Faculty of Information and Communication Technology

BsC Hons in Applied Computer Science

$3rd$ year

PMH

Submitted by:

BHOWANEEDIN Padmalaksmi Pallavi
LAGUETTE Jean Alain William

Submitted to: **Shiam Beeharry**

DATE : 28/02/2024

TABLE OF CONTENT :

Vous pouvez ajouter des titres (Format > Styles de paragraphe) qui apparaîtront dans votre table des matières.

ABSTRACT

The project aims to seamlessly integrate PayPal payment functionality into a Flutter mobile application, enhancing its financial transaction capabilities. Leveraging the PayPal SDKs and APIs, the implementation involves the creation of a secure and user-friendly payment gateway within the Flutter app. Users will be able to make transactions, purchases, and payments using their PayPal accounts, providing a reliable and widely recognized payment solution. The integration process will include setting up the necessary authentication, handling payment requests, and ensuring a smooth user experience. This project not only facilitates secure financial transactions but also contributes to expanding the app's versatility by incorporating a widely used and trusted payment system.

//// *PROJECT NAME* : Integrate Paypal in a Flutter App ////

INTRODUCTION :

In today's dynamic digital landscape, mobile applications have become indispensable tools for various functionalities, including financial transactions. As businesses and developers seek to provide users with convenient and secure payment options, integrating popular and trusted payment gateways has become paramount. This project focuses on the integration of PayPal, a widely recognized and widely used online payment platform, into a Flutter mobile application.

Flutter, known for its cross-platform development capabilities, provides an ideal framework for creating robust and responsive mobile applications. By incorporating PayPal's Software Development Kits (SDKs) and Application Programming Interfaces (APIs), this project aims to empower Flutter applications with the ability to process payments seamlessly. This integration not only enhances the user experience by offering a familiar and trusted payment method but also opens up new possibilities for financial transactions within the app.

Throughout this project, we will delve into the technical aspects of integrating PayPal, addressing authentication, payment request handling, and ensuring the overall security and reliability of the payment gateway. By the end of the implementation, users of the Flutter app will have a streamlined and secure platform to make transactions, reinforcing the application's functionality and user appeal.

METHODOLOGY

The project initiation involved a comprehensive meeting to define the objectives of integrating PayPal into the Flutter app. Following this, the design phase kicked off with research and the creation of mockups and wireflows, consolidating inspiration from existing online solutions. A dedicated research phase focused on exploring diverse approaches to implement PayPal integration via its API and Flutter. The development phase commenced with the setup of the development environment and version control, leading to the coding and implementation of the PayPal integration.

Error Resolution :

- Issue Prioritization:
  - Prioritize and categorize issues based on their impact on the project timeline and functionality.
  - Establish a systematic approach for addressing and resolving errors.
- Collaborative Problem-Solving:
  - Foster collaboration among team members to collectively address challenges and share knowledge.
  - Leverage online forums and resources to seek solutions for issues related to the PayPal API and Android Studio.

Testing, encompassing both unit and integration testing, ensured a seamless user experience. The iterative nature of error resolution addressed challenges arising from the PayPal API and Android Studio. Documentation played a crucial role, including thorough code documentation and an issue log, serving as a valuable resource for future development.

A short description of the project and what you wanted to achieve as main functionalities.

- **User Interface for Payment:**
  - You have created a Flutter app with a MakePayment screen, where users can initiate a payment by pressing the "Pay with PayPal" button.

- **PayPal Payment Process:**

- The PaypalPayment widget is responsible for managing the PayPal payment process.
- It retrieves an access token from PayPal, creates a payment request, and initiates the payment flow by redirecting the user to the PayPal payment page.

❖ **Asynchronous Operations:**

- Your code uses asynchronous programming to handle network requests and responses. It employs the Future and await keywords to manage asynchronous tasks.

❖ **Code Modularity:**

- The code is modularized into separate files (main.dart for the main application, paypal_payment.dart for PayPal-related UI, and paymentservice.dart for PayPal API interactions).

❖ **Error Handling:**

- There is error handling in place, with the use of try, catch, and rethrow to manage exceptions and provide feedback to the user through SnackBars.

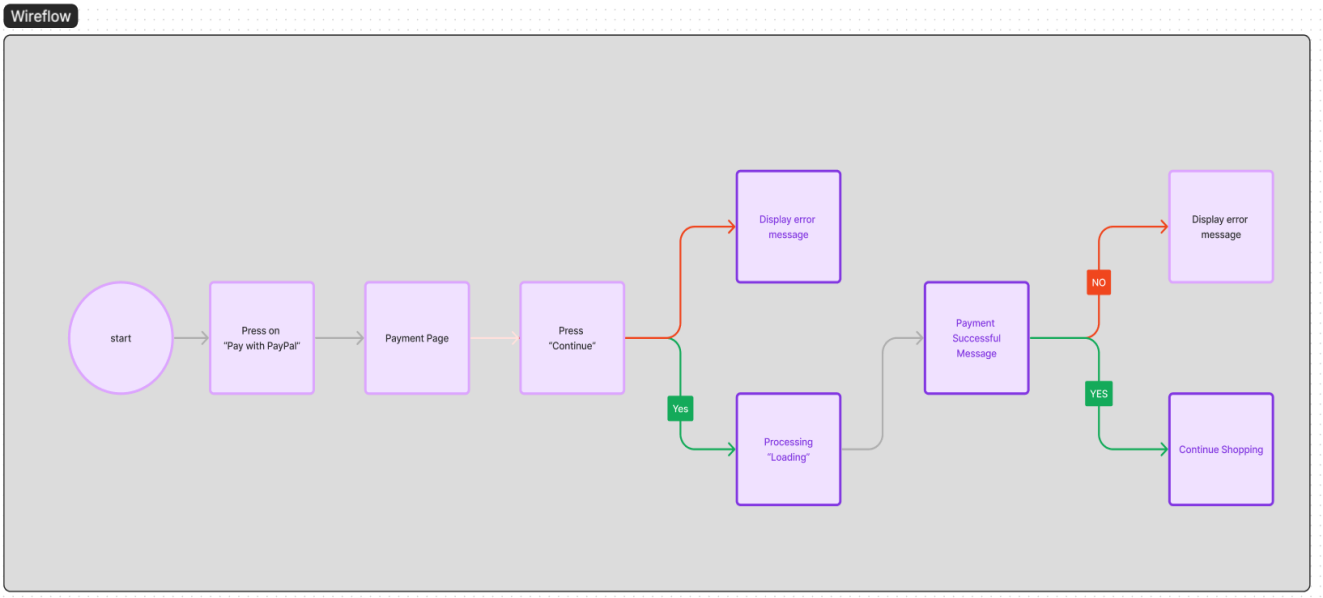
❖ **Webview Integration:**

- The code utilizes the flutter_webview_plugin package to embed a web view for displaying the PayPal payment page within the Flutter app.

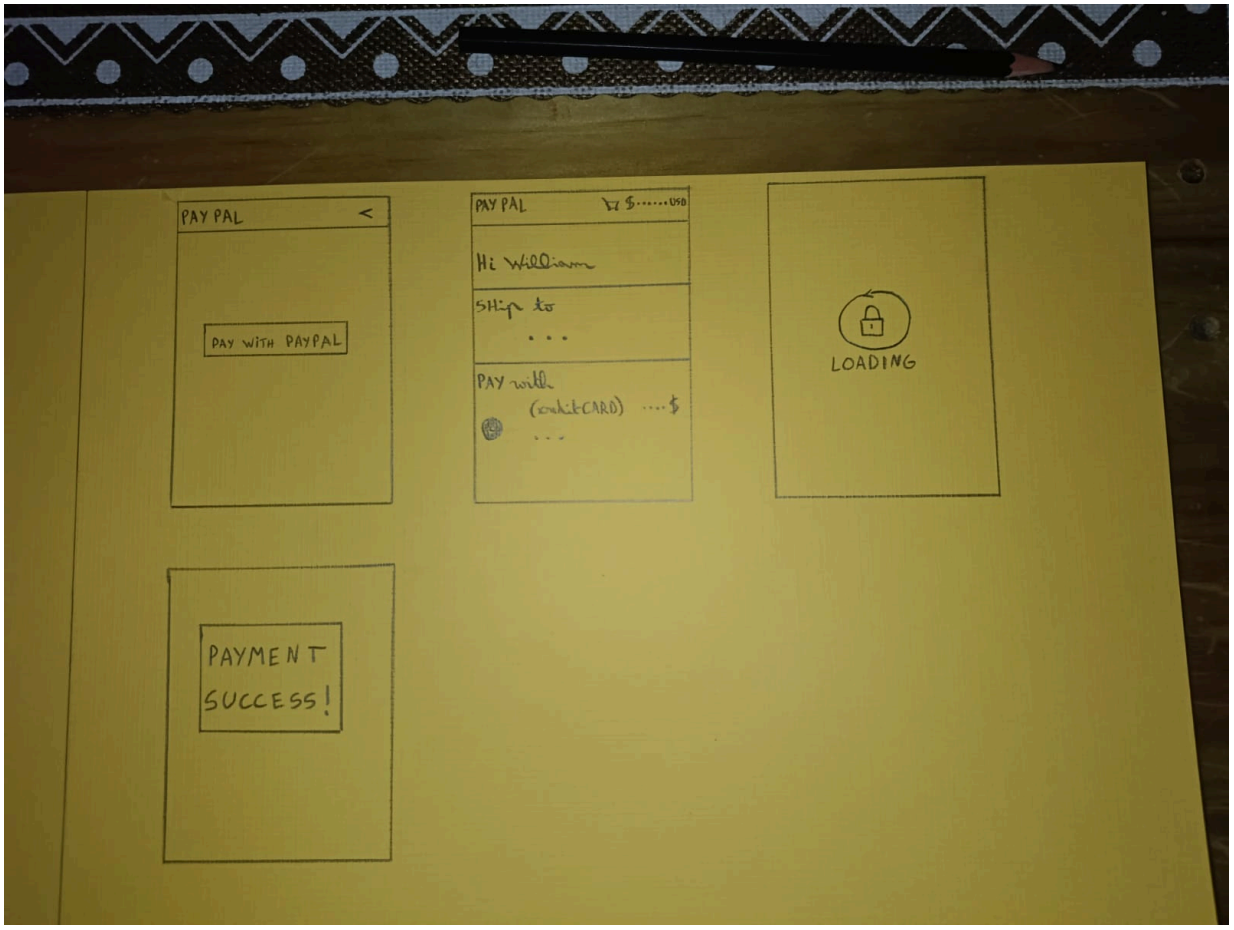
DESIGN

Here are the designs we had in mind for the project, with a mockup, a wireflow and a Storyboard

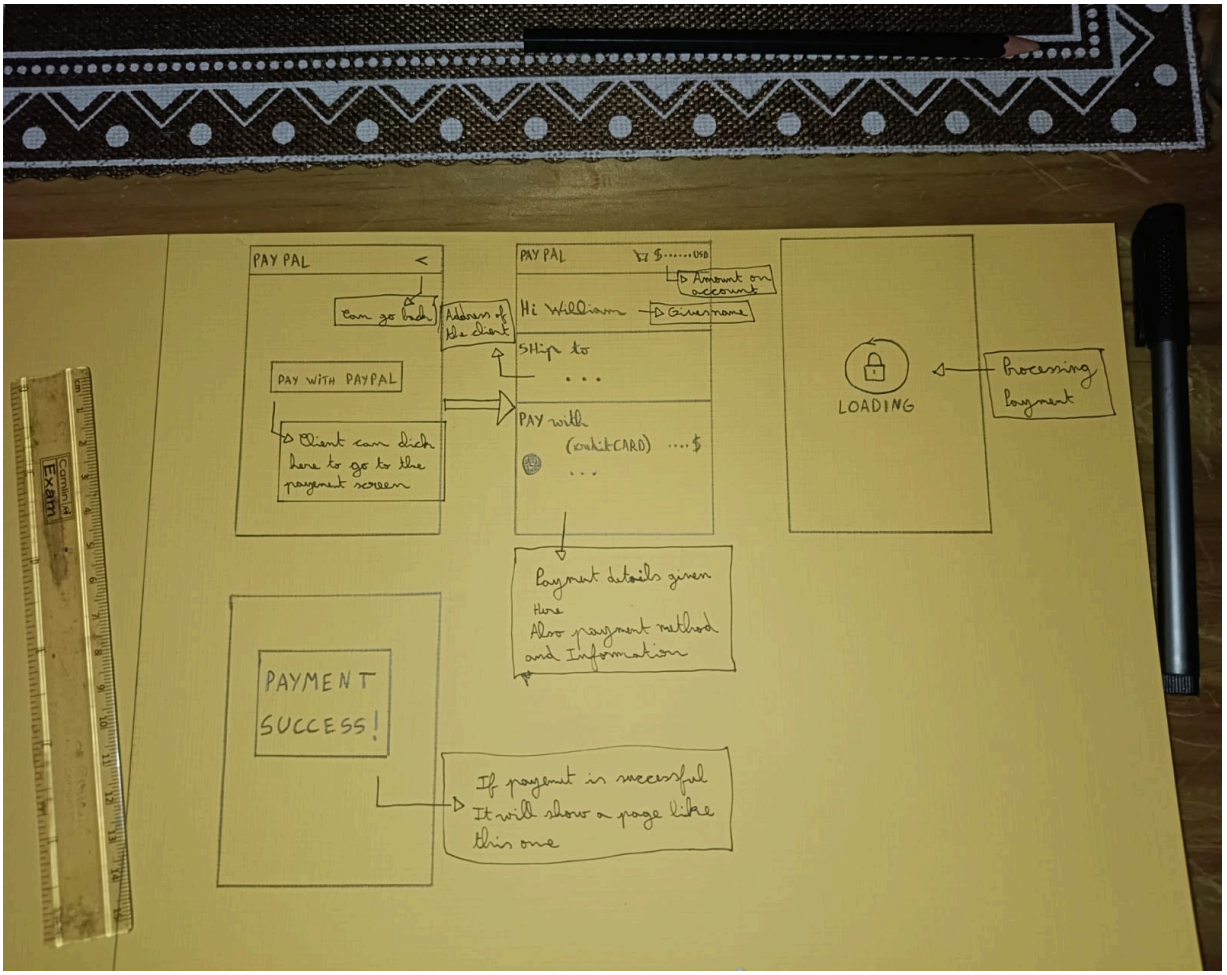
Wireflow :



Mockup :


PAY PAL
PAY WITH PAYPAL
PAY PAL
Hi William
SHip to
PAY with
LOADING
PAYMENT
SUCCESS!

Storyboard :


PAY PAL
can go back
Address of the client
PAY WITH PAYPAL
Client can click here to go to the payment screen
PAY PAL
$.....USD
Amount on account
Hi William
Givenname
SHip to
PAY with
(credit CARD) ....$
Payment details given Here
Also payment method and Information
LOADING
Processing Payment
PAYMENT SUCCESS!
If payment is successful It will show a page like this one

ALGORITHM USED

Explanation of Dart main section of codes if there are any complex functions.

- **Imports:**
  - The necessary packages and files are imported, including flutter/material.dart, constants.dart, flutter_paypal, and ui_helper.dart. These imports bring in the required functionalities for building a Flutter app and integrating PayPal.

- **Main Function:**
  - The main() function is the entry point of the Dart application, responsible for running the app. In this case, it calls runApp() to start the Flutter application, passing an instance of the MyApp widget.

- **MyApp Class:**
  - The MyApp class is a stateless widget that serves as the root of the application. It sets the title and theme for the app and specifies the initial screen (MyHomePage) to be displayed.

- **MyHomePage Class:**
  - MyHomePage is a stateful widget representing the main screen of the app. It includes a counter variable (_counter) and a method (_incrementCounter) to increment the counter. The build method constructs the UI with a Scaffold, AppBar, and a Column containing an ElevatedButton.

- **ElevatedButton:**
  - The ElevatedButton widget is used to create a button labeled "Pay With PayPal." When pressed, it triggers the navigation to another screen (UsePaypal) using Navigator.of(context).push().

- **UsePaypal Screen:**
  - The UsePaypal screen is navigated to when the "Pay With PayPal" button is pressed. It is part of the Flutter PayPal integration and includes parameters such as sandbox mode, client ID, secret key, return and cancel URLs, transaction details, and callback functions for success, error, and cancellation events.

- **Callback Functions:**
  - The onSuccess, onError, and onCancel callback functions define what should happen when a payment is successfully completed, encounters an error, or is canceled, respectively.

- **UIHelper:**
  - The UIHelper class from ui_helper.dart is utilized to show alert dialogs indicating the success, error, or cancellation of the payment.

Algorithms used in the code :

- State Management :
  - The Flutter framework employs its own algorithm for state management. In this code, the _counter variable in the MyHomePage widget is an example of local state management within a widget. Flutter utilizes a reactive programming paradigm to handle changes in state and efficiently update the UI.

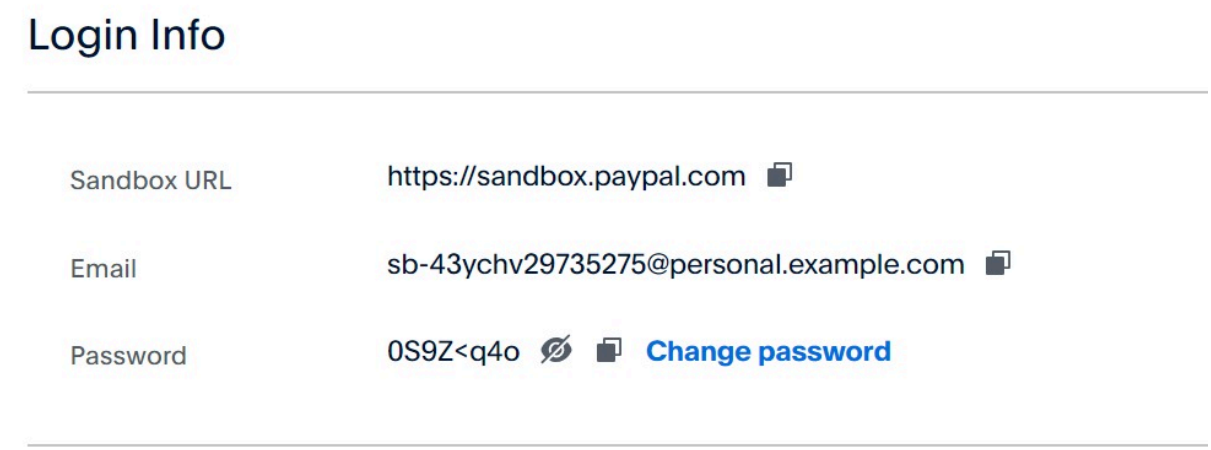
- Routing and Navigation :
  - The navigation from the main screen to the PayPal integration screen (UsePaypal) involves Flutter's navigation system. It uses a stack-based approach, where screens are pushed onto a navigation stack, and you can navigate between them using the Navigator class.

- Asynchronous Programming :
  - Flutter applications often involve asynchronous programming to handle tasks like network requests. The use of async and await keywords in the callback functions (e.g., onSuccess, onError) indicates asynchronous operations, which is a common practice in Flutter app development.

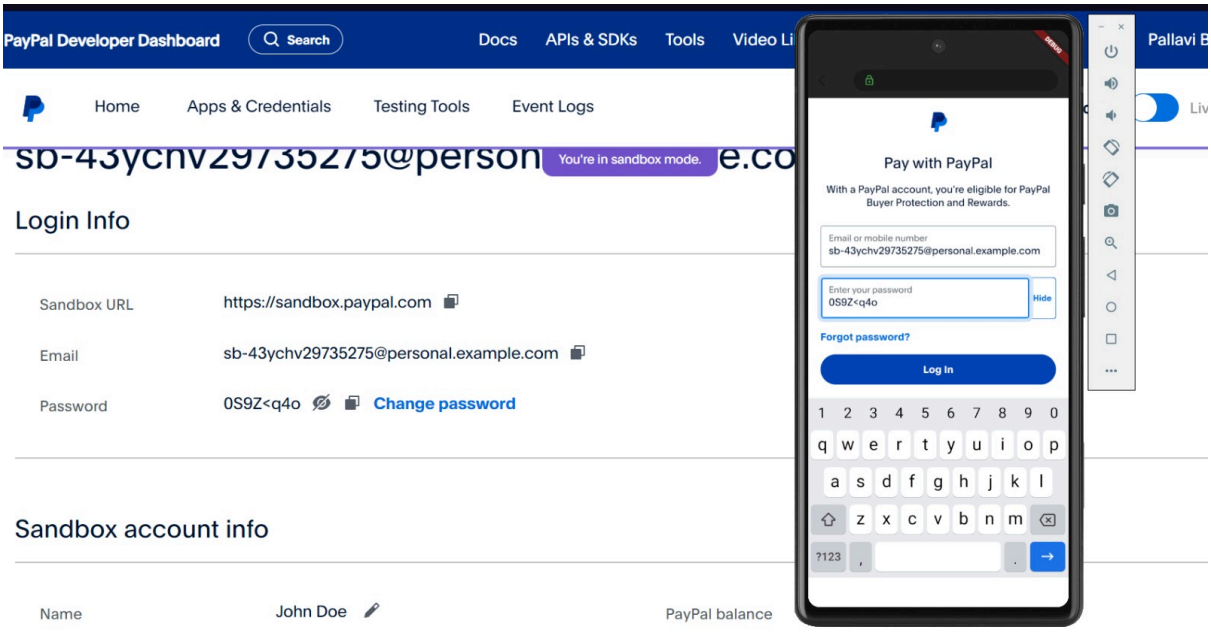
- Callback Pattern :
  - The code uses a callback pattern to handle events such as successful payment, payment errors, and cancellations. Callbacks (onSuccess, onError, onCancel) are functions passed as parameters to the UsePaypal widget and are executed when certain events occur.

- Event Handling :
  - The ElevatedButton widget's onPressed property is set to trigger navigation to the UsePaypal screen. This involves handling the button press event, a fundamental aspect of Flutter's event-driven programming model.

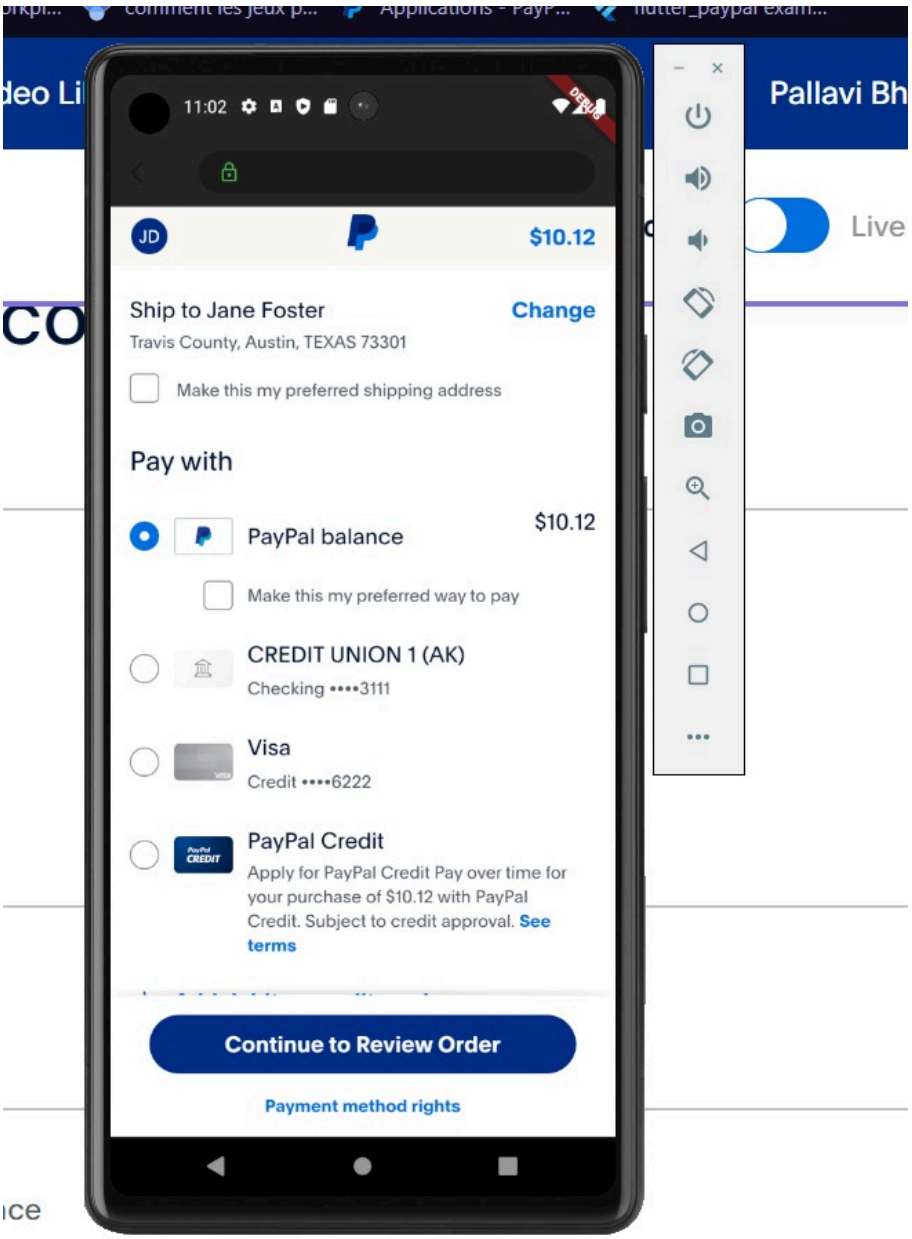
ANALYSIS OF RESULT OBTAINED



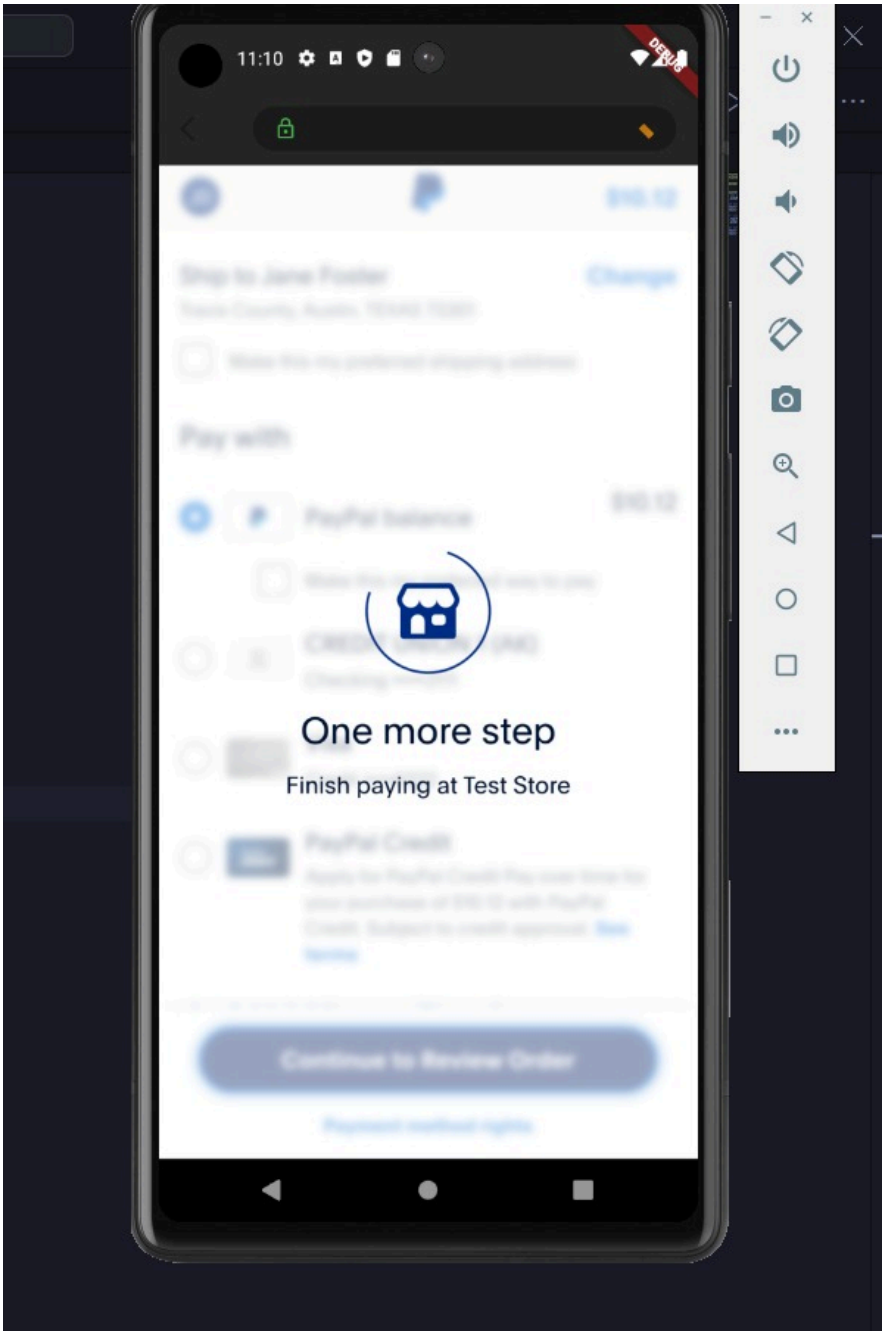
Information about the person that logged-in into the payapal



This is the users information with the login screen



This is the payment screen where you can choose the payment method and the shipping location



A loading page for the processing of the order after the “Continue to review order button is pressed”

CONCLUSION

In conclusion, the project has made substantial progress towards its primary goal of integrating PayPal payments into the Flutter mobile application. The code structure is well-organized, with modular components facilitating maintenance and scalability. The user interface, represented by the MakePayment screen, provides a straightforward means for users to initiate payments through the "Pay with PayPal" button. Asynchronous programming ensures smooth interactions during network requests, contributing to a responsive user experience. The error-handling mechanisms implemented in the code enhance the application's robustness. To confirm the project's success, thorough testing across various payment scenarios and real-world conditions is essential. Incorporating user feedback and testing the application in diverse scenarios will further validate the achievement of the project's objectives and contribute to refining the overall user experience.

REFERENCE :

Dart packages. (n.d.). flutter_paypal example | Flutter package. [online] Available at: https://pub.dev/packages/flutter_paypal/example

Dart packages. (n.d.). one_context | Flutter package. [online] Available at: https://pub.dev/packages/one_context [Accessed 1 Mar. 2024].

www.youtube.com. (n.d.). Developer Dashboard Apps & Credentials. [online] Available at: https://www.youtube.com/watch?v=uNq64tqz7TE

www.youtube.com. (n.d.). Flutter: Paypal Payments | Integrate PaypalGateway complete guide. [online] Available at: https://www.youtube.com/watch?v=GCoiMbybVxw&ab_channel=FlutterAdvance

Yadav, G. (2021). Paypal Payment Gateway Integration in Flutter. [online] Flutter Community. Available at: https://medium.com/flutter-community/paypal-payment-gateway-integration-in-flutter-379fbb3b87f5