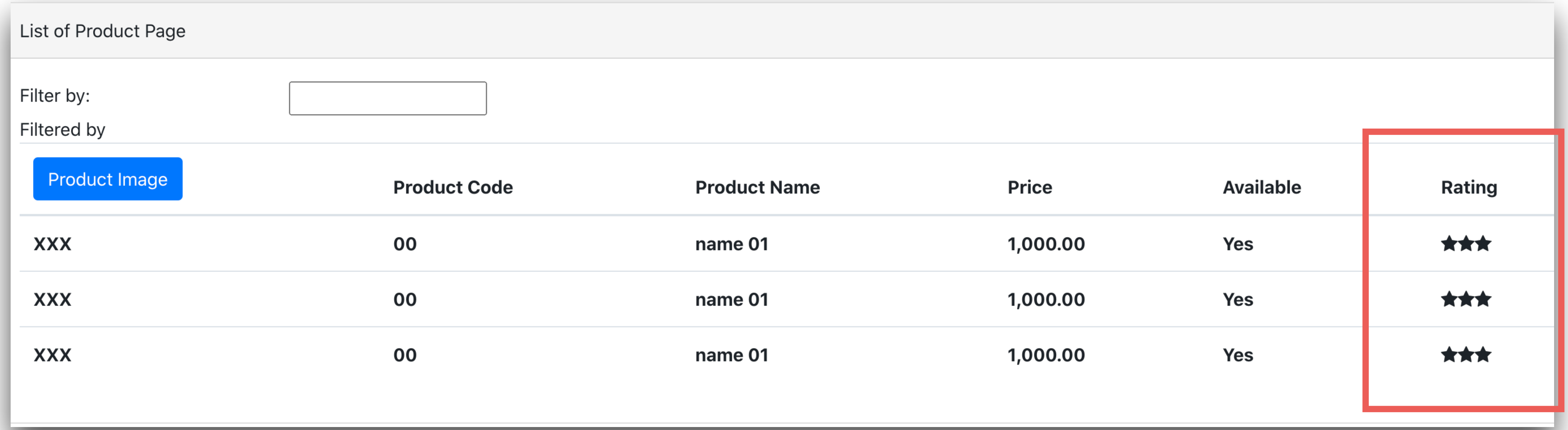
4. Update product list component

Edit file productList.component.ts

```
export class ProductListComponent implements OnInit {
  products: Product[] = [];

  filterData = '';

  constructor(public service: ProductService) {}

  ngOnInit(): void {
    this.getAll();
  }

  getAll(): void {
    this.service.getAllProoduct().subscribe((products) => {
      return (this.products = products);
    });
  }
}
```

Feature 7

Woking with child component (Rating)



List of Product Page

Filter by:

Filtered by

Product Image	Product Code	Product Name	Price	Available	Rating
XXX	00	name 01	1,000.00	Yes	★★★
XXX	00	name 01	1,000.00	Yes	★★★
XXX	00	name 01	1,000.00	Yes	★★★

Create star of rating

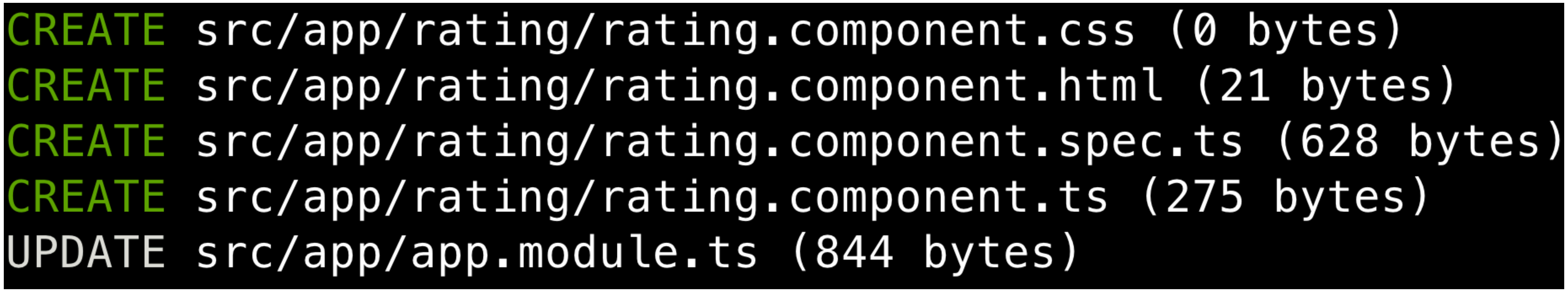
Using CSS from FontAwesome.com

```
<link
      rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css"
    />
```

https://fontawesome.com/v4.7.0/

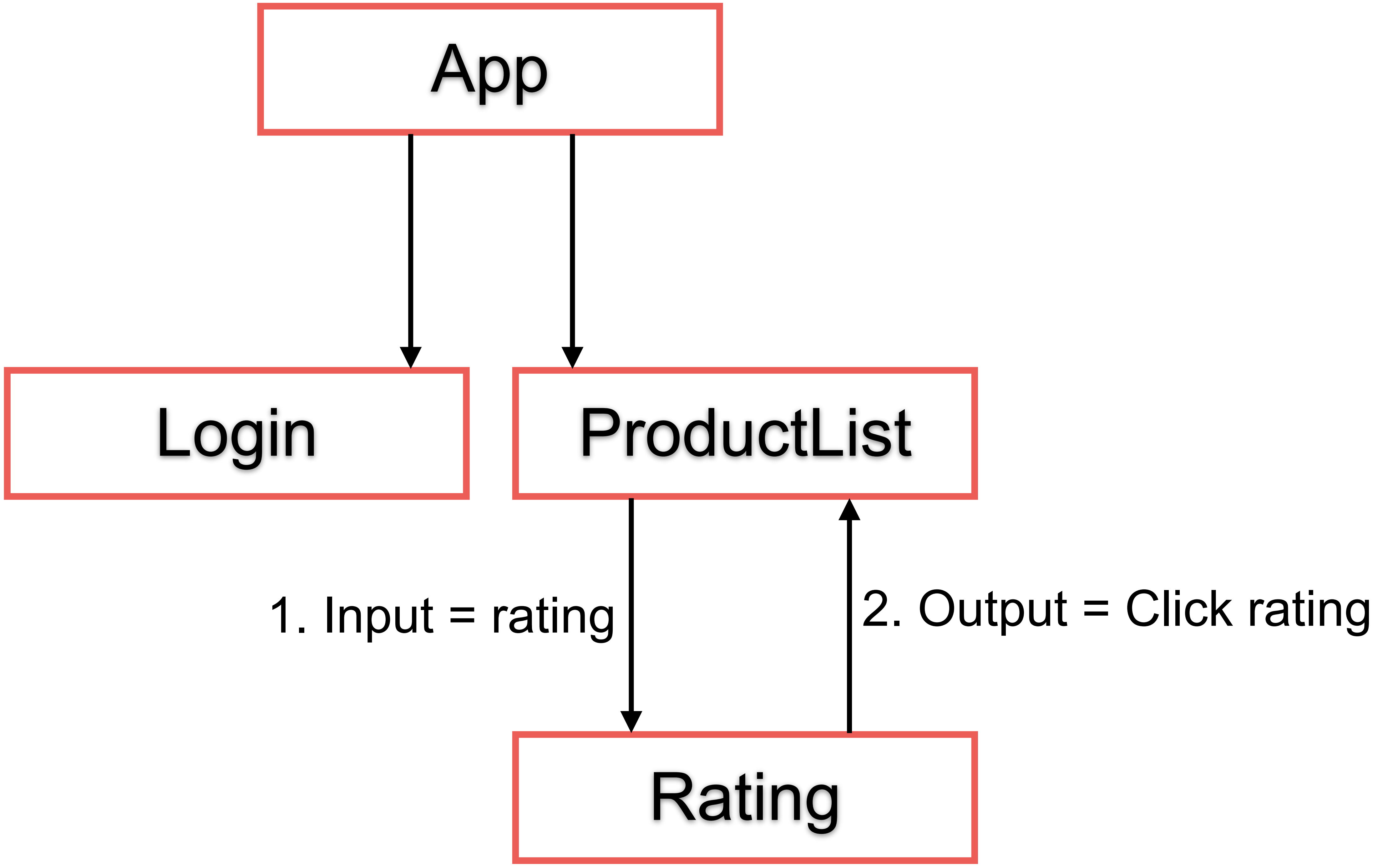
Create rating component

$ng generate component rating



```
CREATE src/app/rating/rating.component.css (0 bytes)
CREATE src/app/rating/rating.component.html (21 bytes)
CREATE src/app/rating/rating.component.spec.ts (628 bytes)
CREATE src/app/rating/rating.component.ts (275 bytes)
UPDATE src/app/app.module.ts (844 bytes)
```

Get all products from service



1. Send data from parent to child

productList.component.html

```
<th>
  <app-rating [rating]="product.rating"> </app-rating>
</th>
```

rating.component.ts

```
export class RatingComponent implements OnChanges {
  @Input() rating: number;

  starWidth: number;

  ngOnChanges(): void {
    console.log(this.rating);
    this.starWidth = (75 / 5) * this.rating;
  }
}
```

Display data

rating.component.html

```html
<div [style.width.px]="starWidth" style="overflow: hidden;">
  <div style="width: 75px;">
    <span class="fa fa-star"></span>
    <span class="fa fa-star"></span>
    <span class="fa fa-star"></span>
    <span class="fa fa-star"></span>
    <span class="fa fa-star"></span>
  </div>
</div>
```

2. Send output from Child to parent

Child = Rating component

rating.component.html

```
<div
  [style.width.px]="starWidth"
  style="overflow: hidden;"
  (click)="onClickRating()"
>
```

1

rating.component.ts

```
@Output() ratingClicked: EventEmitter<string> = new EventEmitter<string>();

onClickRating(): void {
  console.log('Click on rating');
  this.ratingClicked.emit(`Rating ${this.rating} was clicked`);
}
```

2 Emit to output (ratingClicked)

2. Send output from Child to parent

Parent = Product List component

productList.component.html

```
<app-rating
  [rating]="product.rating"
  (ratingClicked)="onRatingClicked($event)">
</app-rating>
```

3 Binding function with output from child

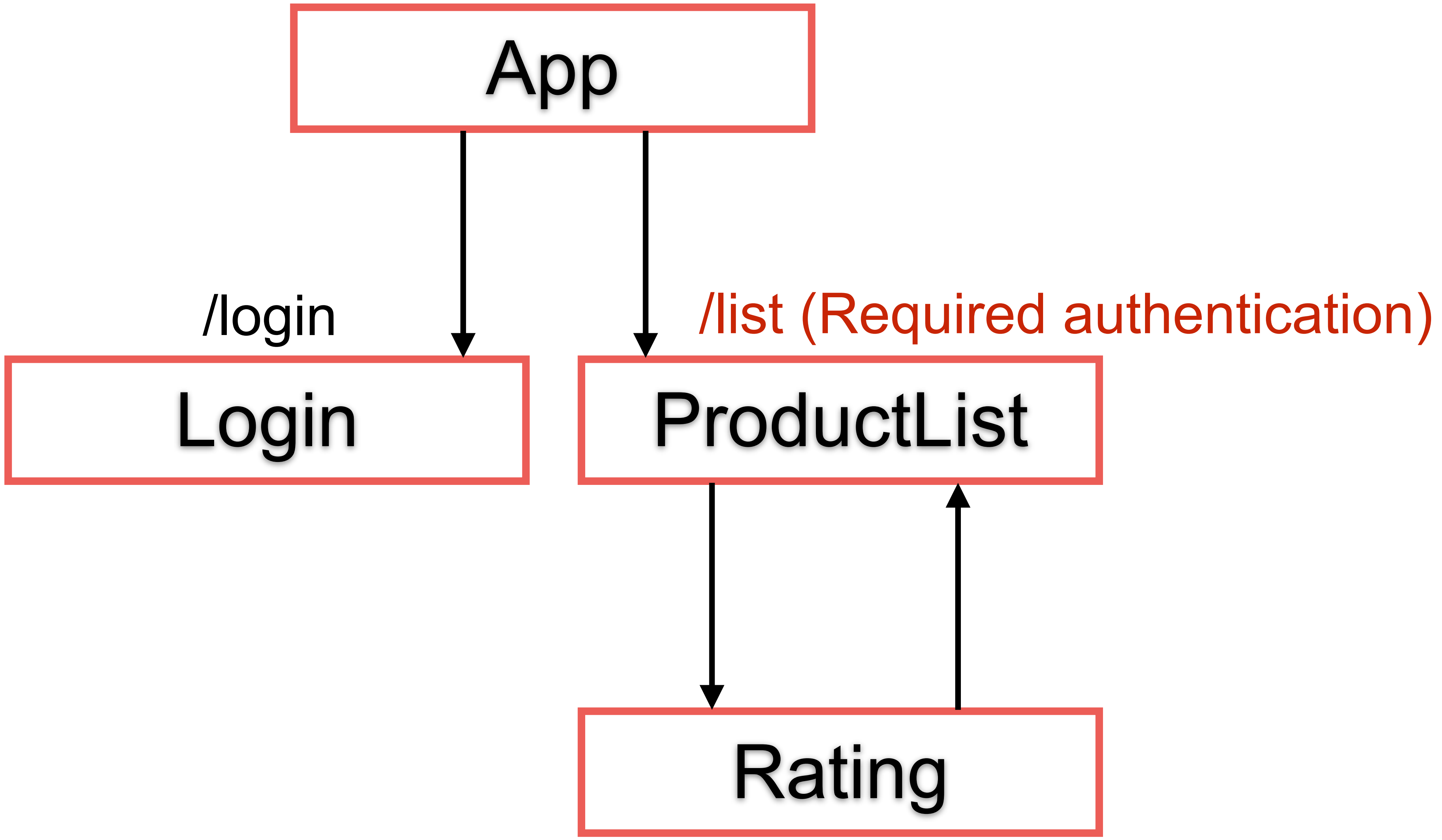
4 Receive output from child

productList.component.ts

```
onRatingClicked(message: string): void {
  console.log(message);
}
```

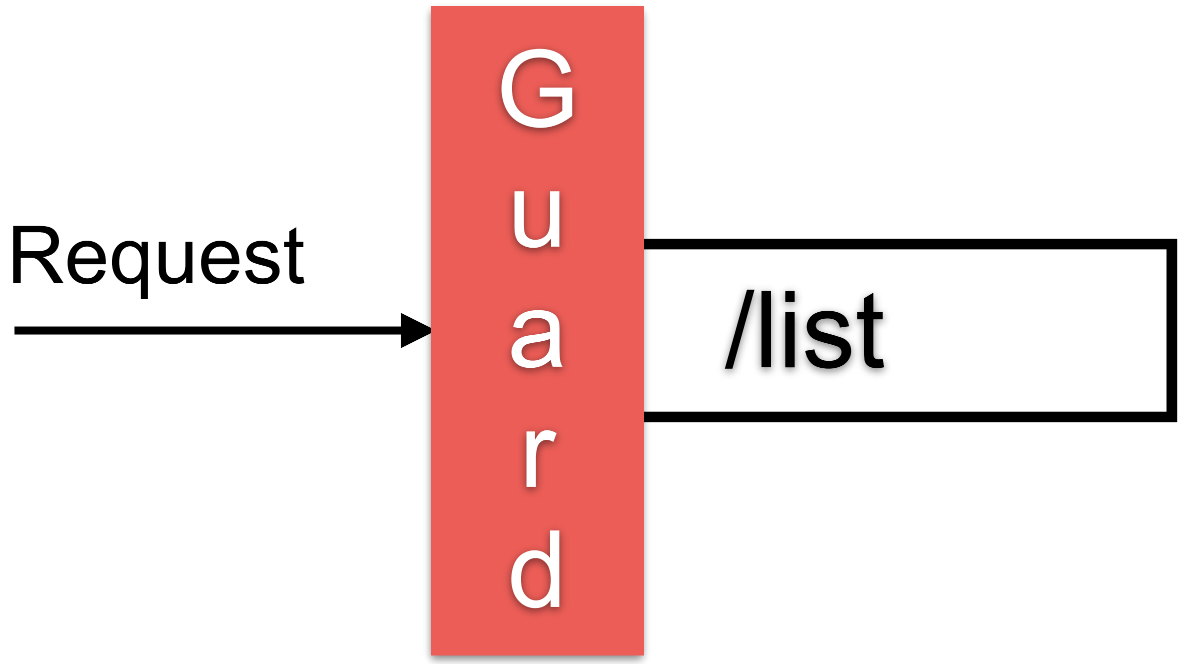
Feature 8

Working with authentication and guard



Working with route guard

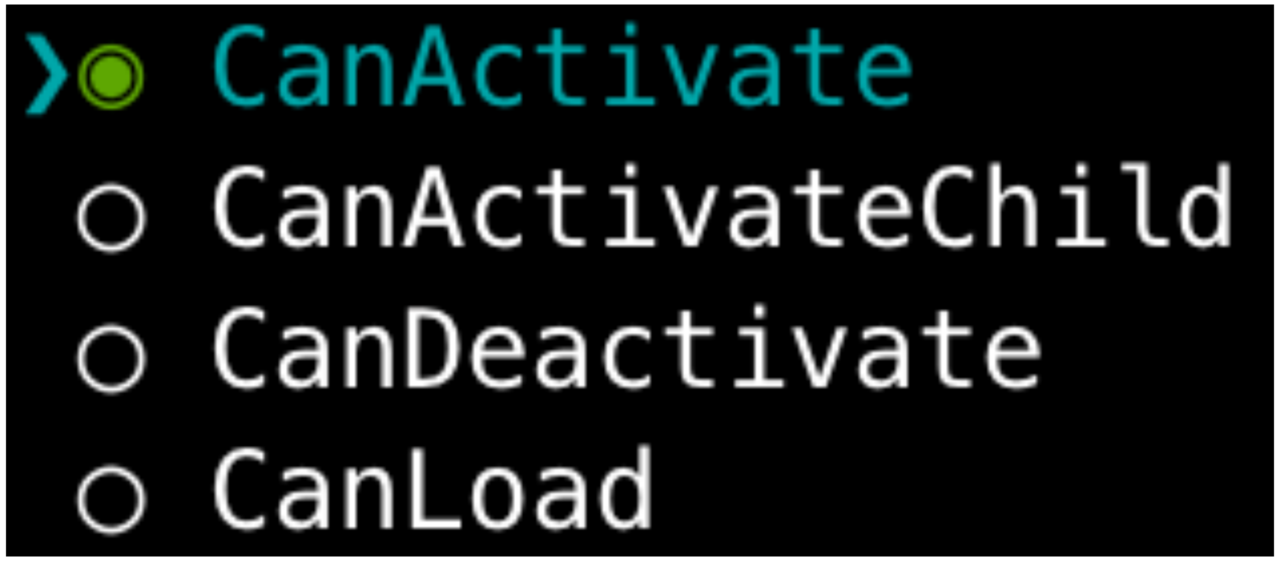
Interfaces which can tell the route
Allow navigation of request to route ?



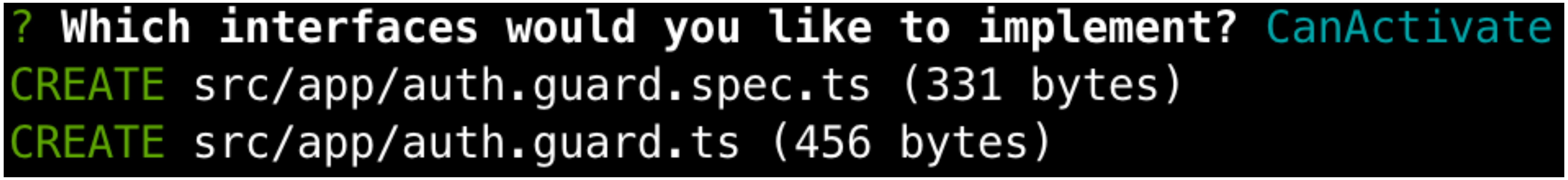
https://angular.io/api/router/CanActivate

Create guard

$ng generate guard auth



```
❯◉ CanActivate
 ◯ CanActivateChild
 ◯ CanDeactivate
 ◯ CanLoad
```



```
? Which interfaces would you like to implement? CanActivate
CREATE src/app/auth.guard.spec.ts (331 bytes)
CREATE src/app/auth.guard.ts (456 bytes)
```

https://angular.io/api/router/CanActivate

auth.guard.th

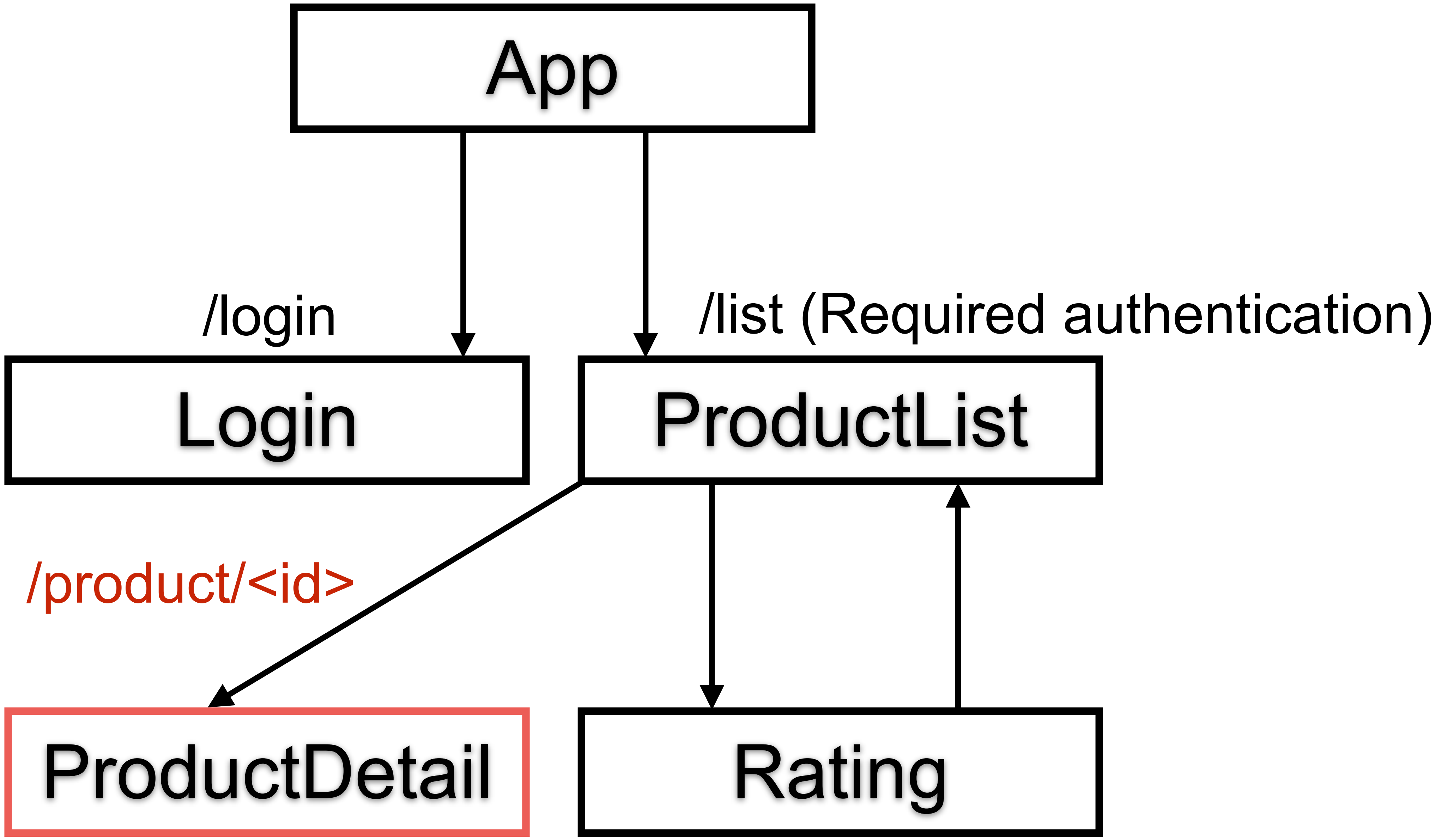
```
export class AuthGuard implements CanActivate {

  constructor(private router: Router) {}

  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot
  ): boolean {
    // TODO :: check authentication

    const param = route.params.name;
    if (!param) {
      return true;
    } else {
      // not logged in : redirect to login page with the return url
      this.router.navigate(['/login'], {
        queryParams: { returnUrl: state.url },
      });
      return false;
    }
  }
}
```

app-routing.module.th

```
const routes: Routes = [
  { path: 'login', component: LoginComponent },
  { path: 'list', component: ProductListComponent },
  {
    path: 'list/:name',
    component: ProductListComponent,
    canActivate: [AuthGuard],
  },
];
```

Feature 9

Show detail of product and add product to the basket


App
/login
/list (Required authentication)
Login
ProductList
/product/<id>
ProductDetail
Rating