**Hanoi University of Science and Technology**  
School of Information Communication and Technology

---------------o0o---------------



**Report: Web Information System**

**Topic: Google authentication and Online Payment**

**Professor:** **Dr. Phạm Huy Hoàng**

Student list:

|  |  |  |
| --- | --- | --- |
| **STT** | **Full Name** | **StudentID** |
| 1 | Nguyễn Trường Giang | 20194751 |
| 2 | Mạc Đình Phú | 20194817 |
| 3 | Phạm Hải Đăng | 20194736 |
| 4 | Đặng Phúc Hiếu | 20194759 |
| 5 | Lê Triệu Sáng | 20194829 |
| 6 | Phạm Ngọc Thắng | 20194840 |

INDEX

[I. Introduction 3](#_Toc109603210)

[**1. Reason for research** 3](#_Toc109603211)

[**2. Research purposes** 3](#_Toc109603212)

[**3. Researching Object** 3](#_Toc109603213)

[II. Google OAuth 3](#_Toc109603214)

[**1.** **What is OAuth?** 3](#_Toc109603215)

[**2.** **How does OAuth work?** 4](#_Toc109603216)

[**3. Google OAuth** 6](#_Toc109603217)

[**3.1 What is Google OAuth?** 6](#_Toc109603218)

[**3.2 Using OAuth 2.0 to Access Google APIs** 7](#_Toc109603219)

[III. PayPal Online Payment 9](#_Toc109603220)

[**1.** **What are online payments?** 9](#_Toc109603221)

[**2.** **What is a Digital Wallet and how does it work?** 9](#_Toc109603222)

[**2.1 What is a Digital Wallet?** 9](#_Toc109603223)

[**2.2** **How do Digital Wallets Work?** 9](#_Toc109603224)

[**3.** **PayPal Payment API** 10](#_Toc109603225)

[**3.1 What is PayPal?** 10](#_Toc109603226)

[**3.2** **How does PayPal work?** 10](#_Toc109603227)

[IV. Make a simple web using Google OAuth and PayPal Online Payment 14](#_Toc109603228)

[**1.** **Requirement of the simple web** 14](#_Toc109603229)

[**2.** **Web source code** 14](#_Toc109603230)

[**3.** **Web demonstration** 15](#_Toc109603231)

[V. Conclusion 15](#_Toc109603232)

# **I. Introduction**

## **1. Reason for research**

Nowadays, when we use an application or web on a smartphone or PC. It’s easy to log in to a user account in many ways. The traditional way is logging in with a username and password. Another method is using Third-party app authentication like Facebook, Google, Google... The reason for multiple log-in methods is to help user can reuse existing accounts. It saves the user time creating an account when using the app or web. Users don’t need to remember several accounts for many apps. For this method, we study Google Open Authentication (Google OAuth) in version Google Auth 2.0.

Online Shopping is one of the most services familiar to people. Online payment is one of the convenient ways for online shopping. To get more knowledge about online payment we consider a popular way is using

e-wallet PayPal.

## **2. Research purposes**

* How to add Google Auth API to app or web
* How to add Online Payment using PayPal e-wallet.

## **3. Researching Object**

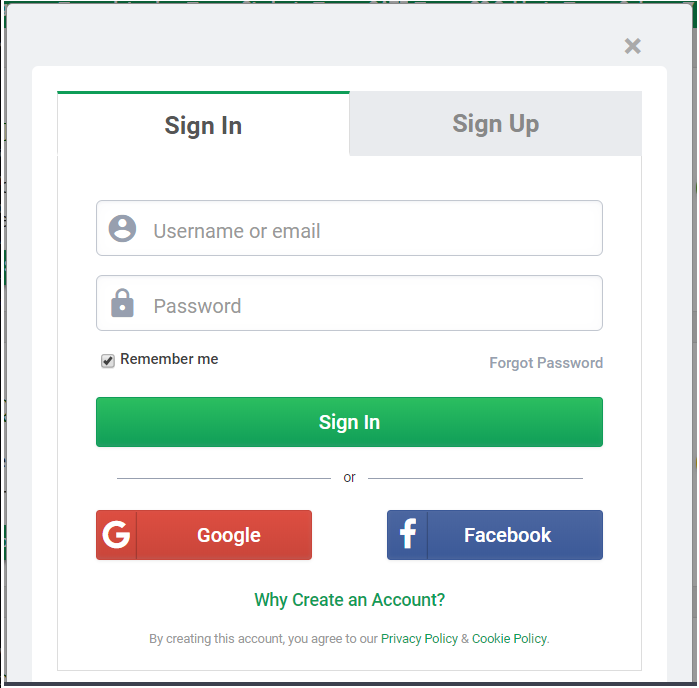
* Google Auth API
* PayPal Payment API

# **II. Google OAuth**

## **What is OAuth?**

OAuth (Open Authorization) is an open standard protocol for authorization of an application for using user information, in general, it allows a third-party application access to user related info like name, DOB, email, or other required data from an application like Facebook, Google etc. without giving the third party app the user password. It is pronounced as oh-auth.

You might have seen a “login with Google” or “login with Facebook” button on the login/signup page of a website that makes it easier to get using the service or website by simply logging into one of the services and granting the client application permission to access your data without giving Password. This is done with the OAuth.



( Log in with the OAuth method on the web)

## **How does OAuth work?**

There are 3 main players in an OAuth transaction: the user, the consumer, and the service provider. This triumvirate has been affectionately deemed the OAuth Love Triangle.

In our example, Joe is the user, Bitly is the consumer, and Google is the service provided that controls Joe’s secure resource (his Google stream). Joe would like Bitly to be able to post shortened links to his stream. Here’s how it works:

Step 1 – The User Shows Intent

Joe (User): “Hey, Bitly, I would like you to be able to post links directly to my Google stream.”

Bitly (Consumer): “Great! Let me go ask for permission.”

Step 2 – The Consumer Gets Permission

Bitly: “I have a user that would like me to post to his stream. Can I have a request token?”

Google (Service Provider): “Sure. Here’s a token and a secret.”

The secret is used to prevent request forgery. The consumer uses the secret to sign each request so that the service provider can verify it is coming from the consumer application.

Step 3 – The User Is Redirected to the Service Provider

Bitly: “OK, Joe. I’m sending you over to Google so you can approve. Take this token with you.”

Joe: “OK!”

- Bitly directs Joe to Google for authorization>

This is the scary part. If Bitly were super-shady Evil Co. it could pop up a window that looked like Google but was phishing for your username and password. Always be sure to verify that the URL you’re directed to is a vice provider (Google, in this case).

Step 4 – The User Gives Permission

Joe: “Google, I’d like to authorize this request token that Bitly gave me.”

Google: “OK, just to be sure, you want to authorize Bitly to do X, Y, and Z with your Google account?”

Joe: “Yes!”

Google: “OK, you can go back to Bitly and tell them they have permission to use their request token.”

Google marks the request token as “good-to-go,” so when the consumer requests access, it will be accepted (so long as it’s signed using their shared secret).

Step 5 – The Consumer Obtains an Access Token

Bitly: “Google, can I exchange this request token for an access token?”

Google: “Sure. Here’s your access token and secret.”

Step 6 – The Consumer Accesses the Protected Resource

Bitly: “I’d like to post this link to Joe’s stream. Here’s my access token!”

Google: “Done!”

In our scenario, Joe never had to share his Google credentials with Bitly. He simply delegated access securely using OAuth. At any time, Joe can log in to Google and review the access he has granted and revoke tokens for specific applications without affecting others. OAuth also allows for granular permission levels. You can give Bitly the right to post to your Google account, but restrict LinkedIn to read-only access.

## **3. Google OAuth**

### **3.1 What is Google OAuth?**

Google OAuth is Google's GSuite version of an OAuth 2 client. OAuth is a standard that apps can use to provide client applications with “secure delegated access”. OAuth works over HTTPS and authorizes devices, APIs, servers, and applications with access tokens rather than credentials.

OAuth allows for:

+ Different access levels: read-only VS read-write. This allows you to grant access to your user list or bi-directional access to automatically synchronize your new LinkedIn friends to your Gmail contacts.

+ Access granularity: you can decide to grant access to only your contact information (username, e-mail, date of birth, etc.) or your entire list of friends, calendar, and whatnot.

+ Access management: It allows you to manage access from the resource provider's application. If the third-party application does not provide a mechanism for canceling access, you would be stuck with them having access to your information. With OAuth, there is a provision for revoking access at any time.

### **3.2 Using OAuth 2.0 to Access Google APIs**

Google APIs use the OAuth 2.0 protocol for authentication and authorization. Google supports common OAuth 2.0 scenarios such as those for web server, client-side, installed, and limited-input device applications.

To begin, obtain OAuth 2.0 client credentials from the Google API Console. Then your client application requests an access token from the Google Authorization Server, extracts a token from the response, and sends the token to the Google API that you want to access. For an interactive demonstration of using OAuth 2.0 with Google (including the option to use your client credentials), experiment with the OAuth 2.0 Playground.

This page gives an overview of the OAuth 2.0 authorization scenarios that Google supports and provides links to more detailed content. For details about using OAuth 2.0 for authentication, see OpenID Connect.

All applications follow a basic pattern when accessing a Google API using OAuth 2.0. At a high level, you follow five steps:

1. Obtain OAuth 2.0 credentials from the Google API Console.

Visit the Google API Console to obtain OAuth 2.0 credentials such as a client ID and client secret that are known to both Google and your application. The set of values varies based on what type of application you are building. For example, a JavaScript application does not require a secret, but a web server application does.

2. Obtain an access token from the Google Authorization Server.

Before your application can access private data using a Google API, it must obtain an access token that grants access to that API. A single access token can grant varying degrees of access to multiple APIs. A variable parameter called scope controls the set of resources and operations that an access token permits. During the access-token request, your application sends one or more values in the scope parameter.

There are several ways to make this request, and they vary based on the type of application you are building. For example, a JavaScript application might request an access token using a browser redirect to Google, while an application installed on a device that has no browser uses web service requests.

Some requests require an authentication step where the user logs in with their Google account. After logging in, the user is asked whether they are willing to grant one or more permissions that your application is requesting. This process is called user consent.

If the user grants at least one permission, the Google Authorization Server sends your application an access token (or an authorization code that your application can use to obtain an access token) and a list of scopes of access granted by that token. If the user does not grant permission, the server returns an error.

It is generally a best practice to request scopes incrementally, at the time access is required, rather than upfront. For example, an app that wants to support saving an event to a calendar should not request Google Calendar access until the user presses the "Add to Calendar" button; see Incremental authorization.

3. Examine scopes of access granted by the user.

Compare the scopes included in the access token response to the scopes required to access features and functionality of your application depending upon access to a related Google API. Disable any features of your app unable to function without access to the related API.

The scope included in your request may not match the scope included in your response, even if the user granted all requested scopes. Refer to the documentation for each Google API for the scopes required for access. An API may map multiple scope string values to a single scope of access, returning the same scope string for all values allowed in the request. For example, the Google People API may return a scope of https://www.googleapis.com/auth/contacts when an app requests a user authorizes a scope of https://www.google.com/m8/feeds/; the Google People API method people.updateContact requires a granted scope of https://www.googleapis.com/auth/contacts.

4. Send the access token to an API.

After an application obtains an access token, it sends the token to a Google API in an HTTP Authorization request header. It is possible to send tokens as URI query-string parameters, but we don't recommend it, because URI parameters can end up in log files that are not completely secure. Also, it is good REST practice to avoid creating unnecessary URI parameter names.

Access tokens are valid only for the set of operations and resources described in the scope of the token request. For example, if an access token is issued for the Google Calendar API, it does not grant access to the Google Contacts API. You can, however, send that access token to the Google Calendar API multiple times for similar operations.

5. Refresh the access token, if necessary.

Access tokens have limited lifetimes. If your application needs access to a Google API beyond the lifetime of a single access token, it can obtain a refresh token. A refresh token allows your application to obtain new access tokens.

# **III. PayPal Online Payment**

1. **What are online payments?**

Online payments are payments that are initiated over the internet for goods or services purchased either online or offline. Common methods to facilitate this include:

+ Bank Debits via online mandate (often referred to as Direct Debit - which is the terminology we’ll use in this guide)

+ Bank transfers (also referred to as wire transfers)

+ Online credit or debit card transactions

+ Digital wallet payments (such as PayPal)

Payments can be one-off (e.g. e-commerce transactions like purchasing clothing) or recurring (e.g. subscriptions to services like Netflix or Spotify).

1. **What is a Digital Wallet and how does it work?**

**2.1 What is a Digital Wallet?**

A digital wallet is a software equivalent of your physical wallet that carries your money. It’s operable from your computer, smartphone, or mobile device, and eliminates the need to carry cash.

Once you load your credit card information onto the digital wallet app, you can keep paying for your purchases with it—both online and offline. Even when you shop in-store, Near-Field Communication (NFC) technology makes it possible for you to checkout using your digital wallet at the point of sale.

* 1. **How do Digital Wallets Work?**

A digital wallet emulates a real, physical wallet by holding currency and keeping it handy for payments. A mechanism authenticates access and keeps transactions secure.

Digital wallets ask users to enter their debit card or credit card numbers which are then encrypted to prevent misuse. After setting up the account, only the customer can access the wallet by entering their valid login credentials. This makes it device-independent.

Even if a user loses the device, no one else can access their wallet. On the other hand, an authorized user can access and use the wallet from any other device.

To make a payment, customers must keep their mobile device close to the card reader at a retailer’s POS. A contactless payments symbol is often visible at these counters which indicates that the store accepts digital wallet payments. Once the payment is authorized, the amount will be quickly transferred to the merchant.

1. **PayPal Payment API**

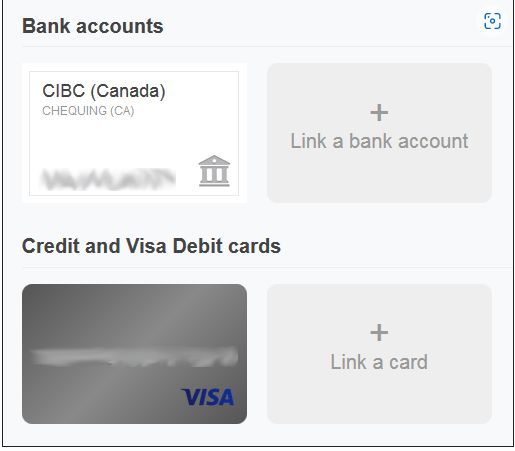
**3.1 What is PayPal?**

PayPal is a financial tool that lets you conduct transactions online without entering your financial details into every website you deal with. Link your credit card and/or bank account to PayPal, and then add to (or withdraw from) a secure money pool, shop at retailers who accept PayPal, or send money to other users.

* 1. **How does PayPal work?**

1. Pay from your credit card or bank account

When you sign up for PayPal, you can link your credit card account, your bank account, or both to your PayPal account. That way, when you pay for something using PayPal, you get to choose where the money comes from.



1. Create a secure pool of money

You can transfer money from your bank account to your PayPal account. That way, when you want to send money or pay for something with PayPal, you can just use the money in your PayPal account. You don’t have to involve your credit card or bank account details at all in the transaction.

Ảnh có chứa văn bản

Mô tả được tạo tự động

1. Draw money from your PayPal account when you need it

If you need to make a purchase that can’t be completed with PayPal, don’t sweat it! It allows you to easily transfer money back into your bank account from your semi-anonymous pool on PayPal if liquid cash would be more useful to you in a certain situation.

Ảnh có chứa văn bản

Mô tả được tạo tự động

1. Seamless online shopping through PayPal

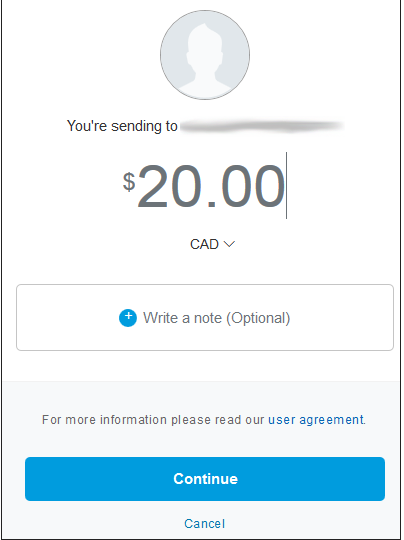
Millions of websites accept PayPal as payment, and shopping with PayPal is a snap on e-commerce websites like eBay! When you go to check out, simply select PayPal as your payment type, log into your account, and select where you want the money to come from your bank account, your credit card, or your balance on PayPal. No credit card details are required.

Ảnh có chứa văn bản

Mô tả được tạo tự động

1. Transfer money quickly and easily to other PayPal users

If your friends or family members use PayPal, too, then you can send them money when they need it with just a few quick clicks! Just type in their email address or phone number, choose how much money you want to give them (and in what currency), write them a note if you want, and select where the money’s going to come from — your credit card, your bank account, or your PayPal balance — and send your gift off.



* 1. **Integrate card payments**

Step 1: Enable your account to accept card payments

Before you can accept card payments on your website, verify your sandbox business account is enabled for advanced credit and debit cards. To verify:

1. Log in to the developer dashboard and go to your sandbox account.

2. Your Account Details > Manage accounts > View/edit accounts > Settings tab shows whether or not your account is enabled or disabled, for example:



Step 2: A client token is required to uniquely identify your buyer. This is required to use card fields.

The following request generates a client token that you'll use for the **data-client-token** property when you integrate the JavaScript SDK **<script>** tag into your HTML page in Step 3.

Step 3: Render JavaScript SDK and PayPal buttons

Add the PayPal JavaScript SDK to your web page and include your app's client ID and a dynamic client token for each request. Also include a **<div>** to render the PayPal buttons. In the included JavaScript file, there are reference routes on the server that you'll add in Step 4.

Step 4: Call the Orders API

To accept card payments directly on your website, create API endpoints on your server that communicate with the PayPal Orders V2 API. Those endpoints are responsible for creatingordersrrk and capturing the payment for an order.

Step 5: Add card form elements

Create a checkout form for your checkout page that includes the card field elements. Connect them to your server using the API endpoints created in Step 4.

Example source code: https://github.com/paypal-examples/docs-examples/tree/main/advanced-integration

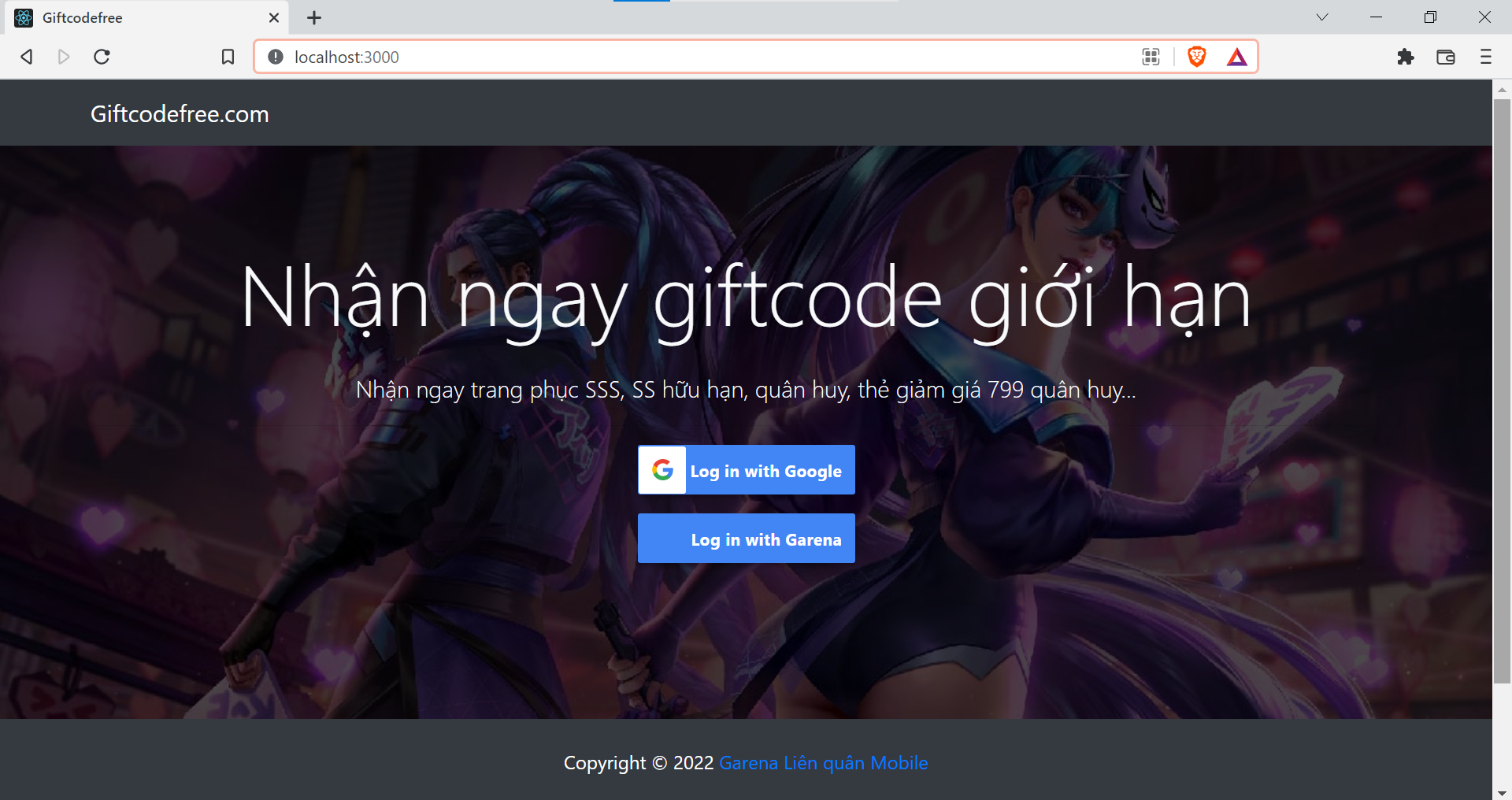
1. **Make a simple web using Google OAuth and PayPal Online Payment**
2. **Requirement of the simple web**

In this website, we provide to users giftcode from Lien quan Mobile – a famous game nowadays. To get giftcode, users need to login with their google account and pay $25 for this giftcode using paypal.

1. **Web source code**

**https://github.com/NeedAvailableName/GG\_Auth**

1. **Web demonstration**

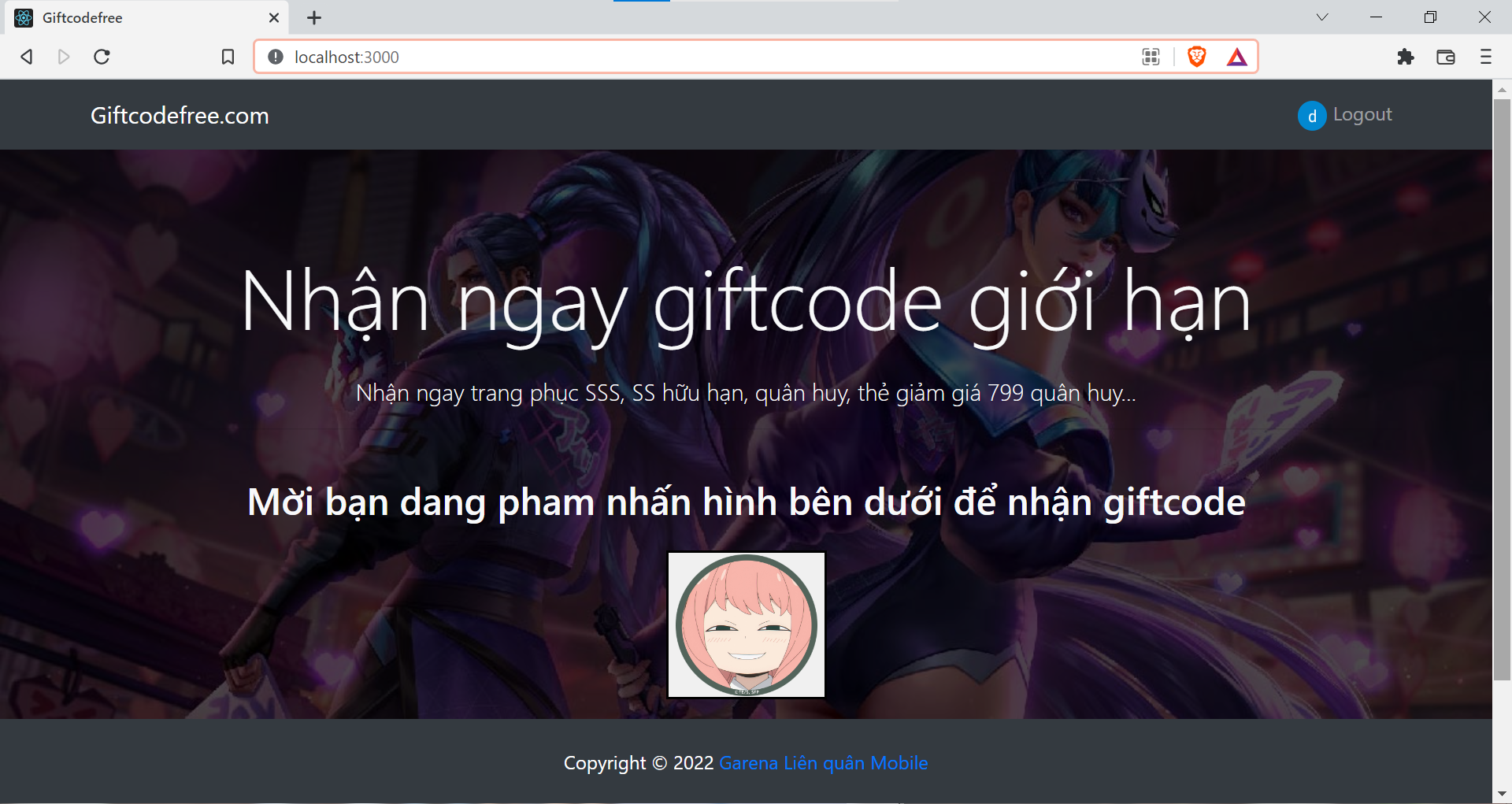
****

Website interface

Graphical user interface, application

Description automatically generated

After choosing login with google, server direct to google login interface



Choose one account to login and then user need to click below image to get giftcode

Graphical user interface, application

Description automatically generated

Users need to login their paypal account to pay for this giftcode

Graphical user interface, application

Description automatically generated

User can choose pay method such as using paypal balance, visa or credit card…

Graphical user interface, text, application

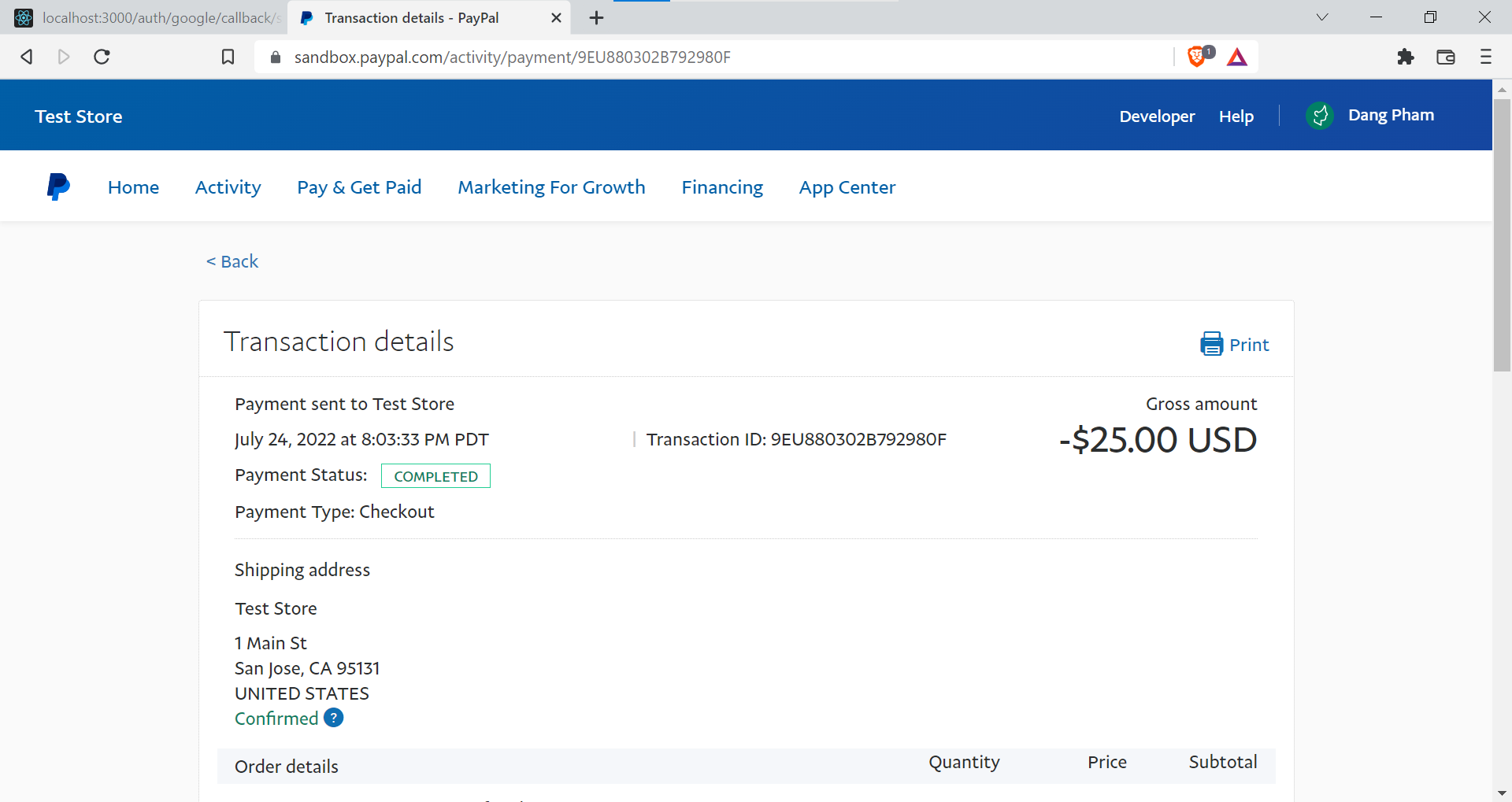
Description automatically generated

After paying successfully, server will return user giftcode

Text

Description automatically generated

In terminal, you can see the detail of the transaction



You can login your account to see the transaction

# **Conclusion**

In this project, our team worked together and get more knowledge about Open Authentication and Online Payment. Although try your best and have a simple product. Because of the knowledge limitation, some function in this product is not complete. But we got to experience using new techniques. In the feature, we can apply this to the next product and improve using API skills.