

# SportyShoes

## An E-Commerce Website

### Source Code

#### **File 1:** Phase4KitchenStoryApplicationTests.java

```
package com.kitchenstory;

import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class Phase4KitchenStoryApplicationTests {

    @Test
    void contextLoads() {
    }

}
```

#### **File 2:** Phase4KitchenStoryApplication.java

```
package com.kitchenstory;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

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

@SpringBootApplication
public class Phase4KitchenStoryApplication implements CommandLineRunner {

    @Autowired
    private AdminCredRepo credRepo;
    @Autowired
    private BCryptPasswordEncoder passwordEncoder;

    public static void main(String[] args) {
        SpringApplication.run(Phase4KitchenStoryApplication.class, args);
    }
}
```

```

    }

    @Override
    public void run(String... args) throws Exception {
        AdminCred cred = new AdminCred();
        cred.setUsername("admin@kitchenstory.com");
        cred.setPassword(this.passwordEncoder.encode("admin123"));
        cred.setRole("ROLE_ADMIN");
        this.credRepo.save(cred);
    }
}

```

## File 3: CustomUserDetails.java

```

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;
}
}

```

## File 4: JwtAuthEntryPoint.java

```

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");
    }

}

```

## File 5: JwtAuthFilter.java

```

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.UsernamePasswordAuthenticationToken;
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.customUserDetailsService.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(usernamePasswordAuthenticationTok
en);

            } else {
                System.out.println("Token is invalid!");
            }

        }

        filterChain.doFilter(request, response);

    }
}

```

## File 6: SecurityConfig.java

```

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.AuthenticationManag
erBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerA
dapter;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import
org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

import com.kitchenstory.services.CustomUserDetailsService;

```

```

@SuppressWarnings("deprecation")

```

```

@Configuration

```

```

@EnableWebSecurity

```

```

public class SecurityConfig extends WebSecurityConfigurerAdapter{

```

```

    @Autowired

```

```

    private CustomUserDetailsService customUserDetailsService;

```

```

    @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(SessionCreationPo

```

```

            licy.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();
    }

}

```

## File 7: AdminLoginController.java

```

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();
        }
    }
}

```

## File 8: JwtController.java

```

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.UsernamePasswordAuthenticationToken;
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.customUserDetailsService.loadUserByUsername(jwtRequest.getUsername());
        String token = this.jwtUtil.generateToken(userDetails);
        System.out.println("Token: "+token);
        return ResponseEntity.ok(new JwtResponse(token));
    }
}

```

## File 9: SignInController.java

```
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);
        }
    }
}

```

```
    }  
}
```

## File 10: UserController.java

```
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);
        }
    }
}

```

## File 11: AdminCred.java

```

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;
}
}

```

## File 12: JwtRequest.java

```

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 +
"]";
    }
}
}

```

## File 12: JwtResponse.java

```

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;
    }

}

```

## File 13: Order.java

```
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;
}
```



```
}  
  
}
```

## File 14: Product.java

```
package com.kitchenstory.entities;  
  
import org.springframework.data.annotation.Id;  
import org.springframework.data.mongodb.core.mapping.Document;  
  
@Document(collection = "Products")  
public class Product {  
  
    @Id  
    private String name;  
    private String pid;  
    private String category;  
    private String ingredient;  
    private int price;  
    private String brand;  
    private String quantity;  
    private String origin;  
    public Product() {  
    }  
    public Product(String name, String pid, String category, String ingredient, int price,  
String brand,  
                    String quantity, String origin) {  
        super();  
        this.name = name;  
        this.pid = pid;  
        this.category = category;  
        this.ingredient = ingredient;  
        this.price = price;  
        this.brand = brand;  
        this.quantity = quantity;  
        this.origin = origin;  
    }  
    public String getName() {  
        return name;  
    }  
    public void setName(String name) {  
        this.name = name;  
    }  
    public String getPid() {  
        return pid;  
    }  
    public void setPid(String pid) {
```

```

        this.pid = pid;
    }
    public String getCategory() {
        return category;
    }
    public void setCategory(String category) {
        this.category = category;
    }
    public String getIngredient() {
        return ingredient;
    }
    public void setIngredient(String ingredient) {
        this.ingredient = ingredient;
    }
    public int getPrice() {
        return price;
    }
    public void setPrice(int price) {
        this.price = price;
    }
    public String getBrand() {
        return brand;
    }
    public void setBrand(String brand) {
        this.brand = brand;
    }
    public String getQuantity() {
        return quantity;
    }
    public void setQuantity(String quantity) {
        this.quantity = quantity;
    }
    public String getOrigin() {
        return origin;
    }
    public void setOrigin(String origin) {
        this.origin = origin;
    }
}

```

## File 15: JwtUtil.java

```

package com.kitchenstory.jwthelper;

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Component;

```

```

import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import java.util.function.Function;

@Component
public class JwtUtil {

    private String SECRET_KEY = "secret";

    public String extractUsername(String token) {
        return extractClaim(token, Claims::getSubject);
    }

    public Date extractExpiration(String token) {
        return extractClaim(token, Claims::getExpiration);
    }

    public <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {
        final Claims claims = extractAllClaims(token);
        return claimsResolver.apply(claims);
    }

    private Claims extractAllClaims(String token) {
        return
Jwts.parser().setSigningKey(SECRET_KEY).parseClaimsJws(token).getBody();
    }

    private Boolean isTokenExpired(String token) {
        return extractExpiration(token).before(new Date());
    }

    public String generateToken(UserDetails userDetails) {
        Map<String, Object> claims = new HashMap<>();
        return createToken(claims, userDetails.getUsername());
    }

    private String createToken(Map<String, Object> claims, String subject) {

        return Jwts.builder().setClaims(claims).setSubject(subject).setIssuedAt(new
Date(System.currentTimeMillis()))
                .setExpiration(new Date(System.currentTimeMillis() + 1000 * 60 * 60 *
10))
                .signWith(SignatureAlgorithm.HS256, SECRET_KEY).compact();
    }

    public Boolean validateToken(String token, UserDetails userDetails) {
        final String username = extractUsername(token);
        return (username.equals(userDetails.getUsername()) && !
isTokenExpired(token));
    }
}

```

```
    }  
}
```

## File 16: AdminCredRepo.java

```
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);  
  
}
```

## File 17: OrderRepo.java

```
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>{  
  
}
```

## File 18: ProductRepo.java

```
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);  
}
```

## File 19: AdminService.java

```
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;
    }
}
```

## File 20: CustomUserDetailService.java

```
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);
    }
}

```

## File 20: OrderService.java

```

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();
    }

}

```

## File 20: ProductService.java

```

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;
    }

}

```

## **File 21:** application.properties

spring.data.mongodb.host=localhost  
spring.data.mongodb.port=27017  
spring.data.mongodb.database=kitchenstory  
server.port=9595

## File 22: 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>2.7.6</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.kitchenstory</groupId>
  <artifactId>Phase4_Kitchen_Story</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>Phase4_Kitchen_Story</name>
  <description>Phase 4 Kitchen Story Backend Project</description>
  <properties>
    <java.version>17</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-mongodb</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.thymeleaf.extras</groupId>
      <artifactId>thymeleaf-extras-springsecurity5</artifactId>
    </dependency>
```



```

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
      <optional>true</optional>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>org.springframework.security</groupId>
      <artifactId>spring-security-test</artifactId>
      <scope>test</scope>
    </dependency>
    <!-- https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt -->
    <dependency>
      <groupId>io.jsonwebtoken</groupId>
      <artifactId>jjwt</artifactId>
      <version>0.9.0</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api -->
    <dependency>
      <groupId>javax.xml.bind</groupId>
      <artifactId>jaxb-api</artifactId>
    </dependency>

  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>

</project>

```

*#Source Code for frontend was too big so did not include it in the document to keep the size less than 10 MB, but all the code files are uploaded in github for reference.*

<https://github.com/AshrafMH22/KitchenStory.git>