Kitchen Story-Backend Source Code

Config

CustomerUserDetails

```
package com.kitchenstory.config;
import java.util.Collection;
import java.util.HashSet;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import com.kitchenstory.entities.AdminCred;
@SuppressWarnings("serial")
public class CustomUserDetails implements UserDetails{

	private AdminCred adminCred;

	public CustomUserDetails(AdminCred adminCred) {
		this.adminCred = adminCred;
	}
@Override
	public Collection<? extends GrantedAuthority> getAuthorities() {
		HashSet<SimpleGrantedAuthority> set = new HashSet<>();
```

```
        set.add(new SimpleGrantedAuthority(this.adminCred.getRole()));
        return set;
}
@Override
public String getPassword() {
        // TODO Auto-generated method stub
        return this.adminCred.getPassword();
}

@Override
public String getUsername() {
        // TODO Auto-generated method stub
        return this.adminCred.getUsername();
}

@Override
public boolean isAccountNonExpired() {
        // TODO Auto-generated method stub
        return true;
}

@Override
public boolean isAccountNonLocked() {
```

```
		// TODO Auto-generated method stub
		return true;
	}

	@Override
	public boolean isCredentialsNonExpired() {
		// TODO Auto-generated method stub
		return true;
	}

	@Override
	public boolean isEnabled() {
		// TODO Auto-generated method stub
		return true;
	}

}
```

JwtAuthEntryPoint

```
package com.kitchenstory.config;

import java.io.IOException;

import javax.servlet.ServletException;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.security.core.AuthenticationException;
import org.springframework.security.web.AuthenticationEntryPoint;
import org.springframework.stereotype.Component;

@Component
public class JwtAuthEntryPoint implements AuthenticationEntryPoint{

    @Override
    public void commence(HttpServletRequest request, HttpServletResponse response,
                         AuthenticationException authException) throws IOException, ServletException {
        response.sendError(401,"Unauthorized");

    }

}
```

JwtAuthFilter

```
package com.kitchenstory.config;
```

```
import java.io.IOException;


import javax.servlet.FilterChain;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;


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

import
org.springframework.security.authentication.UsernamePasswordAuthenticationTo
ken;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.core.userdetails.UserDetails;

import
org.springframework.security.web.authentication.WebAuthenticationDetailsSource
;

import org.springframework.stereotype.Component;

import org.springframework.web.filter.OncePerRequestFilter;


import com.kitchenstory.jwthelper.JwtUtil;

import com.kitchenstory.services.CustomUserDetailService;


@Component

public class JwtAuthFilter extends OncePerRequestFilter {
```

```
@Autowired
CustomUserDetailService customUserDetailService;

@Autowired
private JwtUtil jwtUtil;

@Override
protected void doFilterInternal(HttpServletRequest request,
HttpServletResponse response, FilterChain filterChain)
        throws ServletException, IOException {
    String requestTokenHeader = request.getHeader("Authorization");
    String username = null;
    String jwtToken = null;

    if(requestTokenHeader!=null &&
requestTokenHeader.startsWith("Bearer ")) {
        jwtToken = requestTokenHeader.substring(7);

        try {
            username = this.jwtUtil.extractUsername(jwtToken);

        }catch(Exception e) {
```

```
                e.printStackTrace();
            }

            UserDetails userDetails =
this.customUserDetailService.loadUserByUsername(username);

            if(username!=null &&
SecurityContextHolder.getContext().getAuthentication()==null) {
                UsernamePasswordAuthenticationToken
usernamePasswordAuthenticationToken = new
UsernamePasswordAuthenticationToken(userDetails, null,
userDetails.getAuthorities());
                usernamePasswordAuthenticationToken.setDetails(new
WebAuthenticationDetailsSource().buildDetails(request));

SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenti
cationToken);
            }else {
                System.out.println("Token is invalid!");
            }

        }

        filterChain.doFilter(request, response);

    }
```

}

SecurityConfig

```
package com.kitchenstory.config;


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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import
org.springframework.security.config.annotation.authentication.builders.Authentica
tionManagerBuilder;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import
org.springframework.security.config.annotation.web.configuration.EnableWebSec
urity;

import
org.springframework.security.config.annotation.web.configuration.WebSecurityCo
nfigurerAdapter;

import org.springframework.security.config.http.SessionCreationPolicy;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import
org.springframework.security.web.authentication.UsernamePasswordAuthenticatio
nFilter;


import com.kitchenstory.services.CustomUserDetailService;
```

```
@SuppressWarnings("deprecation")
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter{

	@Autowired
	private CustomUserDetailService customUserDetailService;

	@Autowired
	private JwtAuthEntryPoint entryPoint;

	@Autowired
	JwtAuthFilter authFilter;

	@Override
	protected void configure(HttpSecurity http) throws Exception {
		http
			.csrf()
			.disable()
			.cors().and()
			.authorizeRequests()
```

```
                .antMatchers("/token").permitAll()
                .antMatchers("/changePassword").permitAll()
                .antMatchers("/getProduct/**").permitAll()
                .antMatchers("/getItem/**").permitAll()
                .antMatchers("/order").permitAll()
                .antMatchers("/getOrder/**").permitAll()
                .anyRequest().authenticated()
                .and().sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)
                .and().exceptionHandling().authenticationEntryPoint(entryPoint);
        http.addFilterBefore(authFilter, UsernamePasswordAuthenticationFilter.class);

    }


    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.userDetailsService(this.customUserDetailService).passwordEncoder(passwordEncoder());
    }
```

```
	@Bean
	public BCryptPasswordEncoder passwordEncoder() {
		return new BCryptPasswordEncoder(10);
	}

	@Bean
	public AuthenticationManager authenticationManagerBean() throws
Exception {
		return super.authenticationManagerBean();
	}

}
```

Controller

AdminLoginController

```
package com.kitchenstory.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
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.RestController;

import com.kitchenstory.entities.Product;
import com.kitchenstory.services.ProductService;

@RestController
@CrossOrigin(origins = "*")
public class AdminLoginController {

	@Autowired
	private ProductService prodService;

	@PostMapping("/saveItem")
	public ResponseEntity<?> saveItem(@RequestBody Product product){
		this.prodService.addProduct(product);
		return ResponseEntity.status(HttpStatus.CREATED).build();
	}
```

```
	@GetMapping("/getItems")
	public ResponseEntity<?> getItems() {
		List<Product> allItems = this.prodService.showProducts();
		if(allItems.isEmpty()) {
			return
ResponseEntity.status(HttpStatus.NOT_FOUND).build();
		}else {
			return ResponseEntity.ok(allItems);
		}
	}
	@DeleteMapping("/deleteItem/{name}")
	public ResponseEntity<?> deleteItem(@PathVariable("name") String name)
{
		try {
			this.prodService.deleteProduct(name);
			return ResponseEntity.status(HttpStatus.OK).build();

		}catch(Exception e) {
			System.out.println("Empty Result Data Access Exception.");
			return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
		}

	}
```

```
	@PutMapping("/updateItem/{name}")
	public ResponseEntity<?> updateItem(@PathVariable("name") String name,@RequestBody Product product) {
		try {
			this.prodService.updateProduct(name, product);
			return ResponseEntity.status(HttpStatus.OK).build();

		}catch(Exception e) {
			System.out.println("Empty Result Data Access Exception.");
			return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
		}


	}
	@GetMapping("/getItem/{name}")
	public ResponseEntity<?> getItem(@PathVariable("name") String name){
		Product product = this.prodService.getProduct(name);
		if(product!=null) {
			return ResponseEntity.ok(product);
		}else {
			return ResponseEntity.status(HttpStatus.NOT_FOUND).build();
		}
```

```
    }


}
```

JwtController

```
package com.kitchenstory.controller;


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

import org.springframework.http.ResponseEntity;

import org.springframework.security.authentication.AuthenticationManager;

import
org.springframework.security.authentication.UsernamePasswordAuthenticationTo
ken;

import org.springframework.security.core.userdetails.UserDetails;

import
org.springframework.security.core.userdetails.UsernameNotFoundException;

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

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

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

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

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

import com.kitchenstory.entities.JwtRequest;

import com.kitchenstory.entities.JwtResponse;

import com.kitchenstory.jwthelper.JwtUtil;

import com.kitchenstory.services.CustomUserDetailService;
```

```
@RestController
@CrossOrigin(origins = "*")
public class JwtController {

	@Autowired
	AuthenticationManager authenticationManager;

	@Autowired
	CustomUserDetailService customUserDetailService;

	@Autowired
	JwtUtil jwtUtil;

	@RequestMapping(value="/token",method = RequestMethod.POST)
	public ResponseEntity<?> generateToken(@RequestBody JwtRequest jwtRequest){
		System.out.println(jwtRequest);

		try {
			this.authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(jwtRequest.getUsername(), jwtRequest.getPassword()));
```

```
            }catch(UsernameNotFoundException e) {
                e.printStackTrace();
            }
            UserDetails userDetails =
this.customUserDetailService.loadUserByUsername(jwtRequest.getUsername());
            String token = this.jwtUtil.generateToken(userDetails);
            System.out.println("Token: "+token);
            return ResponseEntity.ok(new JwtResponse(token));
        }

}
```

SignInController

```
package com.kitchenstory.controller;


import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;
```

```
import com.kitchenstory.entities.AdminCred;
import com.kitchenstory.services.AdminService;



@RestController
@CrossOrigin(origins = "*")
public class SignInController {

	@Autowired
	private AdminService adminService;
	@Autowired
	private BCryptPasswordEncoder passwordEncoder;

	@PutMapping("/changePassword")
	public ResponseEntity<?> setPassword(@RequestBody AdminCred admin)
{
		AdminCred adm =
this.adminService.findAdmin(admin.getUsername());
		if(adm == null) {
			return
ResponseEntity.status(HttpStatus.NOT_FOUND).build();
		}
		else {
```

```
        adm.setPassword(this.passwordEncoder.encode(admin.getPassword()));
                        this.adminService.saveCred(adm);
                        return ResponseEntity.ok(adm);
                }
        }

}
```

UserController

```
package com.kitchenstory.controller;
import java.text.DateFormat;
import java.util.Calendar;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
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.RequestBody;
import org.springframework.web.bind.annotation.RestController;
```

```
import com.kitchenstory.entities.Order;
import com.kitchenstory.entities.Product;
import com.kitchenstory.services.OrderService;
import com.kitchenstory.services.ProductService;
@RestController
@CrossOrigin(origins = "*")
public class UserController {

	@Autowired
	private OrderService service;

	@Autowired
	private ProductService pservice;

	@PostMapping("/order")
	public ResponseEntity<?> createOrder(@RequestBody Order order){
		DateFormat df = DateFormat.getDateInstance();
		Calendar cl = Calendar.getInstance();
		String od = df.format(cl.getTime());
		order.setOd(od);
		this.service.saveOrder(order);
		return ResponseEntity.status(HttpStatus.CREATED).build();
```

```
	}
	@GetMapping("/getOrders")
	public ResponseEntity<?> getAllOrders(){
		List<Order> orders = this.service.getOrders();
		if(orders.isEmpty()) {
			return
ResponseEntity.status(HttpStatus.NOT_FOUND).build();
		}
		else {
			return ResponseEntity.ok(orders);
		}
	}

	@DeleteMapping("/delete/{id}")
	public ResponseEntity<?> deleteOrder(@PathVariable("id") String id){
		this.service.deleteOrder(id);
		return ResponseEntity.status(HttpStatus.OK).build();
	}

	@GetMapping("/getProduct/{name}")
	public ResponseEntity<?> getProducts(@PathVariable("name") String
name){
		List<Product> products = this.pservice.getAllByName(name);
		if(products.isEmpty()) {
```

```
                return ResponseEntity.status(HttpStatus.NOT_FOUND).build();
            }else {
                return ResponseEntity.ok(products);
            }
        }

        @GetMapping("/getOrder/{id}")
        public ResponseEntity<?> getOrder(@PathVariable("id") String id){
            Order order = this.service.getOrder(id);
            if(order==null) {
                return ResponseEntity.status(HttpStatus.NOT_FOUND).build();
            }else {
                return ResponseEntity.ok(order);
            }
        }

}
```

Entities

AdminCred

```
package com.kitchenstory.entities;
```

```
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

@Document(collection = "AdminCred")
public class AdminCred {

	@Id
	private String username;
	private String password;
	private String role;
	public AdminCred() {
		super();
		// TODO Auto-generated constructor stub
	}
	public AdminCred(String username, String password, String role) {
		super();
		this.username = username;
		this.password = password;
		this.role = role;
	}
	public String getUsername() {
		return username;
	}
```

```
	public void setUsername(String username) {
		this.username = username;
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}
	public String getRole() {
		return role;
	}
	public void setRole(String role) {
		this.role = role;
	}


}
```

JwtRequest

```
package com.kitchenstory.entities;

public class JwtRequest {
```

```
String username;
String password;
public JwtRequest() {

}
public JwtRequest(String username, String password) {
	super();
	this.username = username;
	this.password = password;
}
public String getUsername() {
	return username;
}
public void setUsername(String username) {
	this.username = username;
}
public String getPassword() {
	return password;
}
public void setPassword(String password) {
	this.password = password;
}
@Override
```

```
	public String toString() {
		return "JwtRequest [username=" + username + ", password=" +
password + "]";
	}




}
```

JwtResponse

```
package com.kitchenstory.entities;


public class JwtResponse {


	String token;


	public JwtResponse() {
		super();
	}


	public JwtResponse(String token) {
		super();
		this.token = token;
	}
```

```
	public String getToken() {
		return token;
	}

	public void setToken(String token) {
		this.token = token;
	}



}
```

Order

```
package com.kitchenstory.entities;


import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.DBRef;
import org.springframework.data.mongodb.core.mapping.Document;


@Document(collection = "Orders")
public class Order {
```

```
@Id
private String id;
    private String name;
    private String address;
    private String state;
    private String contact;
    private String od;
    private int quantity;
    private String payment;
    @DBRef
    private Product product;
    public Order() {
        super();
        // TODO Auto-generated constructor stub
    }
    public Order(String id, String name, String address, String state, String
contact, String od, int quantity,
            String payment, Product product) {
        super();
        this.id = id;
```

```
        this.name = name;
        this.address = address;
        this.state = state;
        this.contact = contact;
        this.od = od;
        this.quantity = quantity;
        this.payment = payment;
        this.product = product;
}
public String getId() {
        return id;
}
public void setId(String id) {
        this.id = id;
}
public String getName() {
        return name;
}
public void setName(String name) {
        this.name = name;
}
public String getAddress() {
        return address;
```

```
}
public void setAddress(String address) {
        this.address = address;
}
public String getState() {
        return state;
}
public void setState(String state) {
        this.state = state;
}
public String getContact() {
        return contact;
}
public void setContact(String contact) {
        this.contact = contact;
}
public String getOd() {
        return od;
}
public void setOd(String od) {
        this.od = od;
}
public int getQuantity() {
```

```
        return quantity;
    }
    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }
    public String getPayment() {
        return payment;
    }
    public void setPayment(String payment) {
        this.payment = payment;
    }
    public Product getProduct() {
        return product;
    }
    public void setProduct(Product product) {
        this.product = product;
    }


}
```

Product

```
package com.kitchenstory.entities;
```

```
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.DBRef;
import org.springframework.data.mongodb.core.mapping.Document;


@Document(collection = "Orders")
public class Order {



    @Id
    private String id;
        private String name;
        private String address;
        private String state;
        private String contact;
        private String od;
        private int quantity;
        private String payment;
        @DBRef
        private Product product;
        public Order() {
```

```
        super();
        // TODO Auto-generated constructor stub
    }
    public Order(String id, String name, String address, String state, String
contact, String od, int quantity,
            String payment, Product product) {
        super();
        this.id = id;
        this.name = name;
        this.address = address;
        this.state = state;
        this.contact = contact;
        this.od = od;
        this.quantity = quantity;
        this.payment = payment;
        this.product = product;
    }
    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
```

```
public String getName() {
        return name;
}
public void setName(String name) {
        this.name = name;
}
public String getAddress() {
        return address;
}
public void setAddress(String address) {
        this.address = address;
}
public String getState() {
        return state;
}
public void setState(String state) {
        this.state = state;
}
public String getContact() {
        return contact;
}
public void setContact(String contact) {
        this.contact = contact;
```

```
}
public String getOd() {
	return od;
}
public void setOd(String od) {
	this.od = od;
}
public int getQuantity() {
	return quantity;
}
public void setQuantity(int quantity) {
	this.quantity = quantity;
}
public String getPayment() {
	return payment;
}
public void setPayment(String payment) {
	this.payment = payment;
}
public Product getProduct() {
	return product;
}
public void setProduct(Product product) {
```

```
        this.product = product;
    }


}
```

Repo

AdminCredRepo

```
package com.kitchenstory.repo;


import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.stereotype.Repository;


import com.kitchenstory.entities.AdminCred;


@Repository
public interface AdminCredRepo extends MongoRepository<AdminCred, String>
{
    public AdminCred findByUsername(String username);


}
```

OrderRepo

```
package com.kitchenstory.repo;
```

```
import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.stereotype.Repository;

import com.kitchenstory.entities.Order;

@Repository
public interface OrderRepo extends MongoRepository<Order,String>{

}
```

ProductRepo

```
package com.kitchenstory.repo;

import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.stereotype.Repository;
import com.kitchenstory.entities.Product;

@Repository
public interface ProductRepo extends MongoRepository<Product, String>{
	public Product findByName(String name);
	public Iterable<Product> findByNameContainingIgnoreCase(String name);
}
```

Services

AdminService

```
package com.kitchenstory.services;

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

import com.kitchenstory.entities.AdminCred;
import com.kitchenstory.repo.AdminCredRepo;

@Service
public class AdminService {

	@Autowired
	private AdminCredRepo adminRepo;

	public void saveCred(AdminCred adminCred) {
		this.adminRepo.save(adminCred);
	}
	public AdminCred findAdmin(String username) {
		AdminCred admin = this.adminRepo.findByUsername(username);
		return admin;
	}

}
```

CustomUserDetailService

```
package com.kitchenstory.services;


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

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.core.userdetails.UserDetailsService;

import
org.springframework.security.core.userdetails.UsernameNotFoundException;

import org.springframework.stereotype.Service;


import com.kitchenstory.config.CustomUserDetails;

import com.kitchenstory.entities.AdminCred;

import com.kitchenstory.repo.AdminCredRepo;


@Service

public class CustomUserDetailService implements UserDetailsService{

	@Autowired

	private AdminCredRepo credRepo;


	@Override

	public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {


		AdminCred admin = this.credRepo.findByUsername(username);
```

```
		if(admin == null) {
			throw new UsernameNotFoundException("NO USER FOUND!");
		}
		return new CustomUserDetails(admin);
	}



}
```

OrderService

```
package com.kitchenstory.services;

import java.util.List;

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

import com.kitchenstory.entities.Order;
import com.kitchenstory.repo.OrderRepo;

@Service
public class OrderService {
```

```
	@Autowired
	private OrderRepo orepo;

	public void saveOrder(Order order) {
		this.orepo.save(order);
	}

	public List<Order> getOrders(){
		return this.orepo.findAll();
	}

	public void deleteOrder(String id) {
		this.orepo.deleteById(id);
	}
	public Order getOrder(String id) {
		return this.orepo.findById(id).get();
	}

}
```

ProductService

```
package com.kitchenstory.services;

import java.util.List;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.util.Streamable;
import org.springframework.stereotype.Service;

import com.kitchenstory.entities.Product;
import com.kitchenstory.repo.ProductRepo;

@Service
public class ProductService {

	@Autowired
	private ProductRepo prodRepo;

	public void addProduct(Product product) {
		this.prodRepo.save(product);
	}
	public List<Product> showProducts(){
		return this.prodRepo.findAll();
	}
	public void deleteProduct(String name) {
		this.prodRepo.deleteById(name);
	}
```

```
	public void updateProduct(String name,Product updateProduct) {
		Product product = this.prodRepo.findByName(name);
		product.setName(updateProduct.getName());
		product.setPid(updateProduct.getPid());
		product.setCategory(updateProduct.getCategory());
		product.setBrand(updateProduct.getBrand());
		product.setIngredient(updateProduct.getIngredient());
		product.setOrigin(updateProduct.getOrigin());
		product.setPrice(updateProduct.getPrice());
		product.setQuantity(updateProduct.getQuantity());
		this.prodRepo.save(product);
	}
	public Product getProduct(String name) {
		return this.prodRepo.findByName(name);
	}

	public List<Product> getAllByName(String name){
		Iterable<Product> products =
this.prodRepo.findByNameContainingIgnoreCase(name);
		List<Product> allProducts = Streamable.of(products).toList();
		return allProducts;
	}
```

}