**Title: compiler.xml**

**Title: encodings.xml**

**Title: jarRepositories.xml**

**Title: misc.xml**

**Title: uiDesigner.xml**

**Title: vcs.xml**

**Title: workspace.xml**

**Title: maven-wrapper.properties**

distributionUrl=https://repo.maven.apache.org/maven2/org/apache/maven/apache-maven/3.9.5/apache-maven-3.9.5-bin.zip
wrapperUrl=https://repo.maven.apache.org/maven2/org/apache/maven/wrapper/maven-wrapper/3.2.0/maven-wrapper-3.2.0.jar

**Title: pom.xml**

**Title: AppConstants.java**

package com.sephora.ecommerce.config;
public class AppConstants {
public static final Long ADMIN\_ID = 101L;
public static final Long USER\_ID = 102L;
public static final long JWT\_TOKEN\_VALIDITY = 5 \* 60 \* 60;
public static final String PAGE\_NUMBER = "0";
public static final String PAGE\_SIZE = "2";
public static final String SORT\_CATEGORIES\_BY = "categoryId";
public static final String SORT\_PRODUCTS\_BY = "productId";
public static final String SORT\_USERS\_BY = "userId";
public static final String SORT\_ORDERS\_BY = "totalAmount";
public static final String SORT\_DIR = "asc";
public static final String[] PUBLIC\_URLS = {"/v3/api-docs/\*\*", "/swagger-ui/\*\*","/api/register/\*\*","/api/login"};
public static final String[] USER\_URLS = {"/api/public/\*\*"};
public static final String[] ADMIN\_URLS = {"/api/admin/\*\*"};
}

**Title: CacheConfig.java**

package com.sephora.ecommerce.config;
import org.springframework.cache.CacheManager;
import org.springframework.cache.annotation.EnableCaching;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.cache.RedisCacheManager;
import org.springframework.data.redis.connection.RedisConnectionFactory;
@Configuration
@EnableCaching
public class CacheConfig {
@Bean
public CacheManager cacheManager(RedisConnectionFactory connectionFactory) {
return RedisCacheManager.builder(connectionFactory).build();
}
}

**Title: MyConfig.java**

package com.sephora.ecommerce.config;
import org.modelmapper.ModelMapper;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
@Configuration
public class MyConfig {
@Bean
public ModelMapper modelMapper() {
return new ModelMapper();
}
}

**Title: RedisConfig.java**

package com.sephora.ecommerce.config;
import org.springframework.context.annotation.Configuration;
import org.springframework.session.data.redis.config.annotation.web.http.EnableRedisHttpSession;
@Configuration
@EnableRedisHttpSession
public class RedisConfig {
}

**Title: AddressController.java**

**Title: CartController.java**

**Title: CategoryController.java**

**Title: OrderController.java**

**Title: ProductController.java**

**Title: UserController.java**

**Title: ECommerceApplication.java**

package com.sephora.ecommerce;
import com.sephora.ecommerce.config.AppConstants;
import com.sephora.ecommerce.entities.Role;
import com.sephora.ecommerce.repositories.RoleRepository;
import jakarta.annotation.PostConstruct;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import java.util.List;
@SpringBootApplication
public class ECommerceApplication implements CommandLineRunner {
private RoleRepository roleRepository;
public ECommerceApplication(RoleRepository roleRepository) {
this.roleRepository = roleRepository;
}
public static void main(String[] args) {
SpringApplication.run(ECommerceApplication.class, args);
}
@Override
//@PostConstruct //to ensure method run only once.
public void run(String... args) throws Exception {
try {
Role adminRole = new Role();
adminRole.setRoleId(AppConstants.ADMIN\_ID);
adminRole.setRoleName("ADMIN");
Role userRole = new Role();
userRole.setRoleId(AppConstants.USER\_ID);
userRole.setRoleName("USER");
List<Role> roles = List.of(adminRole, userRole);
List<Role> savedRoles = roleRepository.saveAll(roles);
savedRoles.forEach(System.out::println);
} catch (Exception e) {
e.printStackTrace();
}
}
}

**Title: Address.java**

package com.sephora.ecommerce.entities;
import jakarta.persistence.\*;
import lombok.\*;
import java.util.ArrayList;
import java.util.List;
@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@Table(name = "addresses")
@EqualsAndHashCode(onlyExplicitlyIncluded = true)
public class Address {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
@EqualsAndHashCode.Include
private Long addressId;
private String localAddress;
private String city;
private String state;
private String country;
private String pinCode;
@ManyToOne
@JoinColumn(name = "user\_id")
@EqualsAndHashCode.Exclude
private User user;
}

**Title: Cart.java**

package com.sephora.ecommerce.entities;
import jakarta.persistence.\*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import java.util.ArrayList;
import java.util.List;
@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@Table(name = "carts")
public class Cart {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long cartId;
private Double totalPrice = 0.0;
@OneToOne
@JoinColumn(name = "user\_id")
private User user;
@OneToMany(mappedBy = "cart", cascade = {CascadeType.PERSIST, CascadeType.MERGE}, orphanRemoval = true)
private List<CartItem> cartItems = new ArrayList<>();
}

**Title: CartItem.java**

package com.sephora.ecommerce.entities;
import jakarta.persistence.\*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@Table(name = "cart\_items")
public class CartItem {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long cartItemId;
private Integer quantity;
private Double discount;
private Double productPrice;
@ManyToOne
@JoinColumn(name = "cart\_id")
private Cart cart;
@ManyToOne
@JoinColumn(name = "product\_id")
private Product product;
}

**Title: Category.java**

package com.sephora.ecommerce.entities;
import com.fasterxml.jackson.annotation.JsonBackReference;
import jakarta.persistence.\*;
import lombok.\*;
import java.util.HashSet;
import java.util.Set;
@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@Table(name = "categories")
@ToString(exclude = "products")
public class Category {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long categoryId;
@Column(unique = true)
private String categoryName;
@ManyToMany(mappedBy = "categories", fetch = FetchType.EAGER)
private Set<Product> products = new HashSet<>();
}

**Title: Order.java**

package com.sephora.ecommerce.entities;
import jakarta.persistence.\*;
import lombok.\*;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;
@Entity
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@Table(name = "orders")
public class Order {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long orderId;
private String email;
private LocalDate orderDate;
@OneToOne
@JoinColumn(name = "payment\_id")
private Payment payment;
@OneToMany(mappedBy = "order", cascade = { CascadeType.PERSIST, CascadeType.MERGE })
private List<OrderItem> orderItems = new ArrayList<>();
private Double totalAmount;
private String orderStatus;
}

**Title: OrderItem.java**

package com.sephora.ecommerce.entities;
import jakarta.persistence.\*;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@Entity
@Table(name = "orderItems")
public class OrderItem {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long orderItemId;
private Integer quantity;
private Double discount;
private Double orderedProductPrice;
@ManyToOne
@JoinColumn(name = "product\_id")
private Product product;
@ManyToOne
@JoinColumn(name = "order\_id")
private Order order;
}

**Title: Payment.java**

package com.sephora.ecommerce.entities;
import jakarta.persistence.\*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@Table(name = "payments")
public class Payment {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long paymentId;
@OneToOne(mappedBy = "payment", cascade = {CascadeType.PERSIST, CascadeType.MERGE})
private Order order;
private String paymentMethod;
}

**Title: Product.java**

**Title: Role.java**

package com.sephora.ecommerce.entities;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
@Entity
@AllArgsConstructor
@NoArgsConstructor
@Table(name = "roles")
@Data
public class Role {
@Id
private Long roleId;
private String roleName;
}

**Title: User.java**

package com.sephora.ecommerce.entities;
import jakarta.persistence.\*;
import lombok.\*;
import java.util.HashSet;
import java.util.Set;
@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@Table(name = "users")
@EqualsAndHashCode(onlyExplicitlyIncluded = true)
public class User {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
@EqualsAndHashCode.Include
private Long userId;
private String userName;
private String mobileNumber;
private String email;
private String password;
@ManyToMany(cascade = {CascadeType.PERSIST, CascadeType.MERGE},
fetch = FetchType.EAGER)
@JoinTable(name = "user\_role",
joinColumns = @JoinColumn(name = "user\_id"),
inverseJoinColumns = @JoinColumn(name = "role\_id"))
private Set<Role> roles = new HashSet<>();
@ToString.Exclude
@OneToMany(mappedBy = "user", cascade = CascadeType.ALL, fetch = FetchType.EAGER)
@EqualsAndHashCode.Exclude
private Set<Address> addresses = new HashSet<>();
@OneToOne(mappedBy = "user", cascade = {CascadeType.PERSIST, CascadeType.MERGE}, orphanRemoval = true)
private Cart cart;
}

**Title: APIException.java**

package com.sephora.ecommerce.exceptions;
public class APIException extends RuntimeException{
private static final long serialVersionUID = 1L;
public APIException(){}
public APIException(String message){
super(message);
}
}

**Title: GlobalExceptionHandler.java**

package com.sephora.ecommerce.exceptions;
import com.sephora.ecommerce.payloads.APIResponse;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RestControllerAdvice;
@RestControllerAdvice
public class GlobalExceptionHandler {
@ExceptionHandler(ResourceNotFoundException.class)
public ResponseEntity<?> resourceNotFoundEException(ResourceNotFoundException e){
String message = e.getMessage();
APIResponse apiResponse = new APIResponse(message, false);
return new ResponseEntity<>(apiResponse, HttpStatus.BAD\_REQUEST);
}
@ExceptionHandler(APIException.class)
public ResponseEntity<?> apiException(APIException e){
String message = e.getMessage();
APIResponse apiResponse = new APIResponse(message, false);
return new ResponseEntity<>(apiResponse, HttpStatus.BAD\_REQUEST);
}
}

**Title: ResourceNotFoundException.java**

package com.sephora.ecommerce.exceptions;
public class ResourceNotFoundException extends RuntimeException {
private static final long serialVersionUID = 1L;
String resourceName;
String field;
String fieldName;
Long fieldId;
public ResourceNotFoundException() {
}
public ResourceNotFoundException(String resourceName, String field, String fieldName) {
super(String.format("%s not found with %s: %s", resourceName, field, fieldName));
this.resourceName = resourceName;
this.field = field;
this.fieldName = fieldName;
}
public ResourceNotFoundException(String resourceName, String field, Long fieldId) {
super(String.format("%s not found with %s: %d", resourceName, field, fieldId));
this.resourceName = resourceName;
this.field = field;
this.fieldId = fieldId;
}
}

**Title: AddressDTO.java**

package com.sephora.ecommerce.payloads;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
@Data
@AllArgsConstructor
@NoArgsConstructor
public class AddressDTO {
private Long addressId;
private String localAddress;
private String city;
private String state;
private String country;
private String pinCode;
}

**Title: AddressWithUserIdDTO.java**

package com.sephora.ecommerce.payloads;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
@Data
@AllArgsConstructor
@NoArgsConstructor
public class AddressWithUserIdDTO {
private Long addressId;
private String localAddress;
private String city;
private String state;
private String country;
private String pinCode;
private Long userId;
}

**Title: APIResponse.java**

package com.sephora.ecommerce.payloads;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
@Data
@NoArgsConstructor
@AllArgsConstructor
public class APIResponse {
private String message;
private boolean status;
}

**Title: CartDTO.java**

package com.sephora.ecommerce.payloads;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import java.util.ArrayList;
import java.util.List;
@Data
@AllArgsConstructor
@NoArgsConstructor
public class CartDTO {
private Long cartId;
private Double totalPrice = 0.0;
private List<ProductDTO> productDTOList = new ArrayList<>();
}

**Title: CartItemsDTO.java**

package com.sephora.ecommerce.payloads;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
@Data
@AllArgsConstructor
@NoArgsConstructor
public class CartItemsDTO {
private Long cartItemId;
private Integer quantity;
private Double discount;
private Double productPrice;
private CartDTO cartDTO;
private ProductDTO productDTO;
}

**Title: CategoryDTO.java**

package com.sephora.ecommerce.payloads;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
@Data
@AllArgsConstructor
@NoArgsConstructor
public class CategoryDTO {
private Long categoryId;
private String categoryName;
}

**Title: CategoryResponse.java**

package com.sephora.ecommerce.payloads;
import java.util.List;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
@Data
@NoArgsConstructor
@AllArgsConstructor
public class CategoryResponse {
private List<CategoryDTO> content;
private Integer pageNumber;
private Integer pageSize;
private Long totalElements;
private Integer totalPages;
private boolean lastPage;
}

**Title: OrderDTO.java**

package com.sephora.ecommerce.payloads;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;
@Data
@AllArgsConstructor
@NoArgsConstructor
public class OrderDTO {
private Long orderId;
private String email;
private LocalDate orderDate;
private PaymentDTO payment;
private Double totalAmount;
private String orderStatus;
private List<OrderItemDTO> orderItems = new ArrayList<>();
}

**Title: OrderItemDTO.java**

package com.sephora.ecommerce.payloads;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
public class OrderItemDTO {
private Long orderItemId;
private Integer quantity;
private double discount;
private double orderedProductPrice;
private ProductDTO product;
}

**Title: OrderResponse.java**

package com.sephora.ecommerce.payloads;
import java.util.List;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
@Data
@NoArgsConstructor
@AllArgsConstructor
public class OrderResponse {
private List<OrderDTO> content;
private Integer pageNumber;
private Integer pageSize;
private Long totalElements;
private Integer totalPages;
private boolean lastPage;
}

**Title: PaymentDTO.java**

package com.sephora.ecommerce.payloads;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
@Data
@AllArgsConstructor
@NoArgsConstructor
public class PaymentDTO {
private Long paymentId;
private String paymentMethod;
}

**Title: ProductDTO.java**

package com.sephora.ecommerce.payloads;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import java.util.Set;
@Data
@AllArgsConstructor
@NoArgsConstructor
public class ProductDTO {
private Long productId;
private String productName;
private String image;
private String description;
private Integer quantity;
private Double price;
private Double discount;
private Double specialPrice;
private Set<String> categoryNames;
}

**Title: ProductResponse.java**

package com.sephora.ecommerce.payloads;
import java.util.List;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
@Data
@NoArgsConstructor
@AllArgsConstructor
public class ProductResponse {
private List<ProductDTO> content;
private Integer pageNumber;
private Integer pageSize;
private Long totalElements;
private Integer totalPages;
private boolean lastPage;
}

**Title: UserDTO.java**

package com.sephora.ecommerce.payloads;
import com.sephora.ecommerce.entities.Role;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import java.util.HashSet;
import java.util.Set;
@Data
@NoArgsConstructor
@AllArgsConstructor
public class UserDTO {
private Long userId;
private String userName;
private String mobileNumber;
private String email;
private String password;
// private Set<Role> roles = new HashSet<>();
private Set<AddressDTO> addresses = new HashSet<>();
}

**Title: UserResponse.java**

package com.sephora.ecommerce.payloads;
import lombok.\*;
import java.util.List;
@Data
@NoArgsConstructor
@AllArgsConstructor
public class UserResponse {
private List<UserWithCartDTO> content;
private Integer pageNumber;
private Integer pageSize;
private Long totalElements;
private Integer totalPages;
private boolean lastPage;
}

**Title: UserWithCartDTO.java**

package com.sephora.ecommerce.payloads;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import java.util.HashSet;
import java.util.Set;
@Data
@NoArgsConstructor
@AllArgsConstructor
public class UserWithCartDTO {
private Long userId;
private String userName;
private String mobileNumber;
private String email;
private String password;
// private Set<Role> roles = new HashSet<>();
private Set<AddressDTO> addresses = new HashSet<>();
private CartDTO cartDTO;
}

**Title: AddressRepository.java**

package com.sephora.ecommerce.repositories;
import com.sephora.ecommerce.entities.Address;
import com.sephora.ecommerce.entities.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import java.util.List;
public interface AddressRepository extends JpaRepository<Address, Long> {
@Query("SELECT COUNT(a) FROM Address a WHERE a.user = :user AND a.localAddress = :localAddress AND a.city = :city AND a.state = :state AND a.country = :country AND a.pinCode = :pinCode")
long countByAddressDetails(User user, String localAddress, String city, String state, String country, String pinCode);
}

**Title: CartItemRepository.java**

package com.sephora.ecommerce.repositories;
import com.sephora.ecommerce.entities.CartItem;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
public interface CartItemRepository extends JpaRepository<CartItem, Long> {
@Query("SELECT ci FROM CartItem ci WHERE ci.cart.id = ?1 AND ci.product.id = ?2")
CartItem findCartItemByProductIdAndCartId(Long cartId, Long productId);
@Modifying
@Query("DELETE FROM CartItem ci WHERE ci.cart.id = ?1 AND ci.product.id = ?2")
void deleteCartItemByProductIdAndCartId(Long productId, Long cartId);
}

**Title: CartRepository.java**

package com.sephora.ecommerce.repositories;
import com.sephora.ecommerce.entities.Cart;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import java.util.List;
public interface CartRepository extends JpaRepository<Cart, Long> {
@Query("SELECT c FROM Cart c JOIN FETCH c.cartItems ci JOIN FETCH ci.product p WHERE p.id = ?1")
List<Cart> findCartsByProductId(Long productId);
@Query("SELECT c FROM Cart c WHERE c.user.email = ?1 AND c.id = ?2")
Cart findCartByEmailAndCartId(String email, Long cartId);
}

**Title: CategoryRepository.java**

package com.sephora.ecommerce.repositories;
import com.sephora.ecommerce.entities.Category;
import org.springframework.data.jpa.repository.JpaRepository;
public interface CategoryRepository extends JpaRepository<Category, Long> {
Category findByCategoryName(String categoryName);
boolean existsByCategoryName(String newCategoryName);
}

**Title: CustomProductRepository.java**

//package com.sephora.ecommerce.repositories;
//
//import org.springframework.stereotype.Repository;
//
//public interface CustomProductRepository {
// void deleteProductCategories(Long productId);
// void addProductCategory(Long productId, Long categoryId);
//}

**Title: CustomProductRepositoryImpl.java**

//package com.sephora.ecommerce.repositories;
//
//import jakarta.persistence.EntityManager;
//import jakarta.persistence.PersistenceContext;
//import org.springframework.stereotype.Repository;
//import org.springframework.transaction.annotation.Transactional;
//
//@Repository
//public class CustomProductRepositoryImpl implements CustomProductRepository{
//
// @PersistenceContext
// private EntityManager entityManager;
//
// @Override
// @Transactional
// public void deleteProductCategories(Long productId) {
// entityManager.createNativeQuery("DELETE FROM product\_category WHERE product\_id = :productId")
// .setParameter("productId", productId)
// .executeUpdate();
// }
//
// @Override
// @Transactional
// public void addProductCategory(Long productId, Long categoryId) {
// entityManager.createNativeQuery("INSERT INTO product\_category (product\_id, category\_id) VALUES (:productId, :categoryId)")
// .setParameter("productId", productId)
// .setParameter("categoryId", categoryId)
// .executeUpdate();
// }
//}

**Title: OrderItemRepo.java**

package com.sephora.ecommerce.repositories;
import com.sephora.ecommerce.entities.Order;
import com.sephora.ecommerce.entities.OrderItem;
import org.springframework.data.jpa.repository.JpaRepository;
import java.util.List;
public interface OrderItemRepo extends JpaRepository<OrderItem, Long> {
}

**Title: OrderRepo.java**

package com.sephora.ecommerce.repositories;
import com.sephora.ecommerce.entities.Order;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import java.util.List;
public interface OrderRepo extends JpaRepository<Order, Long> {
// @Query("SELECT o FROM Order o WHERE o.email = ?1 AND o.id = ?2")
Order findByEmailAndOrderId(String emailId, Long orderId);
List<Order> findAllByEmail(String emailId);
}

**Title: PaymentRepo.java**

package com.sephora.ecommerce.repositories;
import com.sephora.ecommerce.entities.Payment;
import org.springframework.data.jpa.repository.JpaRepository;
public interface PaymentRepo extends JpaRepository<Payment, Long> {
}

**Title: ProductRepository.java**

**Title: RoleRepository.java**

package com.sephora.ecommerce.repositories;
import com.sephora.ecommerce.entities.Role;
import org.springframework.data.jpa.repository.JpaRepository;
public interface RoleRepository extends JpaRepository<Role, Long> {
}

**Title: UserRepository.java**

package com.sephora.ecommerce.repositories;
import com.sephora.ecommerce.entities.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import java.util.List;
public interface UserRepository extends JpaRepository<User, Long> {
// @Query("SELECT u FROM User u JOIN FETCH u.addresses a WHERE a.addressId = ?1")
// List<User> findByAddress(Long addressId);
}

**Title: AddressService.java**

package com.sephora.ecommerce.services;
import com.sephora.ecommerce.entities.Address;
import com.sephora.ecommerce.payloads.AddressDTO;
import com.sephora.ecommerce.payloads.AddressWithUserIdDTO;
import java.util.List;
import java.util.Set;
public interface AddressService {
Set<AddressWithUserIdDTO> createAddresses(Set<AddressWithUserIdDTO> addressDTOList);
AddressWithUserIdDTO createAddress(AddressWithUserIdDTO addressDTO);
List<AddressWithUserIdDTO> getAllAddresses();
AddressWithUserIdDTO getAddressById(Long addressId);
//Here using AddressDTO instead of Address
AddressWithUserIdDTO updateAddressById(Long addressId, AddressWithUserIdDTO addressDTO);
String deleteAddress(Long addressId);
}

**Title: CartService.java**

package com.sephora.ecommerce.services;
import com.sephora.ecommerce.payloads.CartDTO;
import java.util.List;
public interface CartService {
CartDTO addProductToCart(Long cartId, Long productId, Integer quantity);
List<CartDTO> getAllCarts();
CartDTO getCartByCartIdAndEmailId(String emailId, Long cartId);
CartDTO updateProductQuantityInCart(Long cartId, Long productId, Integer quantity);
void updateProductInCarts(Long cartId, Long productId);
String deleteProductFromCart(Long cartId, Long productId);
}

**Title: CategoryService.java**

package com.sephora.ecommerce.services;
import com.sephora.ecommerce.entities.Category;
import com.sephora.ecommerce.payloads.CategoryDTO;
import com.sephora.ecommerce.payloads.CategoryResponse;
import java.util.List;
public interface CategoryService {
CategoryDTO createCategory(CategoryDTO categoryDTO);
CategoryResponse getAllCategories(Integer pageNumber, Integer pageSize, String sortBy, String sortOrder);
CategoryDTO updateCategory(String categoryName, String newCategoryName);
String deleteCategory(String identifier);
}

**Title: FileService.java**

package com.sephora.ecommerce.services;
import org.springframework.web.multipart.MultipartFile;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
public interface FileService {
String uploadImage(String path, MultipartFile file) throws IOException;
InputStream getResource(String path, String fileName) throws FileNotFoundException;
}

**Title: AddressServiceImpl.java**

**Title: AddressServiceImpl2.java**

**Title: CartServiceImpl.java**

**Title: CategoryServiceImpl.java**

package com.sephora.ecommerce.services.impl;
import com.sephora.ecommerce.entities.Category;
import com.sephora.ecommerce.entities.User;
import com.sephora.ecommerce.exceptions.APIException;
import com.sephora.ecommerce.payloads.CategoryDTO;
import com.sephora.ecommerce.payloads.CategoryResponse;
import com.sephora.ecommerce.payloads.UserResponse;
import com.sephora.ecommerce.repositories.CategoryRepository;
import com.sephora.ecommerce.services.CategoryService;
import lombok.extern.slf4j.Slf4j;
import org.modelmapper.ModelMapper;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import java.util.List;
import java.util.stream.Collectors;
@Service
@Transactional
@Slf4j
public class CategoryServiceImpl implements CategoryService {
private CategoryRepository categoryRepository;
private ModelMapper modelMapper;
public CategoryServiceImpl(CategoryRepository categoryRepository, ModelMapper modelMapper) {
this.categoryRepository = categoryRepository;
this.modelMapper = modelMapper;
}
@Override
public CategoryDTO createCategory(CategoryDTO categoryDTO) {
Category category = modelMapper.map(categoryDTO, Category.class);
Category savedCategory = categoryRepository.findByCategoryName(category.getCategoryName());
if (savedCategory != null) {
throw new APIException("Category with the name '" + category.getCategoryName() + "' already exists!!!");
}
savedCategory = categoryRepository.save(category);
return modelMapper.map(savedCategory, CategoryDTO.class);
}
@Override
public CategoryResponse getAllCategories(Integer pageNumber, Integer pageSize, String sortBy, String sortOrder) {
Sort sortByAndOrder = sortOrder.equalsIgnoreCase("asc") ? Sort.by(sortBy).ascending()
: Sort.by(sortBy).descending();
Pageable pageDetails = PageRequest.of(pageNumber, pageSize, sortByAndOrder);
Page<Category> pageCategories = categoryRepository.findAll(pageDetails);
List<Category> categories = pageCategories.getContent();
List<CategoryDTO> categoryDTOs = categories.stream().map(category -> modelMapper.map(category, CategoryDTO.class)).collect(Collectors.toList());
CategoryResponse categoryResponse = new CategoryResponse();
categoryResponse.setContent(categoryDTOs);
categoryResponse.setPageNumber(pageCategories.getNumber());
categoryResponse.setPageSize(pageCategories.getSize());
categoryResponse.setTotalElements(pageCategories.getTotalElements());
categoryResponse.setTotalPages(pageCategories.getTotalPages());
categoryResponse.setLastPage(pageCategories.isLast());
return categoryResponse;
}
@Override
public CategoryDTO updateCategory(String categoryName, String newCategoryName) {
Category savedCategory = categoryRepository.findByCategoryName(categoryName);
if (savedCategory == null) {
throw new APIException("Category with the name '" + categoryName + "' does not exist!!!");
}
if (!newCategoryName.equals(categoryName) && categoryRepository.existsByCategoryName(newCategoryName)) {
throw new APIException("Category with the name '" + newCategoryName + "' already exists!!!");
}
savedCategory.setCategoryName(newCategoryName);
Category updatedCategory = categoryRepository.save(savedCategory);
return modelMapper.map(updatedCategory, CategoryDTO.class);
}
@Override
public String deleteCategory(String identifier) {
Category categoryToDelete = categoryRepository.findByCategoryName(identifier);
if (categoryToDelete != null) {
categoryRepository.delete(categoryToDelete);
return "Category with Name '" + identifier + "' and ID '"+categoryToDelete.getCategoryId()+"' deleted Successfully!";
}
try{
Long categoryId = Long.parseLong(identifier);
categoryToDelete = categoryRepository.findById(categoryId).orElse(null);
if (categoryToDelete != null) {
categoryRepository.delete(categoryToDelete);
return "Category with Name '"+categoryToDelete.getCategoryName()+"' and ID '" + categoryId + "' deleted Successfully!";
}
} catch (NumberFormatException e) {
log.error("The {} cannot be parsed as Long, it's neither categoryName nor categoryId; exception {}", identifier, e);
}
throw new APIException("Category with name/id '" + identifier + "' does not exist!");
}
}

**Title: FileServiceImpl.java**

package com.sephora.ecommerce.services.impl;
import com.sephora.ecommerce.services.FileService;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;
import java.io.\*;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.UUID;
@Service
public class FileServiceImpl implements FileService {
@Override
public String uploadImage(String path, MultipartFile file) throws IOException {
String originalFilename = file.getOriginalFilename();
String randomId = UUID.randomUUID().toString();
String fileName = randomId.concat(originalFilename.substring(originalFilename.lastIndexOf('.')));
String filePath = path + File.separator + fileName;
File folder = new File(path);
if (!folder.exists()){
folder.mkdir();
}
Files.copy(file.getInputStream(), Paths.get(filePath));
return fileName;
}
@Override
public InputStream getResource(String path, String fileName) throws FileNotFoundException {
String filePath = path + File.separator + fileName;
InputStream inputStream = new FileInputStream(filePath);
return inputStream;
}
}

**Title: OrderServiceImpl.java**

**Title: ProductServiceImpl.java**

**Title: UserServiceImpl.java**

//package com.sephora.ecommerce.services.impl;
//
//import com.sephora.ecommerce.config.AppConstants;
//import com.sephora.ecommerce.entities.Address;
//import com.sephora.ecommerce.entities.Role;
//import com.sephora.ecommerce.entities.User;
//import com.sephora.ecommerce.exceptions.APIException;
//import com.sephora.ecommerce.exceptions.ResourceNotFoundException;
//import com.sephora.ecommerce.payloads.AddressDTO;
//import com.sephora.ecommerce.payloads.UserDTO;
//import com.sephora.ecommerce.repositories.AddressRepository;
//import com.sephora.ecommerce.repositories.RoleRepository;
//import com.sephora.ecommerce.repositories.UserRepository;
//import com.sephora.ecommerce.services.UserService;
//import org.modelmapper.ModelMapper;
//import org.springframework.dao.DataIntegrityViolationException;
//import org.springframework.stereotype.Service;
//import org.springframework.transaction.annotation.Transactional;
//
//import java.util.List;
//import java.util.stream.Collectors;
//
//@Service
//@Transactional
//public class UserServiceImpl implements UserService {
//
// private UserRepository userRepository;
// private RoleRepository roleRepository;
// private ModelMapper modelMapper;
// private AddressRepository addressRepository;
//
// public UserServiceImpl(UserRepository userRepository, RoleRepository roleRepository, ModelMapper modelMapper, AddressRepository addressRepository) {
// this.userRepository = userRepository;
// this.roleRepository = roleRepository;
// this.modelMapper = modelMapper;
// this.addressRepository = addressRepository;
// }
//
//
// @Override
// public UserDTO registerUser(UserDTO userDTO) {
// try {
// User user = modelMapper.map(userDTO, User.class);
//
// Role role = roleRepository.findById(AppConstants.USER\_ID).get();
// user.getRoles().add(role);
//
// String localAddress = userDTO.getAddressDTO().getLocalAddress();
// String city = userDTO.getAddressDTO().getCity();
// String state = userDTO.getAddressDTO().getState();
// String country = userDTO.getAddressDTO().getCountry();
// String pinCode = userDTO.getAddressDTO().getPinCode();
//
// Address address = addressRepository.findByLocalAddressAndCityAndStateAndCountryAndPinCode(localAddress, city, state, country, pinCode);
// if(address==null){
// address = new Address(localAddress, city, state, country, pinCode);
// addressRepository.save(address);
// }
//
// user.setAddresses(List.of(address));
//
// User registeredUser = userRepository.save(user);
//
// userDTO = modelMapper.map(registeredUser, UserDTO.class);
//
// userDTO.setAddressDTO(modelMapper.map(user.getAddresses().stream().findFirst().get(), AddressDTO.class));
// return userDTO;
// } catch (DataIntegrityViolationException e){
// throw new APIException("User already exists with emailId: "+userDTO.getEmail());
// }
// }
//
// @Override
// public List<UserDTO> getAllUsers() {
// List<User> users = userRepository.findAll();
// List<UserDTO> usersDTO = users.stream().map(e -> {
// UserDTO dto = modelMapper.map(e, UserDTO.class);
// return dto;
// }).collect(Collectors.toList());
// return usersDTO;
// }
//
// @Override
// public UserDTO getUserById(Long userId) {
// User user = userRepository.findById(userId).orElseThrow(() -> new ResourceNotFoundException("User", "userId", userId));
// UserDTO userDTO = modelMapper.map(user, UserDTO.class);
// return userDTO;
// }
//
// @Override
// public UserDTO updateUser(Long userId, UserDTO userDTO) {
// User user = userRepository.findById(userId).orElseThrow(() -> new ResourceNotFoundException("User", "userId", userId));
// user.setUserName(userDTO.getUserName());
// user.setMobileNumber(userDTO.getMobileNumber());
// user.setEmail(userDTO.getEmail());
// user.setPassword(userDTO.getPassword());
//
// userDTO = modelMapper.map(user, UserDTO.class);
//
// return userDTO;
// }
//
// @Override
// public String deleteUser(Long userId) {
// User user = userRepository.findById(userId).orElseThrow(() -> new ResourceNotFoundException("User", "userId", userId));
// userRepository.delete(user);
// return "User with userId "+userId+" deleted successfully!!!";
// }
//}

**Title: UserServiceImpl2.java**

**Title: OrderService.java**

package com.sephora.ecommerce.services;
import com.sephora.ecommerce.payloads.OrderDTO;
import com.sephora.ecommerce.payloads.OrderResponse;
import java.util.List;
public interface OrderService {
OrderDTO placeOrder(String emailId, Long cartId, String paymentMethod);
OrderDTO getOrder(String emailId, Long orderId);
List<OrderDTO> getOrdersByUser(String emailId);
OrderResponse getAllOrders(Integer pageNumber, Integer pageSize, String sortBy, String sortOrder);
OrderDTO updateOrder(String emailId, Long orderId, String orderStatus);
}

**Title: ProductService.java**

package com.sephora.ecommerce.services;
import com.sephora.ecommerce.entities.Product;
import com.sephora.ecommerce.payloads.ProductDTO;
import com.sephora.ecommerce.payloads.ProductResponse;
import org.springframework.core.io.InputStreamResource;
import org.springframework.core.io.Resource;
import org.springframework.web.multipart.MultipartFile;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.List;
public interface ProductService {
ProductDTO addProduct(ProductDTO productDTO);
ProductResponse getAllProducts(Integer pageNumber, Integer pageSize, String sortBy, String sortOrder);
ProductResponse searchByCategory(String categoryName, Integer pageNumber, Integer pageSize, String sortBy, String sortOrder);
ProductResponse searchByKeyword(String keyword, Integer pageNumber, Integer pageSize, String sortBy, String sortOrder);
ProductDTO updateProduct(Long productId, ProductDTO productDTO);
ProductDTO updateProductImage(Long productId, MultipartFile image) throws IOException;
InputStreamResource showProductImage(String imageName) throws FileNotFoundException;
String deleteProduct(Long productId);
}

**Title: UserService.java**

package com.sephora.ecommerce.services;
import com.sephora.ecommerce.entities.User;
import com.sephora.ecommerce.payloads.UserDTO;
import com.sephora.ecommerce.payloads.UserResponse;
import com.sephora.ecommerce.payloads.UserWithCartDTO;
import java.util.List;
public interface UserService {
UserDTO registerUser(UserDTO userDTO);
UserResponse getAllUsers(Integer pageNumber, Integer pageSize, String sortBy, String sortOrder);
UserDTO getUserById(Long userId);
UserDTO updateUser(Long userId, UserDTO userDTO);
String deleteUser(Long userId);
}

**Title: application.properties**

#db config
spring.datasource.url=jdbc:mysql://localhost:3306/sephora
spring.datasource.username=root
spring.datasource.password=test
#ORM s/w specific properties
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.open-in-view=false
#image
project.image=images/
#spring.servlet.multipart.enabled=true
spring.servlet.multipart.max-file-size=10MB
spring.servlet.multipart.max-request-size=10MB
#spring.jpa.properties.hibernate.show\_sql=true
#spring.jpa.properties.hibernate.use\_sql\_comments=true
#logging.level.org.hibernate.SQL=DEBUG
# Redis configuration
spring.redis.host=localhost
spring.redis.port=6379
logging.file.name=app.log

**Title: ECommerceApplicationTests.java**

package com.sephora.ecommerce;
import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;
@SpringBootTest
class ECommerceApplicationTests {
@Test
void contextLoads() {
}
}

**Title: application.properties**

#db config
spring.datasource.url=jdbc:mysql://localhost:3306/sephora
spring.datasource.username=root
spring.datasource.password=test
#ORM s/w specific properties
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.open-in-view=false
#image
project.image=images/
#spring.servlet.multipart.enabled=true
spring.servlet.multipart.max-file-size=10MB
spring.servlet.multipart.max-request-size=10MB
#spring.jpa.properties.hibernate.show\_sql=true
#spring.jpa.properties.hibernate.use\_sql\_comments=true
#logging.level.org.hibernate.SQL=DEBUG
logging.file.name=app.log