# Angular Furniture System Workshop

Exercise and homework for the ["Angular Fundamentals" course @ "SoftUni"](https://softuni.bg/courses/angular-2-fundamentals)

## Initial Setup

Explore the provided **Angular** **Skeleton** and **server**. Install **dependencies** in both projects and run them. (Start the server with **node index.js**).

You will also need to install **MongoDB** from the [download center](https://www.mongodb.com/download-center/community). Install a GUI for MongoDB like [RoboMongo](https://robomongo.org/)

Use [Postman](https://www.getpostman.com/) to **test** signing up and signing into the server.

You will be provided with all the back-end logic. The following routes are implemented:

* **'http://localhost:5000/auth/register'**
* **'http://localhost:5000/auth/login'**
* **'http://localhost:5000/furniture/create'**
* **'http://localhost:5000/furniture/all'**
* **'http://localhost:5000/furniture/details/:id'**
* **'http://localhost:5000/furniture/mine'**
* **'http://localhost:5000/furniture/delete/:id'**

## Setup Ngx Toastr

Use **Ngx Toastr** to provide **success** and **error** messages on the page

* Install it with **npm install ngx-toastr –save** (angular animations as well if needed).
* Add **toastr.css** to **angular.json** “node\_modules/ngx-toastr/toastr.css”
* Import **ToastrModule** from "**ngx-toastr**" and **BrowserAnimationsModule**
* Call inside imports **ToastrModule.forRoot()**
* Inject **ToastrService** inside interceptors or components and use it.
* More on [Ngx Toastr](https://www.npmjs.com/package/ngx-toastr)

## Interceptors

We need to create **two interceptors** in our project.

One for attaching **authorization jwt’s** to access authenticated routes (“Authorization”: “Bearer {jwt}”) also to **save the token** when signing into the application (local storage). After successful **signing into** the application implement the **logout** functionality.

Another interceptor for **handling errors** that the server returns (it currently returns only **401 Unauthorized** and **400 Bad Request**).

*Don’t forget to provide both interceptors*

## Implement Furniture Store

Create a **folder furniture** and generate **4 components** (furniture-all, create-furniture, furniture-details, furniture-user) ***Use the provided html for all the html files***.

Add **routing** for these 4 components and use **children routes** inside “**app.routing.ts**”. **Protect** the routes with the given **authentication guard**.

We need a **furniture service** to retrieve information from the server. The **URL’s** are as follows:

* Create Furniture (POST): **http://localhost:5000/furniture/create**
* All Furniture (GET): **http://localhost:5000/furniture/all**
* Furniture Details (GET): **http://localhost:5000/furniture/details/{id}**
* User Furniture (GET): **http://localhost:5000/furniture/user**
* Delete Furniture (DELETE): **http://localhost:5000/furniture/delete/{id}**

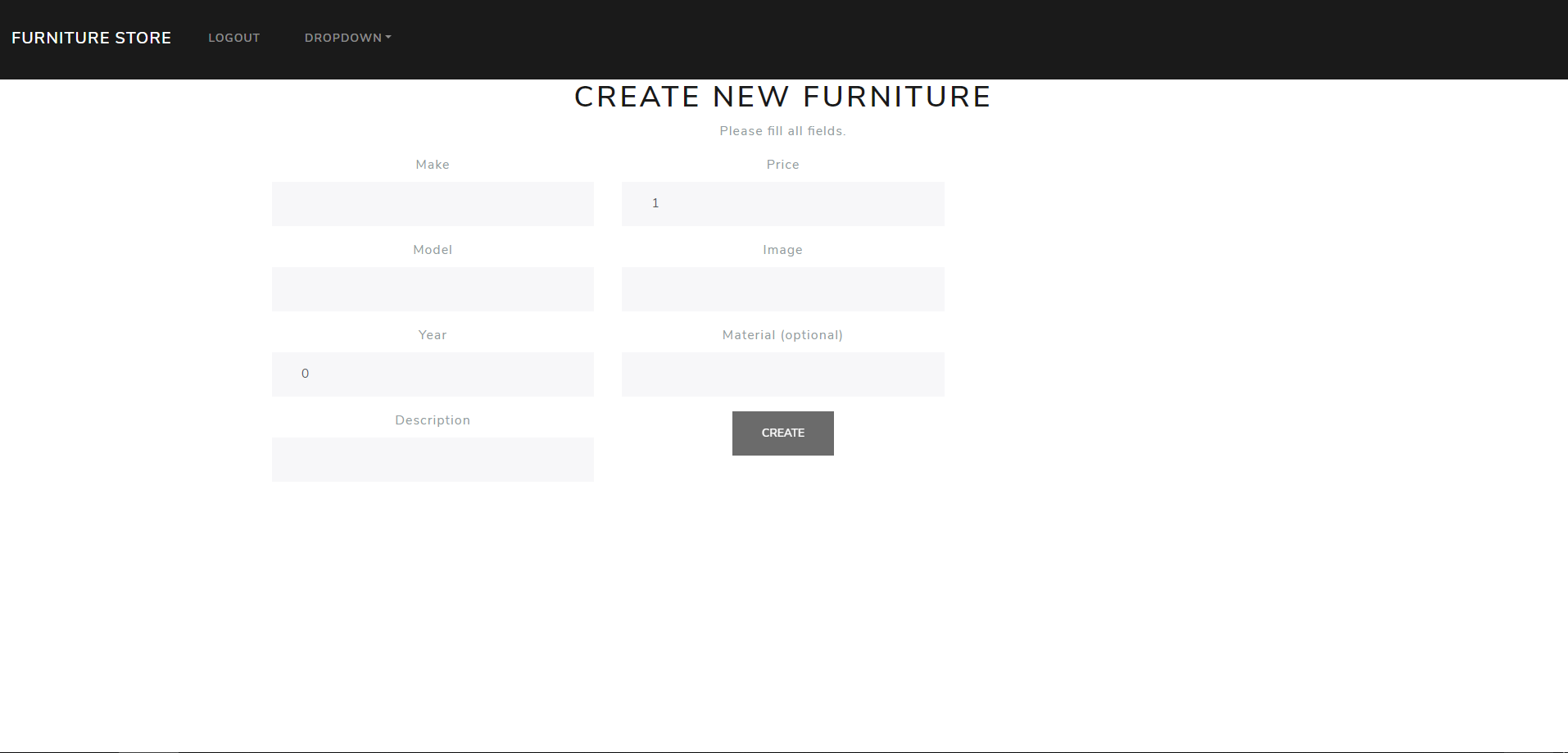
### Create Furniture

**Validate fields:**

* Make and Model must be **at least** **4 symbols** long
* Year must be between **1950** and **2050**
* Description must be **at least** than **10 symbols**
* Price must be a **positive number**
* Image URL is **required**
* Material is **optional**

**Handle errors** that the **server** could return (**400 Bad Request** if some fields are invalid).

If the creation is **successful** show a **message** and **redirect** to all furniture.

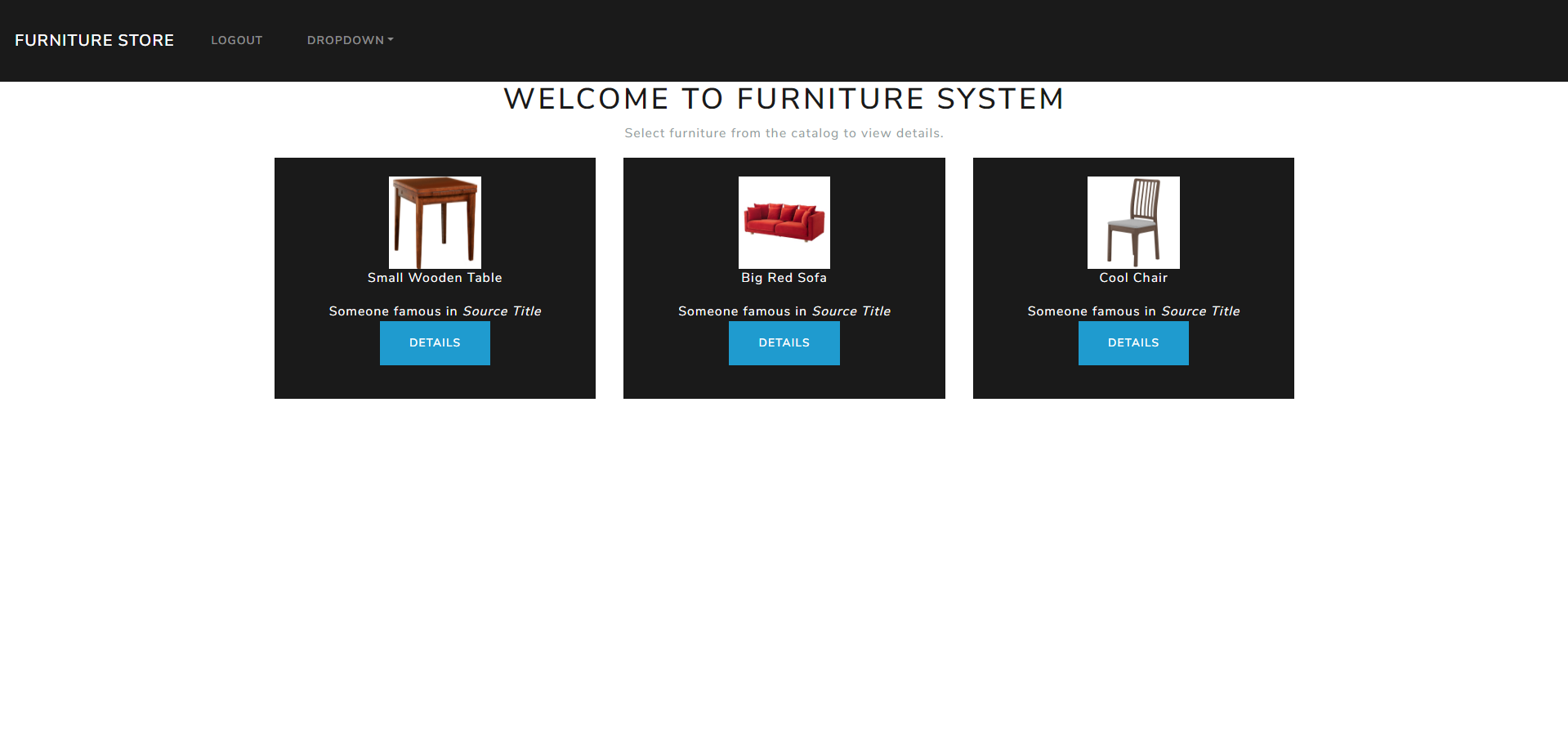


### All Furniture

Create a **model** to hold (id, make, model, year, description, price, and image).

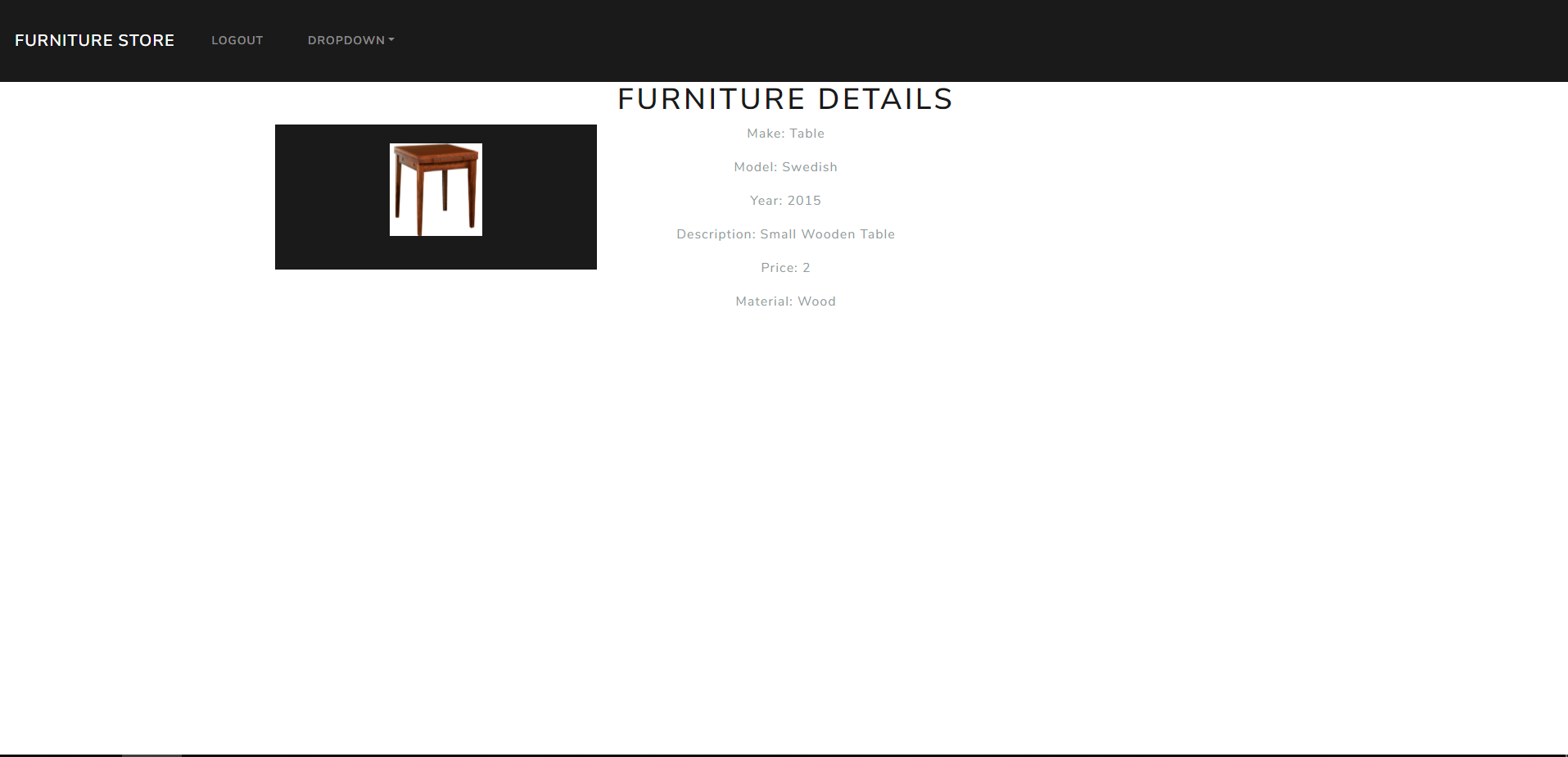
Call the service and **list all furniture** inside the store. Show only the **first 10 symbols** of the description. Use the **async pipe** to load the data.

Create a router link to **redirect to details**.



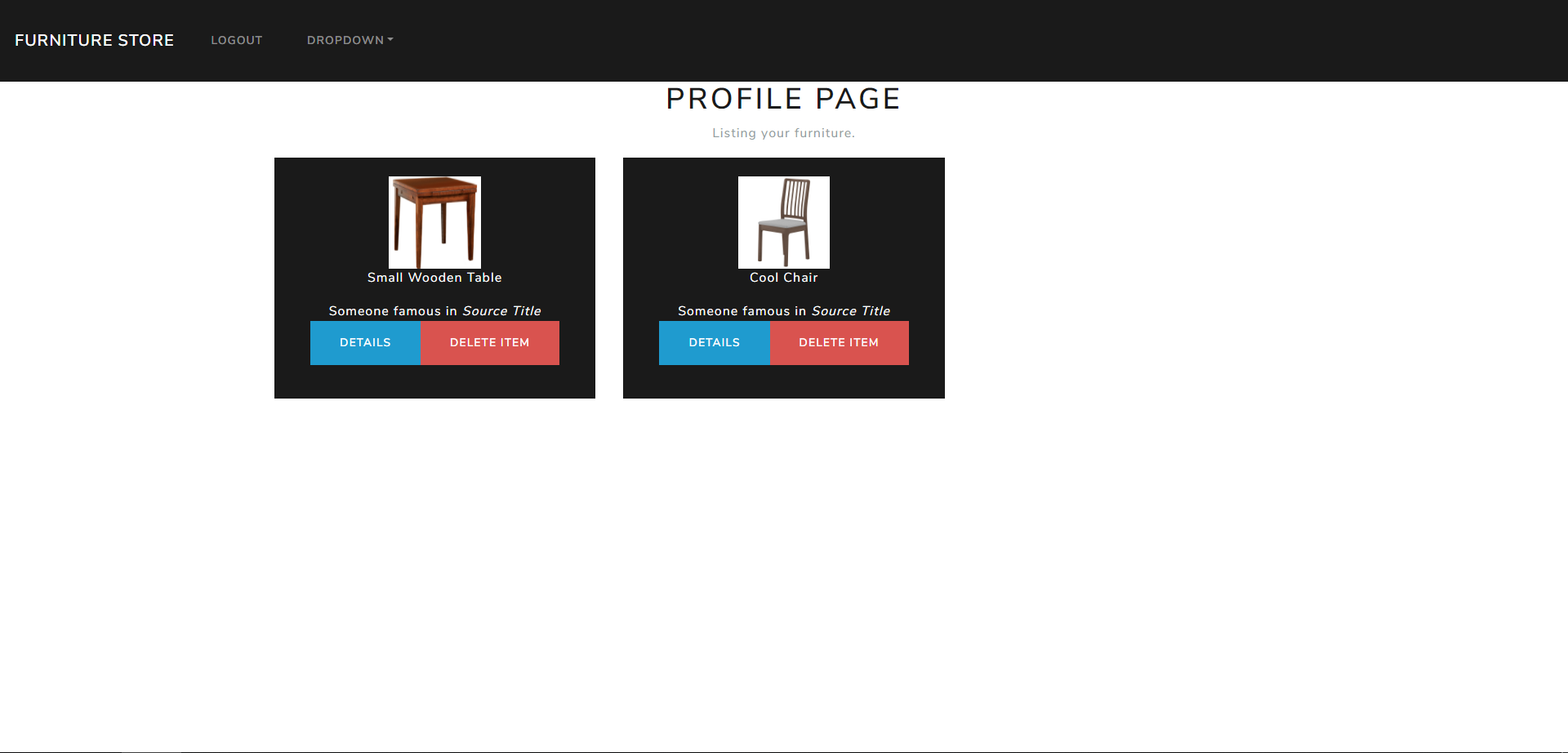
### Furniture Details

Use the **previously** created model, get the **id** from the **URL**, call the service and **display** the information. You can also create a **resolver**.



### My Furniture

List all **user created** furniture, link to **redirect** to details and ability to **delete** your own items.



## \* Admin Functionality

Starting the server adds an admin with **email: "admin@admin.com"** and **password: "Admin"**.

Add admin functionality in your Angular app by allowing the admin to **delete any** furniture at **furniture/all** and **edit** **any** furniture.

* Furniture by id (GET): **http://localhost:5000/furniture/:id**
* Edit furniture (PUT): **http://localhost:5000/furniture/edit/:id**

Copy the HTML template from **create furniture** component and **modify** it to work for the edit component

Now we have a fully functioning **CRUD** application ☺