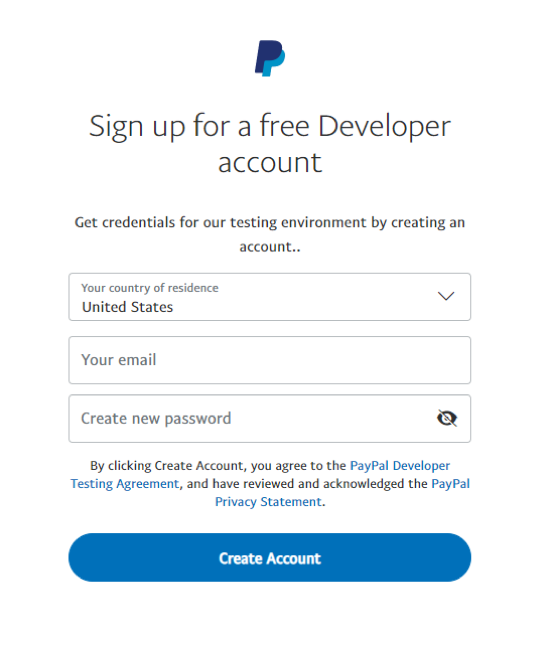
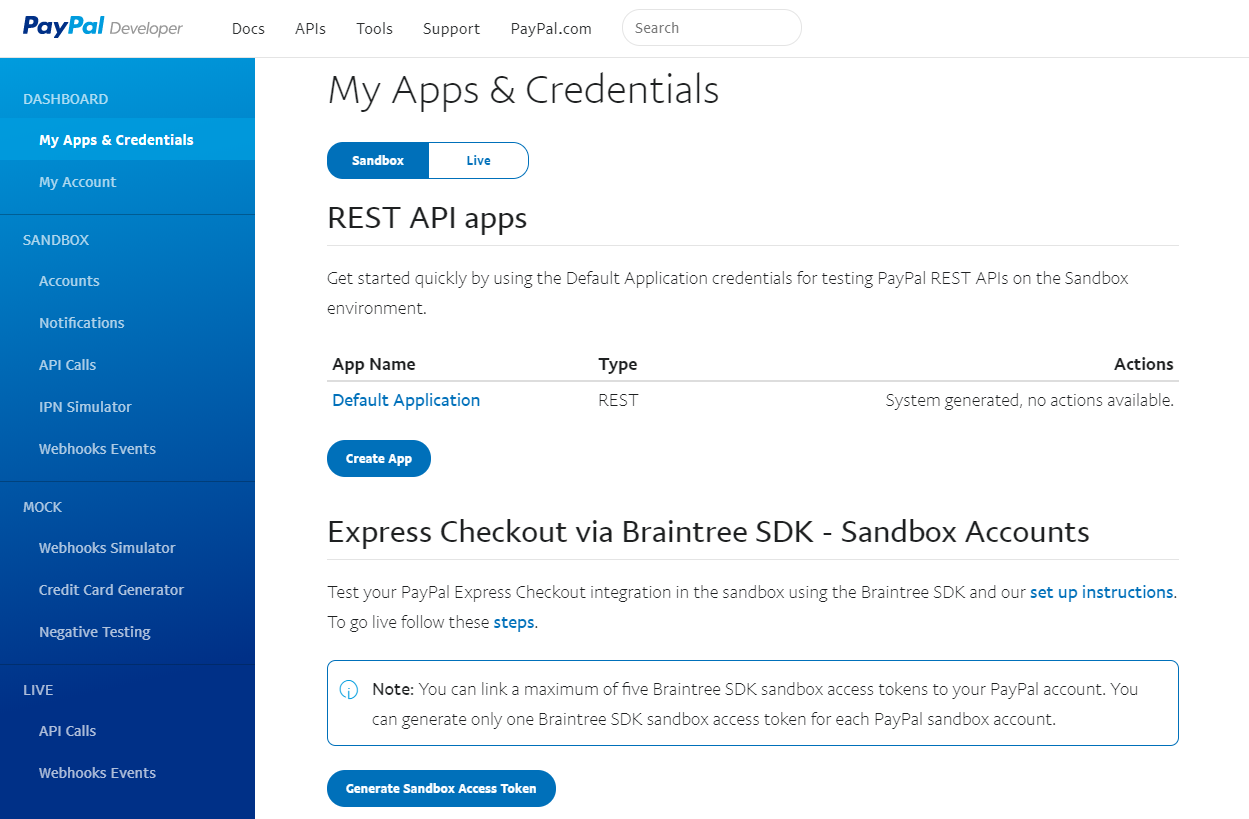
**Setup Development Environment**

1. **Tạo tài khoản PayPal Developer**

Đăng ký tài khoản PayPal Developer tại: <https://www.paypal.com/signin/client?flow=provisionUser>

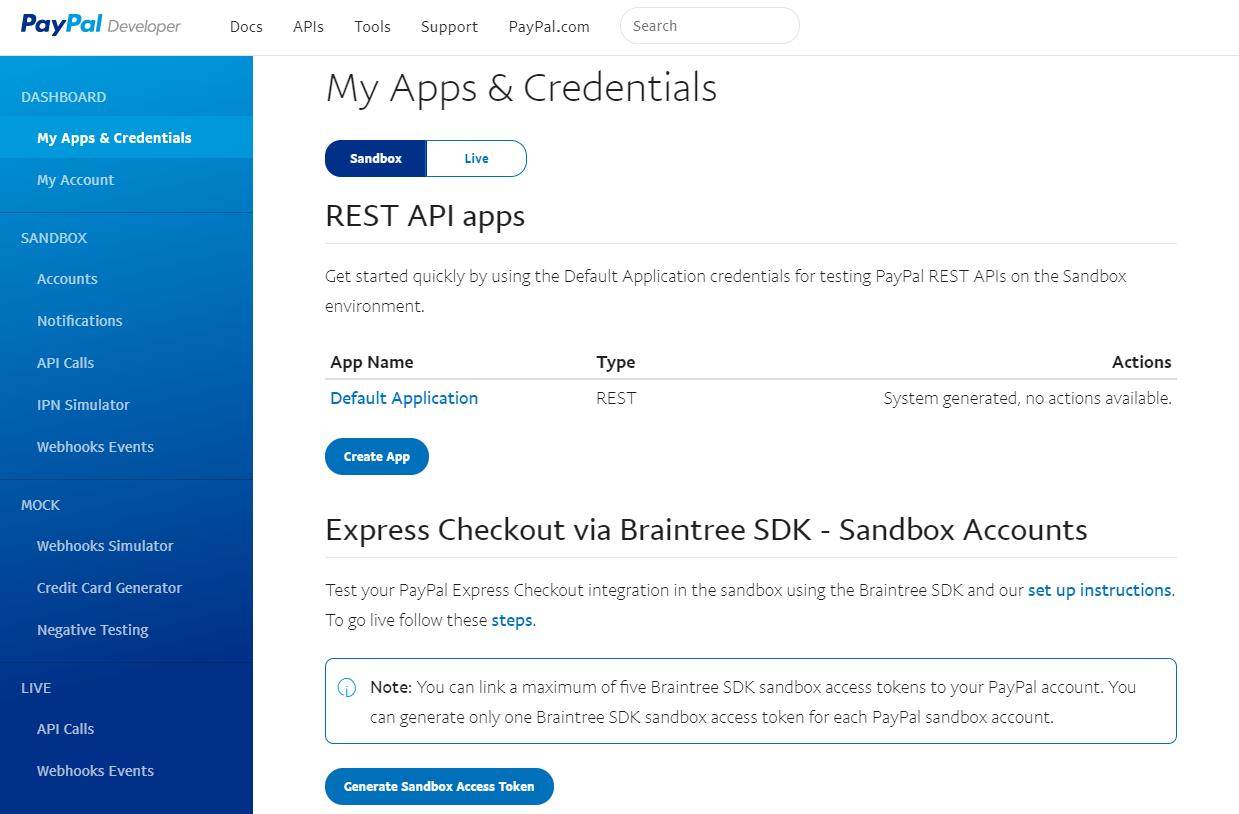
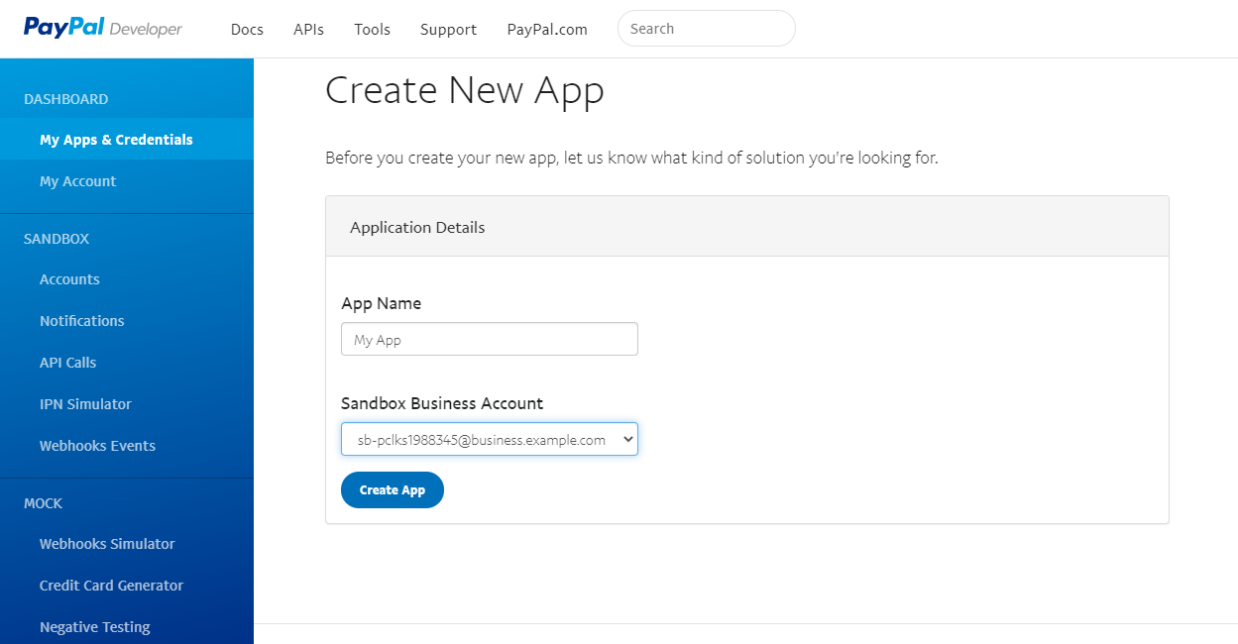
Đăng nhập vào dashboard

1. **Tạo REST API app**

Để gọi được các API của PayPal thì ta phải tạo một ứng dụng REST API (REST API apps), sau đó lấy các thông tin xác thực (credentials).

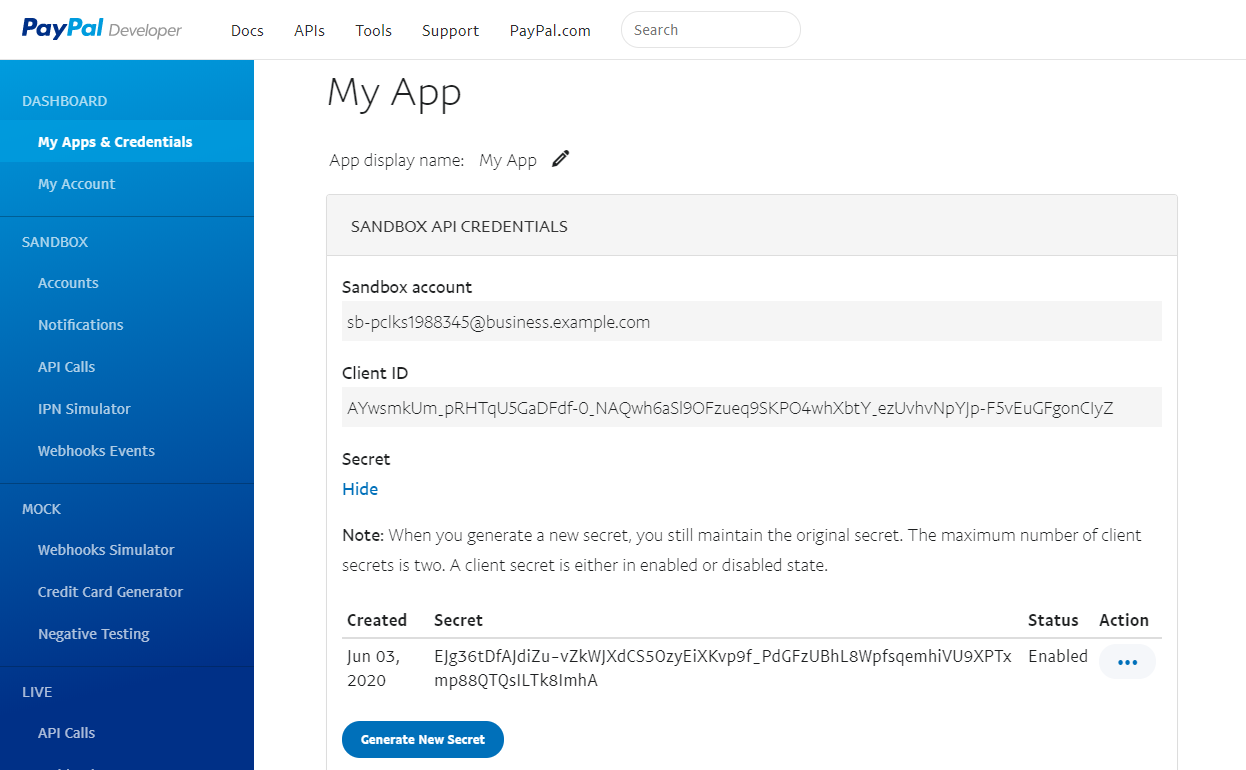
Mặc định PayPal đã tạo sẵn một REST API app với tên Default Application.

Để tạo ứng dụng REST API:

1. Chọn **Create App**
2. Nhập **App Name**
3. Nhấn nút **Create App**
4. **Lấy thông tin xác thực API (API credentials)**

Thông tin xác thực API bao gồm: **Client ID** và **Secret.**

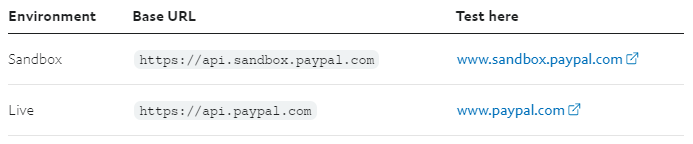
Hai thông tin này bắt buộc phải có khi gọi API của PayPal để giao dịch.

Lưu hai thông tin này lại để sau này sử dụng để gọi API của PayPal.

1. **Tạo tài khoản Sandbox để test**

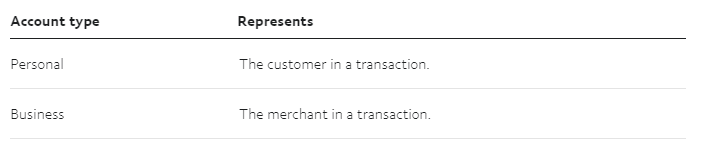
Sau khi get được thông tin xác thực API (API credentials), ta sẽ tiến hành tạo tài khoản cho người mua (buyer) và người bán (merchant/seller) để giao dịch.

PayPal có hai môi trường phục vụ cho việc test (Sandbox Environment) và production (Live Environment).

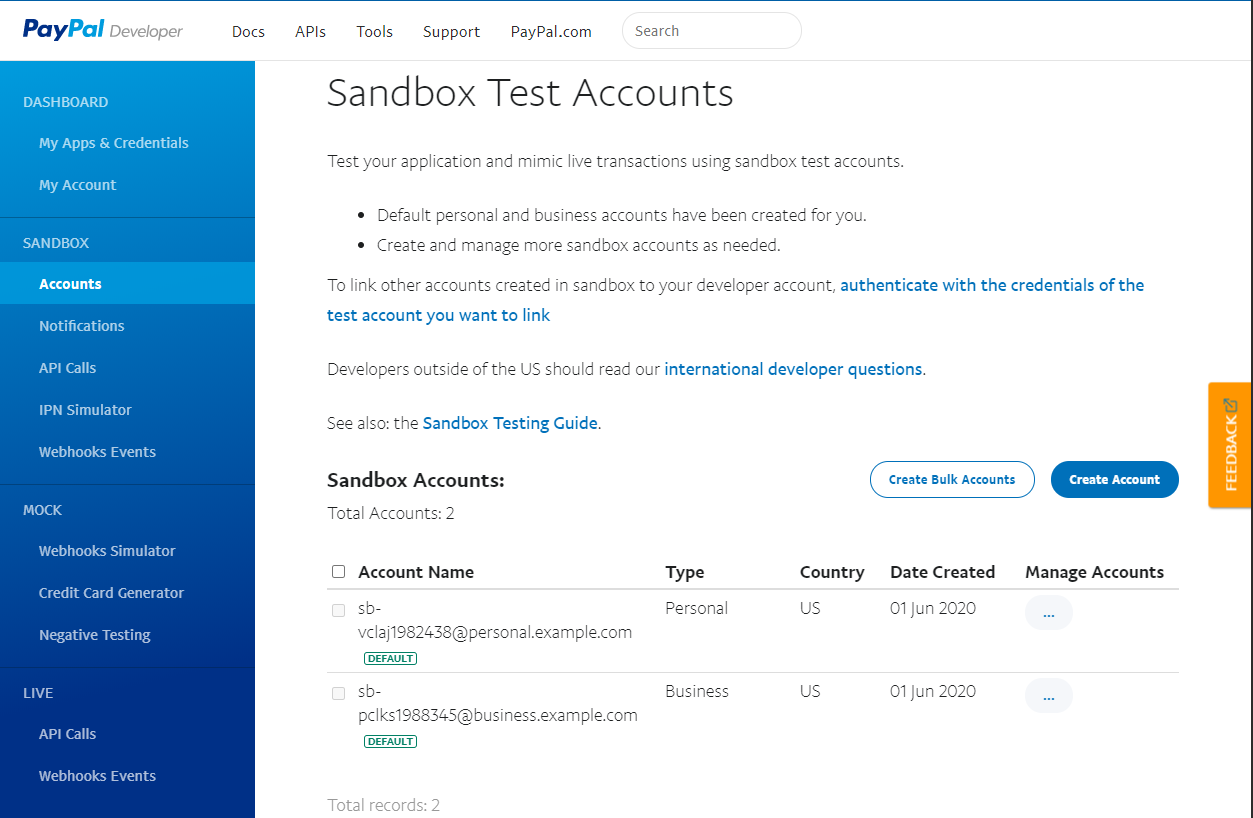


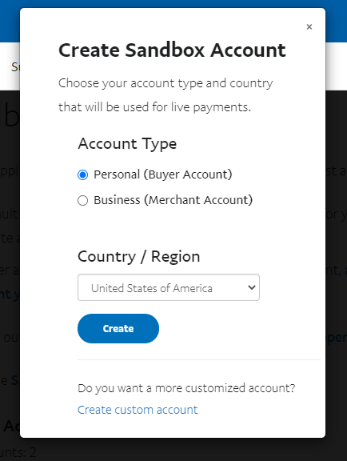
PayPal Sandbox là môi trường để chúng ta có thể phát triển, test thử các giao dịch, đảm bảo việc tích hợp PayPal chính xác trước khi chạy trên production.

Có hai loại tài khoản:

* Personal: tài khoản cho người mua (bên chuyển tiền)
* Business: tài khoản cho người bán (bên nhận tiền)

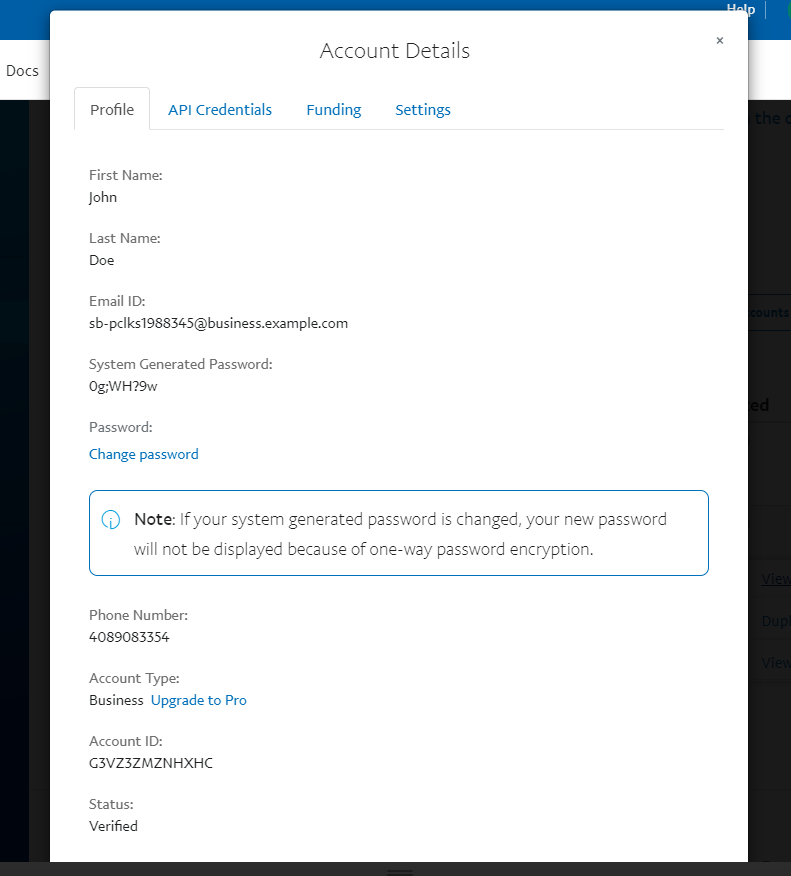
Mặc định thì PayPal đã tạo sẵn 2 tài khoản cho người mua (Personal) và người bán (Business).

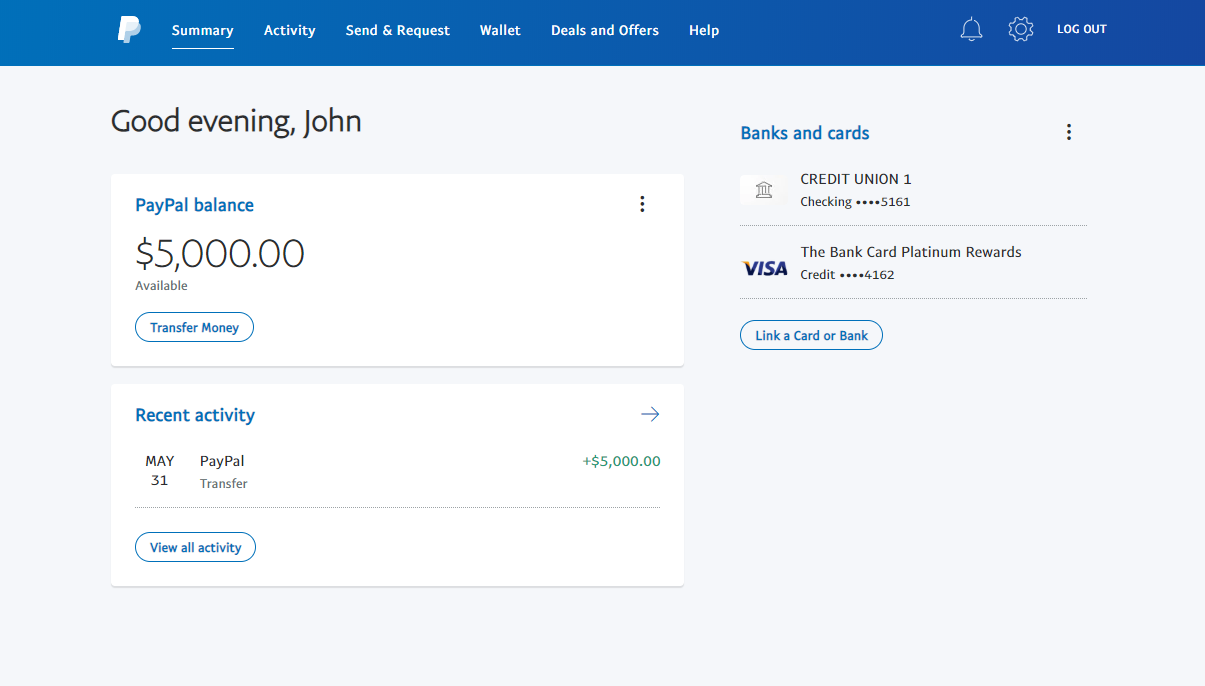
Ta có thể sử dụng tài khoản mặc định này để test các giao dịch hoặc có thể tạo thêm.

Tạo tài khoản Sanbox:

1. Chọn **Accounts** ở menu Sandbox
2. Chọn **Create Account**
3. Chọn **Account Type** và nhấn **Create**

**Đăng nhập vào PayPal Sandbox để xem lịch sử giao dịch, số dư tài khoản:**

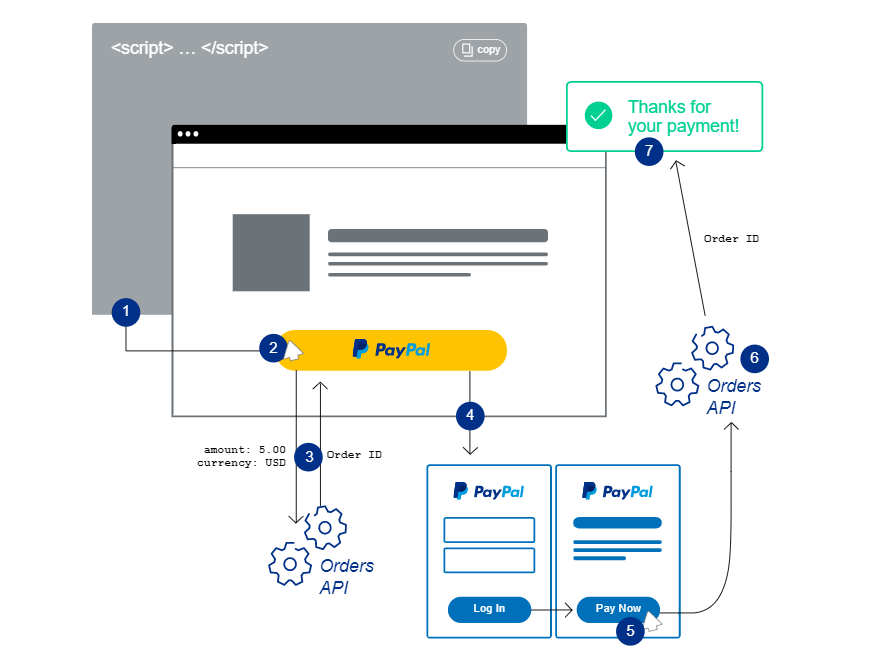
1. Lựa chọn tài khoản sandbox
2. Chọn Manage Accounts -> View/Edit Account
3. Lấy thông tin đăng nhập bao gồm email và password (system generated password)
4. Truy cập <https://www.sandbox.paypal.com/>
5. Đăng nhập với email và password ở trên



Mặc định mỗi tài khoản Sandbox sẽ có sẵn 5000$ để chúng ta có thể giao dịch.

Muốn biết giao dịch thành công hay thất bại (đã chuyển được tiền hay chưa) thì chỉ cần đăng nhập vào tài khoản của người bán và người mua để xem số dư (PayPal balance) và lịch sử giao dịch (Activity).

**PayPal Payment Process**



Qui trình tích hợp thanh toán với PayPal

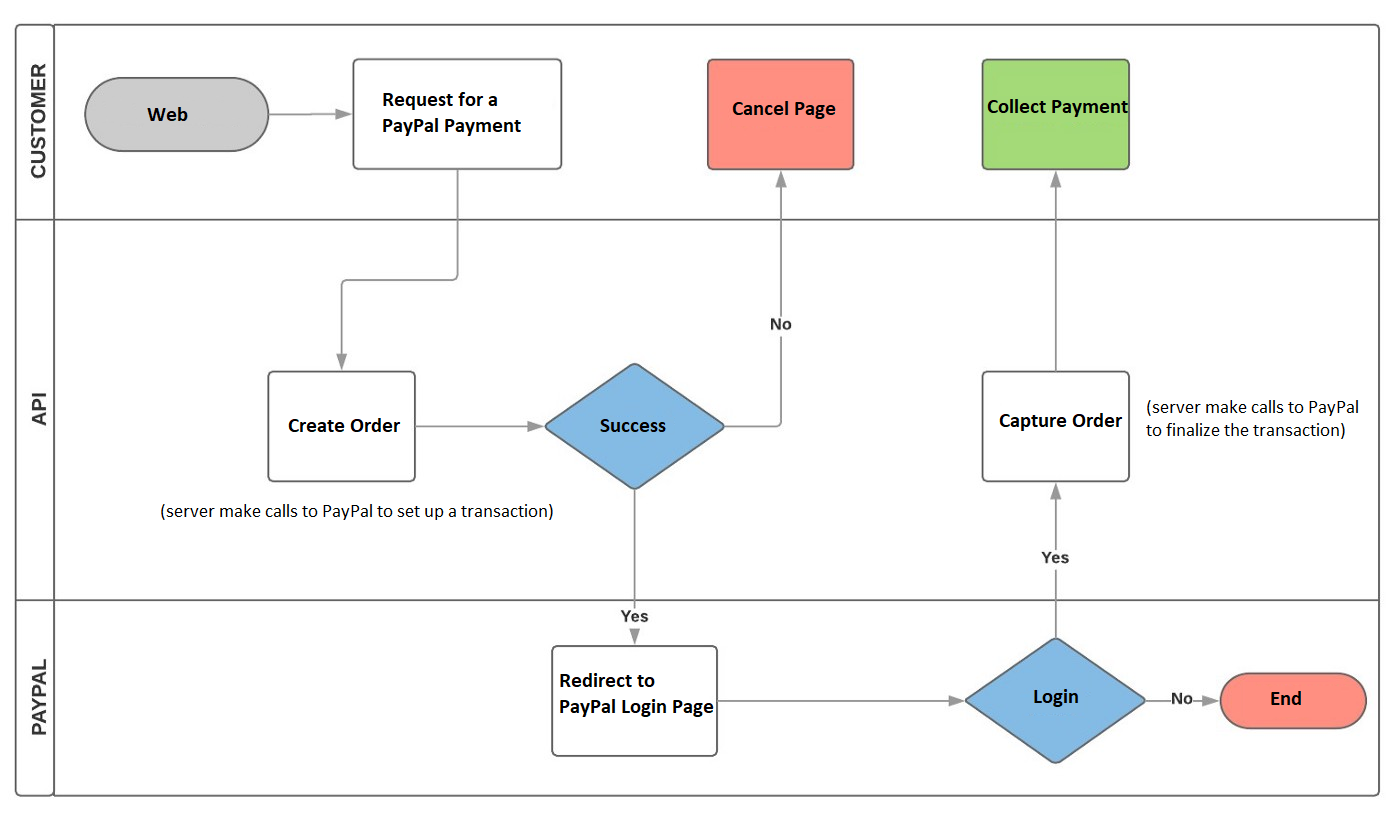
1. Thêm nút thanh toán với PayPal vào web
2. Người mua nhấn vào nút thanh toán với PayPal
3. Gọi PayPal Orders API để tiến hành phiên giao dịch (set up a transaction) (gửi số tiền, đơn vị tiền + các thông tin khác)
4. Hiển thị trang đăng nhập PayPal, người mua đăng nhập vào PayPal và nhấn nút Pay Now
5. Gọi PayPal Orders API để hoàn thành phiên giao dịch (finalize the transaction)
6. Hiển thị trang xác nhận thanh toán thành công đến người mua

Việc call Orders API (hiểu đơn giản là API do PayPal cung cấp để tiến hành giao dịch) có thể thực hiện ở Client-Side hoặc Server-Side.

Call Orders API ở Client-Side giúp việc tích hợp thanh toán với PayPal dễ dàng. Tuy nhiên chúng ta sẽ call Orders API ở phía Server-Side để dễ dàng thao tác với hệ thống, services, database và bảo mật tốt hơn (Client ID và Secret sẽ lưu ở phía backend).

*“PayPal provides client-side shortcuts to help you complete a transaction with minimal server integration. You can also make calls to PayPal APIs directly from your server to integrate PayPal more closely with your backend systems, services, and database.”*

**Integration**



**Set Up PayPal JavaScript SDK Script**

**Thêm PayPal JavaScript SDK vào web**

<!DOCTYPE html>

<head>

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <meta http-equiv="X-UA-Compatible" content="IE=edge" />

</head>

<body>

  <script src="https://www.paypal.com/sdk/js?client-id=CLIENT\_ID">

  </script>

</body>

*Thay thế CLIENT\_ID với Sandbox Client Id đã tạo*

Xem thêm cách customize PayPal Javascript SDK tại đây: <https://developer.paypal.com/docs/checkout/reference/customize-sdk/>

**Render button PayPal**

<body>

  <script src="https://www.paypal.com/sdk/js?client-id=CLIENT\_ID">

  </script>

  <div id="paypal-button-container"></div>

  <script>

    paypal.Buttons().render('#paypal-button-container');

  </script>

</body>

Xem thêm cách customize button tại đây:

<https://developer.paypal.com/docs/checkout/integration-features/customize-button/>

**Set Up Server-Side SDK**

Ta sẽ không gọi trực tiếp đến API của PayPal mà sẽ dùng thư viện hỗ trợ (hay còn gọi là SDK).

PayPal REST SDKs cung cấp cho chúng ta các model, method cần thiết để có thể sử dụng để gọi đến PayPal API (nếu không sử dụng SDK thì ta sẽ phải tự tạo model, call http, xử lý authentication…)

PayPal REST SDKs có sẵn cho: Java, PHP, Node, Python, Ruby, .NET.

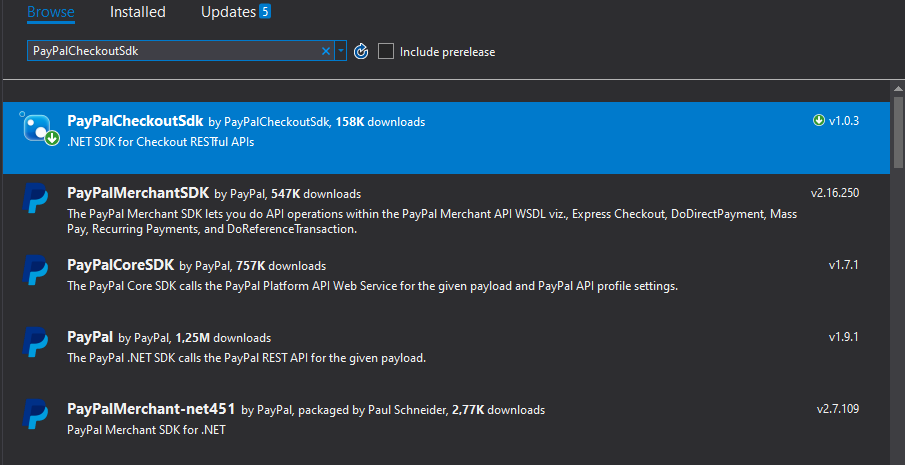
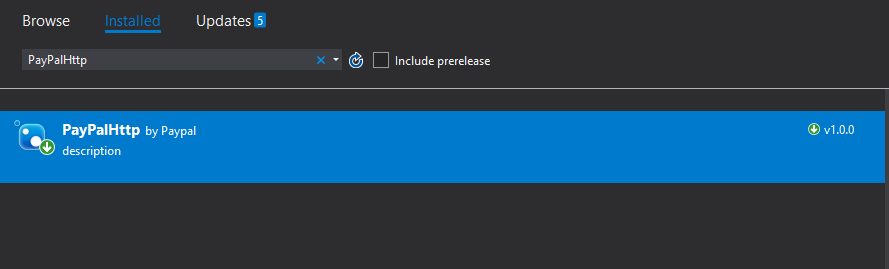
**Cài đặt SDK cho .NET Core**

* Cài đặt qua .NET CLI

dotnet add package PayPalCheckoutSdk

dotnet add package PayPalHttp

* Cài đặt qua Nuget packges



Sau khi cài đặt SDK thành công, tại project đã cài đặt SDK, tạo file **PayPalClient.cs** để chứa những config bao gồm: config PayPal environment (development, testing dùng Sandbox environment, production dùng Live Enviroment), setup PayPal Credentials bao gồm Client ID, Client Secret.

using PayPalCheckoutSdk.Core;

using PayPalHttp;

public class PayPalClient

{

    /\*\*

        Set up PayPal environment with sandbox credentials.

        In production, use LiveEnvironment.

        \*/

    public static PayPalEnvironment Environment()

    {

        return new SandboxEnvironment("CLIENT\_ID","CLIENT\_SECRET");

    }

    /\*\*

        Returns PayPalHttpClient instance to invoke PayPal APIs.

        \*/

    public static HttpClient Client()

    {

        return new PayPalHttpClient(Environment());

    }

}

*Thay thế CLIENT\_ID và CLIENT\_SECRET với Sandbox Client Id và Sandbox Client Secret đã tạo*

**Set Up Transaction**

**Ở phía server:**

**CheckoutController.cs**

[HttpPost("api/checkout/orders ")]

public async Task<IActionResult> Create()

{

    var order = await checkoutService.CreateOrder();

    return Ok(order);

}

**CheckoutService.cs**

public async Task<Order> CreateOrder()

{

    var request = new OrdersCreateRequest();

    request.Prefer("return=representation");

    request.RequestBody(GetOrderRequest());

    // Call PayPal to set up a transaction

    var response = await PayPalClient.Client().Execute(request);

    return response.Result<Order>();

}

private OrderRequest GetOrderRequest() {

    return new OrderRequest {

        CheckoutPaymentIntent = "CAPTURE",

        ApplicationContext = new ApplicationContext {

            BrandName = "<-Your Brand Name->",

            LandingPage = "NO\_PREFERENCE",

            UserAction = "CONTINUE",

            ShippingPreference = "NO\_SHIPPING"

        },

        PurchaseUnits = new List<PurchaseUnitRequest> {

                new PurchaseUnitRequest {

                    AmountWithBreakdown = new AmountWithBreakdown {

                        CurrencyCode = "USD",

                        Value = "100",

                        AmountBreakdown = new AmountBreakdown {

                                ItemTotal = new Money {

                                CurrencyCode = "USD",

                                Value = "100"

                            }

                        }

                    },

                    Items = new List<Item> {

                        new Item {

                            Name = "<-Item name->",

                            Description = "<-Item description->",

                            UnitAmount = new Money {

                                CurrencyCode = "USD",

                                Value = "100"

                            },

                            Quantity = "1"

                        }

                }

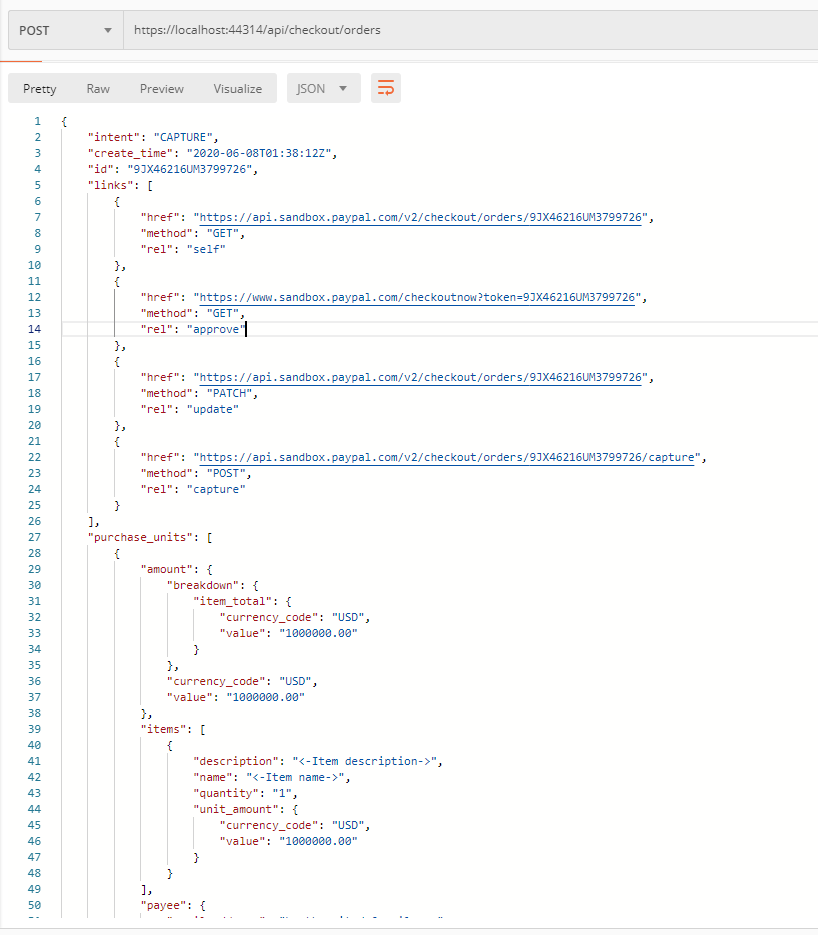
            }

        }

    };

}

**Test Checkout API với Postman**



**Ở phía client:**

<script>

    paypal.Buttons({

        createOrder: function () {

            return fetch('/my-server/api/checkout/orders', {

                method: 'post',

                headers: {

                    'content-type': 'application/json'

                }

            }).then(function (*res*) {

                return res.json();

            }).then(function (*data*) {

                return data.id;

            });

        }

    }).render('#paypal-button-container');

</script>

**Capture Transaction**

**Ở phía server:**

**CheckoutController.cs**

[HttpPost("api/checkout/orders/{id}/capture")]

public async Task<IActionResult> Capture([FromRoute] string *id*)

{

    var order = await checkoutService.CaptureOrder(id);

    return Ok(order);

}

**CheckoutService.cs**

public async Task<Order> CaptureOrder(string *orderId*)

{

    var request = new OrdersCaptureRequest(orderId);

    request.Prefer("return=representation");

    request.RequestBody(new OrderActionRequest());

    // Call PayPal to capture an order

    var response = await PayPalClient.Client().Execute(request);

    return response.Result<Order>();

}

**Ở phía client:**

<script>

    paypal.Buttons({

        createOrder: function () {

            return fetch('/my-server/api/checkout/orders', {

                method: 'post',

                headers: {

                    'content-type': 'application/json'

                }

            }).then(function (*res*) {

                return res.json();

            }).then(function (*data*) {

                return data.id;

            });

        },

        onApprove: function (*data*) {

         return fetch(`/my-server/api/checkout/orders/${data.orderID}/capture`, {

                method: 'post',

                headers: {

                    'content-type': 'application/json'

                }

            }).then(function (*res*) {

                return res.json();

            }).then(function (*details*) {

                alert('Transaction funds captured');

            })

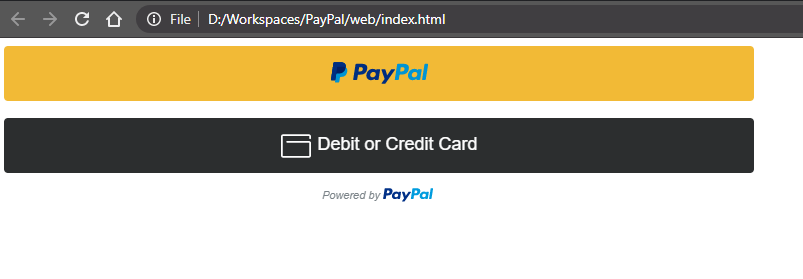
        }

    }).render('#paypal-button-container');

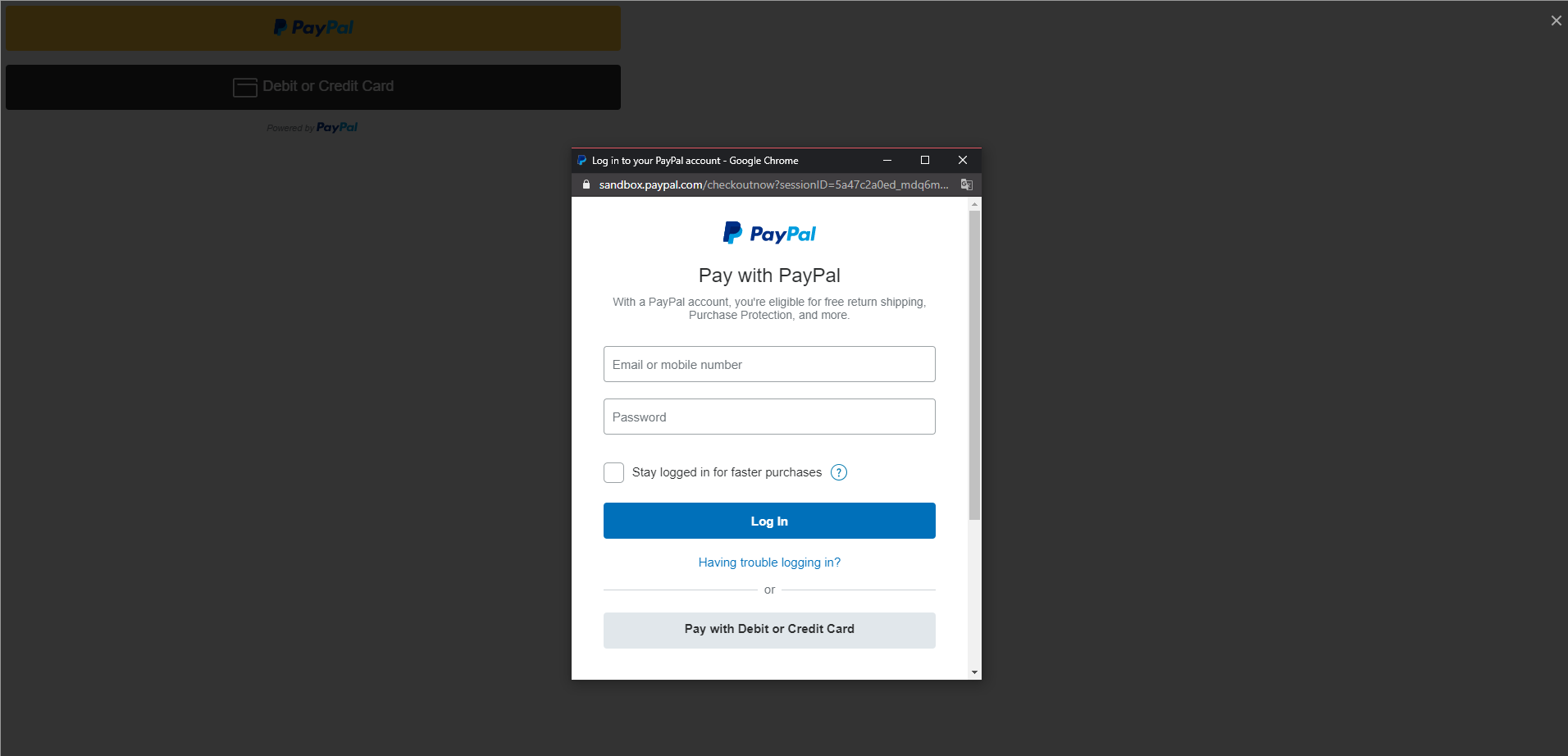
</script>

**Testing**

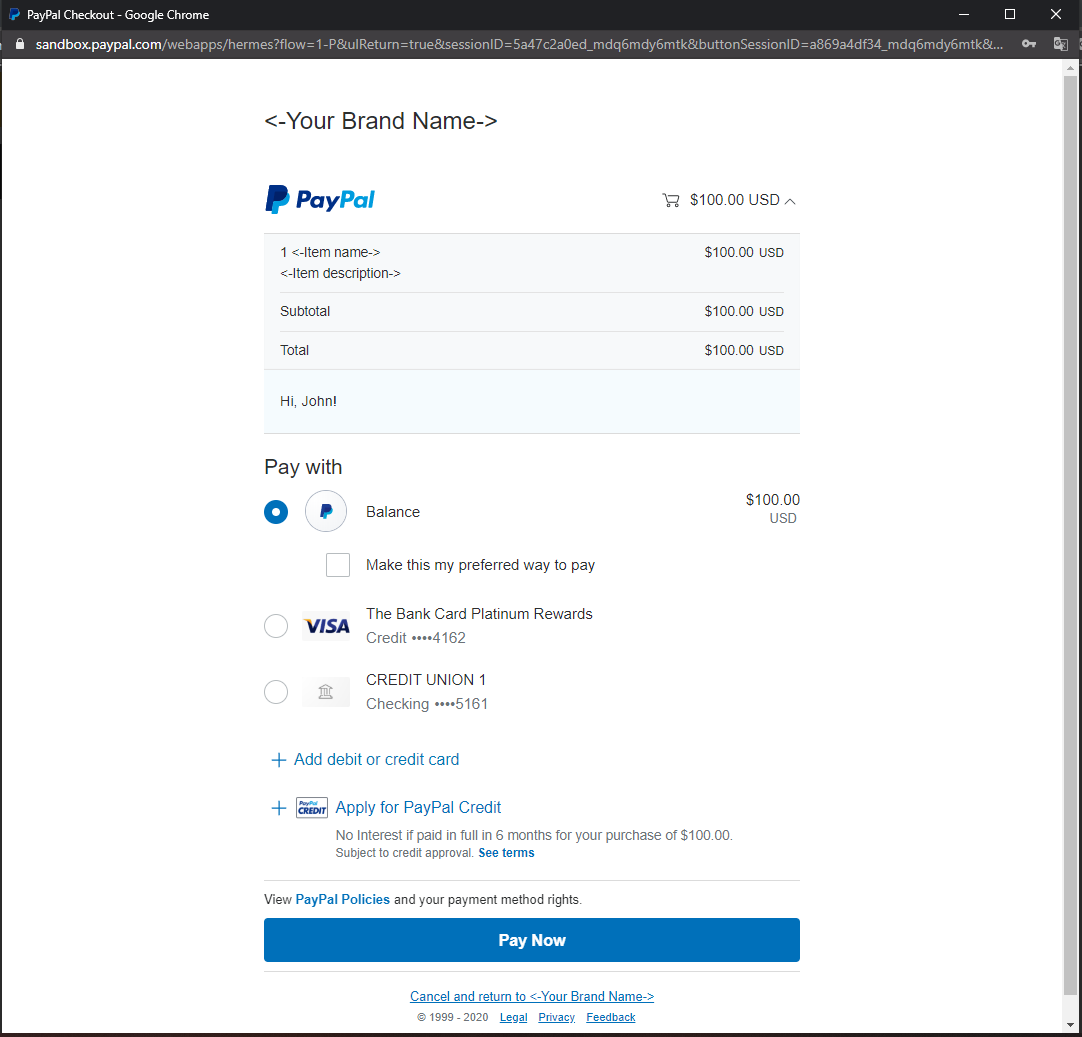
**PayPal button**



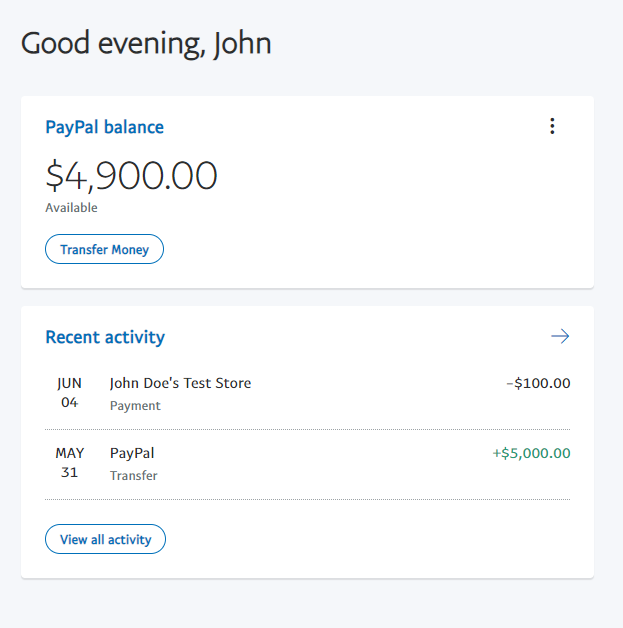
**Click PayPal -> Đăng nhập với Personal Account**

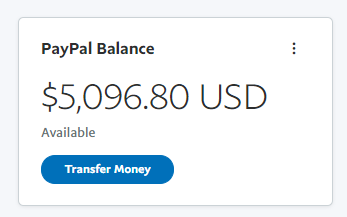
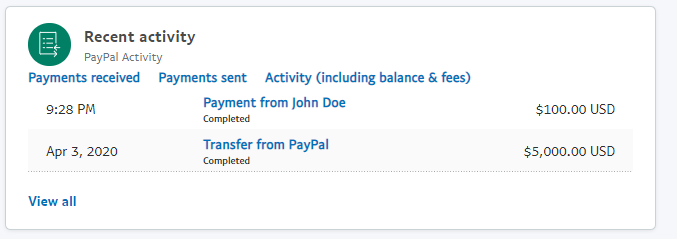


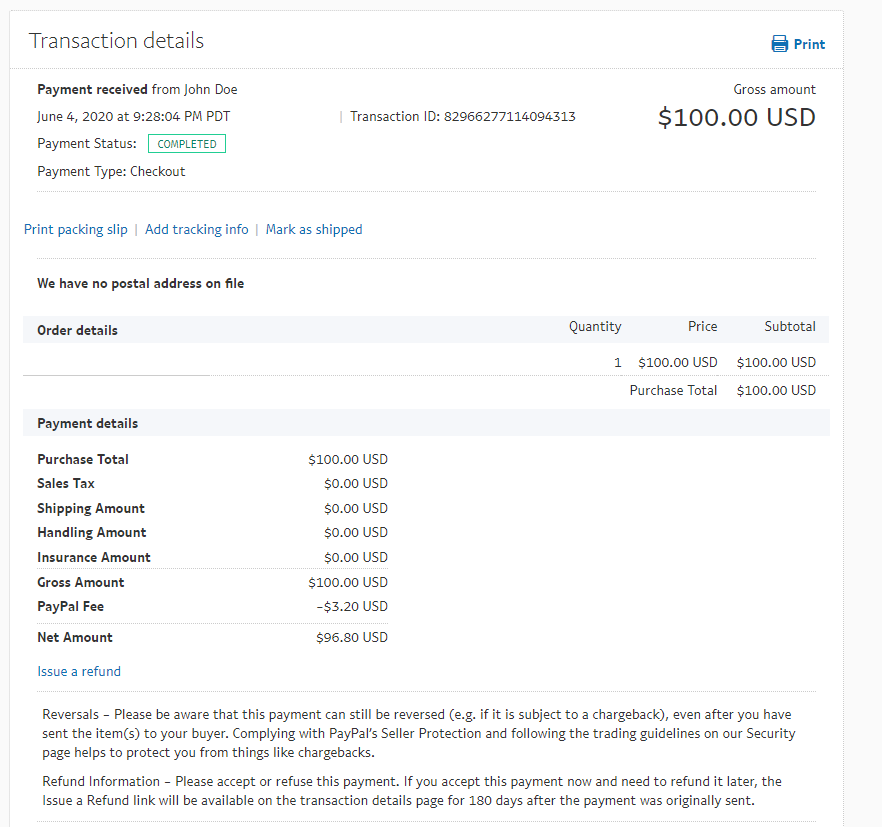
**Sau khi đăng nhập thành công PayPal -> Click Pay Now**



**Đăng nhập Sandbox với Personal Account**



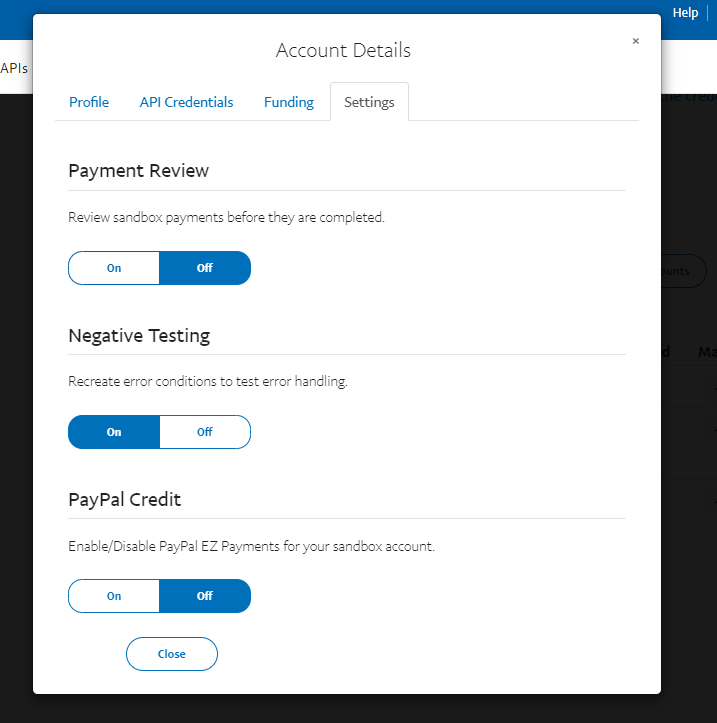
**Đăng nhập Sandbox với Business Account**



**Test các trường hợp lỗi với negative testing**

**Bật negative testing**

1. Truy cập  <https://developer.paypal.com> và đăng nhập vào Dashboard với tài khoản PayPal Developer
2. Chọn **Accounts** ở menu **Sandbox**
3. Chọn tài khoản **Business Sandbox** đã tạo
4. Tại cột **Manage Account** -> **View/Edit Account**
5. Chọn **Settings** tab
6. Tại option **Negative Testing** -> click **ON**



**Test với negative testing**

REST API apps dùng request header để thực thi negative testing ở sandbox.

*“REST API apps use a request header to invoke negative testing in the sandbox. This header configures the sandbox into a negative testing state for transactions that include the merchant.”*

Thêm keyPayPal-Mock-Response vào request header khi gọi PayPal API.

Ở Server:

**CheckoutService.cs**

public async Task<Order> CaptureOrder(string *orderId*)

{

    var request = new OrdersCaptureRequest(orderId);

    request.Prefer("return=representation");

    request.Headers.Add("PayPal-Mock-Response", "{\"mock\_application\_codes\": \"DUPLICATE\_INVOICE\_ID\"}");

    request.RequestBody(new OrderActionRequest());

    // Call PayPal to capture an order

    var response = await PayPalClient.Client().Execute(request);

    return response.Result<Order>();

}

**CheckoutController.cs**

[HttpPost("api/checkout/orders/{id}/capture")]

public async Task<IActionResult> Capture([FromRoute] string *id*)

{

    try

    {

        var order = await checkoutService.CaptureOrder(id);

        return Ok(new

        {

            Success = true,

            Result = order

        });

    }

    catch (Exception exception)

    {

        return BadRequest(new

        {

            Success = false,

            Error = JsonConvert.DeserializeObject(exception.Message)

        });

    }

}

Ở Client:

<script>

    paypal.Buttons({

        onApprove: function (*data*, *actions*) {

         return fetch(`/my-server/api/checkout/orders/${data.orderID}/capture`, {

                method: 'post',

                headers: {

                    'content-type': 'application/json'

                }

            }).then(function (*res*) {

                return res.json();

            }).then(function (*details*) {

                console.log(details);

/\* If your buyer's funding source fails,

\* the Orders API returns an INSTRUMENT\_DECLINED error.

\* A funding source might fail for these reasons:

\* - The billing address associated with the financial instrument could not be

\* confirmed.

\* - The transaction exceeds the card limit.

\* - The card issuer denied the transaction.

\* \*/

            if (details.error && details.error.name === 'INSTRUMENT\_DECLINED') {

                    return actions.restart();

                }

            })

        },

/\* When a buyer cancels a payment, they typically return to the parent page.

\* You can instead use the onCancel function to show a cancellation page

\* or return to the shopping cart

\* \*/

        onCancel: function (*data*) {

            // Show a cancel page, or return to cart

        },

/\* If an error prevents buyer checkout,

\* show an error page using the onError callback

\* \*/

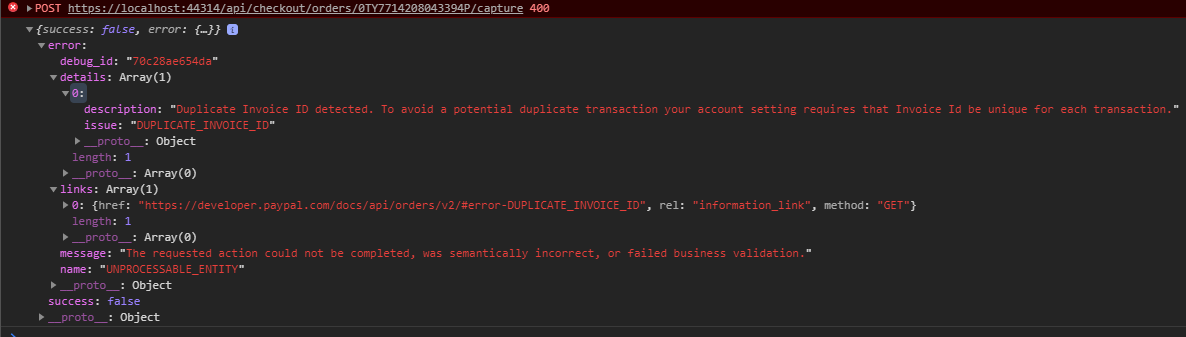
        onError: function (*err*) {

            // Show an error page here, when an error occurs

        }

    }).render('#paypal-button-container');

</script>



**Custom credit and debit card payments**

**Subscriptions**

With PayPal Subscriptions, you can bill customers for physical and digital goods or services at regular intervals.

**Integration features**

Use the Subscriptions API or Subscriptions Dashboard to:

* Create plans that charge users a fixed amount at regular intervals, or based on the quantity to which they subscribe.
* Offer your subscribers free or discounted trials to get more subscription signups.
* Easily modify the price of your plans.
* Accept currencies for over 250 countries.
* Enable subscribers to upgrade and downgrade their plans or change the quantity to which they subscribe.
* Automate payment recovery for failed payments.

<https://developer.paypal.com/docs/subscriptions>