package com.medicare.controller;

import java.io.IOException;

import java.util.List;

import javax.validation.Valid;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.security.access.prepost.PreAuthorize;

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

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

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

import org.springframework.web.bind.annotation.RestController;

import org.springframework.web.multipart.MultipartFile;

import com.fasterxml.jackson.core.JsonProcessingException;

import com.fasterxml.jackson.databind.JsonMappingException;

import com.fasterxml.jackson.databind.ObjectMapper;

import com.medicare.config.ImageUtil;

import com.medicare.entities.Product;

import com.medicare.entities.ProductImage;

import com.medicare.services.ProductService;

@RestController

@CrossOrigin(origins = "\*")

public class ProductController {

@Autowired

private ProductService productService;

@Autowired

private ObjectMapper objectMapper;

//add new product

@PreAuthorize("hasAuthority('ADMIN')")

@PostMapping("/add/product")

public ResponseEntity<?> addNewProduct(@RequestParam("product") String product,

@RequestParam("image") MultipartFile file) throws IOException{

ProductImage img = new ProductImage();

img.setName(file.getOriginalFilename());

img.setType(file.getContentType());

img.setImageData(ImageUtil.compressImage(file.getBytes()));

Product p = null;

try {

p = objectMapper.readValue(product,Product.class);

p.setProductImage(img);

} catch (JsonMappingException e) {

e.printStackTrace();

} catch (JsonProcessingException e) {

e.printStackTrace();

return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body("Invalid Request");

}

Product saveProduct = this.productService.addProduct(p);

return ResponseEntity.ok(saveProduct);

}

//update existing product

@PreAuthorize("hasAuthority('ADMIN')")

@PutMapping("/update/product/{id}")

public ResponseEntity<?> updateProduct(@PathVariable("id") Long id,@Valid @RequestBody Product product){

Product updateProduct = this.productService.findProduct(id);

updateProduct.setName(product.getName());

updateProduct.setBrand(product.getBrand());

updateProduct.setCategory(product.getCategory());

updateProduct.setDescription(product.getDescription());

updateProduct.setSalt(product.getSalt());

updateProduct.setTotalAvailable(product.getTotalAvailable());

updateProduct.setPrice(product.getPrice());

this.productService.addProduct(updateProduct);

return ResponseEntity.status(HttpStatus.CREATED).build();

}

//find product by id

@GetMapping("get-product/{id}")

public ResponseEntity<?> getProductById(@PathVariable("id") Long id){

Product product = this.productService.findProduct(id);

ProductImage img = new ProductImage();

img.setImageData(ImageUtil.decompressImage(product.getProductImage().getImageData()));

img.setImgId(product.getProductImage().getImgId());

img.setName(product.getProductImage().getName());

img.setType(product.getProductImage().getType());

product.setProductImage(img);

return ResponseEntity.ok(product);

}

//find all products

@GetMapping("/get/all-products")

public ResponseEntity<?> getAllProducts(){

List<Product> allProducts = this.productService.findAllProducts();

allProducts.forEach(product -> {

ProductImage img = new ProductImage();

img.setImageData(ImageUtil.decompressImage(product.getProductImage().getImageData()));

img.setImgId(product.getProductImage().getImgId());

img.setName(product.getProductImage().getName());

img.setType(product.getProductImage().getType());

product.setProductImage(img);

});

if(allProducts.isEmpty()) {

return ResponseEntity.status(HttpStatus.NOT\_FOUND).build();

}else {

return ResponseEntity.ok(allProducts);

}

}

@GetMapping(value = {"/get/products/{name}"})

public ResponseEntity<?> getProductByName(@PathVariable("name") String name,@PathVariable("name") String salt){

List<Product> products = this.productService.findByNameOrSalt(name, salt);

products.forEach(product -> {

ProductImage img = new ProductImage();

img.setImageData(ImageUtil.decompressImage(product.getProductImage().getImageData()));

img.setImgId(product.getProductImage().getImgId());

img.setName(product.getProductImage().getName());

img.setType(product.getProductImage().getType());

product.setProductImage(img);

});

if(products.isEmpty()) {

return ResponseEntity.status(HttpStatus.NOT\_FOUND).build();

}else {

return ResponseEntity.ok(products);

}

}

@GetMapping("/get/products-by-category/{category}")

public ResponseEntity<?> getProductsByCategory(@PathVariable("category") String category){

List<Product> products = this.productService.findProductByCategory(category);

products.forEach(product -> {

ProductImage img = new ProductImage();

img.setImageData(ImageUtil.decompressImage(product.getProductImage().getImageData()));

img.setImgId(product.getProductImage().getImgId());

img.setName(product.getProductImage().getName());

img.setType(product.getProductImage().getType());

product.setProductImage(img);

});

if(products.isEmpty()) {

return ResponseEntity.status(HttpStatus.NOT\_FOUND).build();

}else {

return ResponseEntity.ok(products);

}

}

@PreAuthorize("hasAuthority('ADMIN')")

@DeleteMapping("/delete/product/{id}")

public ResponseEntity<?> deleteProduct(@PathVariable("id") Long id){

this.productService.deleteProductById(id);

return ResponseEntity.status(HttpStatus.OK).build();

}

@PreAuthorize("hasAuthority('ADMIN')")

@PutMapping("/set-availability/product/{id}")

public ResponseEntity<?> setAvailability(@PathVariable("id") Long id, @RequestBody Product product){

Product updateProduct = this.productService.findProduct(id);

updateProduct.setAvailable(product.isAvailable());

this.productService.addProduct(updateProduct);

return ResponseEntity.status(HttpStatus.CREATED).build();

}

@GetMapping("/get/{name}")

public ResponseEntity<?> getAvailable(@PathVariable("name") String name){

List<Product> products = this.productService.findTrueProduct(name);

if(products.isEmpty()) {

return ResponseEntity.status(HttpStatus.NOT\_FOUND).build();

}else {

return ResponseEntity.ok(products);

}

}

}