BANK INTEGRATE

Contents

[**1.** **Introduction** 2](#_Toc177551199)

[**2.** **Screen flow and work flow** 2](#_Toc177551200)

[**2.1.** **Screen flow** 2](#_Toc177551201)

[**2.2.** **Work flow** 2](#_Toc177551202)

[**3.** **How to integrate (Java)** 4](#_Toc177551203)

[**3.1.** **Register with bank** 4](#_Toc177551204)

[**3.2.** **Implement to Java project** 4](#_Toc177551205)

[**3.2.1.** **Config application file** 4](#_Toc177551206)

[**3.2.2.** **Create a class for call API from bank (okhttp3 library)** 5](#_Toc177551207)

[**** **SCB:** 5](#_Toc177551208)

[**** **KBANK** 10](#_Toc177551209)

[**4.** **Document** 13](#_Toc177551210)

1. **Introduction**

This demonstration showcases how to seamlessly integrate a banking payment gateway into a web application. It simulates a real-world payment flow, from collecting customer card details to processing payments via the bank’s API.

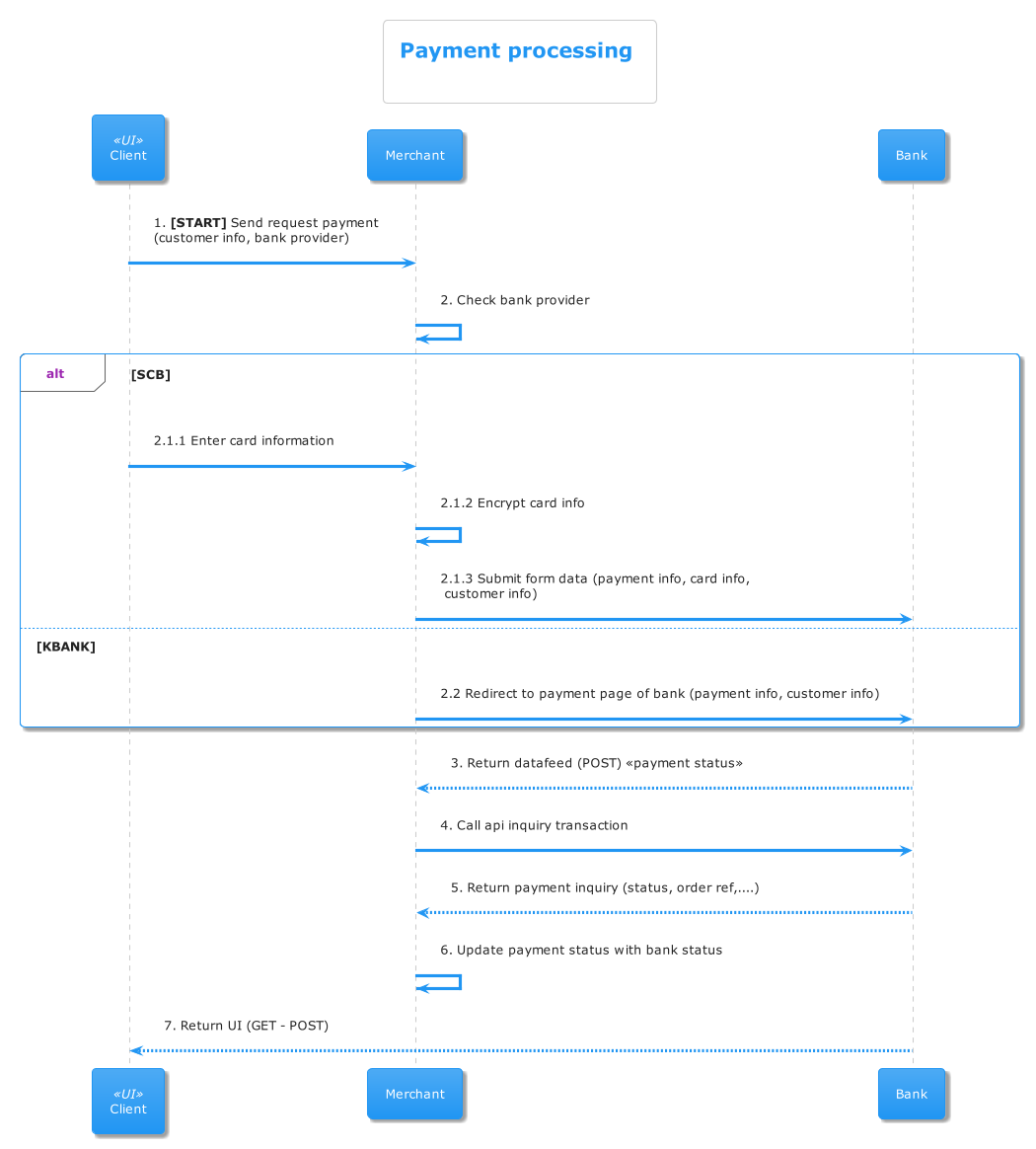
In this demo, we will walk through the entire process, from submitting payment details to handling successful and failed transactions. You’ll see how our application interacts with the bank's API to:

* **Capture payment information** (e.g., cardholder details, card number, expiration date, and security code).
* **Submit payment data** to the bank’s payment gateway for processing.
* **Handle different transaction outcomes**, including success, failure, and cancellation.
* **Redirect users** to appropriate pages after completing the transaction.

1. **Screen flow and work flow**
   1. **Screen flow**

Figma: <https://www.figma.com/design/7nsbLTsKUOO8EDPL1s7ELg/Untitled?node-id=0-1&t=dbrYlkb01RBvsVJb-1>

* 1. **Work flow**



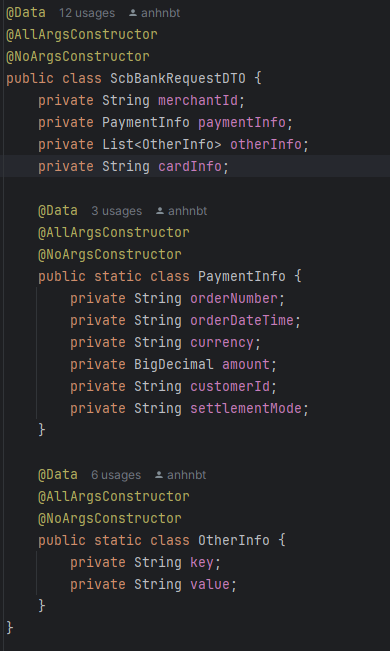
1. **How to integrate (Java)**
   1. **Register with bank**

Register with bank to get:

* **Merchant Id**: a unique identifier assigned to a business (merchant) when they register for a payment gateway or API integration with a bank or payment processor. It is used to distinguish the transactions belonging to a specific merchant and to ensure that payments are routed correctly to the merchant's account.
* **API Key**: usually has 2 components (apiKey, apiSecret or other name). It typically sent in the header or body of each API request to authenticate your business with the bank’s API. It ensures that the requests are coming from a valid and authorized merchant.
* **Document**:used to show the request and response details of each API
  1. **Implement to Java project**
     1. **Config application file**



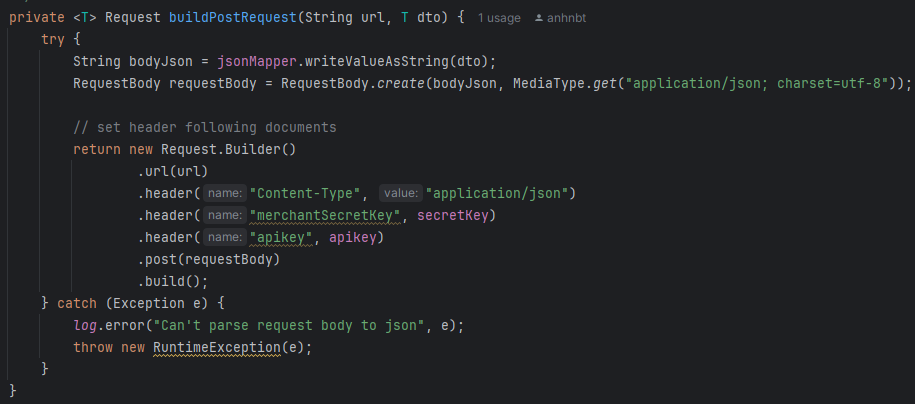
* MerchantId: identifier the business
* Payment url: use for call api payment with bank
* Fe return url (POST - GET): use for callback UI of merchant (web) with payment status
* Be return url (POST): use for callback datafeed (BE), usually config in bank
* API key: use for authorize API
* Inquiry url: used for **querying the status** of a payment or transaction after it has been initiated.
  + 1. **Create a class for call API from bank (okhttp3 library)**
* **SCB:**
  + **API confirm payment:**
    - Request:

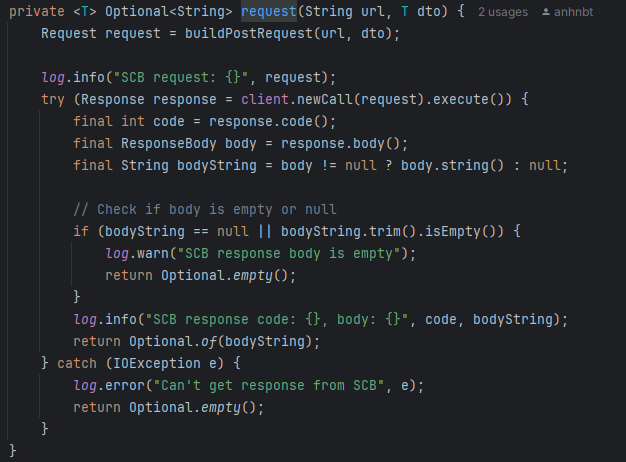


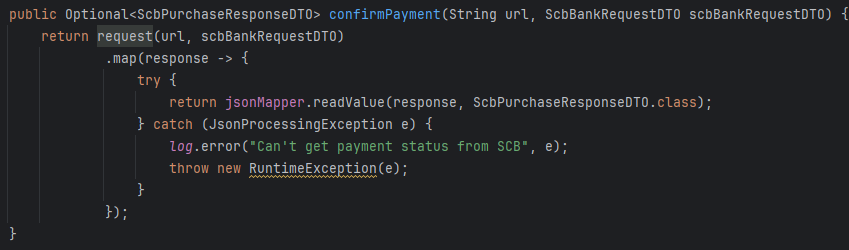
* + - Response:



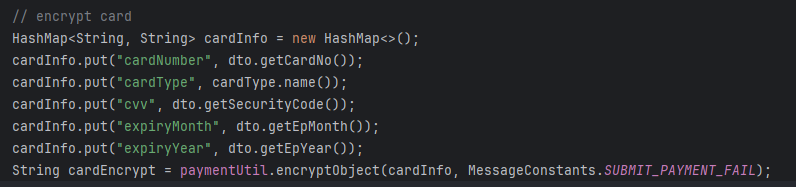
* + - Perform request: Add apiKey and secretKey into header

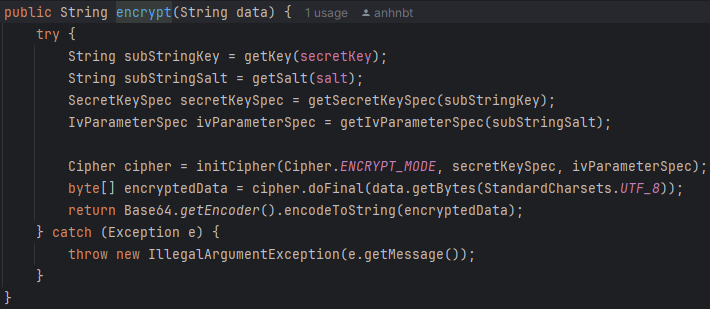






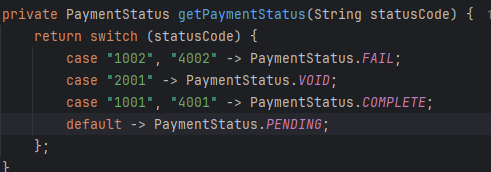
* + - Prepare data: customer info, payment info, card info (use AES encrypt)
    - Encrypted card info: use hash secret key bank provider to hash



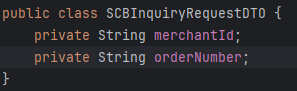


Detail: go into source code -> src/main/java/com/example/integratebank/util/AESHelper.java

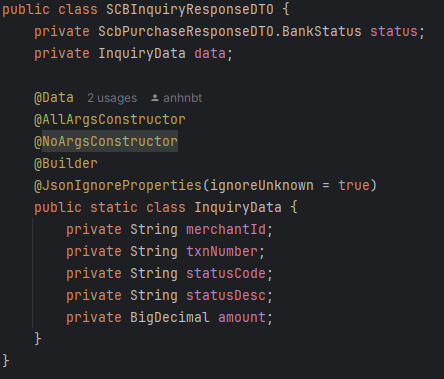
* + - Process response
* No response from bank: transaction **FAIL**
* Status code equal “1001”: call API inquiry to get payment status -> update status transaction
* Remaining: transaction **PENDING**
* Mapper bank status with transaction status (following document)



* + **API Inquiry:**
    - Request:

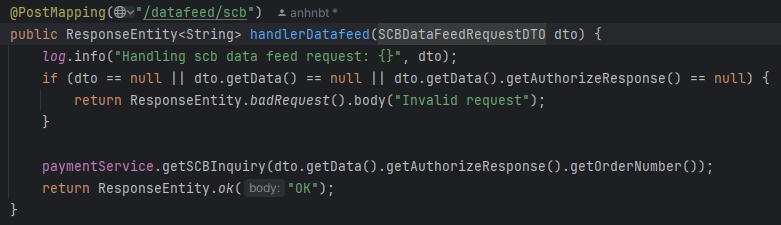


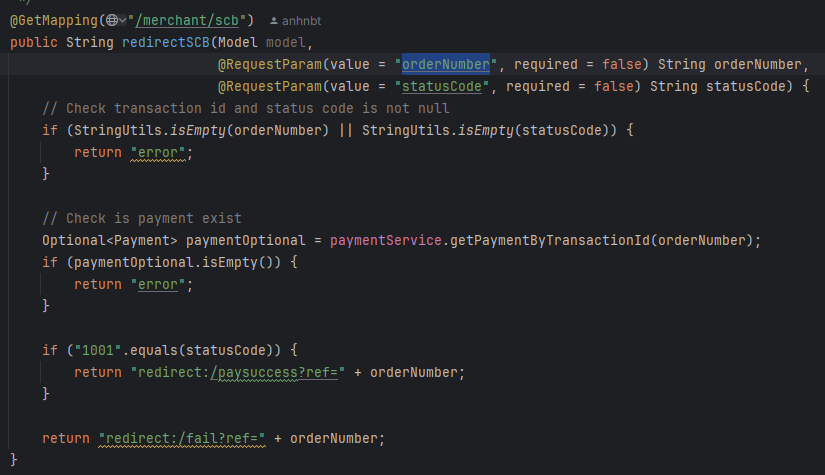
* + - Response:



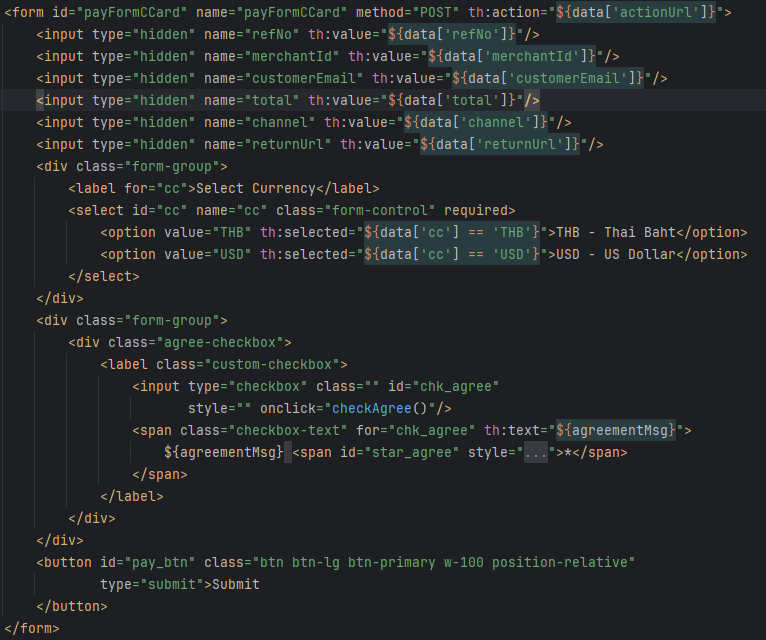
* + - Perform request: like confirm payment API
  + **Create API handle datafeed from bank to handle payment status**
    - Flow: Bank confirm payment -> callback datafeed to merchant (web) -> get bank status from Inquiry API -> update transaction status
    - Request:



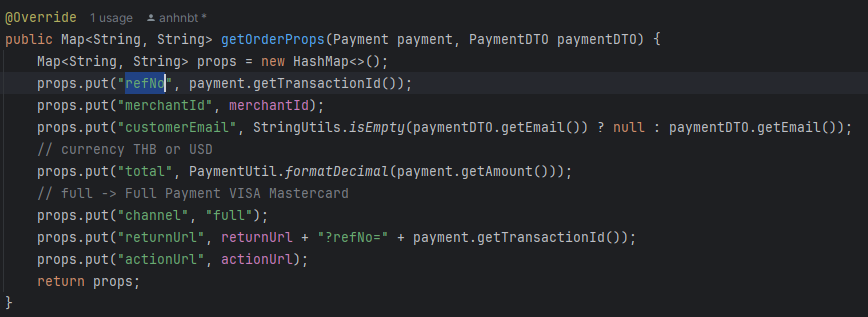
* + - API:  
      
  + **Create API to handle UI (like fe return url config)**
    - Request param: orderNumber (String), statusCode (String)



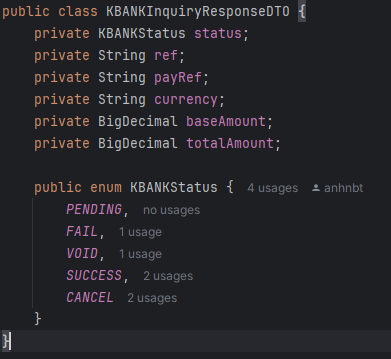
* **KBANK**
  + Flow: Submit form data (payment info) -> payment in bank -> return bank status with datafeed, UI
  + Form submit payment:



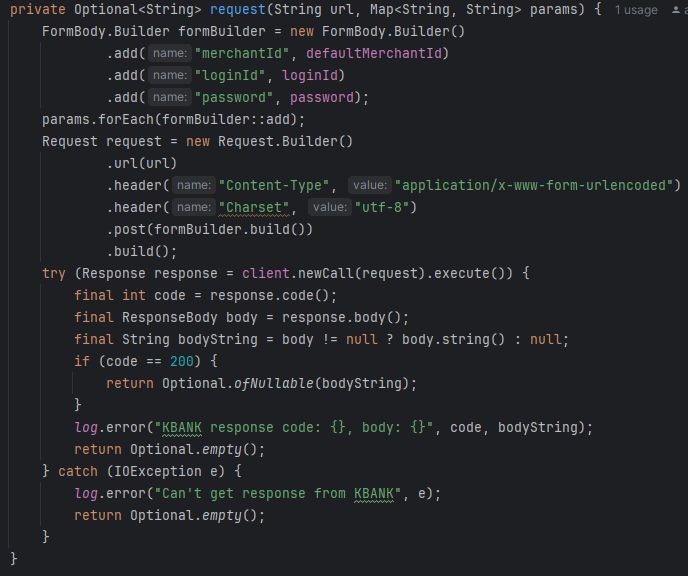
* + - Prepare data to payment:
* Following the document to define param:
  + refNo: id of transaction
  + merchantId: id of identifier the business
  + customerEmail (optional): email of customer
  + total: amount user input
  + channel:
  + returnUrl: FE return url
  + actionUrl: url to submit form payment

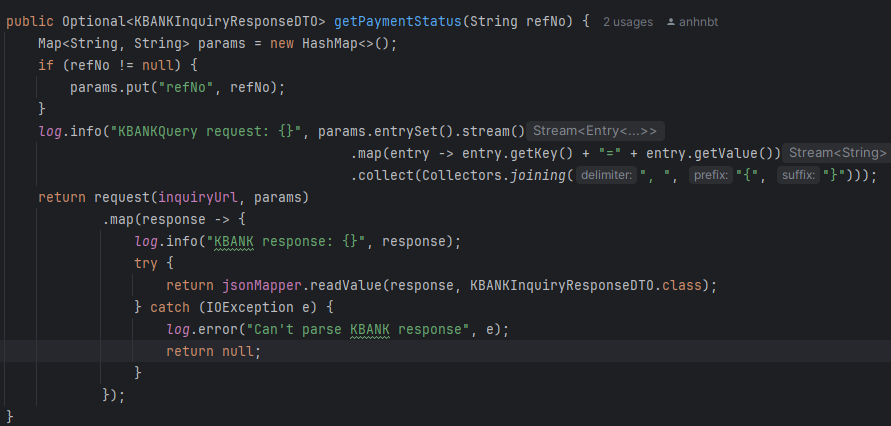


* + **API Inquiry:**
    - Request param: refNo (String)
    - Resposne:

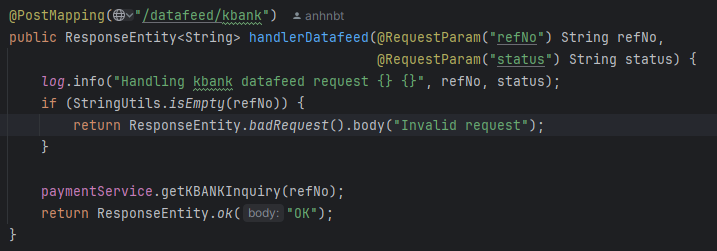


* + - Perform request:

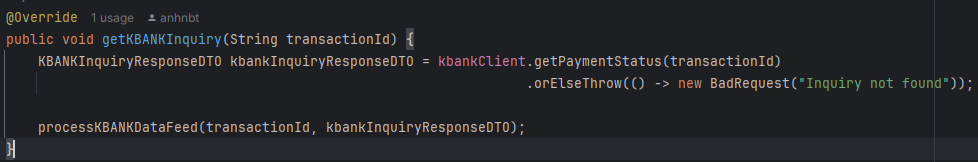




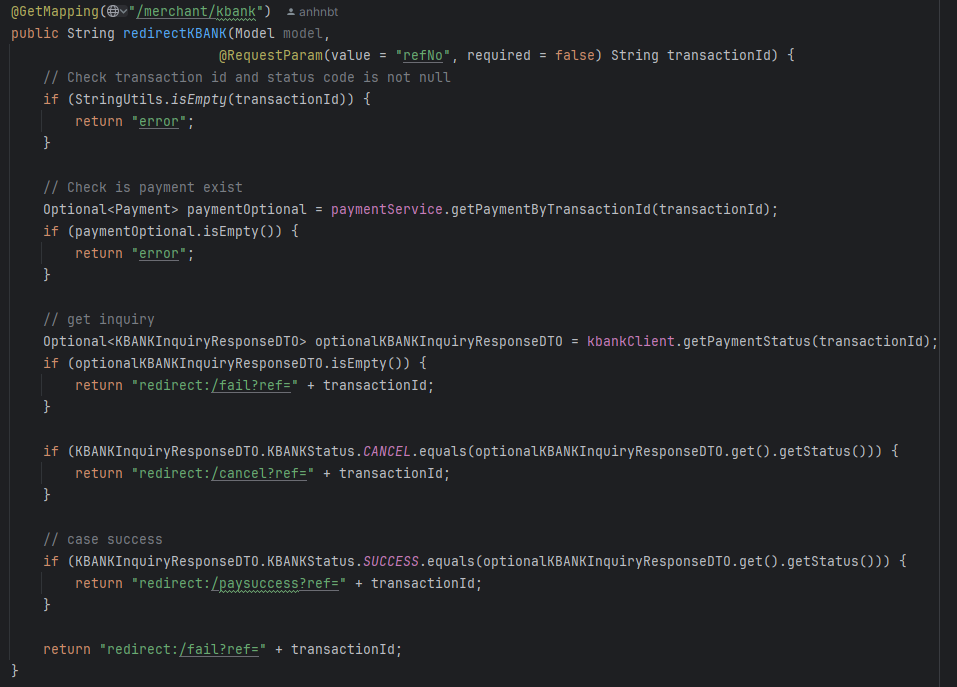
* + **Create API handle datafeed from bank**
    - Config in bank: url datafeed
    - Request param: refNo (String), status (String)



* + - Flow: Handle datafeed -> call API Inquiry from bank -> process datafeed (update status, amount, ..)



* + **Create API handle UI (use returnUrl)**
    - Request param: refNo (String)
    - Flow: Bank -> call redirect URL with param -> call API Inquiry get payment status  
      -> return UI by match status



* **Note:** Request and response must flow the document that bank provided. The above code is just an example.

1. **Document**

* Link source: <https://github.com/Kabuto-Hido/bank-integration>
* Link Figma: <https://www.figma.com/design/7nsbLTsKUOO8EDPL1s7ELg/Untitled?node-id=0-1&t=dbrYlkb01RBvsVJb-1>
* API docs example:

