

OrderRequest.java

package com.example.BE\_PBL6\_FastOrderSystem.request;

import lombok.Data;

import java.util.List;

@Data

public class OrderRequest {

private Long userId;

private Long storeId;

private double totalAmount;

private String status;

private String paymentMethod; // "CASH" or "MOMO"

private String deliveryAddress;

private List<OrderDetailRequest> orderDetails;

}

OrderResponse.java

package com.example.BE\_PBL6\_FastOrderSystem.response;

import lombok.Data;

import java.util.List;

@Data

public class OrderResponse {

private Long orderId;

private Long userId;

private Long storeId;

private double totalAmount;

private String status;

private String paymentMethod;

private String deliveryAddress;

private List<OrderDetailResponse> orderDetails;

private boolean paymentSuccess;

}

MoMoPaymentRequest.java

package com.example.BE\_PBL6\_FastOrderSystem.request;

import lombok.AllArgsConstructor;

import lombok.Data;

@Data

@AllArgsConstructor

public class MoMoPaymentRequest {

private String partnerCode;

private String accessKey;

private double amount;

private Long userId;

private String orderId;

private String requestId;

private String signature;

}

MoMoPaymentResponse.java

package com.example.BE\_PBL6\_FastOrderSystem.response;

import lombok.Data;

@Data

public class MoMoPaymentResponse {

private boolean success;

}

HmacSHA256Util.java

package com.example.BE\_PBL6\_FastOrderSystem.util;

import javax.crypto.Mac;

import javax.crypto.spec.SecretKeySpec;

import java.util.Base64;

public class HmacSHA256Util {

public static String generateSignature(String data, String key) throws Exception {

Mac sha256\_HMAC = Mac.getInstance("HmacSHA256");

SecretKeySpec secret\_key = new SecretKeySpec(key.getBytes(), "HmacSHA256");

sha256\_HMAC.init(secret\_key);

return Base64.getEncoder().encodeToString(sha256\_HMAC.doFinal(data.getBytes()));

}

}

OrderServiceImpl.java

package com.example.BE\_PBL6\_FastOrderSystem.service.Impl;

import com.example.BE\_PBL6\_FastOrderSystem.exception.ResourceNotFoundException;

import com.example.BE\_PBL6\_FastOrderSystem.model.Order;

import com.example.BE\_PBL6\_FastOrderSystem.model.OrderDetail;

import com.example.BE\_PBL6\_FastOrderSystem.model.Product;

import com.example.BE\_PBL6\_FastOrderSystem.model.User;

import com.example.BE\_PBL6\_FastOrderSystem.repository.OrderRepository;

import com.example.BE\_PBL6\_FastOrderSystem.repository.ProductRepository;

import com.example.BE\_PBL6\_FastOrderSystem.repository.StoreRepository;

import com.example.BE\_PBL6\_FastOrderSystem.repository.UserRepository;

import com.example.BE\_PBL6\_FastOrderSystem.request.MoMoPaymentRequest;

import com.example.BE\_PBL6\_FastOrderSystem.request.OrderRequest;

import com.example.BE\_PBL6\_FastOrderSystem.response.MoMoPaymentResponse;

import com.example.BE\_PBL6\_FastOrderSystem.response.OrderResponse;

import com.example.BE\_PBL6\_FastOrderSystem.service.IOrderService;

import com.example.BE\_PBL6\_FastOrderSystem.util.HmacSHA256Util;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

import org.springframework.web.client.RestTemplate;

import java.time.LocalDateTime;

import java.util.List;

import java.util.UUID;

import java.util.stream.Collectors;

@Service

@RequiredArgsConstructor

public class OrderServiceImpl implements IOrderService {

private final OrderRepository orderRepository;

private final UserRepository userRepository;

private final ProductRepository productRepository;

private final StoreRepository storeRepository;

private static final String PARTNER\_CODE = "your\_partner\_code";

private static final String ACCESS\_KEY = "your\_access\_key";

private static final String SECRET\_KEY = "your\_secret\_key";

@Override

public OrderResponse createOrder(OrderRequest orderRequest) {

if ("MOMO".equalsIgnoreCase(orderRequest.getPaymentMethod())) {

boolean paymentSuccess = processMoMoPayment(orderRequest);

if (!paymentSuccess) {

throw new RuntimeException("MoMo payment failed");

}

}

User user = userRepository.findById(orderRequest.getUserId())

.orElseThrow(() -> new ResourceNotFoundException("User not found"));

Order order = new Order();

order.setUser(user);

order.setStore(storeRepository.findById(orderRequest.getStoreId())

.orElseThrow(() -> new ResourceNotFoundException("Store not found")));

order.setOrderDate(LocalDateTime.now());

order.setTotalAmount(orderRequest.getTotalAmount());

order.setStatus(orderRequest.getStatus());

order.setPaymentMethod(orderRequest.getPaymentMethod());

order.setDeliveryAddress(orderRequest.getDeliveryAddress());

order.setCreatedAt(LocalDateTime.now());

order.setUpdatedAt(LocalDateTime.now());

List<OrderDetail> orderDetails = orderRequest.getOrderDetails().stream().map(detailRequest -> {

Product product = productRepository.findById(detailRequest.getProductId())

.orElseThrow(() -> new ResourceNotFoundException("Product not found"));

OrderDetail orderDetail = new OrderDetail();

orderDetail.setOrder(order);

orderDetail.setProduct(product);

orderDetail.setQuantity(detailRequest.getQuantity());

orderDetail.setUnitPrice(detailRequest.getUnitPrice());

orderDetail.setTotalPrice(detailRequest.getTotalPrice());

return orderDetail;

}).collect(Collectors.toList());

order.setOrderDetails(orderDetails);

orderRepository.save(order);

OrderResponse orderResponse = new OrderResponse(order);

orderResponse.setPaymentSuccess(true);

return orderResponse;

}

private boolean processMoMoPayment(OrderRequest orderRequest) {

RestTemplate restTemplate = new RestTemplate();

String momoApiUrl = "https://test-payment.momo.vn/gw\_payment/transactionProcessor";

String orderId = UUID.randomUUID().toString();

String requestId = UUID.randomUUID().toString();

String rawData = "partnerCode=" + PARTNER\_CODE + "&accessKey=" + ACCESS\_KEY + "&amount=" + orderRequest.getTotalAmount() + "&orderId=" + orderId + "&requestId=" + requestId;

try {

String signature = HmacSHA256Util.generateSignature(rawData, SECRET\_KEY);

MoMoPaymentRequest momoRequest = new MoMoPaymentRequest(PARTNER\_CODE, ACCESS\_KEY, orderRequest.getTotalAmount(), orderRequest.getUserId(), orderId, requestId, signature);

MoMoPaymentResponse momoResponse = restTemplate.postForObject(momoApiUrl, momoRequest, MoMoPaymentResponse.class);

return momoResponse != null && momoResponse.isSuccess();

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

}



