**package** com.cpt.payments.service.impl;

**import** **static** org.junit.jupiter.api.Assertions.*assertEquals*;

**import** **static** org.junit.jupiter.api.Assertions.*assertThrows*;

**import** **static** org.mockito.ArgumentMatchers.*any*;

**import** **static** org.mockito.ArgumentMatchers.*eq*;

**import** **static** org.mockito.Mockito.*when*;

**import** org.junit.jupiter.api.Test;

**import** org.junit.jupiter.api.extension.ExtendWith;

**import** org.mockito.InjectMocks;

**import** org.mockito.Mock;

**import** org.mockito.junit.jupiter.MockitoExtension;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** com.cpt.payments.cache.ValidationRulesCache;

**import** com.cpt.payments.dto.PaymentDTO;

**import** com.cpt.payments.dto.PaymentRequestDTO;

**import** com.cpt.payments.dto.PaymentResponseDTO;

**import** com.cpt.payments.dto.UserDTO;

**import** com.cpt.payments.exception.ValidationException;

**import** com.cpt.payments.http.HttpServiceEngine;

**import** com.cpt.payments.processing.ProcessingPaymentError;

**import** com.google.gson.Gson;

**import** lombok.extern.slf4j.Slf4j;

@Slf4j

@ExtendWith(MockitoExtension.**class**)

**public** **class** PaymentServiceImplTest {

@Mock

**private** ApplicationContext applicationContext;

@Mock

**private** ValidationRulesCache validationRulesCache;

@Mock

**private** HttpServiceEngine httpServiceEngine;

@Mock

**private** Gson gson;

@InjectMocks

**private** PaymentServiceImpl paymentServiceImpl;

**private** PaymentRequestDTO preparePaymentRequest() {

// Create and populate UserDTO

UserDTO user = **new** UserDTO();

user.setEndUserID("123456789");

user.setFirstname("John");

user.setLastname("Doe");

user.setEmail("john.doe@example.com");

user.setMobilePhone("1234567890");

// Create and populate PaymentDTO

PaymentDTO payment = **new** PaymentDTO();

payment.setCurrency("USD");

payment.setAmount(1.01);

payment.setBrandName("Client Brand Name");

payment.setLocale("en\_US");

payment.setReturnUrl("https://example.com/success");

payment.setCancelUrl("https://example.com/fail");

payment.setCountry("US");

payment.setMerchantTxnRef("5014");

payment.setPaymentMethod("APM");

payment.setProviderId("TRUSTLY");

payment.setPaymentType("SALE");

// Create and populate PaymentRequestDTO

PaymentRequestDTO paymentRequest = **new** PaymentRequestDTO();

paymentRequest.setUser(user);

paymentRequest.setPayment(payment);

**return** paymentRequest;

}

//@Test

**public** **void** testValidateAndProcessPaymentSuccess() {

***log***.info("Test method");

//Arrange

PaymentRequestDTO request = preparePaymentRequest();

ResponseEntity<String> resEntity = ResponseEntity.*status*(

HttpStatus.***CREATED***).body("");

*when*(httpServiceEngine.makeHttpRequest(*any*())).thenReturn(resEntity);

//Act

PaymentResponseDTO response = paymentServiceImpl.validateAndProcessPayment(request);

***log***.info("Response: {}", response);

//Assert

*assertEquals*("TX123", response.getTxnId());

*assertEquals*("http://www.google.com", response.getRedirectUrl());

***log***.info("Test method End");

}

@Test

**public** **void** testValidateAndProcessPaymentAPICallFailedCase() {

***log***.info("Test method");

//Arrange

PaymentRequestDTO request = preparePaymentRequest();

ResponseEntity<String> resEntity = ResponseEntity.*status*(

HttpStatus.***BAD\_REQUEST***).body("");

*when*(httpServiceEngine.makeHttpRequest(*any*())).thenReturn(resEntity);

ProcessingPaymentError processingPaymentErrorRef = **new** ProcessingPaymentError(

"123", "Error message");

*when*(gson.fromJson(*any*(String.**class**), *eq*(ProcessingPaymentError.**class**))

).thenReturn(processingPaymentErrorRef);

//Act

ValidationException validationExpResponse = *assertThrows*(ValidationException.**class**,

() -> paymentServiceImpl.validateAndProcessPayment(request));

***log***.info("Response: {}", validationExpResponse);

//Assert

*assertEquals*(processingPaymentErrorRef.getErrorCode(),

validationExpResponse.getErrorCode());

*assertEquals*(processingPaymentErrorRef.getErrorMessage(),

validationExpResponse.getErrorMessage());

*assertEquals*(HttpStatus.***BAD\_REQUEST***, validationExpResponse.getHttpStatus());

***log***.info("Test method End");

}

}