**SOURCECODE : FOODBOX**

**SOURCE CODE : BACKEND**

**FoodboxApplication.java**

package com.simplilearn.foodbox;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class FoodboxApplication {

public static void main(String[] args) {

SpringApplication.run(FoodboxApplication.class, args);

}

}

**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);

}

}

**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);

}

}

**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);

}

**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;

}

}

**Application Properties**

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

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

spring.datasource.username=root

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

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

**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>2.5.4</version>

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

</parent>

<groupId>com.simplilearn</groupId>

<artifactId>foodbox</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>foodbox</name>

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

<properties>

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

<maven-jar-plugin.version>3.1.1</maven-jar-plugin.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>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>

<dependency>

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

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

<optional>true</optional>

</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>

**SOURCE CODE : FRONTEND**

/\* You can add global styles to this file, and also import other style files \*/

/\*

\* ----------------------------

\* Common CSS

\* ----------------------------

\*/

\* {

margin: 0;

padding: 0;

-webkit-box-sizing: border-box;

-moz-box-sizing: border-box;

box-sizing: border-box;

}

body {

font-family: Poppins, sans-serif;

font-weight: 400;

font-size: 16px;

line-height: 1.625;

color: #666;

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

}

ul {

margin: 0;

}

button,

input[type="button"] {

cursor: pointer;

}

button:focus,

input:focus {

outline: 0;

}

input {

border: none;

}

button {

border: none;

background: 0 0;

}

img {

max-width: 100%;

height: auto;

}

p {

margin: 0;

}

h1,

h2,

h3,

h4,

h5,

h6 {

color: #333;

font-weight: 700;

margin: 0;

line-height: 1.2;

}

h1 {

font-size: 36px;

}

h2 {

font-size: 30px;

}

h3 {

font-size: 24px;

}

h4 {

font-size: 18px;

}

h5 {

font-size: 15px;

}

h6 {

font-size: 13px;

}

a {

display: inline-block;

}

a:active,

a:focus,

a:hover {

text-decoration: none;

outline: 0;

}

a,

a:hover {

-webkit-transition: all 0.3s ease;

-o-transition: all 0.3s ease;

-moz-transition: all 0.3s ease;

transition: all 0.3s ease;

}

/\*Layout Css\*/

.section-content-p35 {

padding: 0 35px;

}

.page-container {

background: lightcyan;

padding-left: 150px;

}

.main-content {

padding-top: 116px;

min-height: 100vh;

}

aside.menu-sidebar.d-none.d-lg-block.clr {

background: 0 0;

z-index: 99999999;

height: 75px;

}

/\*

\* ----------------------------

\* Header Desktop Section

\* ----------------------------

\*/

.header-desktop {

background: skyblue;

-webkit-box-shadow: 0 2px 5px 0 rgba(0, 0, 0, 0.1);

-moz-box-shadow: 0 2px 5px 0 rgba(0, 0, 0, 0.1);

box-shadow: 0 2px 5px 0 rgba(0, 0, 0, 0.1);

position: fixed;

top: 0;

right: 0;

left: 150px;

height: 75px;

z-index: 3;

}

.header-desktop .section-content {

overflow: visible;

position: absolute;

top: 50%;

left: 0;

right: 0;

-webkit-transform: translateY(-50%);

-moz-transform: translateY(-50%);

-ms-transform: translateY(-50%);

-o-transform: translateY(-50%);

transform: translateY(-50%);

}

.header-wrap {

display: -webkit-box;

display: -webkit-flex;

display: -moz-box;

display: -ms-flexbox;

display: flex;

-webkit-box-align: center;

-webkit-align-items: center;

-moz-box-align: center;

-ms-flex-align: center;

align-items: center;

-webkit-box-pack: justify;

-webkit-justify-content: space-between;

-moz-box-pack: justify;

-ms-flex-pack: justify;

justify-content: space-between;

}

.header-button {

display: -webkit-box;

display: -webkit-flex;

display: -moz-box;

display: -ms-flexbox;

display: flex;

-webkit-box-align: center;

-webkit-align-items: center;

-moz-box-align: center;

-ms-flex-align: center;

align-items: center;

}

.number {

padding: 6px 0;

}

.arrows i {

font-size: 30px;

padding: 4px 4px;

}

section {

position: relative;

}

.section-content {

position: relative;

margin: 0 auto;

z-index: 1;

}

.section-content-p30 {

padding: 0 30px;

}

.au-input-xl {

min-width: 935px;

}

input.au-input.au-input-xl {

border-radius: 40px !important;

margin-right: 15px;

}

.au-input {

line-height: 43px;

border: 1px solid #e5e5e5;

font-size: 14px;

color: #666;

padding: 0 17px;

-webkit-border-radius: 3px;

-moz-border-radius: 3px;

border-radius: 3px;

-webkit-transition: all 0.5s ease;

-o-transition: all 0.5s ease;

-moz-transition: all 0.5s ease;

transition: all 0.5s ease;

}

.cart-area.d-n {

display: block;

}

.cart-area i {

font-size: 22px;

float: left;

margin-top: 16px;

color: #fff;

margin-left: 15px;

}

.total {

background: #205b8d;

line-height: 43px;

color: #fff;

border-radius: 30px;

border: 1px solid #0da8e4;

float: left;

margin-top: 7px;

padding-left: 17px;

}

.total span {

display: inline-block;

background: #fff;

padding: 0 13px;

color: #000;

border-top-right-radius: 20px;

border-bottom-right-radius: 20px;

margin-left: 11px;

}

.form-header {

display: -webkit-box;

display: -webkit-flex;

display: -moz-box;

display: -ms-flexbox;

display: flex;

}

.au-btn-submit {

position: relative;

right: 0;

min-width: 95px;

color: #fff;

border: 1px solid #205b8d;

-webkit-border-radius: 3px;

-moz-border-radius: 3px;

border-radius: 43px;

background: #205b8d;

}

.au-btn-submit:hover {

background: #05143f;

}

.au-btn-submit > i {

font-size: 20px;

color: #fff;

position: absolute;

top: 50%;

left: 50%;

-webkit-transform: translate(-50%, -50%);

-moz-transform: translate(-50%, -50%);

-ms-transform: translate(-50%, -50%);

-o-transform: translate(-50%, -50%);

transform: translate(-50%, -50%);

}

/\*

\* ----------------------------

\* Left Menu Bar

\* ----------------------------

\*/

.menu-sidebar {

width: 150px;

position: fixed;

left: 0;

top: 0;

bottom: 0;

background: #fff;

overflow-y: auto;

}

.menu-sidebar .logo {

background: skyblue;

height: 75px;

padding: 0 35px;

display: -webkit-box;

display: -webkit-flex;

display: -moz-box;

display: -ms-flexbox;

display: flex;

-webkit-box-align: center;

-webkit-align-items: center;

-moz-box-align: center;

-ms-flex-align: center;

align-items: center;

position: relative;

z-index: 3;

}

.menu-sidebar .navbar-list .navbar-sub-list {

display: none;

padding-left: 34px;

}

.menu-sidebar .navbar-list .navbar-sub-list li a {

padding: 11.5px 0;

}

.menu-sidebar-content {

position: relative;

height: -webkit-calc(100vh - 75px);

height: -moz-calc(100vh - 75px);

height: calc(100vh - 75px);

background: skyblue;

}

.navbar-sidebar {

padding: 35px;

padding-bottom: 0;

}

.navbar-sidebar .navbar-list li a {

color: black;

font-size: 18px;

padding-left: 2%;

}

.navbar-sidebar .navbar-list li a i {

margin-right: 19px;

}

.navbar-sidebar .navbar-list li a:hover {

color: skyblue;

}

/\*

\* ----------------------------

\* Header Mobile Section

\* ----------------------------

\*/

.header-mobile {

position: absolute;

top: 0;

left: 0;

right: 0;

}

.header-mobile .header-mobile-bar {

padding: 15px 0;

background: #fff;

}

.header-mobile .header-mobile-inner {

display: -webkit-box;

display: -webkit-flex;

display: -moz-box;

display: -ms-flexbox;

display: flex;

-webkit-box-align: center;

-webkit-align-items: center;

-moz-box-align: center;

-ms-flex-align: center;

align-items: center;

-webkit-box-pack: justify;

-webkit-justify-content: space-between;

-moz-box-pack: justify;

-ms-flex-pack: justify;

justify-content: space-between;

}

.header-mobile .hamburger {

width: 36px;

height: 36px;

padding: 0;

line-height: 1;

vertical-align: top;

background: #fff;

display: -webkit-box;

display: -webkit-flex;

display: -moz-box;

display: -ms-flexbox;

display: flex;

-webkit-box-pack: center;

-webkit-justify-content: center;

-moz-box-pack: center;

-ms-flex-pack: center;

justify-content: center;

-webkit-box-align: center;

-webkit-align-items: center;

-moz-box-align: center;

-ms-flex-align: center;

align-items: center;

}

.header-mobile .hamburger .hamburger-box {

width: 20px;

height: 15px;

}

.header-mobile .hamburger .hamburger-box .hamburger-inner {

width: 20px;

height: 2px;

-webkit-border-radius: 0;

-moz-border-radius: 0;

border-radius: 0;

}

.header-mobile .hamburger .hamburger-box .hamburger-inner:before {

width: 20px;

height: 2px;

top: 6px;

}

.header-mobile .hamburger .hamburger-box .hamburger-inner:after {

top: 12px;

width: 20px;

height: 2px;

}

.header-mobile .navbar-mobile {

display: none;

position: absolute;

width: 100%;

top: 88px;

z-index: 20;

}

.header-mobile .navbar-mobile .navbar-mobile-list {

background: #f8f8f8;

}

.header-mobile .navbar-mobile .navbar-mobile-list > li > a {

padding-left: 15px !important;

}

.header-mobile .navbar-mobile .navbar-mobile-list li a {

color: #555;

display: block;

padding: 10px 15px;

padding-right: 25px;

padding-left: 0;

border-bottom: 1px solid #e6e6e6;

text-transform: capitalize;

line-height: inherit;

}

.header-mobile .navbar-mobile .navbar-mobile-list li a:hover {

color: #0da8e4;

}

.header-mobile .navbar-mobile .navbar-mobile-list li a > i {

margin-right: 19px;

}

.navbar-mobile-sub-list {

display: none;

padding-left: 30px;

background: #fff;

}

.header-mobile .navbar-mobile .navbar-mobile-sub-list li a {

padding-left: 15px;

}

.header-mobile-2 {

background: #393939;

position: static;

}

.header-mobile-2.header-mobile .navbar-mobile {

top: 82px;

}

.header-mobile-2.header-mobile .hamburger {

background: 0 0;

}

.header-mobile-2.header-mobile .hamburger .hamburger-box .hamburger-inner {

background: #fff;

}

.header-mobile-2.header-mobile

.hamburger

.hamburger-box

.hamburger-inner::before {

background: #fff;

}

.header-mobile-2.header-mobile

.hamburger

.hamburger-box

.hamburger-inner::after {

background: #fff;

}

.main-content.page-m {

padding: 10% 15% !important;

}

/\*

\* ----------------------------

\* Products List

\* ----------------------------

\*/

.product-box img {

width: 150px;

}

.items select {

width: 100px;

height: 40px;

border-radius: 40px;

padding-left: 15px;

margin-bottom: 10px;

outline: 0;

margin-left: 5px;

}

.product-box {

margin-bottom: 60px;

}

.price {

margin: 15px 0 0;

font-weight: 800;

color: #0da8e4;

}

.product-box h1 {

font-size: 18px;

border-bottom: 1px solid #ccc;

padding-bottom: 5px;

font-weight: 600;

padding-top: 12px;

margin-bottom: 7px;

}

.product-box h2 {

font-size: 15px;

padding-bottom: 0;

font-weight: 500;

color: #585858;

}

/\*

\* ----------------------------

\* Shopping Cart Checkout

\* ----------------------------

\*/

.form-area {

background: #e5e5e5;

position: relative;

margin-bottom: 45px;

padding: 50px 20px;

border: 3px solid #ccc;

}

.form-area h3 {

position: absolute;

top: -26px;

left: 13px;

background: #e5e5e5;

padding: 10px 20px;

font-weight: 500;

z-index: 23;

display: inline-block;

}

.input-space {

margin-bottom: 20px;

}

.input-space label {

padding-left: 31px;

font-size: 14px;

margin-bottom: 50px;

}

.input-space input[type="text"] {

width: 100%;

height: 40px;

border-radius: 30px;

outline: 0;

padding: 10px;

border: 1px solid #ccc;

}

.input-space select {

width: 100%;

height: 40px;

border-radius: 30px;

outline: 0;

padding: 0 10px;

border: 1px solid #ccc;

}

.au-checkbox {

display: block;

position: relative;

cursor: pointer;

font-size: 22px;

-webkit-user-select: none;

-moz-user-select: none;

-ms-user-select: none;

user-select: none;

}

.au-checkbox input {

position: absolute;

opacity: 0;

cursor: pointer;

}

.au-checkmark {

position: absolute;

top: 0;

left: 0;

height: 22px;

width: 22px;

background-color: #fff !important;

border: 2px solid #e5e5e5;

-webkit-border-radius: 2px;

-moz-border-radius: 2px;

border-radius: 2px;

}

.au-checkbox:hover input ~ .au-checkmark {

background-color: transparent;

}

.au-checkbox input:checked ~ .au-checkmark {

background-color: #fff;

}

.au-checkmark:after {

content: "";

position: absolute;

display: none;

}

.au-checkbox input:checked ~ .au-checkmark:after {

display: block;

}

label.date {

padding-top: 8px;

}

.au-checkbox .au-checkmark:after {

left: 5px;

top: -1px;

width: 9px;

height: 15px;

border: solid #0da8e4;

border-width: 0 4px 4px 0;

-webkit-border-radius: 2px;

-moz-border-radius: 2px;

border-radius: 2px;

-webkit-transform: rotate(45deg);

-moz-transform: rotate(45deg);

-ms-transform: rotate(45deg);

-o-transform: rotate(45deg);

transform: rotate(45deg);

}

option:first {

color: #ccc;

}

select {

color: #828482;

}

/\*

\* ----------------------------

\* Shopping Cart Details

\* ----------------------------

\*/

.table {

margin: 0;

}

table.table.table-bordered {

border: 1px solid #ccc;

background: #fff;

}

.table-bordered td,

.table-bordered th {

border: 1px solid #dee2e6;

}

table.table.table-bordered th {

background: #0da8e4;

color: #fff;

font-size: 14px;

padding: 7px;

}

.table-responsive {

padding-right: 1px;

}

.items select {

width: 100px;

height: 40px;

border-radius: 40px;

padding-left: 15px;

margin-bottom: 10px;

outline: 0;

margin-left: 5px;

}

/\*

\* ----------------------------

\* Product Details

\* ----------------------------

\*/

.price {

margin: 15px 0 0;

font-weight: 800;

color: #0da8e4;

}

.detail-section {

margin-top: 60px;

}

.detail-img {

width: 30%;

}

/\*\*/

/\*

\* ----------------------------

\* Buttons

\* ----------------------------

\*/

.btn-info {

color: #fff;

background-color: #0da8e4 !important;

border-color: #0da8e4 !important;

}

a.primary-btn {

background: #05143f;

padding: 7px 20px;

text-align: center;

display: inline-block;

color: #fff;

border-radius: 2px;

font-size: 14px;

margin-top: 5px;

}

a.primary-btn:hover {

background: #0da8e4;

color: #fff;

}

.active-link {

font-weight: bold;

}

/\*

\* ----------------------------

\* Footer section

\* ----------------------------

\*/

footer ul li a {

color: #fff;

}

footer {

background: #232324;

z-index: 1;

position: fixed;

width: 100%;

height: 40px;

bottom: 0%;

text-align: center;

}

footer li a {

color: #313131;

font-weight: 500;

}

footer ul li {

display: inline-block;

font-size: 14px;

padding: 10px 20px;

}

/\*

\* ----------------------------

\* Media Queries

\* ----------------------------

\*/

@media (max-width: 1600px) {

.au-input-xl {

min-width: 290px;

}

}

@media (max-width: 1315px) and (min-width: 992px) {

.header-desktop3 .section-content {

padding: 0 15px;

}

}

@media (max-width: 1570px) and (min-width: 992px) {

.header-navbar ul li a {

padding: 25px 15px;

}

}

@media (max-width: 1315px) and (min-width: 992px) {

.header-navbar ul li a {

font-size: 13px;

padding: 27px 15px;

}

.header-navbar ul li a i {

margin-right: 5px;

}

}

@media (min-width: 1200px) {

.container {

max-width: 1320px;

}

}

@media (min-width: 992px) and (max-width: 1199px) {

.cart-area i {

font-size: 22px;

float: left;

margin-top: 16px;

color: #205b8d;

margin-left: 15px;

}

.header-mobile .header-mobile-bar {

padding: 10px 0;

background: #fff;

}

.header-wrap .account-item > .content {

display: none;

}

}

@media (max-width: 991px) {

.section-content-p30 {

padding: 0;

}

.page-container {

position: relative;

top: 88px;

padding-left: 0;

}

.page-wrapper {

background: #e5e5e5;

padding-bottom: 12vh;

}

.main-content {

padding-top: 50px;

padding-bottom: 100px;

}

.cart-area.d-n {

display: none;

}

.form-header {

-webkit-box-pack: center;

-webkit-justify-content: center;

-moz-box-pack: center;

-ms-flex-pack: center;

justify-content: center;

}

.au-input-xl {

min-width: 350px;

}

.logo {

text-align: center;

}

.header-desktop {

position: relative;

top: 0;

left: 0;

height: 80px;

}

.header-wrap {

-webkit-box-orient: vertical;

-webkit-box-direction: normal;

-webkit-flex-direction: column;

-moz-box-orient: vertical;

-moz-box-direction: normal;

-ms-flex-direction: column;

flex-direction: column;

}

.header-button {

margin-top: 30px;

width: 100%;

-webkit-box-pack: justify;

-webkit-justify-content: space-between;

-moz-box-pack: justify;

-ms-flex-pack: justify;

justify-content: space-between;

}

.menu-sidebar2 {

top: 0;

right: -300px;

left: auto;

-webkit-box-shadow: 0 5px 10px 0 rgba(0, 0, 0, 0.1);

-moz-box-shadow: 0 5px 10px 0 rgba(0, 0, 0, 0.1);

box-shadow: 0 5px 10px 0 rgba(0, 0, 0, 0.1);

}

.header-desktop2 {

left: 0;

position: relative;

}

.header-desktop4 {

position: static;

}

.section-content.section-content-p30.e-spc {

padding-top: 100px;

}

header.header-desktop.extra {

top: 93px !important;

}

.cart-area i {

font-size: 22px;

float: left;

margin-top: 16px;

color: #205b8d;

margin-left: 15px;

}

.header-wrap2 {

-webkit-box-pack: justify;

-webkit-justify-content: space-between;

-moz-box-pack: justify;

-ms-flex-pack: justify;

justify-content: space-between;

}

}

@media (max-width: 767px) {

.au-input-xl {

min-width: 150px;

}

.header-button-item {

font-size: 22px;

margin-right: 15px;

}

.main-content.page-m {

padding: 10% 0 !important;

}

.header-mobile .navbar-mobile {

display: none;

position: absolute;

width: 100%;

top: 66px;

z-index: 20;

}

.section-content.section-content-p30.e-spc {

padding-top: 100px;

}

header.header-desktop.extra {

top: 64px !important;

}

.header-desktop {

position: relative;

top: 0;

left: 0;

height: 80px;

}

a.logo img {

width: 110px;

}

.cart-area i {

font-size: 22px;

float: left;

margin-top: 16px;

color: #205b8d;

margin-left: 15px;

}

.footer-pagination {

text-align: center;

}

.product-box {

text-align: centerg;

}

.page-container {

position: relative;

top: 66px;

padding-left: 0;

}

.au-btn-submit {

position: relative;

right: 10px;

min-width: 85px;

color: #fff;

border: 1px solid #205b8d;

-webkit-border-radius: 3px;

-moz-border-radius: 3px;

border-radius: 0;

background: #205b8d;

}

input.au-input.au-input-xl {

border-radius: 0 !important;

margin-right: 0;

}

.cart-area.d-b {

display: block;

}

.cart-area.d-n {

display: none;

}

.cart-area i {

font-size: 18px;

float: left;

margin-top: 16px;

color: #03a9f4;

margin-left: 15px;

}

.total span {

display: inline-block;

background: #fff;

padding: 0 10px;

color: #000;

border-top-right-radius: 20px;

border-bottom-right-radius: 20px;

margin-left: 6px;

}

.total {

background: #205b8d;

line-height: 30px;

color: #fff;

border-radius: 30px;

border: 1px solid #205b8d;

float: left;

margin-top: 9px;

padding-left: 12px;

}

.header-mobile .header-mobile-bar {

padding: 6px 0;

background: #fff;

border: 0;

padding: 8px 0;

}

}

.banner {

display: flex;

align-items: center;

}

.banner-start h3 {

display: inline-block;

}

.banner-start i {

padding: 0 6px;

}

.banner-middle {

flex: 1;

}

<div class="page-wrapper">

<!-- MENU SIDEBAR-->

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

<div class="logo">

<a routerLink="/products">

<img

src="assets/food2.png"

alt="food box"

class="img-responsive"

style="width: 350px; height: 50px"

/>

</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>

<p></p>

<br />

</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>

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

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

describe('AppComponent', () => {

beforeEach(async () => {

await TestBed.configureTestingModule({

declarations: [

AppComponent

],

}).compileComponents();

});

it('should create the app', () => {

const fixture = TestBed.createComponent(AppComponent);

const app = fixture.componentInstance;

expect(app).toBeTruthy();

});

it(`should have as title 'foodbox'`, () => {

const fixture = TestBed.createComponent(AppComponent);

const app = fixture.componentInstance;

expect(app.title).toEqual('foodbox');

});

it('should render title', () => {

const fixture = TestBed.createComponent(AppComponent);

fixture.detectChanges();

const compiled = fixture.nativeElement;

expect(compiled.querySelector('.content span').textContent).toContain('foodbox app is running!');

});

});

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';

}

import { Address } from './address';

describe('Address', () => {

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

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

});

});

export class Address {

id: string;

zipCode: string;

street: string;

city: string;

}

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

describe('CartItem', () => {

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

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

});

});

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;

}

}

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

describe('ProductCategory', () => {

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

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

});

});

export class ProductCategory {

id: number;

cuisine\_name: string;

}

import { Product } from './product';

describe('Product', () => {

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

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

});

});

export class Product {

id: string;

name: string;

description: string;

price: number;

imageUrl: string;

dateCreated: Date;

lastUpdate: Date;

cuisine: number;

}

import { User } from './user';

describe('User', () => {

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

expect(new User()).toBeTruthy();

});

});

export class User {

id: number;

username: string;

password: string;

email: string;

type: number;

phone: number;

zipcode: string;

street: string;

city: string;

}

ADD-PRODUCTS

.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;

}

<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>

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();

});

});

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

<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>

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();

});

});

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

<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>

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();

});

});

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 = 0.0;

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

<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>

<br />

Thank you <b>{{ username }}</b> for purchasing our products.

</p>

<p>

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

</p>

<br /><br />

<p>The products will be delivered to you soon.</p>

<p>Your address is as follow :</p>

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

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

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

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

describe('CheckOutComponent', () => {

let component: CheckOutComponent;

let fixture: ComponentFixture<CheckOutComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

declarations: [CheckOutComponent],

}).compileComponents();

});

beforeEach(() => {

fixture = TestBed.createComponent(CheckOutComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

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

expect(component).toBeTruthy();

});

});

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-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;

}

<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>

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

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

describe('LoginComponent', () => {

let component: LoginComponent;

let fixture: ComponentFixture<LoginComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

declarations: [LoginComponent],

}).compileComponents();

});

beforeEach(() => {

fixture = TestBed.createComponent(LoginComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

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

expect(component).toBeTruthy();

});

});

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{

background:#F0FFFF;

color:black;

height: 46px;

}

.logout{

background:black;

color:white;

height: 46px;

}

<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>

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

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

describe('LoginStatusComponent', () => {

let component: LoginStatusComponent;

let fixture: ComponentFixture<LoginStatusComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

declarations: [LoginStatusComponent],

}).compileComponents();

});

beforeEach(() => {

fixture = TestBed.createComponent(LoginStatusComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

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

expect(component).toBeTruthy();

});

});

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

<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>

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

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

describe('ProductCategoryMenuComponent', () => {

let component: ProductCategoryMenuComponent;

let fixture: ComponentFixture<ProductCategoryMenuComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

declarations: [ProductCategoryMenuComponent],

}).compileComponents();

});

beforeEach(() => {

fixture = TestBed.createComponent(ProductCategoryMenuComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

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

expect(component).toBeTruthy();

});

});

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

<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>

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

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

describe('ProductDetailsComponent', () => {

let component: ProductDetailsComponent;

let fixture: ComponentFixture<ProductDetailsComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

declarations: [ProductDetailsComponent],

}).compileComponents();

});

beforeEach(() => {

fixture = TestBed.createComponent(ProductDetailsComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

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

expect(component).toBeTruthy();

});

});

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

<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>

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

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

describe('ProductListComponent', () => {

let component: ProductListComponent;

let fixture: ComponentFixture<ProductListComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

declarations: [ProductListComponent],

}).compileComponents();

});

beforeEach(() => {

fixture = TestBed.createComponent(ProductListComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

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

expect(component).toBeTruthy();

});

});

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

.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;

}

<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>

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

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

describe('RegisterComponent', () => {

let component: RegisterComponent;

let fixture: ComponentFixture<RegisterComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

declarations: [RegisterComponent],

}).compileComponents();

});

beforeEach(() => {

fixture = TestBed.createComponent(RegisterComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

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

expect(component).toBeTruthy();

});

});

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

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

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

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

describe('RemoveProductComponent', () => {

let component: RemoveProductComponent;

let fixture: ComponentFixture<RemoveProductComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

declarations: [ RemoveProductComponent ]

})

.compileComponents();

});

beforeEach(() => {

fixture = TestBed.createComponent(RemoveProductComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

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

expect(component).toBeTruthy();

});

});

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

<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>

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

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

describe('SearchComponent', () => {

let component: SearchComponent;

let fixture: ComponentFixture<SearchComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

declarations: [SearchComponent],

}).compileComponents();

});

beforeEach(() => {

fixture = TestBed.createComponent(SearchComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

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

expect(component).toBeTruthy();

});

});

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

.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;

}

<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>

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

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

describe('UpdateProductComponent', () => {

let component: UpdateProductComponent;

let fixture: ComponentFixture<UpdateProductComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

declarations: [UpdateProductComponent],

}).compileComponents();

});

beforeEach(() => {

fixture = TestBed.createComponent(UpdateProductComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

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

expect(component).toBeTruthy();

});

});

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

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();

});

});

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();

}

}

}

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();

});

});

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[];

};

}

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();

});

});

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[];

};

}