DOCUMENT TYPE:

TECHNICAL DOCUMENT

DOCUMENT TITLE:

COHERENTPLUS – E-commerce – Online Payment Integration

TARGET AUDIENCE:

Project management team, software, and hardware development team. Internal use only

(PRIVATE AND CONFIDENTIAL)

2025

Table of Contents

DOCUMENT VERSION

Version	Date	Name of Author	Description
1.0.0	21/10/2025	Liew Yee Shian	Document Creation

1 Abstract

This document outlines the steps and overview on how the online payment API is integrated into the flutter application system.

SCOPE:

1) Overview of the flutter ecommerce
2) Overview of the online payment system
3) How is the online payment system integrated
4) Command responses
5) Sample request and response message

1.1 Abbreviation

Code	Description
&Pay	AmbersandPay
ecom	Flutter E-commerce Application
MId	Merchant ID
TxId	Transaction ID

2 Overview

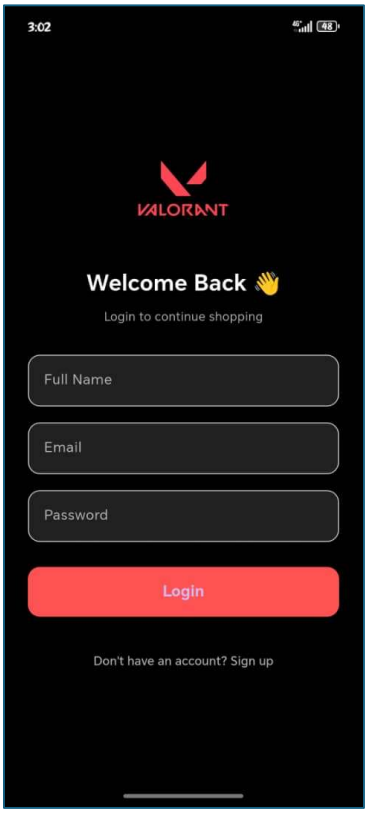
2.1 Overview of flutter ecommerce

2.1.1 Flutter Application

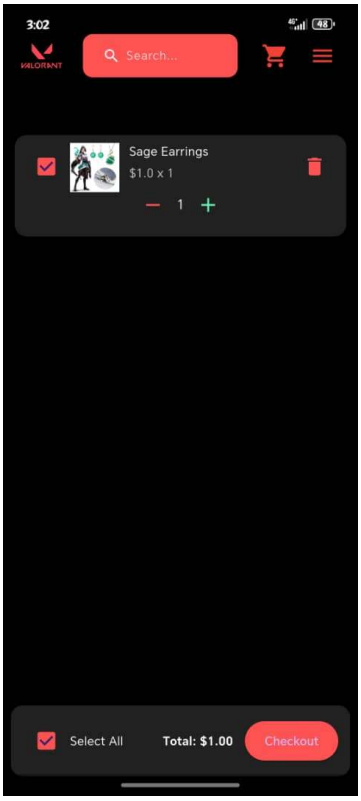
This ecom is a test file for showing user how the &pay online payment will look like after implementing it in your application.

2.1.2 How to use the ecom

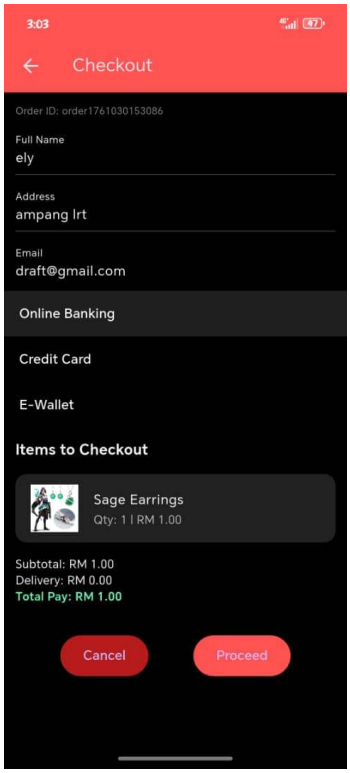
1) Sign up and then login according to what you sign up with your credentials



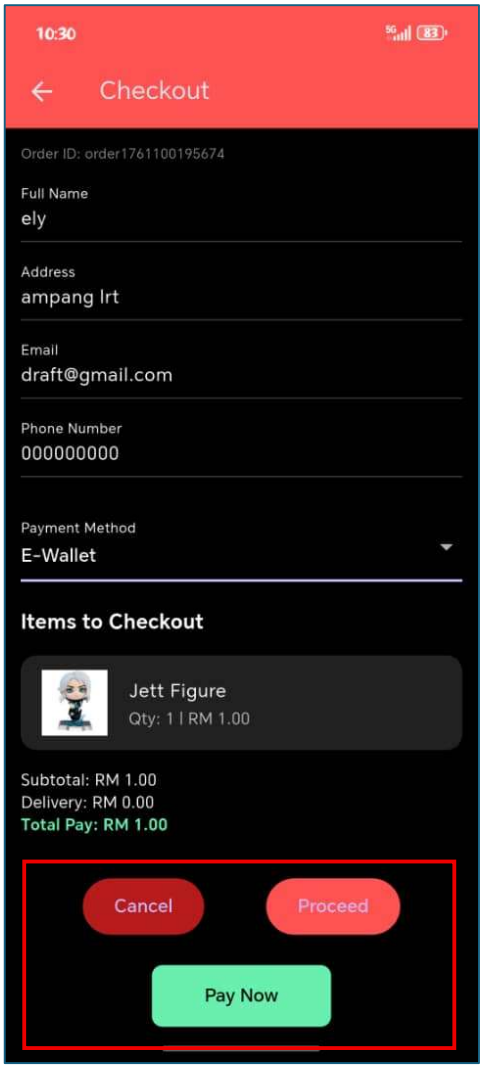
2) Add item to cart and checkout



3) Enter your information and choose any payment method of your preferred choice



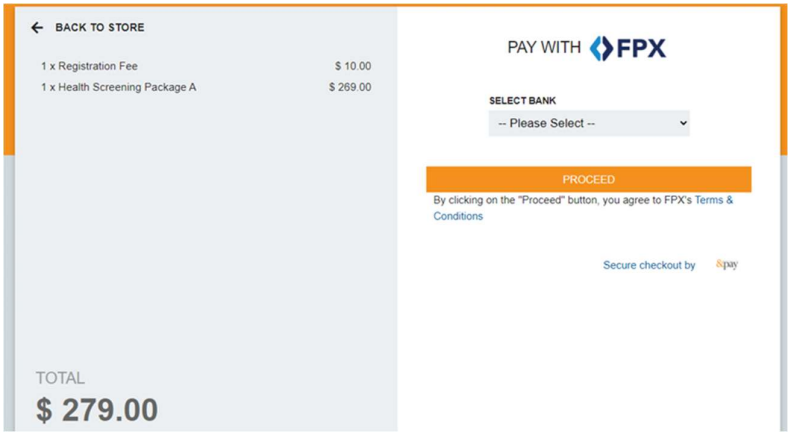
4) Click proceed and pay, it will redirect you to an in-app payment page of &pay



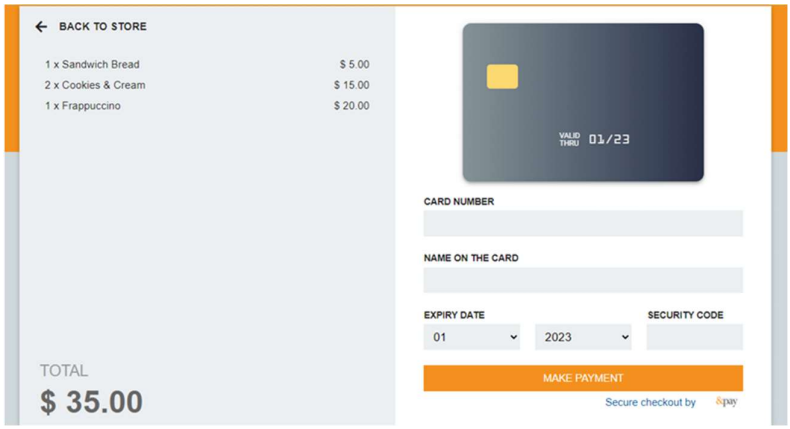
2.2 Overview of &pay Online Payment

2.2.1 Transaction channels

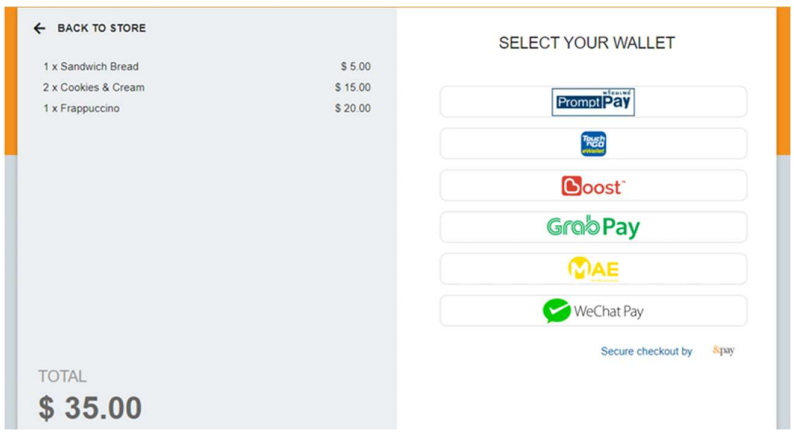
1) Online Payment



2) Credit Card



3) E-Wallet



3 Implementing Online Payment API

3.1 Generating Signature

3.1.1 Postman View

To generate signature, we use the sha512 SHA512 - Online Tools.

Sha512 [{json}+integration key] = signature key

Example:

1) Json body:


```
{"custContact":"01139998925","custEmail":"custEmail","custName":"Ryan","merchantId":"91012387",
    "txType":"SALE","txChannel":"CC","orderId":"INV6374802","orderRef":"INV-10001","productList":
    [{"amount":"5.00","name":"Booking","qty":1}],"tid":"20003761","txAmount":"5.00","txCurrency":"MYR"}
```


Figure 3.1.1 json body

2) Input sha512 {json}+key integration

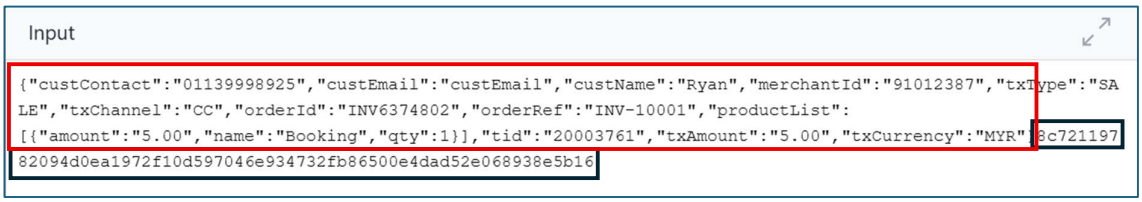


Figure 3.1.1 generate key signature

3) And after you succeed the output should be this:


```
"ret": 0,
"msg": "Success",
"merchantId": "91012387",
"orderId": "INV6374802",
"txId": "176103137124211498",
"checkoutUrl": "https://stg-pay2.ampersandpay.com/checkout?
    payload=ck5Kam9XRWlkS0VhSk5aUVJhLy9PZz09OkYrUjI3a0dwK0xHcXVWM0hla00vRVpRUkpxTXZkYWJ4NUJtVUdxUFM2emVycnN5VUxTeEFNQ0JnK2VTUTZUanpQVUxvd3E2W
    TVaU1ZwZW15RzRBNk4yZC9KWVB4OGpaS3RSMFhSV3lhTTNDY2Zobi9RM1QwWU9Ec1d0cVF2RjNtUWY4SXViY3pISHdWTkxWKzRWc2JtYVpKZXJvZHdEa0h4NG5jLzFSZ3Z0U2pnWX
    hLRG94RUoyNzZ4Ym56NC8rSHpKQVMralYwVzdrRzV3S3dqVldESkYvYko4MFRBUVlTd2Q2UG1XTzV6bWZuNGFvWHVwVHFzY1Y3WGc1emNlQ21IYXBQeDJRbTVqUDNaMFovTEdrR3B
    LZUNCYVR2Zy9WSDVBVGp1azc3bjNMbW5jU3V0OW9RWGxwNmxZKzIvYXpNbVNCSDk3eTlkbi84NndwaDh5K0Z2SEVYK2xUY0g4eEpEV0phYWp0aG8yYVdQMUtlQVBWZEh4VG5BY1dE
    c3hqTGhLK1lycnErWGhEcFhKWE9BMmZrdTNHNzBpOTBaaXozakprQUZ1Sjltb4VjNXYjBhV0hXamlXQXk0RFVldTJXUmVpTGsvOUNYL2wzN1FWbUs3L1dKL0taQXo3TjBXUVUvS
    nIrUmxuU3BWbkFYWWZycWdEK2pEN1dqMTFJYUw0NzZJTFloYUE1M3hCMnp3bUU2MkdkUUlZdW5tZUN6MCtsT05JWk0yQzI3T0xXRU9JV25SMGdpS2JiMFJ2NGZjS1RCSXQ0UjA1MV
    JMdUxQZGs1bmlrUHNlOVczNnhLNTlTRndPZEY2Y2ZyVFRjemtSK3hJNXNmdi9YcHpTd29nd3JJT1QvQnZITW1JY2ZudWxSYjBNSEF4QmRINVBLaUJ3N3dEREExnYTZFaTV4aHF6d2N
    sUG1NRnN6SU1CcG5CSk5mOFZvcGhNbERNc0V2NWswRkVNeThwZkI5YUNmSENQQ1hKczU0ZFJwVDM5TE91cnZtL2xUVEt1dTZhaWIzNFR2V0NUeW1vZDg3aFc0RTFGSHVOQ1lZcGU4
    TzV6NE83azdtdnJjWXRZMDd5K3FYUlRoS3RpT3lPMnhxdEVzZVJnTXNIY1dFR3NKSzdZUWlDZWZRdFFVYXArMjNxZDBGS3BhaEVTMHM0W1lGdnZxS2xDenEvODkxVklvZ2hWbGczM
    XNiOEdaeHhON3YydmwvY3lEUHhzV2xCQkdLSXdGWHVCcENZL1B0METtZm42eDQ3cnloaWVLVFd5Q2J6M2s2N0xBVnlmQ3UrYmU4T1hWRDVQVDBrcnM1b0lqQ1VNeEpmRmtDbXRYbm
    o2MHZvUlNFK21yWFQrbUFFT05aQWtkbC9nTU5GQXU1L25mOHNwd0xYUFlRYWxXZzBlZ2hiYTkzSy92ZGk2NWQrbjlwbFZYUVR3dDVXQjVlb1IxVjBJK01FaVZXREhqQjVSdmlWQk4
    4UEk3bDU1S0FGVktVMU1HMnk4alowZUtIQ0x2cXJiQlpIaitpYmpBVUdxcnZnNXdycXkrN0xJT3Y0allxaEo3MnZwdFBOQnUyNHlLeGFYOUVJMXJFNWFWT05majVjTFRtYVMvL01n
    NjJ0LzlnKzVQcWh0K3p3bUNvU2dmc2diOC9tb3B0anA3d05vbzdhRmtQWWw1WjgxeFlTV2ZpWkt5WUpzcUl5b2tqcytTQ1huMTd1c21OY2xTV29NdWxxWUIxQ09oM1NuSU9hRSs4d
    WpWQ29DdGJqZjF5ZDRQR1N2T0xJb1hPMmxNWE5FK3YrSTVPclIreGtOdjhTUUJVWWtsQm5aSTk1bVFhYkRjVlJRSU1uQ1RkZHVuW1hKeW96clovL0VNK3BmemdWUk8zYmYvdUJUS1
    gvcWplaElTUDlKTmJJOG5UNFNIczY0cndINzBHUklXS3kyT1hXeUh6MDhvPQ&timestamp=2025-10-21T07:22:51",
"checkoutDt": "2025-10-21T07:22:51"
```


Figure 3.1.1 output of json and key signature

A payment link and TxId will be generated based on the information entered by the user.

This process is handled in the backend, which processes the input data and returns the corresponding payment details.

3.1.2 Transaction Request for Generating Key Signature

1) &pay allows merchant to request transaction with checkout URL. Merchant has to redirect end user to &pay hosted checkout page to complete the checkout process

Key Name	Description
Request URL	Staging: https://stg-ipg.ampersandpay.com/tx/request Production: https://ipg.ampersandpay.com/tx/request
Request Method	POST
Content-Type	Application/Json
Character Set	UTF8

Table 3.1.2 &pay transaction API request properties.

2) Request Parameter (json body specs)

	Key Name	Type	Length	Description
Header	signature	Alphanumeric (M)	128	API authorization signature. Please refer to section 3.1.
Body	merchantId	Numeric (M)	8	&Pay merchant ID. This will be provided by &Pay once onboarded
	txType	alpha (M)	10	Transaction type, eg. “SALE”. Please refer to Appendix A.
	txAmount	alphanumeric (M)	16	Transaction amount eg. “15.00”, 2 decimal points only and comma (,) is not allowed.
	txCurrency	alpha (M)	3	Transaction currency code in ISO4217 format. Please refer to Appendix B.
	txChannel	alpha (M)	2	Transaction channel. Please refer to Appendix C.
	orderId	alphanumeric (M)	32	Merchant unique transaction ID.

	orderRef	alphanumeric (O)	100	Merchant transaction reference.
	productList	array (O)		Transaction product list. Please refer to section 3)
	custName	alphanumeric (O)	50	Customer name.
	custEmail	alphanumeric (O)	128	Customer email (Payment receipt will be sent to).
	custContact	alphanumeric (O)	20	Customer contact number.

Table 3.1.2 &pay transaction request parameter.

3) productList

Key Name	Type	Length	Description
name	alphanumeric (M)	50	Product name
qty	numeric (M)	6	Product quantity
amount	alphanumeric (M)	16	Product amount eg. “15.00”, 2 decimal points only and comma (,) is not allowed.

Table 3.1.2 product list request structure

3.2 Implementing Sha512 in Flutter

3.2.1 Calling API

First we have to write how is flutter calling the api. The Mid and integration key is given by our &pay.

```
class CallApi {
  static const String merchantId = "91012387";
  static const String integrationKey =
      "8c72119782094d0ea1972f10d597046e934732fb86500e4dad52e068938e5b16";
  static const String apiUrl = "https://stg-ipg.ampersandpay.com/tx/request";
```

Figure 3.2.1 call API

3.2.2 Payment Method

This is where we have our mapping for payment method. Based on which the user input on the checkout page, it will trigger the code to link with the correct Transaction Channel (appendix c).

```
/// Map Flutter payment method to Ampersand code
static String _getChannelCode(String method) {
  switch (method) {
    case "Online Banking":
      return "DD";
    case "Credit Card":
      return "CC";
    case "E-Wallet":
      return "EW";
    default:
      return "CC";
  }
}
```

Figure 3.2.2 payment method mapping

3.2.3 Generate Sha512

This is where we use sha512 converter to change Json body and integration key into a string of key signature. The key signature will then place to the header and Json in the body.

```
/// Create signature required by API
static String _generateSignature(String jsonString) {
  final bytes = utf8.encode(jsonString + integrationKey);
  final digest = sha512.convert(bytes);
  return digest.toString();
}
```

Figure 3.2.3 Sha512 key signature

```
// Send API request
final response = await http.post(
  Uri.parse(apiUrl),
  headers: {
    "Content-Type": "application/json",
    "signature": signature,
  },
  body: jsonString,
);
```

Figure 3.2.3 key signature and Json

3.2.4 Json Body

This is the json body, it will get the user's information based on this mapping, and this mapping will then be saved and keep it as the 'body'. This is based on the &pay spec for posting transaction request. (refer to 3.1.2)

```
// Payload based on Ampersand spec
final Map<String, dynamic> payload = {
  "merchantId": merchantId,
  "txType": "SALE",
  "txChannel": txChannel,
  "orderId": orderId,
  "orderRef": orderId,
  "txCurrency": "MYR",
  "txAmount": totalAmount.toStringAsFixed(2),
  "custName": userInfo["name"],
  "custEmail": userInfo["email"],
  "custContact": userInfo["number"],
  "productList": items
      .map((item) => {
            "name": item["name"],
            "qty": item["quantity"],
            "amount": item["price"].toStringAsFixed(2),
          })
      .toList(),
};
```

Figure 3.2.4 Json body

3.2.5 Checkout URL

This is where it will generate the checkout URL code, when there is no problem, it will return the URL link in app.

```
if (response.statusCode == 200) {
  final Map<String, dynamic> data = jsonDecode(response.body);

  if (data["ret"] == 0 && data["checkoutUrl"] != null) {
    final txId = data["txId"] ??
        "TXN-${DateTime.now().millisecondsSinceEpoch}";
```

Figure 3.2.5 URL checkout

4 Implementing Query API

4.1 Transaction Query

Merchant can re-query the payment status via this API and a direct response message will be returned. Query to check whether is paid or unpaid.

Key Name	Decription
Request URL	Staging: https://stg-ipg.ampersandpay.com/tx/query Production: https://ipg.ampersandpay.com/tx/query
Request Method	POST
Content Type	Application/Json
Character Set	UTF8

Figure 4.1 Command Request Properties

4.1.1 Query Parameter

	Key Name	Type	Length	Description
Header	Signature	alphanumeric (M)	128	API authorization signature. Please refer to section 3.1.
Body	merchantId	numeric (M)	8	&Pay merchant ID. This will be provided by &Pay once onboarded
	txId	numeric (M)	18	&Pay transaction ID.

Figure 4.1.1 Transaction Query Request Parameter

4.1.2 Polling Query

Unlike Transaction Request, Transaction Query need to poll request so that the application itself knows whether payment has been made or not. Here it gets the polling every 2 seconds after 1 minute, to avoid error records.

```
static Future<void> pollTransaction({
  required String txId,
  required String orderId,
  int intervalSeconds = 2,
  int maxAttempts = 60,
}) async {
  // ✅ Start polling after 1 minute to avoid "no record found" early
  await Future.delayed(const Duration(minutes: 1));
```

Figure 4.1.2 polling request

4.1.3 Paid Query

When Transaction is made, and the query request poll until paid, it will update the firebase so that we know that this order and transaction is paid, hence no need for more polling and user can go back to the main page of the application and transaction will be approved.

```
final status = statusData["txStatus"]?.toString().toUpperCase() ?? "";
log("🌐 Polli                , status: $status");
                Type: String
if (status == "SUCCESS" || status == "PAID") {
  t.cancel();

  // ✅ Update Firestore order status
  await FirebaseFirestore.instance
      .collection('orders')
      .doc(orderId)
      .update({
    "status": "paid",
    "updatedAt": DateTime.now(),
  });

  log("✅ Order $orderId marked as PAID");
}
```

Figure 4.1.3 Paid Query

4.1.4 Generate Query

This will be the POST for getting the query request

```
/// Query Ampersand API for a specific transaction
static Future<Map<String, dynamic>?> queryTransaction(String txId) async {
  try {
    final body = {"merchantId": merchantId, "txId": txId};
    final jsonBody = jsonEncode(body);
    final signature =
        sha512.convert(utf8.encode(jsonBody + integrationKey)).toString();

    final response = await http.post(
      Uri.parse(queryUrl),
      headers: {
        "Content-Type": "application/json",
        "charset": "UTF-8",
        "signature": signature,
      },
      body: jsonBody,
    );

    if (response.statusCode == 200) {
      final data = jsonDecode(response.body);
      log("✅ Query Response: $data");
      return data;
    } else {
      log("❌ HTTP Error: ${response.statusCode} - ${response.body}");
      return null;
    }
```

Figure 4.1.4 POST Query Request

5 API Packages

5.1 Crypto Package

This library is used for data encryption and hashing — mainly for creating secure signatures (e.g. SHA512) when sending data to an API.

1) First you have to import the crypto dependency and its version. Here the version is 3.0.6, it may update overtime. So make sure the version is the newest version. Add the dependency in your pubspec.yaml file.

```
dependencies:
  flutter:
    sdk: flutter

  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.8
  carousel_slider: ^5.1.1
  url_launcher: ^6.3.2
  firebase_core: ^4.1.1
  cloud_firestore: ^6.0.2
  http: ^1.5.0
  crypto: ^3.0.6
  flutter_inappwebview: ^6.1.5
  webview_flutter: ^4.13.0
  firebase_auth: ^6.1.1
  shared_preferences: ^2.5.3
```

Figure 5.1 Crypto Package and version

2) Then import it on top of your API code file

```
import 'dart:async';
import 'dart:convert';
import 'dart:developer';
import 'package:crypto/crypto.dart';
import 'package:http/http.dart' as http;
import 'package:cloud_firestore/cloud_firestore.dart';
```

Figure 5.1 Import Crypto Package

5.2 HTTP Package

This library is used for opening https links such as the API request and Query links.

Same as the cypto package, make sure you have the newest version and add it to your dependency at your pubspec.yaml file, then import it on top of your page.

You can get the dependencies and version here:

https://pub.dev/

or type your dependency first:

crypto: https://pub.dev/packages/crypto/versions

http: https://pub.dev/packages/http/versions

```
dependencies:
  flutter:
    sdk: flutter

  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.8
  carousel_slider: ^5.1.1
  url_launcher: ^6.3.2
  firebase_core: ^4.1.1
  cloud_firestore: ^6.0.2
  http: ^1.5.0
  crypto: ^3.0.6
  flutter_inappwebview: ^6.1.5
  webview_flutter: ^4.13.0
  firebase_auth: ^6.1.1
  shared_preferences: ^2.5.3
```

```
import 'dart:async';
import 'dart:convert';
import 'dart:developer';
import 'package:crypto/crypto.dart';
import 'package:http/http.dart' as http;
import 'package:cloud_firestore/cloud_firestore.dart';
```

Figure 5.2 http package and import

5.3 WebView Package

The &pay's payment page is through a link, however, instead of redirecting it to a website, we make the payment page to open in app.

1) First to need to import an in app view dependency, similarly, get the newest version.

```
dependencies:
  flutter:
    sdk: flutter

  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.8
  carousel_slider: ^5.1.1
  url_launcher: ^6.3.2
  firebase_core: ^4.1.1
  cloud_firestore: ^6.0.2
  http: ^1.5.0
  crypto: ^3.0.6
  flutter_inappwebview: ^6.1.5
  webview_flutter: ^4.13.0
  firebase_auth: ^6.1.1
  shared_preferences: ^2.5.3
```

Figure 5.3 in app view page

2) Import them on top of your payment web page, in this case we created a Payment_webview.dart solely for the payment page

```
import 'dart:async';
import 'package:flutter/material.dart';
import 'package:flutter_inappwebview/flutter_inappwebview.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'finish_checkout.dart';
import 'query_transaction.dart'; // AmpersandPayQuery
```

Figure 5.3 Import App View Dependency

6 Sample Request and Response

6.1 Transaction Request

6.1.1 Request Message

```
{
     "merchantId":"83193488",
     "txType":"SALE",
     "txAmount":"15.00",
     "txCurrency":"MYR",
     "txChannel":"CC",
     "orderId":"order10002",
     "orderRef":"Purchase OID:10002",
     "productList":[
     {"name":"Frappuccino","qty":"1","amount":"5.00"},
     {"name":"Espresso","qty":"1","amount":"10.00"}
     ],
     "custName":"Jonathan Lim",
     "custEmail":"jonathan.lim@ampersandpay.com",
     "custContact":"60191234567"
}
```

6.1.2 Response Message

```
{
     "ret":"0000",
     "msg":"Success",
     "merchantId":"83193488",
     "orderId":"order10002",
     "txId":"168171586207510004",
     "checkoutUrl:" https://stg-
pay.ampersandpay.com/online/checkout?payload=OWExODBkMzM4ZWEhOTYxNTY4N2ExYjAxN2VlN2MyYWQ6NT
hlN2EzNmE5NGJiZmZlOWJjZjkxZjNkMmU0Y2VhMTZjMGYzYzU2NDk0NjAzMTI0MzBmOWI0ZGU4MzBjN2IxYjNlMGQ1N
GQ3OTdjODMwNDQ5NWU4YmM4NmU1NzQ0YWMxN2ExZTQ0YmQ5ZTljZmRkMjU1ZjJiZGEyNGVjMDI5NGQ5NTEyOWEzZGEy
ZWRkMzdiZjIyNDU3Mjg0ZDdjMDU4ODA2YzUwNjcxMTA1NjI2YzQ0MDAwNjUzYzBlYzk5YzEzMGFiYjFkOTVjZjBjODV
iYmJlMzQ2MDA3NWMwODdhMTRiODNjYzJkNzA4YjU0Y2M3ZDIwMzcyZWRiZDg3MWYzZDVkZTg3OTU5OWJlOThmNTIxNj
MzMTQ1OWM0YjJjOTAwMjhkYjQ1NDE5OWQwODhmMjQyMDJmZWE1MTg1NDI2NTU0ZjZmMTMwZWZkMjc4Zjc4MTY2MTgzN
DYwZTM3NTUzMDA5ZjNhOWVkMzY1MzUzYzNhNTUyMDVkMTM2ZWFmMDczMTEyYjY3ODQ4NTVkMGEzNDA3MWVlNTM2Mzc4
NmEyZDBkMTk4OWMxYjE2NzZkMTJiMmI3NmY1ZDAzYzI1NWE2ZGYxNjI1MDY1NGQ3MTMxY2E0NmE4NmZjZGIzODhkZjA
5ZDUyOWU2ZmQ2ZWIwNTRkZGVkZjFlZDllY2Y5ZDFjMjYyYjNhZWM0N2YyZWRiNTgwOGZiNjhiMGFlNTU4ZGU2MjU0Zj
M0ZGMxODJlNmYyMTUwZGFkMzdlNjdhMDQ2MjM4ODBhODRmMDhhZjE3OTFkY2EwMzI4NzUwYTZlZDg3Y2QzNTljOGM4M
DYxYjZmZmM3ZWYzZDY0MjM2NjllYzU4MTM5YmQ3NDEwNTBhNGEwMDFjZWFhMzhjNmU0ODFiY2JlN2VkZTNhYzJlZDQy
YThkMTg1N2UyODdjMGVmOTI1YmI4YmQ3OThhNjBhZDczYjY2OGIxNDg3YzY2NzQ4MmNmZmZmZThhNTEyZTMwOTczYzN
hNWY1YjU4NzgzYzllNDY3Yjk2NWU0M2Q1NDVjNTg0YjhkOTI3ZDA5YjgxYzAzYjM0ZTA0YmY2MWQzYjc0NmUyN2FhMD
IwYWZkYTQ2ZjE0ZjcwMWI1NGQxYTkzY2RkNGFiNjMyNzcwZjNkNjEzMWQ5YTE4MjJjNjFkZDA3ODkzMzBiZTdjMDlkZ
mJlNDFhZmI4ZDI3Y2MwYzY5MjE2NGI1OTljNWU0ZmM4NDUxZmU1ZTA0ZmJlMzEwNTA0MjhkNzA0MzY5NTExN2I1OTRl
MDliZTQ1NTMxYjk2N2VmNTg4NmI2NzI5ZmJiYmY3NzM0M2I&timestamp=2023-04-17T15:17:41",
```

6.2 Transaction Query

6.2.1 Request Message


```
{
     "merchantId":"83193488",
     "txId":"168171586207510004"
}
```


6.2.2 Response Message


```
{
     "ret":"0000",
     "msg":"Transaction completed",
     "merchantId":"83193488",
     "orderId":"order10002",
     "txId":"168171342461010016",
     "txType":"SALE",
     "txStatus":"SUCCESS",
     "txDt":"2023-04-17T15:24:04",
     "txAmount":"15.00",
     "txCurrency":"MYR",
     "txChannel":"CC",
     "respCode":"00",
     "apprCode":"980131",
     "apprId":"2304175274620049",
     "schemeId":"S93"
```

Appendix A. Transaction Type

Type	Request Value	Remark
Sale	SALE	To request sale transaction.
Pre-authorization	PREAUTH	To pre-authorize transaction.
Sale completion	CAPTURE	To capture pre-authorized transaction.
Void	VOID	To void transaction before settlement.
Refund	REFUND	To refund transaction after settlement.

Appendix B. Transaction Currency

Type	Request Value	Remark
Malaysia Ringgit	MYR	Applicable for local acquirer only.
Singapore Dollar	SGD	Applicable for local acquirer only.
Thailand Baht	THB	Applicable for local acquirer only.

Appendix C. Transaction Channel

Channel	Request Value	Description
Credit Card	CC	Credit card checkout channel.
E-Wallet	EW	E-wallet checkout channel.
Direct Debit	DD	Online banking checkout channel, eg. FPX or DOBW.

Appendix D. Transaction Scheme

Channel	Request Value	Description
CC	S01	Visa & Master card.
CC	S02	Visa card only.
CC	S03	Master card only.
CC	S07	American express.
DD	S21	FPX online banking.
DD	S22	Duitnow online banking.
EW	S92	Touch n Go Digital wallet.
EW	S93	Boost wallet.
EW	S94	GrabPay wallet.
EW	S95	MAE wallet.
EW	S96	ShopeePay wallet.
EW	S97	Alipay wallet.
EW	S98	WechatPay wallet.
EW	S99	LiquidPay wallet.

Appendix E. Transaction Status

Status	Description
PENDING	Transaction initialized and pending on authorization.
PROCESSING	Transaction authorization in progress.
SUCCESS	Transaction approved/completed.
FAILED	Transaction failed.
VOIDED	Transaction voided.
REFUNDED	Transaction refunded.

Appendix F. Error Code

Error Code	Description
0	Success
8001	General failure.
8011	Invalid API format.
8012	Invalid API parameter.
8031	Invalid signature.
8041	Invalid merchant account.
8051	Invalid transaction scheme.
8061	Invalid transaction status.
8071	Invalid transaction channel.
8081	Invalid transaction type.
8101	Request value not supported.
8102	Transaction not found.
8103	Transaction request cannot be initialized.
8201	Unauthorized. (Probably due to invalid signature)
8301	Order ID duplicated.
8401	Transaction request timeout.
8999	Unknown error.
9999	Exceptional error.

Appendix G. Response Code

Error Code	Description
00	Approved or completed successfully
01	Refer to card issuer
02	Refer to card issuer's special conditions
03	Invalid merchant
04	Pick-up
05	Do not honor
06	Error
07	Pick-up card, special condition
08	Honour with identification
09	Request in progress
10	Approved for partial amount
11	Approved (VIP)
12	Invalid transaction
13	Invalid amount
14	Invalid card number (no such number)
15	No such issuer
16	Approved, update track 3
17	Customer cancellation
18	Customer dispute
19	Re-enter transaction
20	Invalid response
21	No action taken

22	Suspected malfunction
23	Unacceptable transaction fee
24	File update not supported by receiver
25	Unable to locate record on file
26	Duplicate file update record, old record replaced
27	File update field edit error
28	File update file locked out
29	File update not successful, contact acquirer
30	Format error
31	Bank not supported by switch
32	Completed partially
33	Expired card
34	Suspected fraud
35	Card acceptor contact acquirer
36	Restricted card
37	Card acceptor call acquirer security
38	Allowable PIN tries exceeded
39	No credit account
40	Requested function not supported
41	Lost card
42	No universal account
43	Stolen card, pick-up
44	No investment account
45-50	Reserved for ISO use
51	Not sufficient funds

52	No checking account
53	No savings account
54	Expired card
55	Incorrect personal identification number
56	No card record
57	Transaction not permitted to cardholder
58	Transaction not permitted to terminal
59	Suspected fraud
60	Card acceptor contact acquirer
61	Exceeds withdrawal amount limit
62	Restricted card
63	Security violation
64	Original amount incorrect
65	Exceeds withdrawal frequency limit
66	Card acceptor call acquirer's security department
67	Hard capture (requires that card be picked up at ATM)
68	Response received too late
75	Allowable number of PIN tries exceeded
90	Cutoff is in process
91	Issuer or switch is inoperative
92	Financial institution or intermediate network facility cannot be found for routing
93	Transaction cannot be completed. Violation of law
94	Duplicate transmission
95	Reconcile error

96	System malfunction
TO	Transaction timeout.
NS	Transaction not supported.
FF	Unknown failure.
EX	Exceptional error.

< END OF DOCUMENT >