**Foodbox**

**Back-End Code:**

pom.xml

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.6.4</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.simplilearn</groupId>

<artifactId>foodbox</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>Foodbox-Backend</name>

<description>Full Stack Java Developer Capstone Project</description>

<properties>

<java.version>11</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-rest</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<configuration>

<excludes>

<exclude>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</exclude>

</excludes>

</configuration>

</plugin>

</plugins>

</build>

</project>

**Entity package**

Address.java

**package** com.simplilearn.foodbox.entity;

**import** lombok.Data;

**import** javax.persistence.\*;

@Entity

@Table(name="address")

@Data

**public** **class** Address {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "id")

**private** Long id;

}

Cuisine.java

**package** com.simplilearn.foodbox.entity;

**import** lombok.Getter;

**import** lombok.Setter;

**import** javax.persistence.\*;

@Entity

@Table(name="cuisine")

// @Data -- known bug

@Getter

@Setter

**public** **class** Cuisine {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "id")

**private** Long id;

@Column(name = "cuisine\_name")

**private** String cuisine\_name;

**public** String getCuisine\_name() {

**return** cuisine\_name;

}

**public** **void** setCuisine\_name(String cuisine\_name) {

**this**.cuisine\_name = cuisine\_name;

}

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

}

Product.java

**package** com.simplilearn.foodbox.entity;

**import** lombok.Data;

**import** org.hibernate.annotations.CreationTimestamp;

**import** org.hibernate.annotations.UpdateTimestamp;

**import** javax.persistence.\*;

**import** java.math.BigDecimal;

**import** java.util.Date;

@Entity

@Table(name="product")

@Data

**public** **class** Product {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "id")

**private** Long id;

@Column(name = "cuisine\_id")

**private** **int** cuisine;

@Column(name = "name")

**private** String name;

@Column(name = "description")

**private** String description;

@Column(name = "price")

**private** BigDecimal price;

@Column(name = "image\_url")

**private** String imageUrl;

@Column(name = "date\_created")

@CreationTimestamp

**private** Date dateCreated;

@Column(name = "last\_updated")

@UpdateTimestamp

**private** Date lastUpdated;

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getDescription() {

**return** description;

}

**public** **void** setDescription(String description) {

**this**.description = description;

}

**public** BigDecimal getPrice() {

**return** price;

}

**public** **void** setPrice(BigDecimal price) {

**this**.price = price;

}

**public** String getImageUrl() {

**return** imageUrl;

}

**public** **void** setImageUrl(String imageUrl) {

**this**.imageUrl = imageUrl;

}

**public** Date getDateCreated() {

**return** dateCreated;

}

**public** **void** setDateCreated(Date dateCreated) {

**this**.dateCreated = dateCreated;

}

**public** Date getLastUpdated() {

**return** lastUpdated;

}

**public** **void** setLastUpdated(Date lastUpdated) {

**this**.lastUpdated = lastUpdated;

}

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

}

User.java

**package** com.simplilearn.foodbox.entity;

**import** lombok.Data;

**import** javax.persistence.\*;

@Entity

@Table(name="users")

@Data

**public** **class** User {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "id")

**private** Long id;

@Column(name = "username")

**private** String username;

@Column(name = "email")

**private** String email;

@Column(name = "password")

**private** String password;

@Column(name = "phone")

**private** **int** phone;

@Column(name = "street")

**private** String street;

@Column(name = "city")

**private** String city;

@Column(name = "zipcode")

**private** String zipcode;

**public** String getUsername() {

**return** username;

}

**public** **void** setUsername(String username) {

**this**.username = username;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** **int** getPhone() {

**return** phone;

}

**public** **void** setPhone(**int** phone) {

**this**.phone = phone;

}

**public** String getStreet() {

**return** street;

}

**public** **void** setStreet(String street) {

**this**.street = street;

}

**public** String getCity() {

**return** city;

}

**public** **void** setCity(String city) {

**this**.city = city;

}

**public** String getZipcode() {

**return** zipcode;

}

**public** **void** setZipcode(String zipcode) {

**this**.zipcode = zipcode;

}

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

}

**Config package**

MyDataRestConfig.java

**package** com.simplilearn.foodbox.config;

**import** com.simplilearn.foodbox.entity.User;

**import** com.simplilearn.foodbox.entity.Cuisine;

**import** com.simplilearn.foodbox.entity.Product;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.data.rest.core.config.RepositoryRestConfiguration;

**import** org.springframework.data.rest.webmvc.config.RepositoryRestConfigurer;

**import** org.springframework.http.HttpMethod;

**import** org.springframework.web.servlet.config.annotation.CorsRegistry;

**import** javax.persistence.EntityManager;

**import** javax.persistence.metamodel.EntityType;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.Set;

@Configuration

**public** **class** MyDataRestConfig **implements** RepositoryRestConfigurer {

**private** EntityManager entityManager;

@Autowired

**public** MyDataRestConfig(EntityManager theEntityManager) {

entityManager = theEntityManager;

}

@Override

**public** **void** configureRepositoryRestConfiguration(RepositoryRestConfiguration config, CorsRegistry cors) {

HttpMethod[] theUnsupportedActions = {HttpMethod.***PUT***, HttpMethod.***POST***, HttpMethod.***DELETE***};

// disable HTTP methods for Product: PUT, POST and DELETE

config.getExposureConfiguration()

.forDomainType(Product.**class**)

.withItemExposure((metdata, httpMethods) -> httpMethods.disable(theUnsupportedActions))

.withCollectionExposure((metdata, httpMethods) -> httpMethods.disable(theUnsupportedActions));

// disable HTTP methods for Cuisine: PUT, POST and DELETE

config.getExposureConfiguration()

.forDomainType(Cuisine.**class**)

.withItemExposure((metdata, httpMethods) -> httpMethods.disable(theUnsupportedActions))

.withCollectionExposure((metdata, httpMethods) -> httpMethods.disable(theUnsupportedActions));

// disable HTTP methods for User: PUT, POST and DELETE

config.getExposureConfiguration()

.forDomainType(User.**class**)

.withItemExposure((metdata, httpMethods) -> httpMethods.disable(theUnsupportedActions))

.withCollectionExposure((metdata, httpMethods) -> httpMethods.disable(theUnsupportedActions));

// call an internal helper method

exposeIds(config);

}

**private** **void** exposeIds(RepositoryRestConfiguration config) {

// expose entity ids

// - get a list of all entity classes from the entity manager

Set<EntityType<?>> entities = entityManager.getMetamodel().getEntities();

// - create an array of the entity types

List<Class> entityClasses = **new** ArrayList<>();

// - get the entity types for the entities

**for** (EntityType tempEntityType : entities) {

entityClasses.add(tempEntityType.getJavaType());

}

// - expose the entity ids for the array of entity/domain types

Class[] domainTypes = entityClasses.toArray(**new** Class[0]);

config.exposeIdsFor(domainTypes);

}

}

**DAO package**

CuisineRepository.java

**package** com.simplilearn.foodbox.dao;

**import** com.simplilearn.foodbox.entity.Cuisine;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** org.springframework.data.rest.core.annotation.RepositoryRestResource;

**import** org.springframework.web.bind.annotation.CrossOrigin;

@CrossOrigin("http://localhost:4200")

**public** **interface** CuisineRepository **extends** JpaRepository<Cuisine, Long> {

}

ProductRepository.java

**package** com.simplilearn.foodbox.dao;

**import** com.simplilearn.foodbox.entity.Product;

**import** org.springframework.data.domain.Page;

**import** org.springframework.data.domain.Pageable;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** org.springframework.web.bind.annotation.CrossOrigin;

**import** org.springframework.web.bind.annotation.RequestParam;

@CrossOrigin("http://localhost:4200")

**public** **interface** ProductRepository **extends** JpaRepository<Product, Long> {

// all products

Page<Product> findAllByOrderByNameAsc(Pageable pageable);

Page<Product> findAllByOrderByDateCreatedDesc(Pageable pageable);

// by Cuisines

Page<Product> findByCuisine(@RequestParam("cuisine") **int** cuisine, Pageable pageable);

Page<Product> findByCuisineOrderByNameAsc(@RequestParam("cuisine") **int** cuisine, Pageable pageable);

Page<Product> findByCuisineOrderByDateCreatedDesc(@RequestParam("cuisine") **int** cuisine, Pageable pageable);

// for searching

Page<Product> findByNameContaining(@RequestParam("name") String name, Pageable pageable);

Page<Product> findByNameContainingOrderByNameAsc(@RequestParam("name") String name, Pageable pageable);

Page<Product> findByNameContainingOrderByDateCreatedDesc(@RequestParam("name") String name, Pageable pageable);

}

UserRepository.java

**package** com.simplilearn.foodbox.dao;

**import** com.simplilearn.foodbox.entity.User;

**import** org.springframework.data.domain.Page;

**import** org.springframework.data.domain.Pageable;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** org.springframework.web.bind.annotation.CrossOrigin;

**import** org.springframework.web.bind.annotation.RequestParam;

@CrossOrigin("http://localhost:4200")

**public** **interface** UserRepository **extends** JpaRepository<User, Long> {

Page<User> findByUsername(@RequestParam("username") String username, Pageable pageable);

}

**Controller package**

ProductController.java

**package** com.simplilearn.foodbox.controller;

**import** com.simplilearn.foodbox.dao.ProductRepository;

**import** com.simplilearn.foodbox.entity.Product;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.data.rest.webmvc.ResourceNotFoundException;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.\*;

**import** java.util.HashMap;

**import** java.util.List;

**import** java.util.Map;

@CrossOrigin(origins = "http://localhost:4200")

@RestController

@RequestMapping("/api/v1/")

**public** **class** ProductController {

@Autowired

**private** ProductRepository productRepository;

// get all products

@GetMapping("/products")

**public** List<Product> getAllproducts(){

**return** productRepository.findAll();

}

// create product rest api

@PostMapping("/products")

**public** Product createProduct(@RequestBody Product product) {

**return** productRepository.save(product);

}

// get product by id rest api

@GetMapping("/products/{id}")

**public** ResponseEntity<Product> getproductById(@PathVariable Long id) {

Product product = productRepository.findById(id)

.orElseThrow(() -> **new** ResourceNotFoundException("product not exist with id :" + id));

**return** ResponseEntity.*ok*(product);

}

// update product rest api

@PutMapping("/products/{id}")

**public** ResponseEntity<Product> updateProduct(@PathVariable Long id, @RequestBody Product productDetails){

Product product = productRepository.findById(id)

.orElseThrow(() -> **new** ResourceNotFoundException("product not exist with id :" + id));

product.setName(productDetails.getName());

product.setDescription(productDetails.getDescription());

product.setPrice(productDetails.getPrice());

product.setImageUrl(productDetails.getImageUrl());

Product updatedProduct = productRepository.save(product);

**return** ResponseEntity.*ok*(updatedProduct);

}

// delete product rest api

@DeleteMapping("/products/{id}")

**public** ResponseEntity<Map<String, Boolean>> deleteProduct(@PathVariable Long id){

Product product = productRepository.findById(id)

.orElseThrow(() -> **new** ResourceNotFoundException("product not exist with id :" + id));

productRepository.delete(product);

Map<String, Boolean> response = **new** HashMap<>();

response.put("deleted", Boolean.***TRUE***);

**return** ResponseEntity.*ok*(response);

}

}

UserController.java

**package** com.simplilearn.foodbox.controller;

**import** com.simplilearn.foodbox.dao.UserRepository;

**import** com.simplilearn.foodbox.entity.User;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.data.rest.webmvc.ResourceNotFoundException;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.\*;

**import** java.util.List;

@CrossOrigin(origins = "http://localhost:4200")

@RestController

@RequestMapping("/api/v1/")

**public** **class** UserController {

@Autowired

**private** UserRepository userRepository;

// get all users

@GetMapping("/users")

**public** List<User> getAllUsers(){

**return** userRepository.findAll();

}

// create user rest api

@PostMapping("/users")

**public** User createUser(@RequestBody User user) {

**return** userRepository.save(user);

}

// get user by id rest api

@GetMapping("/users/{id}")

**public** ResponseEntity<User> getUserById(@PathVariable Long id) {

User user = userRepository.findById(id)

.orElseThrow(() -> **new** ResourceNotFoundException("user not exist with id :" + id));

**return** ResponseEntity.*ok*(user);

}

}

**application.properties**

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/foodbox

spring.datasource.username=Work

spring.datasource.password=Simplilearn@123

spring.data.rest.base-path=/api

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

**Front-End code:**

**Common:**

address.spec.ts

import { Address } from './address';

describe('Address', () => {

  it('should create an instance', () => {

    expect(new Address()).toBeTruthy();

  });

});

address.ts

export class Address {

    id: string;

    zipCode: string;

    street: string;

    city: string;

}

cart-item.spec.ts

import { CartItem } from './cart-item';

describe('CartItem', () => {

  it('should create an instance', () => {

    expect(new CartItem()).toBeTruthy();

  });

});

cart-item.ts

import { Product } from './product';

export class CartItem {

    id: string;

    name: string;

    imageUrl: string;

    price: number;

quantity: number;

    constructor( product: Product) {

        this.id = product.id;

        this.name = product.name;

        this.imageUrl = product.imageUrl;

        this.price = product.price;

        this.quantity = 1;

    }

}

product-category.spec.ts

import { ProductCategory } from './product-category';

describe('ProductCategory', () => {

  it('should create an instance', () => {

    expect(new ProductCategory()).toBeTruthy();

  });

});

product-category.ts

export class ProductCategory {

    id: number;

    cuisine\_name: string;

}

product.spec.ts

import { Product } from './product';

describe('Product', () => {

  it('should create an instance', () => {

    expect(new Product()).toBeTruthy();

  });

});

product.ts

export class Product {

    id: string;

    name: string;

    description: string;

    price: number;

    imageUrl: string;

    dateCreated: Date;

    lastUpdate: Date;

    cuisine: number;

}

user.spec.ts

import { User } from './user';

describe('User', () => {

  it('should create an instance', () => {

    expect(new User()).toBeTruthy();

  });

});

user.ts

export class User {

    id: number;

    username: string;

    password: string;

    email: string;

    type: number;

    phone: number;

    zipcode: string;

    street: string;

    city: string;

}

**Components:**

**add-product**

add-product.component.css

.login-form {

    width: 340px;

    margin: 50px auto;

}

.login-form form {

    margin-bottom: 15px;

    background: #f7f7f7;

    box-shadow: 0px 2px 2px rgba(0, 0, 0, 0.3);

    padding: 30px;

}

.login-form h2 {

    margin: 0 0 15px;

}

.form-control, .btn {

    min-height: 38px;

    border-radius: 2px;

}

.btn {

    font-size: 15px;

    font-weight: bold;

}

add-product.component.html

<br><br>

<div class="login-form" style="width:44%" >

    <form   (ngSubmit) = "handleAddProduct()">

        <h2 class="text-center">Add a New Product</h2>

        <div class="form-group">

            Name:<input [(ngModel)] = "product.name" name = "name" type="text" class="form-control"   required="required">

        </div>

        <br>

        <div class="form-group">

            Description:

            <input [(ngModel)] = "product.description" name = "description" type="text" class="form-control"   required="required" >

        </div>

        <br>

        <div class="form-group">

            Price:

            <input [(ngModel)] = "product.price" name = "price" type="text" class="form-control"   required="required">

        </div>

        <br>

        <div class="form-group">

            Image URL:

            <input [(ngModel)] = "product.imageUrl" name = "imageUrl" type="text" class="form-control" required="required">

        </div>

        <br>

        <div class="form-group">

            Cuisine:

            <br>

            <select [(ngModel)]="product.cuisine" name = "selectedCuisine">

                <option \*ngFor="let cuisine of productCuisines" [ngValue]="cuisine.id">{{cuisine.cuisine\_name}}</option>

            </select>

        </div>

        <br>

        <div class="form-group">

            <button type="submit" class="btn btn-success btn-block">Add</button>

        </div>

    </form>

</div>

add-product.component.spec.ts

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { AddProductComponent } from './add-product.component';

describe('AddProductComponent', () => {

  let component: AddProductComponent;

  let fixture: ComponentFixture<AddProductComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ AddProductComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(AddProductComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

add-product.component.ts

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

import { LoginService } from 'src/app/services/login.service';

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

import {Router} from "@angular/router"

import { Product } from 'src/app/common/product';

import { ActivatedRoute } from '@angular/router';

import { ProductService } from 'src/app/services/product.service';

import { ProductCategory } from 'src/app/common/product-category';

@Component({

  selector: 'app-add-product',

  templateUrl: './add-product.component.html',

  styleUrls: ['./add-product.component.css']

})

export class AddProductComponent implements OnInit {

  product: Product = new Product();

  cuisine: ProductCategory = new ProductCategory();

  id: number = 0;

  productCuisines: ProductCategory[];

  selectedCuisine:number = 1;

  constructor(

    private productService: ProductService,

    private route: ActivatedRoute) { }

  ngOnInit(): void {

     this.listProductCategories();

  }

  handleAddProduct() {

    this.productService.addProduct(this.product).subscribe( data =>{

    },

    error => console.log(error));

    console.log(this.product);

    window.location.replace('/products');

  }

  listProductCategories() {

    this.productService.getProductCategories().subscribe(

      data => {

        console.log('Product Categories=' + JSON.stringify(data));

        this.productCuisines = data;

      }

    );

  }

}

**Cart-details:**

cart-details.component.html

<div class="main-content">

    <div class="section-content section-content-p30">

        <div class="container-fluid">

            <div \*ngIf="username == null" class="alert alert-danger col-md-12" role="alert">

                You need to be registered in the system.

            </div>

            <div \*ngIf="username != null && cartItems.length > 0">

                <table class="table table-bordered">

                    <tr>

                        <th width="20%">Product Image</th>

                        <th width="50%">Product Detail</th>

                        <th width="30%"></th>

                    </tr>

                    <tr \*ngFor="let tempCartItem of cartItems">

                        <td>

                            <img src="{{ tempCartItem.imageUrl }}" class="img-responsive" width="150px" />

                        </td>

                        <td>

                            <p>{{ tempCartItem.name }}</p>

                            <p>{{ tempCartItem.unitPrice | currency: 'USD' }}</p>

                        </td>

                        <td>

                            <div class="items">

                                <label>Quantity:</label>

                                <div class="row no-gutters">

                                    <div class="col">

                                        <button (click)="incrementQuantity(tempCartItem)" class="btn btn-primary btn-sm">

                                            <i class="fas fa-plus"></i>

                                        </button>

                                    </div>

                                    <div class="col ml-4 mr-2">

                                        {{ tempCartItem.quantity }}

                                    </div>

                                    <div class="col">

                                        <button (click)="decrementQuantity(tempCartItem)" class="btn btn-primary btn-sm">

                                            <i class="fas fa-minus"></i>

                                        </button>

                                    </div>

                                    <div class="col-8"></div>

                                </div>

                            </div>

                            <button (click)="remove(tempCartItem)" class="au-btn-submit" style="width:22px; background: red; color: white"  >Remove</button>

                            <p class="mt-2"></p>

                        </td>

                    </tr>

                    <tr>

                        <td colspan="2"></td>

                        <td style="font-weight: bold">

                            <p>Total Quantity: {{ totalQuantity }}</p>

                            <p>Total Price: {{ totalPrice | currency: 'USD' }}</p>

                            <a (click)="getTotalPrice()" routerLink="/checkout"><button class="au-btn-submit" style="width:181px; height: 55px; background: green; color: white"  >Confirm</button></a>

                            <br>

                        </td>

                    </tr>

                </table>

            </div>

            <!-- if cart is empty then display a message -->

            <div \*ngIf="cartItems.length == 0" class="alert alert-warning col-md-12" role="alert">

                Your shopping cart is empty.

            </div>

        </div>

    </div>

</div>

cart-details.component.spec.ts

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { CartDetailsComponent } from './cart-details.component';

describe('CartDetailsComponent', () => {

  let component: CartDetailsComponent;

  let fixture: ComponentFixture<CartDetailsComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ CartDetailsComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(CartDetailsComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

cart-details.component.ts

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

import { CartItem } from 'src/app/common/cart-item';

import { CartService } from 'src/app/services/cart.service';

@Component({

  selector: 'app-cart-details',

  templateUrl: './cart-details.component.html',

  styleUrls: ['./cart-details.component.css']

})

export class CartDetailsComponent implements OnInit {

  cartItems: CartItem[] = [];

  totalPrice: number = 0;

  totalQuantity: number = 0;

  username:string = localStorage.getItem('username');

  constructor(private cartService: CartService) { }

  ngOnInit(): void {

    this.listCartDetails();

  }

  listCartDetails() {

    // get a handle to the cart items

    this.cartItems = this.cartService.cartItems;

    // subscribe to the cart totalPrice

    this.cartService.totalPrice.subscribe(

      data => this.totalPrice = data

    );

    // subscribe to the cart totalQuantity

    this.cartService.totalQuantity.subscribe(

      data => this.totalQuantity = data

    );

    // compute cart total price and quantity

    this.cartService.computeCartTotals();

  }

  incrementQuantity(theCartItem: CartItem) {

    this.cartService.addToCart(theCartItem);

  }

  decrementQuantity(theCartItem: CartItem) {

    this.cartService.decrementQuantity(theCartItem);

  }

  remove(theCartItem: CartItem) {

    this.cartService.remove(theCartItem);

  }

  getTotalPrice(){

    localStorage.setItem('totalPrice', this.totalPrice.toString());

  }

}

**Cart-status:**

cart-status.component.html

<div class="cart-area d-n">

    <a routerLink="/cart-details">

        <div class="total">{{ totalPrice | currency: 'USD' }}

            <span>{{ totalQuantity }}</span>

        </div>

        <i class="fa fa-shopping-cart" aria-hidden="true"></i>

    </a>

</div>

cart-status.component.spec.ts

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { CartStatusComponent } from './cart-status.component';

describe('CartStatusComponent', () => {

  let component: CartStatusComponent;

  let fixture: ComponentFixture<CartStatusComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ CartStatusComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(CartStatusComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

cart-status.component.ts

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

import { CartService } from 'src/app/services/cart.service';

@Component({

  selector: 'app-cart-status',

  templateUrl: './cart-status.component.html',

  styleUrls: ['./cart-status.component.css']

})

export class CartStatusComponent implements OnInit {

  totalPrice: number = 0.00;

  totalQuantity: number = 0;

  constructor(private cartService: CartService) { }

  ngOnInit(): void {

    this.updateCartStatus();

  }

  updateCartStatus() {

    // subscribe to the cart totalPrice

    this.cartService.totalPrice.subscribe(

      data => this.totalPrice = data

    );

    // subscribe to the cart totalQuantity

    this.cartService.totalQuantity.subscribe(

      data => this.totalQuantity = data

    );

  }

}

**Check-out:**

check-out.component.html

<div class="main-content">

    <div class="section-content section-content-p30">

        <div class="container-fluid">

        <div style="margin-left: 15px;">

        <h6  style="font-size:22px;">

            <p>Thank you <b>{{username}}</b> for purchasing our products. Your support and trust in us are much appreciated. </p>

            <p>The total price of your bill is <b>${{totalPrice}}</b> </p>

            <p>The products will be dilivred to you soon to your address as following:</p>

            <p><b>city:</b> {{this.users[0]?.city}}</p>

            <p><b>street:</b> {{this.users[0]?.street}}</p>

            <p><b>zipcode:</b> {{this.users[0]?.zipcode}}</p>

        </h6>

    </div>

        <a (click)="backHome()"><button class="au-btn-submit" style="width:181px; height: 55px; background: green; color: white"  >Confirm</button></a>

        </div>

    </div>

</div>

check-out.component.ts

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

import { CartService } from 'src/app/services/cart.service';

import { User } from 'src/app/common/user';

import { LoginService } from 'src/app/services/login.service';

@Component({

  selector: 'app-check-out',

  templateUrl: './check-out.component.html',

  styleUrls: ['./check-out.component.css']

})

export class CheckOutComponent implements OnInit {

  username:string = localStorage.getItem('username');

  totalPrice:string = localStorage.getItem('totalPrice');

  users: Array<User> = new Array();

  constructor(private loginService: LoginService) { }

  ngOnInit(): void {

    this.getUser()

  }

  getUser() {

    this.loginService.getUser(this.username).subscribe(

      data => {

       this.users = data;

      }

      ,error => console.log(error));

}

backHome(){

  window.location.replace('/products');

}

}

**login:**

login.component.html

<br>

<br>

<div class="login-form">

    <form  (ngSubmit)="onSubmit()" >

        <h2 class="text-center">Log in</h2>

        <div>

        <div class="form-group">

            <input [(ngModel)] = "user.username" name = "username" type="text" class="form-control" placeholder="Username" required="required">

        </div>

        <br>

        <div class="form-group">

            <input [(ngModel)] = "user.password" name = "password"  type="password" class="form-control" placeholder="Password" required="required">

        </div>

        <br>

        <div class="form-group">

            <button type="submit" class="btn btn-primary btn-block">Log in</button>

        </div>

    </div>

    </form>

    <p class="text-center"><a routerLink="/register">Create an Account</a></p>

</div>

login.component.ts

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

import { LoginService } from 'src/app/services/login.service';

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

import {Router} from "@angular/router"

import { User } from 'src/app/common/user';

@Component({

  selector: 'app-login',

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

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

})

export class LoginComponent implements OnInit {

  username:string = localStorage.getItem('username');

  user:User = new User();

  user2:User = new User();

  users: Array<User> = new Array();

  username\_value:string;

  constructor(private loginService: LoginService,

              private router: Router) { }

  ngOnInit(): void {

  }

  handleLogin() {

    this.loginService.getUser(this.user.username).subscribe(

      data => {

       this.users = data;

      }

      ,error => console.log(error));

    // check data entered

    if(this.users[0]?.username == this.user.username){

      localStorage.setItem('username', this.user.username);

      console.log("YES");

      window.location.replace('/products');

    }

    else{

      console.log("No");

    }

  }

  onSubmit(){

    console.log(this.user);

    this.handleLogin();

  }

}

**Login-status:**

login-status.component.html

<div class="form-header">

<button \*ngIf="username == null"  routerLink="/login"    class="au-btn-submit login">Login</button>

<button \*ngIf="username != null" routerLink="/products" (click)="logout()"  class="au-btn-submit logout">Logout</button>

</div>

login-status.component.ts

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

import { Router } from '@angular/router';

@Component({

  selector: 'app-login-status',

  templateUrl: './login-status.component.html',

  styleUrls: ['./login-status.component.css']

})

export class LoginStatusComponent implements OnInit {

  username:string = localStorage.getItem('username');

  constructor(private route: Router) { }

  ngOnInit(): void {

  }

  logout(){

    this.username = null;

    localStorage.removeItem("username");

    location.reload();

  }

}

**product-category-menu:**

product-category-menu.component.html

<div class="menu-sidebar-content js-scrollbar1">

    <nav class="navbar-sidebar">

        <ul class="list-unstyled navbar-list">

            <li style="margin-left:-17px;font-size:27px; font-family: Helvetica, sans-serif;"> Cuisines:</li>

            <li style="line-height: 2.3;">

                <i class="fa fa-angle-double-right" style="font-size:14px"></i>

                <a style="text-decoration:none" routerLink="/category" [ngClass]="{'active-link':clicked}"  (click)="Clicked()">

                 All

                </a>

            </li>

            <li style="line-height: 2.3;" \*ngFor="let tempProductCategory of productCategories">

                <i class="fa fa-angle-double-right" style="font-size:14px"></i>

                <a style="text-decoration:none" routerLink="/category/{{ tempProductCategory.id }}" routerLinkActive="active-link" (click)="UnClicked()">

                    {{ tempProductCategory.cuisine\_name }}

                </a>

                <br>

            </li>

            <br>

            <li>

                <button \*ngIf="username == 'admin'"  routerLink="/add-product"    class="btn btn-primary btn-sm">Add Product</button>

            </li>

        </ul>

    </nav>

</div>

product-category-menu.component.ts

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

import { ProductCategory } from 'src/app/common/product-category';

import { ProductService } from 'src/app/services/product.service';

@Component({

  selector: 'app-product-category-menu',

  templateUrl: './product-category-menu.component.html',

  styleUrls: ['./product-category-menu.component.css']

})

export class ProductCategoryMenuComponent implements OnInit {

  username:string = localStorage.getItem('username');

  productCategories: ProductCategory[];

  clicked: boolean = false;

  constructor(private productService: ProductService) { }

  ngOnInit() {

    this.listProductCategories();

  }

  listProductCategories() {

    this.productService.getProductCategories().subscribe(

      data => {

        console.log('Product Categories=' + JSON.stringify(data));

        this.productCategories = data;

      }

    );

  }

  Clicked() {

    this.clicked = true;

}

  UnClicked() {

    this.clicked = false;

    }

}

**product-details:**

product-details.component.html

<div class="detail-section">

    <div class="container-fluid">

        <img src="{{ product.imageUrl }}" class="detail-img">

        <h3>{{ product.name }}</h3>

        <br>

        <h6><i>{{ cuisine.cuisine\_name }} Cuisine </i></h6>

        <div class="price">{{ product.price | currency:'USD' }}</div>

        <button (click)="addToCart()" class="btn btn-primary btn-sm">Add to cart</button>

        <hr>

        <h4>Description</h4>

        <p>{{ product.description }}</p>

        <a routerLink="/products" class="mt-5">Back to Product List</a>

    </div>

</div>

product-details.component.ts

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

import { Product } from 'src/app/common/product';

import { ProductCategory } from 'src/app/common/product-category';

import { ProductService } from 'src/app/services/product.service';

import { ActivatedRoute } from '@angular/router';

import { CartService } from 'src/app/services/cart.service';

import { CartItem } from 'src/app/common/cart-item';

@Component({

  selector: 'app-product-details',

  templateUrl: './product-details.component.html',

  styleUrls: ['./product-details.component.css']

})

export class ProductDetailsComponent implements OnInit {

  product: Product = new Product();

  cuisine: ProductCategory = new ProductCategory();

  constructor(private productService: ProductService,

              private cartService: CartService,

              private route: ActivatedRoute) { }

  ngOnInit(): void {

    //const theProductId: number = +this.route.snapshot.paramMap.get('id');

    this.route.paramMap.subscribe(() => {

      this.handleProductDetails();

    })

  }

  handleProductDetails() {

    // get the "id" param string. convert string to a number using the "+" symbol

    const theProductId: number = +this.route.snapshot.paramMap.get('id');

    this.productService.getProduct(theProductId).subscribe(

      data => {

        this.product = data;

      }

    )

    this.productService.getProductCuisine(theProductId).subscribe(

      data => {

        this.cuisine = data;

      }

    )

  }

  addToCart() {

    console.log(`Adding to cart: ${this.product.name}, ${this.product.price}`);

    const theCartItem = new CartItem(this.product);

    this.cartService.addToCart(theCartItem);

  }

}

**product-list:**

product-list.component.html

<div class="main-content" >

    <div class="section-content section-content-p30">

        <div class="container-fluid">

            <div style="margin-top: -23px;" \*ngIf="username != null" class="alert alert-warning col-xs-12" role="alert">

                Welecome {{username}}. </div>

            <div class="row">

                <!-- loop over the collection of products -->

                <div \*ngFor="let tempProduct of products" class="col-md-3">

                    <div class="card" style="width: 12rem; margin: 21px 18px;" >

                        <a routerLink="/products/{{ tempProduct.id }}">

                            <img style="width:240px;height:160px;" class="card-img-top"

                                src="{{ tempProduct.imageUrl }}">

                        </a>

                        <div >

                            <a style="margin-left: 13px; margin-top: 23px;" routerLink="/products/{{ tempProduct.id }}">

                                <h6 style="font-size:20px;"> {{ tempProduct.name }}</h6>

                            </a>

                        </div>

                        <div class="card-body" style="margin-top: -23px;"  >

                            <div class="price card-text">{{ tempProduct.price | currency:'USD' }}</div>

                            <button \*ngIf="username != 'admin'" (click)="addToCart(tempProduct)"

                                class="btn btn-primary btn-sm">Add to cart</button>

                            <button \*ngIf="username == 'admin'" routerLink="/update-product/{{ tempProduct.id }}"

                                class="btn btn-success btn-sm">Edit</button>

                                &nbsp;

                            <button \*ngIf="username == 'admin'" (click)="deleteProduct(tempProduct.id)"

                                class="btn btn-danger btn-sm">Remove</button>

                        </div>

                    </div>

                </div>

                <!-- if products empty then display a message-->

                <div \*ngIf="products?.length == 0" class="alert alert-danger col-md-12" role="alert">

                    No products found.

                </div>

            </div>

            <!-- begin footer -->

            <div class="footer-pagination">

                <div class="row">

                    <div class="col-md-6"></div>

                    <div class="col-md-12">

                        <div class="row">

                            <div class="col-md-6" style="float: right;">

                                <ngb-pagination [(page)]="thePageNumber" [pageSize]="thePageSize"

                                    [collectionSize]="theTotalElements" [maxSize]="5" [boundaryLinks]="true"

                                    (pageChange)="listProducts()">

                                </ngb-pagination>

                            </div>

                            <div class="col-md-3  " style="float: left;">

                                <span class="mr-2">Sort by</span>

                                <select (change)="updateSortingMethod($event.target.value)">

                                    <option>Name</option>

                                    <option>First Created</option>

                                    <option selected="true">Last Created</option>

                                </select>

                            </div>

                            <div class="col-md-3  " style="float: left;">

                                <span class="mr-2">Page Size</span>

                                <select (change)="updatePageSize($event.target.value)">

                                    <option>2</option>

                                    <option selected="true">5</option>

                                    <option>10</option>

                                    <option>20</option>

                                    <option>50</option>

                                </select>

                            </div>

                        </div>

                    </div>

                </div>

            </div>

            <!-- end footer -->

        </div>

    </div>

</div>

product-list.component.ts

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

import { ProductService } from 'src/app/services/product.service';

import { Product } from 'src/app/common/product';

import { ActivatedRoute } from '@angular/router';

import { timeoutWith } from 'rxjs/operators';

import { CartItem } from 'src/app/common/cart-item';

import { CartService } from 'src/app/services/cart.service';

@Component({

  selector: 'app-product-list',

  templateUrl: './product-list.component.html',

  styleUrls: ['./product-list.component.css']

})

export class ProductListComponent implements OnInit {

  username:string = localStorage.getItem('username');

  products: Product[] = [];

  currentCategoryId: number = 1;

  previousCategoryId: number = 1;

  searchMode: boolean = false;

  // new properties for pagination

  thePageNumber: number = 1;

  thePageSize: number = 5;

  theTotalElements: number = 0;

  previousKeyword: string = null;

  sortingMethod: string = "last\_created";

  constructor(private productService: ProductService,

              private cartService: CartService,

              private route: ActivatedRoute) { }

  ngOnInit() {

    this.username = localStorage.getItem('username');

    this.route.paramMap.subscribe(() => {

      this.listProducts();

    });

  }

  listProducts() {

this.searchMode = this.route.snapshot.paramMap.has('keyword');

    if (this.searchMode) {

      this.handleSearchProducts();

    }

    else {

      this.handleListProducts();

    }

  }

  handleSearchProducts() {

const theKeyword: string = this.route.snapshot.paramMap.get('keyword');

    // if we have a different keyword than previous

// then set thePageNumber to 1

    if (this.previousKeyword != theKeyword) {

      this.thePageNumber = 1;

}

this.previousKeyword = theKeyword;

console.log(`keyword=${theKeyword}, thePageNumber=${this.thePageNumber}`);

    // now search for the products using keyword

    this.productService.searchProductsPaginate(this.thePageNumber - 1,

                                               this.thePageSize,

                                               theKeyword).subscribe(this.processResult());

  }

  handleListProducts() {

    // check if "id" parameter is available

const hasCategoryId: boolean = this.route.snapshot.paramMap.has('id');

    if (hasCategoryId) {

      // get the "id" param string. convert string to a number using the "+" symbol

      this.currentCategoryId = +this.route.snapshot.paramMap.get('id');

    //

    // Check if we have a different category than previous

    // Note: Angular will reuse a component if it is currently being viewed

    //

    // if we have a different category id than previous

    // then set thePageNumber back to 1

    if (this.previousCategoryId != this.currentCategoryId) {

      this.thePageNumber = 1;

}

    this.previousCategoryId = this.currentCategoryId;

    console.log(`currentCategoryId=${this.currentCategoryId}, thePageNumber=${this.thePageNumber}`);

    // now get the products for the given category id

    this.productService.getProductListPaginate(this.thePageNumber - 1,

                                               this.thePageSize,

                                               this.currentCategoryId,

                                               this.sortingMethod)

                                               .subscribe(this.processResult());

    }

    else {

      // not category id available ... default to category id 1

      this.productService.getAllProductListPaginate(this.thePageNumber - 1,

        this.thePageSize,

        this.currentCategoryId,

        this.sortingMethod)

        .subscribe(this.processResult());

}}

  processResult() {

    return data => {

      this.products = data.\_embedded.products;

      this.thePageNumber = data.page.number + 1;

      this.thePageSize = data.page.size;

      this.theTotalElements = data.page.totalElements;

    };

  }

  updatePageSize(pageSize: number) {

    this.thePageSize = pageSize;

    this.thePageNumber = 1;

    this.listProducts();

  }

  updateSortingMethod(sortingMethod: string) {

    this.sortingMethod = sortingMethod;

    this.listProducts();

  }

  addToCart(theProduct: Product) {

    console.log(`Adding to cart: ${theProduct.name}, ${theProduct.price}`);

    // TODO ... do the real work

    const theCartItem = new CartItem(theProduct);

    this.cartService.addToCart(theCartItem);

  }

  deleteProduct(theProductId:number){

     console.log(theProductId);

    this.productService.deleteProduct(theProductId).subscribe( data =>{

    }

    , error => console.log(error));

    window.location.replace('/products');

  }

}

**register:**

register.component.html

<br><br>

<div class="login-form" >

    <form   (ngSubmit) = "saveUser()">

        <h2 class="text-center">Register</h2>

        <div class="form-group">

            <input [(ngModel)] = "user.username" name = "username" type="text" class="form-control" placeholder="Username" required="required">

        </div>

        <br>

        <div class="form-group">

            <input [(ngModel)] = "user.email" name = "email" type="text" class="form-control" placeholder="Email" required="required">

        </div>

        <br>

        <div class="form-group">

            <input [(ngModel)] = "user.password" name = "password" type="password" class="form-control" placeholder="Password" required="required">

        </div>

        <br>

        <div class="form-group">

            <input [(ngModel)] = "user.phone" name = "phone" type="text" class="form-control" placeholder="phone" required="required">

        </div>

        <br>

        <h6>Address</h6>

        <div class="form-group">

            <input [(ngModel)] = "user.city" name = "city" type="text" class="form-control" placeholder="city" required="required">

        </div>

        <br>

        <div class="form-group">

            <input [(ngModel)] = "user.street" name = "street" type="text" class="form-control" placeholder="street" required="required">

        </div>

        <br>

        <div class="form-group">

            <input [(ngModel)] = "user.zipcode" name = "zipcode" type="text" class="form-control" placeholder="zipCode" required="required">

        </div>

        <br>

        <div class="form-group">

            <button type="submit" class="btn btn-primary btn-block">Register</button>

        </div>

    </form>

    <p class="text-center"><a routerLink="/login">Back to Login</a></p>

</div>

register.component.ts

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

import { LoginService } from 'src/app/services/login.service';

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

import {Router} from "@angular/router"

import { User } from 'src/app/common/user';

import { Address } from 'src/app/common/address';

@Component({

  selector: 'app-register',

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

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

})

export class RegisterComponent implements OnInit {

  registerFormGroup: FormGroup;

  user: User = new User();

  username\_value:string;

  theusername: string;

  thepassword: string;

  constructor(private loginService: LoginService,

              private formBuilder: FormBuilder,

              private router: Router) { }

  ngOnInit(): void {

  }

  saveUser(){

    this.loginService.createUser(this.user).subscribe( data =>{

      console.log(data);

    },

    error => console.log(error));

    this.router.navigate(['/products']);

  }

}

**remove-product:**

remove-product.component.html

<p>remove-product works!</p>

remove-product.component.ts

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

import { Product } from 'src/app/common/product';

import { ActivatedRoute } from '@angular/router';

import { ProductService } from 'src/app/services/product.service';

import { ProductCategory } from 'src/app/common/product-category';

@Component({

  selector: 'app-remove-product',

  templateUrl: './remove-product.component.html',

  styleUrls: ['./remove-product.component.css']

})

export class RemoveProductComponent implements OnInit {

  constructor(

    private productService: ProductService,

private route: ActivatedRoute) { }

  product: Product = new Product();

  ngOnInit(): void {

    this.deleteProduct();

  }

  deleteProduct(){

    const theProductId: number = +this.route.snapshot.paramMap.get('id');

    this.productService.deleteProduct(theProductId).subscribe( data =>{

      window.location.replace('/products');

    }

    , error => console.log(error));

    window.location.replace('/products');

  }

}

**search:**

search.component.html

<div class="form-header">

    <input #myInput type="text"

           placeholder="Search for products ..."

           class="au-input au-input-xl"

           (keyup.enter)="doSearch(myInput.value)"  />

    <button (click)="doSearch(myInput.value)" class="au-btn-submit">

        Search

    </button>

</div>

search.component.ts

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

import { Router } from '@angular/router';

@Component({

  selector: 'app-search',

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

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

})

export class SearchComponent implements OnInit {

  constructor(private router: Router) { }

  ngOnInit() {

  }

  doSearch(value: string) {

    console.log(`value=${value}`);

    this.router.navigateByUrl(`/search/${value}`);

  }

}

**update-product:**

update-product.component.html

<br><br>

<div class="login-form" style="width:44%" >

    <form   (ngSubmit) = "updateProduct()">

        <h2 class="text-center">Update Product</h2>

        <div class="form-group">

            Name:<input [(ngModel)] = "product.name" name = "name" type="text" class="form-control" placeholder={{product.name}} required="required">

        </div>

        <br>

        <div class="form-group">

            Description:

            <input [(ngModel)] = "product.description" name = "description" type="text" class="form-control" placeholder={{product.description}} required="required" >

        </div>

        <br>

        <div class="form-group">

            Price:

            <input [(ngModel)] = "product.price" name = "price" type="text" class="form-control" placeholder={{product.price}} required="required">

        </div>

        <br>

        <div class="form-group">

            Image URL:

            <input [(ngModel)] = "product.imageUrl" name = "imageUrl" type="text" class="form-control" placeholder={{product.imageUrl}} required="required">

        </div>

        <br>

        <div class="form-group">

            Cuisine:

            <br>

            <select [(ngModel)]="product.cuisine" name = "cuisine">

                <option \*ngFor="let cuisine of productCuisines" [ngValue]="cuisine.id">{{cuisine.cuisine\_name}}</option>

            </select>

        </div>

        <br>

        <div class="form-group">

            <button type="submit" class="btn btn-success btn-block">Update</button>

        </div>

    </form>

    <p class="text-center"><a routerLink="/login">Back to Login</a></p>

</div>

update-product.component.ts

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

import { LoginService } from 'src/app/services/login.service';

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

import {Router} from "@angular/router"

import { Product } from 'src/app/common/product';

import { ActivatedRoute } from '@angular/router';

import { ProductService } from 'src/app/services/product.service';

import { ProductCategory } from 'src/app/common/product-category';

@Component({

  selector: 'app-update-product',

  templateUrl: './update-product.component.html',

  styleUrls: ['./update-product.component.css']

})

export class UpdateProductComponent implements OnInit {

  product: Product = new Product();

  cuisine: ProductCategory = new ProductCategory();

  id: number = 0;

  productCuisines: ProductCategory[];

  constructor(private loginService: LoginService,

    private productService: ProductService,

private route: ActivatedRoute) { }

  ngOnInit(): void {

    this.listProductCategories()

    this.handleProductDetails()

  }

  handleProductDetails() {

    // get the "id" param string. convert string to a number using the "+" symbol

const theProductId: number = +this.route.snapshot.paramMap.get('id');

    this.productService.getProduct(theProductId).subscribe(

      data => {

        this.product = data;

      }

    )

  }

  updateProduct(){

    console.log(this.product.cuisine)

    const theProductId: number = +this.route.snapshot.paramMap.get('id');

    this.productService.updateProduct(theProductId, this.product).subscribe( data =>{

      console.log(data)

    }

    , error => console.log(error));

  //window.location.replace('/products');

  }

  listProductCategories() {

    this.productService.getProductCategories().subscribe(

      data => {

        console.log('Product Categories=' + JSON.stringify(data));

        this.productCuisines = data;

      }

    );

  }

}

**Services:**

cart.service.spec.ts

import { TestBed } from '@angular/core/testing';

import { CartService } from './cart.service';

describe('CartService', () => {

  let service: CartService;

  beforeEach(() => {

    TestBed.configureTestingModule({});

    service = TestBed.inject(CartService);

  });

  it('should be created', () => {

    expect(service).toBeTruthy();

  });

});

cart.service.ts

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

import { CartItem } from '../common/cart-item';

import { Subject } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class CartService {

  cartItems: CartItem[] = [];

  totalPrice: Subject<number> = new Subject<number>();

  totalQuantity: Subject<number> = new Subject<number>();

  constructor() { }

  addToCart(theCartItem: CartItem) {

    // check if we already have the item in our cart

    let alreadyExistsInCart: boolean = false;

let existingCartItem: CartItem = undefined;

    if (this.cartItems.length > 0) {

      // find the item in the cart based on item id

      existingCartItem = this.cartItems.find( tempCartItem => tempCartItem.id === theCartItem.id );

      // check if we found it

      alreadyExistsInCart = (existingCartItem != undefined);

}

    if (alreadyExistsInCart) {

      // increment the quantity

      existingCartItem.quantity++;

    }

    else {

      // just add the item to the array

      this.cartItems.push(theCartItem);

}

    // compute cart total price and total quantity

    this.computeCartTotals();

  }

  computeCartTotals() {

    let totalPriceValue: number = 0;

let totalQuantityValue: number = 0;

    for (let currentCartItem of this.cartItems) {

      totalPriceValue += currentCartItem.quantity \* currentCartItem.price;

      totalQuantityValue += currentCartItem.quantity;

}

    // publish the new values ... all subscribers will receive the new data

    this.totalPrice.next(totalPriceValue);

this.totalQuantity.next(totalQuantityValue);

    // log cart data just for debugging purposes

    this.logCartData(totalPriceValue, totalQuantityValue);

  }

  logCartData(totalPriceValue: number, totalQuantityValue: number) {

    console.log('Contents of the cart');

    for (let tempCartItem of this.cartItems) {

      const subTotalPrice = tempCartItem.quantity \* tempCartItem.price;

      console.log(`name: ${tempCartItem.name}, quantity=${tempCartItem.quantity}, price=${tempCartItem.price}, subTotalPrice=${subTotalPrice}`);

}

    console.log(`totalPrice: ${totalPriceValue.toFixed(2)}, totalQuantity: ${totalQuantityValue}`);

    console.log('----');

  }

  decrementQuantity(theCartItem: CartItem) {

theCartItem.quantity--;

    if (theCartItem.quantity === 0) {

      this.remove(theCartItem);

    }

    else {

      this.computeCartTotals();

    }

  }

  remove(theCartItem: CartItem) {

    // get index of item in the array

const itemIndex = this.cartItems.findIndex( tempCartItem => tempCartItem.id === theCartItem.id );

    // if found, remove the item from the array at the given index

    if (itemIndex > -1) {

      this.cartItems.splice(itemIndex, 1);

      this.computeCartTotals();

    }

  }

}

login.service.spec.ts

import { TestBed } from '@angular/core/testing';

import { LoginService } from './login.service';

describe('LoginService', () => {

  let service: LoginService;

  beforeEach(() => {

    TestBed.configureTestingModule({});

    service = TestBed.inject(LoginService);

  });

  it('should be created', () => {

    expect(service).toBeTruthy();

  });

});

login.service.ts

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

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

import { User } from '../common/user';

import { Observable } from 'rxjs';

import { map } from 'rxjs/operators';

@Injectable({

  providedIn: 'root'

})

export class LoginService {

  private baseUrl = 'http://localhost:8080/api/users';

  private baseUrlController = "http://localhost:8080/api/v1/users";

  constructor(private httpClient: HttpClient) { }

  getUser(username: string): Observable<User[]> {

    // search url

const searchUrl = `${this.baseUrl}/search/findByUsername?username=${username}`;

    return this.httpClient.get<GetResponseUser>(searchUrl).pipe(

      map(response => response.\_embedded.users)

    );

  }

  createUser(user: User): Observable<Object>{

    return this.httpClient.post(`${this.baseUrlController}`, user);

  }

}

interface GetResponseUser {

  \_embedded: {

    users: User[];

  }}

product.service.spec.ts

import { TestBed } from '@angular/core/testing';

import { ProductService } from './product.service';

describe('ProductService', () => {

  let service: ProductService;

  beforeEach(() => {

    TestBed.configureTestingModule({});

    service = TestBed.inject(ProductService);

  });

  it('should be created', () => {

    expect(service).toBeTruthy();

  });

});

product.service.ts

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

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

import { Product } from '../common/product';

import { Observable } from 'rxjs';

import { map } from 'rxjs/operators';

import { ProductCategory } from '../common/product-category';

@Injectable({

  providedIn: 'root'

})

export class ProductService {

  private baseUrl = 'http://localhost:8080/api/products';

  private categoryUrl = 'http://localhost:8080/api/cuisines';

  private newURL = "http://localhost:8080/api/v1/products";

  getListURL:string = null;

  constructor(private httpClient: HttpClient) { }

  updateProduct(id: number, product: Product): Observable<Object>{

    return this.httpClient.put(`${this.newURL}/${id}`, product);

  }

  addProduct(product: Product): Observable<Object>{

    return this.httpClient.post(`${this.newURL}`, product);

  }

  deleteProduct(id: number): Observable<Object>{

    return this.httpClient.delete(`${this.newURL}/${id}`);

  }

  getProduct(theProductId: number): Observable<Product> {

    // need to build URL based on product id

const productUrl = `${this.baseUrl}/${theProductId}`;

    return this.httpClient.get<Product>(productUrl);

  }

  getProductCuisine(theProductId: number): Observable<ProductCategory> {

    // need to build URL based on product id

const productUrl = `${this.baseUrl}/${theProductId}/cuisine`;

    return this.httpClient.get<ProductCategory>(productUrl);

  }

  getProductListPaginate(thePage: number,

                         thePageSize: number,

                         theCategoryId: number,

                         sortingMethod: string): Observable<GetResponseProducts> {

    if(sortingMethod == "First Created"){

        // need to build URL based on category id, page and size

      this.getListURL = `${this.baseUrl}/search/findByCuisine?cuisine=${theCategoryId}`

        + `&page=${thePage}&size=${thePageSize}`;

    }

    else if(sortingMethod == "Last Created"){

      this.getListURL = `${this.baseUrl}/search/findByCuisineOrderByDateCreatedDesc?cuisine=${theCategoryId}`

      + `&page=${thePage}&size=${thePageSize}`;

}

    else{

      this.getListURL = `${this.baseUrl}/search/findByCuisineOrderByNameAsc?cuisine=${theCategoryId}`

      + `&page=${thePage}&size=${thePageSize}`;

}

    return this.httpClient.get<GetResponseProducts>(this.getListURL);

  }

  getAllProductListPaginate(thePage: number,

    thePageSize: number,

    theCategoryId: number,

sortingMethod: string): Observable<GetResponseProducts> {

      if(sortingMethod == "First Created"){

        // need to build URL based on category id, page and size

      this.getListURL = `${this.baseUrl}`;

    }

    else if(sortingMethod == "Last Created"){

      this.getListURL = `${this.baseUrl}/search/findAllByOrderByDateCreatedDesc?page=${thePage}&size=${thePageSize}`;

}

    else{

      this.getListURL = `${this.baseUrl}/search/findAllByOrderByNameAsc?page=${thePage}&size=${thePageSize}`;

}

    return this.httpClient.get<GetResponseProducts>(this.getListURL);

}

  getProductList(theCategoryId: number): Observable<Product[]> {

    // need to build URL based on category id

const searchUrl = `${this.baseUrl}/search/findByCategoryId?id=${theCategoryId}`;

    return this.getProducts(searchUrl);

  }

  searchProducts(theKeyword: string): Observable<Product[]> {

    // need to build URL based on the keyword

const searchUrl = `${this.baseUrl}/search/findByNameContaining?name=${theKeyword}`;

    return this.getProducts(searchUrl);

  }

  searchProductsPaginate(thePage: number,

                        thePageSize: number,

                        theKeyword: string): Observable<GetResponseProducts> {

    // need to build URL based on keyword, page and size

    const searchUrl = `${this.baseUrl}/search/findByNameContaining?name=${theKeyword}`

                    + `&page=${thePage}&size=${thePageSize}`;

    return this.httpClient.get<GetResponseProducts>(searchUrl);

  }

  private getProducts(searchUrl: string): Observable<Product[]> {

    return this.httpClient.get<GetResponseProducts>(searchUrl).pipe(map(response => response.\_embedded.products));

  }

  getProductCategories(): Observable<ProductCategory[]> {

    return this.httpClient.get<GetResponseProductCategory>(this.categoryUrl).pipe(

      map(response => response.\_embedded.cuisines)

    );

  }

}

interface GetResponseProducts {

  \_embedded: {

    products: Product[];

  },

  page: {

    size: number,

    totalElements: number,

    totalPages: number,

    number: number

  }

}

interface GetResponseProductCategory {

  \_embedded: {

    cuisines: ProductCategory[];

  }

}

app.component.html

<div class="page-wrapper">

  <!-- MENU SIDEBAR-->

  <aside class="menu-sidebar d-none d-lg-block">

    <div class="logo">

      <a routerLink="/products">

        <img src="assets/logo-new.png" alt="food box" class="img-responsive">

      </a>

    </div>

    <app-product-category-menu></app-product-category-menu>

  </aside>

  <!-- END MENU SIDEBAR-->

  <div class="page-container">

    <!-- HEADER DESKTOP-->

    <header class="header-desktop">

      <div class="section-content section-content-p30">

        <div class="container-fluid">

          <div class="header-wrap">

            <app-search></app-search>

            <app-cart-status></app-cart-status>

            <app-login-status></app-login-status>

          </div>

        </div>

      </div>

    </header>

    <!-- END HEADER DESKTOP-->

    <!-- MAIN CONTENT-->

<router-outlet></router-outlet>

  </div>

</div>

<!-- ENDS PAGE CONTAINER -->

<footer>

  <ul>

    <li><a href="#">About Us</a></li>

    <li><a href="#">Contact Us</a></li>

    <li><a href="#">Help</a></li>

  </ul>

</footer>

app.component.ts

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

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  username:string = localStorage.getItem('username');

  title = 'foodbox';

}

app.module.ts

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

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

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

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

import { ProductListComponent } from './components/product-list/product-list.component';

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

import { ProductService } from './services/product.service';

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

import { ProductCategoryMenuComponent } from './components/product-category-menu/product-category-menu.component';

import { SearchComponent } from './components/search/search.component';

import { ProductDetailsComponent } from './components/product-details/product-details.component';

import { NgbModule } from '@ng-bootstrap/ng-bootstrap';

import { CartStatusComponent } from './components/cart-status/cart-status.component';

import { CartDetailsComponent } from './components/cart-details/cart-details.component';

import { RegisterComponent } from './components/register/register.component';

import { LoginComponent } from './components/login/login.component';

import { LoginStatusComponent } from './components/login-status/login-status.component';

import { UpdateProductComponent } from './components/update-product/update-product.component';

import { RemoveProductComponent } from './components/remove-product/remove-product.component';

import { AddProductComponent } from './components/add-product/add-product.component';

import { CheckOutComponent } from './components/check-out/check-out.component';

const routes: Routes = [

  {path: 'login', component: LoginComponent},

  {path: 'register', component: RegisterComponent},

  {path: 'checkout', component: CheckOutComponent},

  {path: 'add-product', component: AddProductComponent},

  {path: 'remove-product/:id', component: RemoveProductComponent},

  {path: 'update-product/:id', component: UpdateProductComponent},

  {path: 'cart-details', component: CartDetailsComponent},

  {path: 'products/:id', component: ProductDetailsComponent},

  {path: 'search/:keyword', component: ProductListComponent},

  {path: 'category/:id', component: ProductListComponent},

  {path: 'category', component: ProductListComponent},

  {path: 'products', component: ProductListComponent},

  {path: '', redirectTo: '/products', pathMatch: 'full'},

  {path: '\*\*', redirectTo: '/products', pathMatch: 'full'}

];

@NgModule({

  declarations: [

    AppComponent,

    ProductListComponent,

    ProductCategoryMenuComponent,

    SearchComponent,

    ProductDetailsComponent,

    CartStatusComponent,

    CartDetailsComponent,

    RegisterComponent,

    LoginComponent,

    LoginStatusComponent,

    UpdateProductComponent,

    RemoveProductComponent,

    AddProductComponent,

    CheckOutComponent

  ],

  imports: [

    RouterModule.forRoot(routes),

    BrowserModule,

    FormsModule,

    ReactiveFormsModule,

    HttpClientModule,

    NgbModule

  ],

  providers: [ProductService],

  bootstrap: [AppComponent]

})

export class AppModule { }

index.html

<!doctype html>

<html lang="en">

<head>

  <meta charset="utf-8">

  <title>Foodbox</title>

  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <link rel="icon" type="image/x-icon" href="favicon.ico">

</head>

<body>

  <app-root></app-root>

</body>

</html>