```java
package com.sportshoes.ecom;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import springfox.documentation.builders.ParameterBuilder;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.schema.ModelRef;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

import java.util.Arrays;

@SpringBootApplication
@EnableSwagger2
public class EcomApplication {

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

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

    @Bean
    public Docket productApi() {
        return new Docket(DocumentationType.SWAGGER_2)
            .globalOperationParameters(
                Arrays.asList(new ParameterBuilder()
```

```
35                              .name("Authorization")
36                              .description("JWT
   Authentication token")
37                              .modelRef(new ModelRef("
   string"))
38                              .parameterType("header")
39                              .required(true)
40                              .build()));
41      }
42 }
43
```

File - F:\SportsShoesEcomApp\src\main\java\com\sportshoes\ecom\ApplicationRunner.java

```java
package com.sportshoes.ecom;

import com.sportshoes.ecom.entity.Category;
import com.sportshoes.ecom.entity.Customers;
import com.sportshoes.ecom.entity.Products;
import com.sportshoes.ecom.services.CategoryService;
import com.sportshoes.ecom.services.CustomerService;
import com.sportshoes.ecom.services.ProductService;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.autoconfigure.kafka.KafkaProperties;
import org.springframework.boot.context.event.ApplicationReadyEvent;
import org.springframework.context.ApplicationListener;
import org.springframework.stereotype.Component;

import java.awt.event.ActionListener;

import static com.sportshoes.ecom.entity.Customers.Role.ROLE_ADMIN;

@Component
@Slf4j
public class ApplicationRunner implements ApplicationListener<ApplicationReadyEvent> {

    @Autowired
    CustomerService customerService;
    @Autowired
    CategoryService categoryService;
    @Autowired
    ProductService productService;
    @Override
    public void onApplicationEvent(ApplicationReadyEvent applicationReadyEvent) {
        seedData();
    }

    private void seedData() {
        Customers c = Customers.builder().emailId("admin@gmail.com").fname("admin").lname("admin")
```

```
37                  .age(21).address("myAddress").role(
   ROLE_ADMIN).isActiveUser(true).password("password").
   build();
38
39          long customer = 0;
40          try {
41              customer = customerService.addNewCustomer
   (c).getID();
42          }catch (Exception e) {
43              log.info("could not load initial Data..
   Exiting");
44              return;
45          }
46          log.info("Loaded date " + c.getEmailId());
47          long sports = categoryService.addNewCategory(
   new Category(customer, "sports")).getID();
48          long students = categoryService.
   addNewCategory(new Category(customer, "students")).
   getID();
49          long homeUtensils = categoryService.
   addNewCategory(new Category(customer, "home_utensils"
   )).getID();
50          Long cricketBat = productService.
   addNewProduct(new Products(customer, sports,"Cricket
   Bat", 1000)).getID();
51          Long cricket_boll = productService.
   addNewProduct(new Products(customer, sports,"
   cricket_boll", 600)).getID();
52          Long tennis_ball = productService.
   addNewProduct(new Products(customer, sports,"
   tennis_boll", 40)).getID();
53          Long stove = productService.addNewProduct(new
    Products(customer, homeUtensils,"stove", 5000)).
   getID();
54          Long vessel = productService.addNewProduct(
   new Products(customer, homeUtensils,"vessel", 500)).
   getID();
55          Long Ref = productService.addNewProduct(new
   Products(customer, homeUtensils,"Refrigerators", 1200
   )).getID();
56          Long test_book = productService.addNewProduct
   (new Products(customer, students,"pen_set_of_8", 100
   )).getID();
57          Long pen_set_of_8 = productService.
```

```
57 addNewProduct(new Products(customer, students,"
   vessel", 500)).getID();
58         Long map = productService.addNewProduct(new
   Products(customer, students,"map", 149)).getID();
59
60
61
62     }
63 }
64
```

File - F:\SportsShoesEcomApp\src\main\java\com\sportshoes\ecom\ServletInitializer.java

```java
package com.sportshoes.ecom;

import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

public class ServletInitializer extends SpringBootServletInitializer {

    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
        return application.sources(EcomApplication.class);
    }

}

```

```java
package com.sportshoes.ecom.repos;

import com.sportshoes.ecom.entity.Category;
import com.sportshoes.ecom.entity.Products;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;
import org.w3c.dom.stylesheets.LinkStyle;

import javax.transaction.Transactional;
import java.util.List;

@Repository
public interface ProductRepo extends JpaRepository<Products,Long> {
    List<Products> findAllByCategoryId(Category category);

    @Query("select p.price from Products p where p.ID = :id")
    int getProductPrice(@Param("id") Long id);

    @Modifying
    @Transactional
    @Query("update Products p set p.deletedFlag = true where p.ID = :id")
    void softDeleteProduct(@Param("id") Long id);


}

```

```java
package com.sportshoes.ecom.repos;


import com.sportshoes.ecom.entity.Category;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import javax.transaction.Transactional;

@Repository
public interface CategoryRepo extends JpaRepository<Category, Long> {
    @Modifying
    @Transactional
    @Query("update Category c set c.activeCategory = false where c.ID = :id")
    void softDeleteCategory(@Param("id") Long id);
}

```

```java
package com.sportshoes.ecom.repos;

import com.sportshoes.ecom.entity.Customers;
import com.sportshoes.ecom.entity.Products;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;
import org.springframework.web.bind.annotation.RequestBody;

import javax.persistence.EntityManager;
import javax.transaction.Transactional;
import java.util.List;

@Repository
public interface CustomerRepo extends JpaRepository<Customers, Long> {
    @Query("select c from Customers c where c.emailId = :email")
    Customers getUserByUserName(@Param("email") String email);

    @Modifying(clearAutomatically = true)
    @Query("update Customers c set c.password = :newPassword where c.ID = :userId ")
    void changeMyPassword(@Param("newPassword") String newPassword, @Param("userId") Long userId);

    @Query("from Customers")
    List<Customers> getAllRegisteredUsers();

    @Modifying
    @Transactional
    @Query("update Customers c set c.isActiveUser = false where c.ID = :id")
    void softDeleteAccount(@Param("id") Long id);

}

```

```java
package com.sportshoes.ecom.repos;

import com.sportshoes.ecom.entity.Purchase;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

import java.time.LocalDate;
import java.util.List;

public interface PurchaseRepo extends JpaRepository<Purchase, Long> {

    @Query("select p from Purchase p where p.customer.ID = :userId")
    List<Purchase> getPurchasesByCustomerId(@Param("userId") Long userId);

    @Query("select p from Purchase p where p.purchaseDate = :purchaseDate")
    List<Purchase> filterProductsByDate(@Param("purchaseDate") LocalDate purchaseDate);


}

```

```java
package com.sportshoes.ecom.entity;

import lombok.Getter;
import lombok.Setter;
import lombok.ToString;

import java.util.Objects;

@Getter
@Setter
@ToString
public final class Cart {
    private Long productId;
    private int quantity;

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Cart cart = (Cart) o;

        return getProductId() != null ? getProductId().equals(cart.getProductId()) : cart.getProductId() == null;
    }

    @Override
    public int hashCode() {
        return getProductId() != null ? getProductId().hashCode() : 0;
    }
}

```

File - F:\SportsShoesEcomApp\src\main\java\com\sportshoes\ecom\entity\Category.java

```java
package com.sportshoes.ecom.entity;

import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonProperty;
import jdk.jfr.Enabled;
import lombok.Builder;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import org.hibernate.annotations.Where;

import javax.persistence.*;
import javax.validation.constraints.Size;

@Entity
@Table(name = "category")
@NoArgsConstructor
@Getter
@Setter
@Where(clause = "is_Active=true")
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long ID;

    @Size(min = 5, max = 20, message = "size should be 5-20 characters")
    @Column(name = "name", nullable = false)
    private String name;

    @Column(name = "is_Active", nullable = false)
    private boolean activeCategory;

    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    @ManyToOne
    Customers admin;

    public Category(long ID) {
        this.ID = ID;
    }

    public Category(Long adminID, String name) {
        this.admin = new Customers(adminID);
```

```
43          this.name =  name;
44      }
45
46      @PrePersist
47      private void activateCategory() {
48          this.activeCategory = true;
49      }
50  }
51
```

File - F:\SportsShoesEcomApp\src\main\java\com\sportshoes\ecom\entity\Products.java

```java
package com.sportshoes.ecom.entity;

import com.fasterxml.jackson.annotation.JsonProperty;
import com.sun.istack.NotNull;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;
import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.Where;

import javax.persistence.*;
import java.time.LocalDateTime;
@Getter
@Setter
@ToString
@NoArgsConstructor
@Entity
@Where(clause = "deletion_flag = false")
@Table(name = "products")
public class Products {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "ID")
    private Long ID;

    @NotNull
    @Column(name = "name", nullable = false, length = 30)
    private String name;

    @Column(name = "price", nullable = false)
    private int price;

    @CreationTimestamp
    private LocalDateTime dateAdded;

    @Column(name = "deletion_flag", nullable = false)
    private boolean deletedFlag;

    @ManyToOne
    @JoinColumn(name = "category", nullable = false)
    private Category categoryId;
```

```
43
44      @JsonProperty(access = JsonProperty.Access.
    WRITE_ONLY)
45      @ManyToOne(fetch = FetchType.LAZY, optional =
    false)
46      Customers Admin;
47
48      public Products(Long adminID, Long categoryId,
    String name, int price) {
49          this.Admin = new Customers(adminID);
50          this.categoryId = new Category(categoryId);
51          this.name = name;
52          this.price = price;
53      }
54
55      public Products(Long categoryId, String name, int
     price) {
56          this.categoryId = new Category(categoryId);
57          this.name = name;
58          this.price = price;
59      }
60
61
62
63      public Products(Long ID) {
64          this.ID = ID;
65      }
66
67      @Override
68      public boolean equals(Object o) {
69          if (this == o) return true;
70          if (o == null || getClass() != o.getClass())
    return false;
71
72          Products products = (Products) o;
73
74          return getID() != null ? getID().equals(
    products.getID()) : products.getID() == null;
75      }
76
77      @Override
78      public int hashCode() {
79          return getID() != null ? getID().hashCode
    () : 0;
```

```
80      }
81
82      @PrePersist
83      public void setDeletionFlag() {
84          this.deletedFlag = false;
85
86      }
87  }
88
```

File - F:\SportsShoesEcomApp\src\main\java\com\sportshoes\ecom\entity\Purchase.java

```java
package com.sportshoes.ecom.entity;

import com.fasterxml.jackson.annotation.JsonIgnore;
import com.sun.istack.NotNull;
import com.sun.istack.Nullable;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;
import org.hibernate.annotations.CreationTimestamp;
import org.springframework.security.config.annotation
.method.configuration.EnableGlobalMethodSecurity;

import javax.persistence.*;
import java.math.BigInteger;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.util.*;

@Entity
@Getter
@Setter
@NoArgsConstructor
@ToString
public class Purchase {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long purchaseId;

    @OneToOne
    @NotNull
    @JoinColumn(nullable = false)
    private Customers customer;

    @ElementCollection
    @CollectionTable(name = "products_quantity")
    @MapKeyJoinColumn(name = "product_id")
    Map<Products, Integer> product = new HashMap<>();

    @Nullable
    Long totalPrice;

    @CreationTimestamp
    LocalDate purchaseDate;
```

```
44
45 }
46
```

File - F:\SportsShoesEcomApp\src\main\java\com\sportshoes\ecom\entity\Customers.java

```java
package com.sportshoes.ecom.entity;

import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonInclude;
import com.fasterxml.jackson.annotation.JsonProperty;
import lombok.*;
import org.hibernate.annotations.ColumnDefault;
import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.Where;

import javax.persistence.*;
import java.time.LocalDateTime;
import java.util.Date;
import java.util.List;

@Entity
@Table(name = "customers")
@Getter
@Setter
@ToString
@NoArgsConstructor
@Builder
@AllArgsConstructor
@Where(clause = "is_active_user=true")
@JsonInclude(JsonInclude.Include.NON_NULL)
public class Customers {

    @Id
    @GeneratedValue(strategy = GenerationType.
IDENTITY)
    @Column(name = "ID")
    private long ID;


    @Column(name = "emailid", unique = true, length
 = 50, nullable = false)
    private String emailId;

    @Column(name = "fname", nullable = false, length
 = 20)
    private String fname;

    @Column(name = "lname", nullable = false, length
 = 30)
```

```
41      private String lname;
42
43
44      @Column(name = "address", nullable = false,
    length = 100)
45      private String address;
46
47      @Column(name = "age", nullable = false, length =
    3)
48      private int age;
49
50      @CreationTimestamp
51      @Column(name = "date_added")
52      private LocalDateTime dateAdded;
53
54      @Enumerated(EnumType.STRING)
55      @Column(nullable = false)
56      private Role role;
57
58      @Column(name = "is_active_user")
59      private boolean isActiveUser;
60
61      @OneToMany(fetch = FetchType.LAZY, cascade =
    CascadeType.ALL)
62      private List<Purchase> purchases;
63
64      @JsonProperty(access = JsonProperty.Access.
    WRITE_ONLY)
65      @Column(name = "password", nullable = false)
66      private String password;
67
68      public Customers(long ID) {
69          this.ID = ID;
70      }
71
72      public enum Role {
73          ROLE_USER, ROLE_ADMIN
74      }
75
76      @PrePersist
77      private void activateUser() {
78          this.isActiveUser = true;
79      }
80
```

```
81
82 }
83
```

```java
package com.sportshoes.ecom.entity.JSONMappers;

import com.sportshoes.ecom.entity.Customers;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import javax.validation.constraints.*;

@Getter
@Setter
@NoArgsConstructor
public class customerMapper {

    @Size(min = 6, max = 30, message = "Invalid email ")
    @Pattern(regexp = "^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,6}$", message = "Invalid email address")
    private String emailId;

    @Size(min = 5, max = 20, message = "size(5-20")
    private String fname;

    @Size(min = 2, message = "minimun length 2")
    private String lname;

    @Size(min = 5, max = 30)
    private String address;

    @Digits(fraction = 0, integer = 3)
    private int age;

    @NotNull
    private boolean isActiveUser;

    @NotNull
    @Size(min = 5, max = 20, message = "min length 5 , max length 20")
    private String password;

    private Customers.Role role;

    public Customers mapToCustomer() {
```

```
41          return Customers.builder()
42                  .emailId(this.emailId)
43                  .fname(this.fname)
44                  .lname(this.lname)
45                  .age(this.age)
46                  .isActiveUser(this.isActiveUser)
47                  .password(this.password)
48                  .role(this.role)
49                  .address(this.address)
50                  .build();
51      }
52
53
54  }
55
```

```java
package com.sportshoes.ecom.entity.JSONMappers;

import com.fasterxml.jackson.annotation.JsonFilter;
import com.sportshoes.ecom.entity.Category;
import com.sun.istack.NotNull;
import lombok.Data;
import org.hibernate.annotations.CreationTimestamp;

import javax.persistence.Column;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import java.time.LocalDateTime;

@Data
public class ProductJSONFilter {

    private final String name;

    private final int price;

    private final Long categoryId;

}

```

```java
package com.sportshoes.ecom.entity.JSONMappers;

import lombok.Data;
import lombok.Getter;
import lombok.Setter;

import javax.validation.constraints.Size;

@Getter
@Setter
public class CategoryJSONMapper {
    @Size(min = 3, max = 20, message = "Category name size 3-20 chars")
    private  String name;
}

```

File - F:\SportsShoesEcomApp\src\main\java\com\sportshoes\ecom\entity\JSONMappers\PurchaseObjectMapper.java

```java
package com.sportshoes.ecom.entity.JSONMappers;

public class PurchaseObjectMapper {
}

```

File - F:\SportsShoesEcomApp\src\main\java\com\sportshoes\ecom\security\JWTUtil.java

```java
package com.sportshoes.ecom.security;

import com.sportshoes.ecom.entity.Customers;
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.function.Function;
@Component
public class JWTUtil {
    private final String mySecretKey="my_secret_key";
    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(
mySecretKey).parseClaimsJws(token).getBody();
    }

    public boolean isTokenExpired(String token) {
        return extractExpiration(token).before(new
Date());
    }
    //generating new token
    public String generateToken(UserDetails customer
```

```
37 ) {
38         HashMap<String, Object> map = new HashMap
   <>();
39         return createToken(map, customer.getUsername
   ());
40     }
41
42     private String createToken(HashMap<String, Object
   > map, String emailId) {
43         return Jwts.builder().setClaims(map)
44                 .setSubject(emailId)
45                 .setIssuedAt(new Date(System.
   currentTimeMillis()))
46                 .setExpiration(new Date(System.
   currentTimeMillis()*60*1000*60*10))
47                 .signWith(SignatureAlgorithm.HS256,
   mySecretKey).compact();
48     }
49
50     public boolean validateToken(String token,
   UserDetails userDetails) {
51         String extractedUsername = extractUserName(
   token);
52         String actualUsername = userDetails.
   getUsername();
53         String token1 = token;
54         boolean expired = isTokenExpired(token);
55         return actualUsername.equals(
   extractedUsername) && !isTokenExpired(token);
56     }
57 }
58
```

File - F:\SportsShoesEcomApp\src\main\java\com\sportshoes\ecom\security\SecurityConfig.java

```java
package com.sportshoes.ecom.security;

import com.sportshoes.ecom.entity.Customers;
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.AuthenticationManagerBuilder;
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.WebSecurityConfigurerAdapter;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    private UserDetailsService userDetailsService;
    private BCryptPasswordEncoder passwordEncoder;

    @Autowired
    public void setUserDetailsService(UserDetailsService userDetailsService) {
        this.userDetailsService = userDetailsService;
    }
    @Autowired
    JWTRequestFilter jwtRequestFilter;

```

```
31
32      @Autowired
33      public void setPasswordEncoder(
    BCryptPasswordEncoder passwordEncoder) {
34          this.passwordEncoder = passwordEncoder;
35      }
36
37      @Override
38      protected void configure(HttpSecurity http)
    throws Exception {
39              http.csrf().disable().authorizeRequests()
40          /*.antMatchers("/v2/api-docs",
41                  "/configuration/ui",
42                  "/swagger-resources/**",
43                  "/configuration/security",
44                  "/swagger-ui.html",
45                  "/webjars/**").permitAll()*/
46                      .antMatchers("/api/admin/admin/*"
    ).hasRole("ADMIN")
47                      .antMatchers("/api/customer/*").
    hasAnyRole("USER", "ADMIN")
48                      .anyRequest().permitAll();
49              http.addFilterBefore(jwtRequestFilter,
    UsernamePasswordAuthenticationFilter.class);
50      }
51
52      @Override
53      protected void configure(
    AuthenticationManagerBuilder auth) throws Exception {
54          auth.userDetailsService(userDetailsService);
55      }
56
57      @Bean
58      @Override
59      public AuthenticationManager
    authenticationManagerBean() throws Exception {
60          return super.authenticationManagerBean();
61      }
62  }
63
```

```java
package com.sportshoes.ecom.security;

import lombok.extern.slf4j.Slf4j;
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.core.userdetails.UserDetailsService;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@Slf4j
@Component
public class JWTRequestFilter extends OncePerRequestFilter {

    private JWTUtil jwtUtil;
    private UserDetailsService service;
    @Autowired
    public void setJwtUtil(JWTUtil jwtUtil) {
        this.jwtUtil = jwtUtil;
    }
    @Autowired
    public void setApplicationUserDetails(UserDetailsService service) {
        this.service = service;
    }

    @Override
    protected void doFilterInternal(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, FilterChain
```

```
34 filterChain) throws ServletException, IOException {
35         logger.info("Entering JWT Filter");
36         String header = httpServletRequest.getHeader(
   "Authorization");
37         String uname = null;
38         String token = null;
39         if(header != null && header.startsWith("
   Bearer ")) {
40              token= header.substring(7);
41             uname = jwtUtil.extractUserName(token);
42         }
43         if(uname != null && SecurityContextHolder.
   getContext().getAuthentication() == null) {
44             UserDetails userDetails = this.service.
   loadUserByUsername(uname);
45             boolean validateToken = jwtUtil.
   validateToken(token, userDetails);
46
47             logger.debug("Is valid token + " +
   validateToken);
48             if(jwtUtil.validateToken(token,
   userDetails)) {
49                 UsernamePasswordAuthenticationToken
   authenticationToken
50                         = new
   UsernamePasswordAuthenticationToken(userDetails, null
   , userDetails.getAuthorities());
51             SecurityContextHolder.getContext().
   setAuthentication(authenticationToken);
52             }
53         }
54         logger.debug("End of JUTIl Filter for " +
   uname + " with token " + token );
55         logger.debug("Exiting JWT filter");
56         filterChain.doFilter(httpServletRequest,
   httpServletResponse);
57     }
58 }
59
```

```java
package com.sportshoes.ecom.security;

import com.sportshoes.ecom.entity.Customers;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import java.util.Arrays;
import java.util.Collection;
import java.util.Collections;


public class ApplicationUserDetails implements UserDetails{

    private final Customers customers;

    public ApplicationUserDetails(Customers customers) {
        this.customers = customers;
    }




    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        System.out.println(customers.getRole().name());
        SimpleGrantedAuthority authority = new SimpleGrantedAuthority(customers.getRole().name());
        System.out.println(authority);
        return Collections.singletonList(authority);
    }

    @Override
    public String getPassword() {
        return customers.getPassword();
    }

    @Override
```

```
37      public String getUsername() {
38          return customers.getEmailId();
39      }
40
41      @Override
42      public boolean isAccountNonExpired() {
43          return true;
44      }
45
46      @Override
47      public boolean isAccountNonLocked() {
48          return true;
49      }
50
51      @Override
52      public boolean isCredentialsNonExpired() {
53          return true;
54      }
55
56      @Override
57      public boolean isEnabled() {
58          return true;
59      }
60
61      public Long getUserID() {
62          return this.customers.getID();
63      }
64  }
65
```

```java
package com.sportshoes.ecom.security;

import com.sportshoes.ecom.entity.Customers;
import com.sportshoes.ecom.exceptions.ProductNotFoundException;
import com.sportshoes.ecom.repos.CustomerRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Primary;
import org.springframework.context.annotation.Profile;
import org.springframework.security.authentication.BadCredentialsException;
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 javax.security.auth.login.CredentialException;

@Service
@Primary
@Profile("!test")
public class UserDetailsServiceImpl implements UserDetailsService {
    @Autowired
    private CustomerRepo customerRepo;

    @Override
    public UserDetails loadUserByUsername(String s) throws UsernameNotFoundException {
        Customers user = customerRepo.getUserByUserName(s);
        //Exception to be implemented and changed
        if(user == null) {
            System.out.println("Could not find user " + " + s );
            throw new BadCredentialsException("Bad username : " + s);
        }
```

```
32          return new ApplicationUserDetails(user);
33      }
34  }
35
```