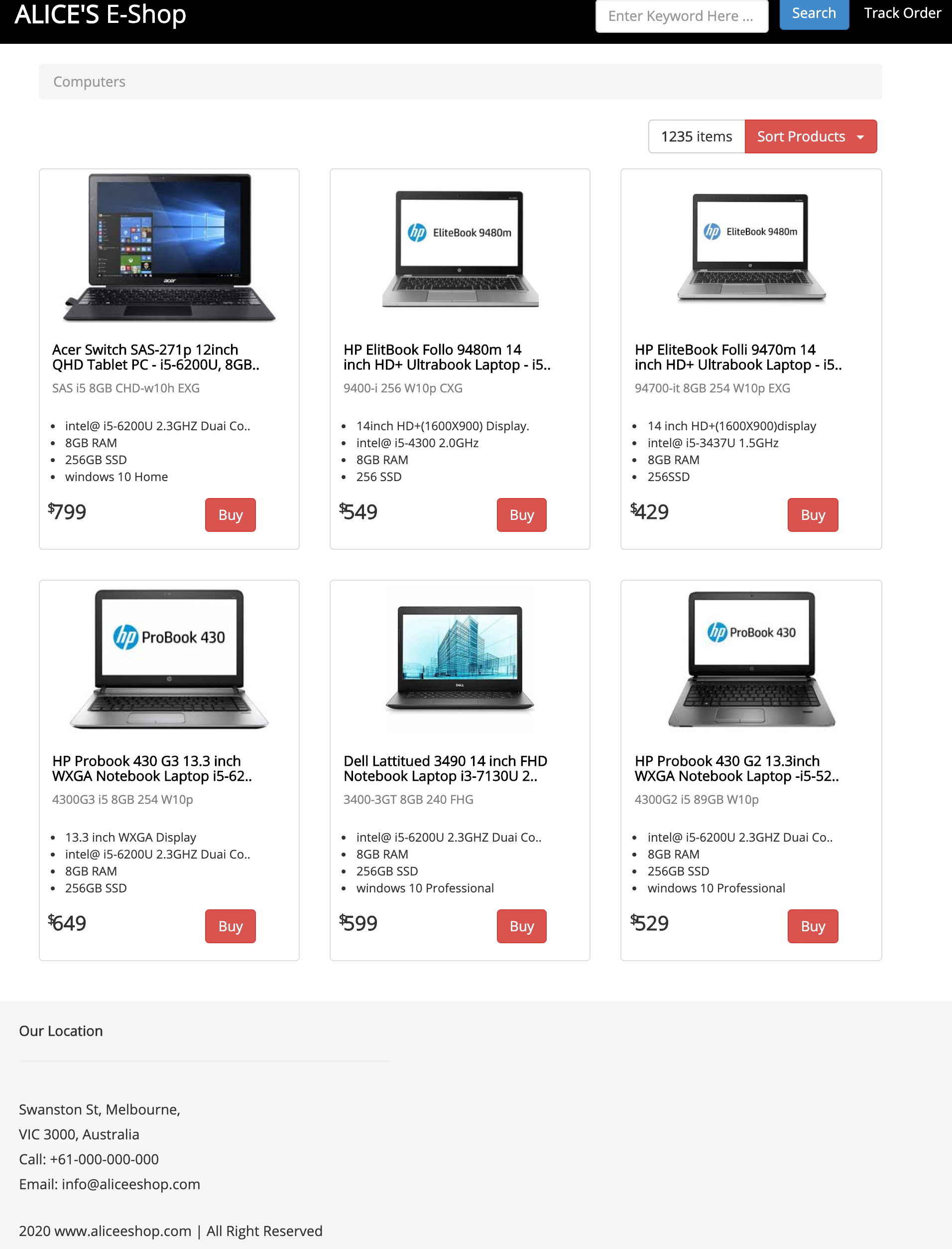
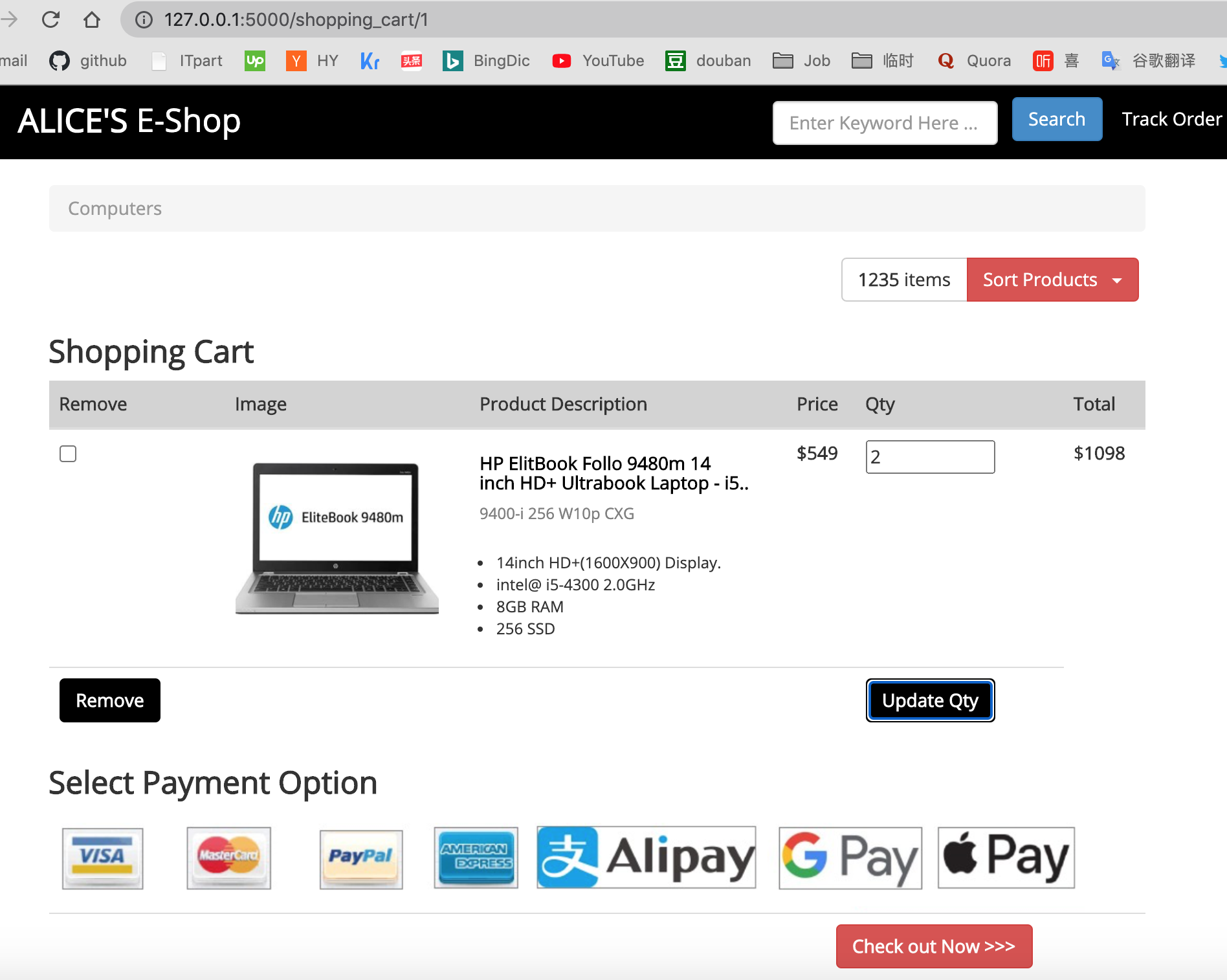
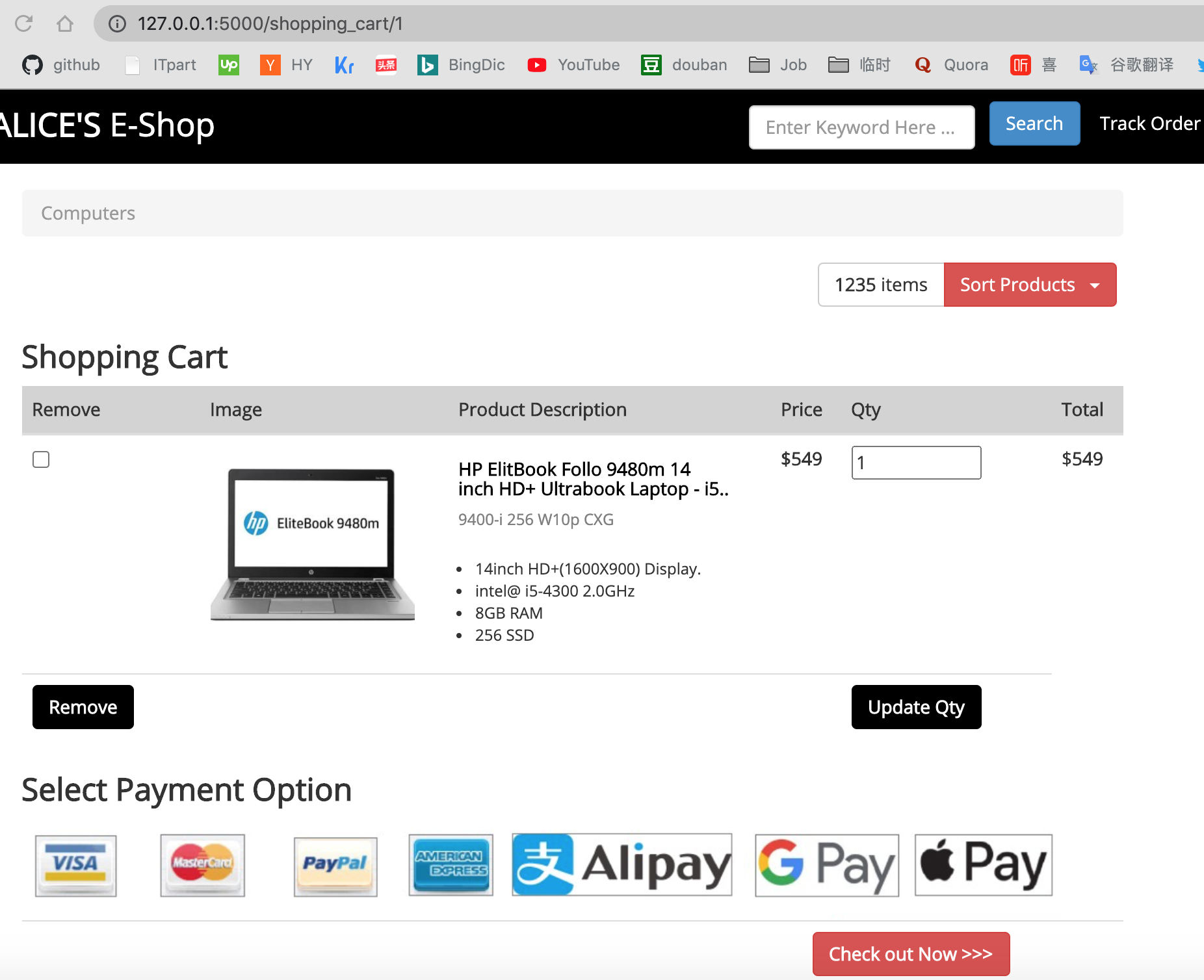
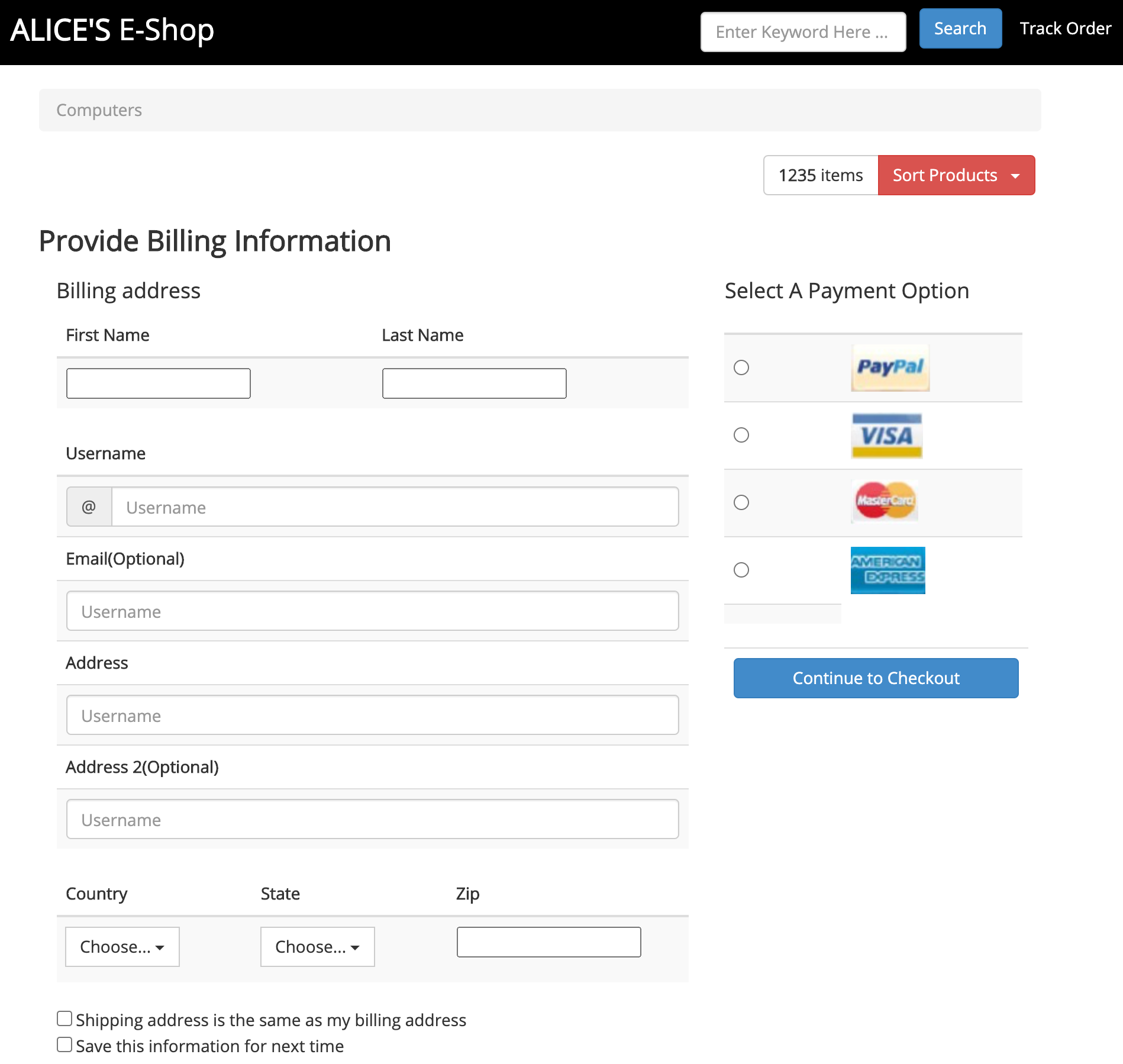
**Q1(a):**

All realted files in templates folder and static folder





**Q1 (b)**

Paypal , mastercard, visa, America\_express integrated by folowing way:

1. goto setting of soundbox.paypal , add all the payment (mastercard, visa, America\_express )options into pay options
2. and set one of card/bank to be defualt of paypal payment.
3. goto sdk of paypal , generate related javascript codes and css files.
4. Add related javascript and css files into my html template files:

<div id="smart-button-container">

<div style="text-align: center;">

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

</div>

</div>

<script src="https://www.paypal.com/sdk/js?client-id=sb&currency=AUD" data-sdk-integration-source="button-factory"></script>

<script>

function initPayPalButton() {

paypal.Buttons({

style: {

shape: 'rect',

color: 'gold',

layout: 'vertical',

label: 'paypal',

},

createOrder: function(data, actions) {

var total = $('#total').val();

return actions.order.create({

purchase\_units: [{"amount":{"currency\_code":"AUD","value":total}}]

});

},

onApprove: function(data, actions) {

return actions.order.capture().then(function(details) {

alert('Transaction completed by ' + details.payer.name.given\_name + '!');

});

},

onError: function(err) {

console.log(err);

}

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

}

initPayPalButton();

</script>

</div>

<!-- /.row -->

</div>

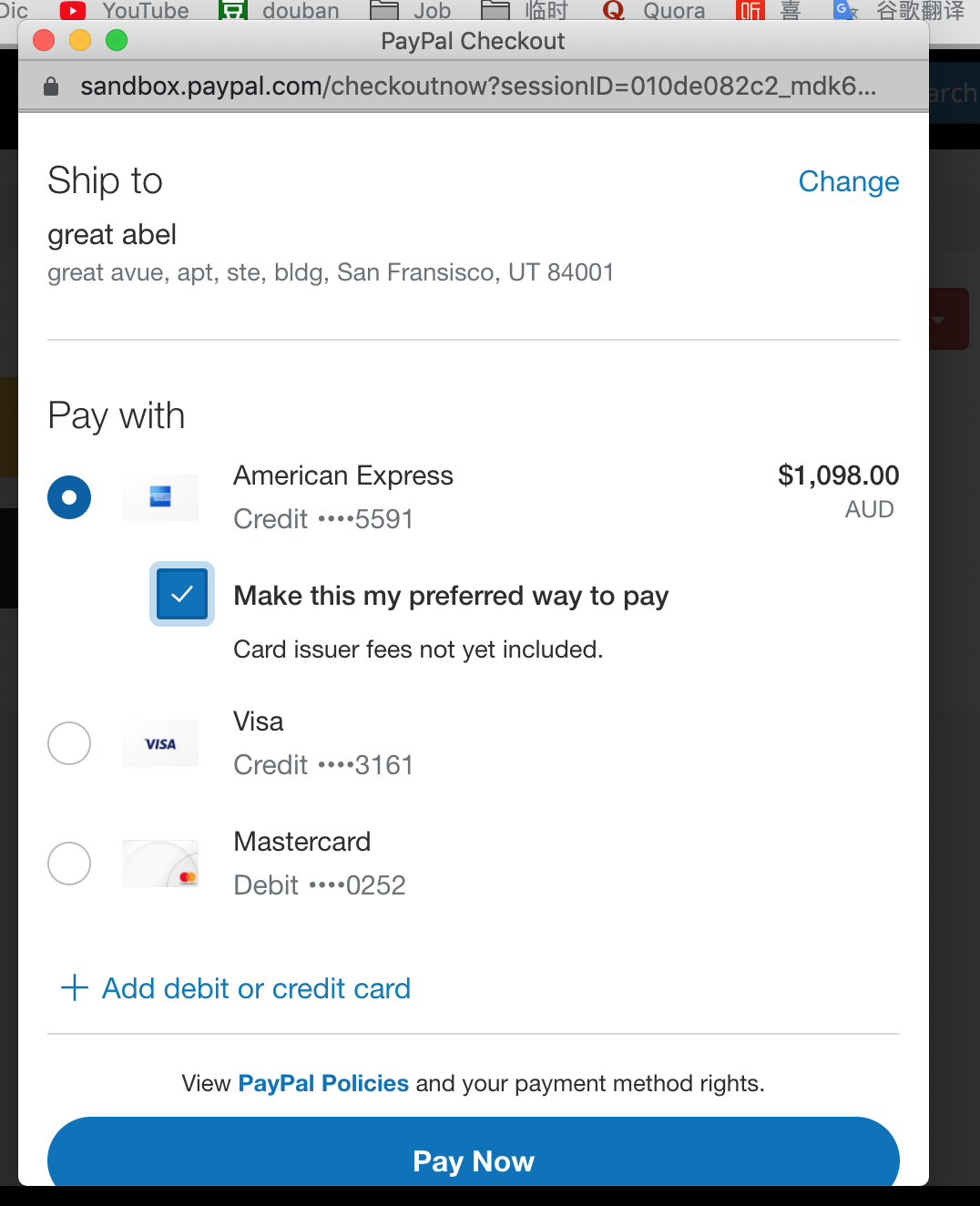
<!-- /.col -->

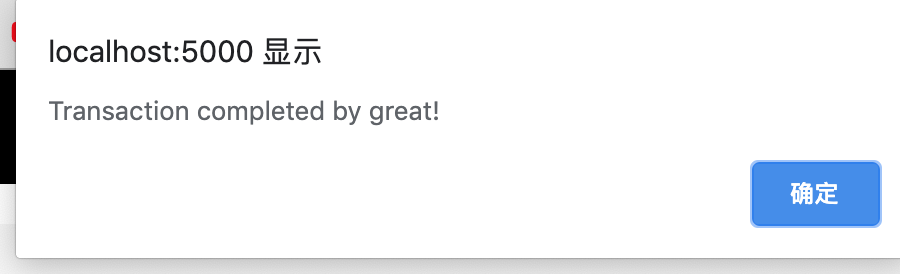
</div>

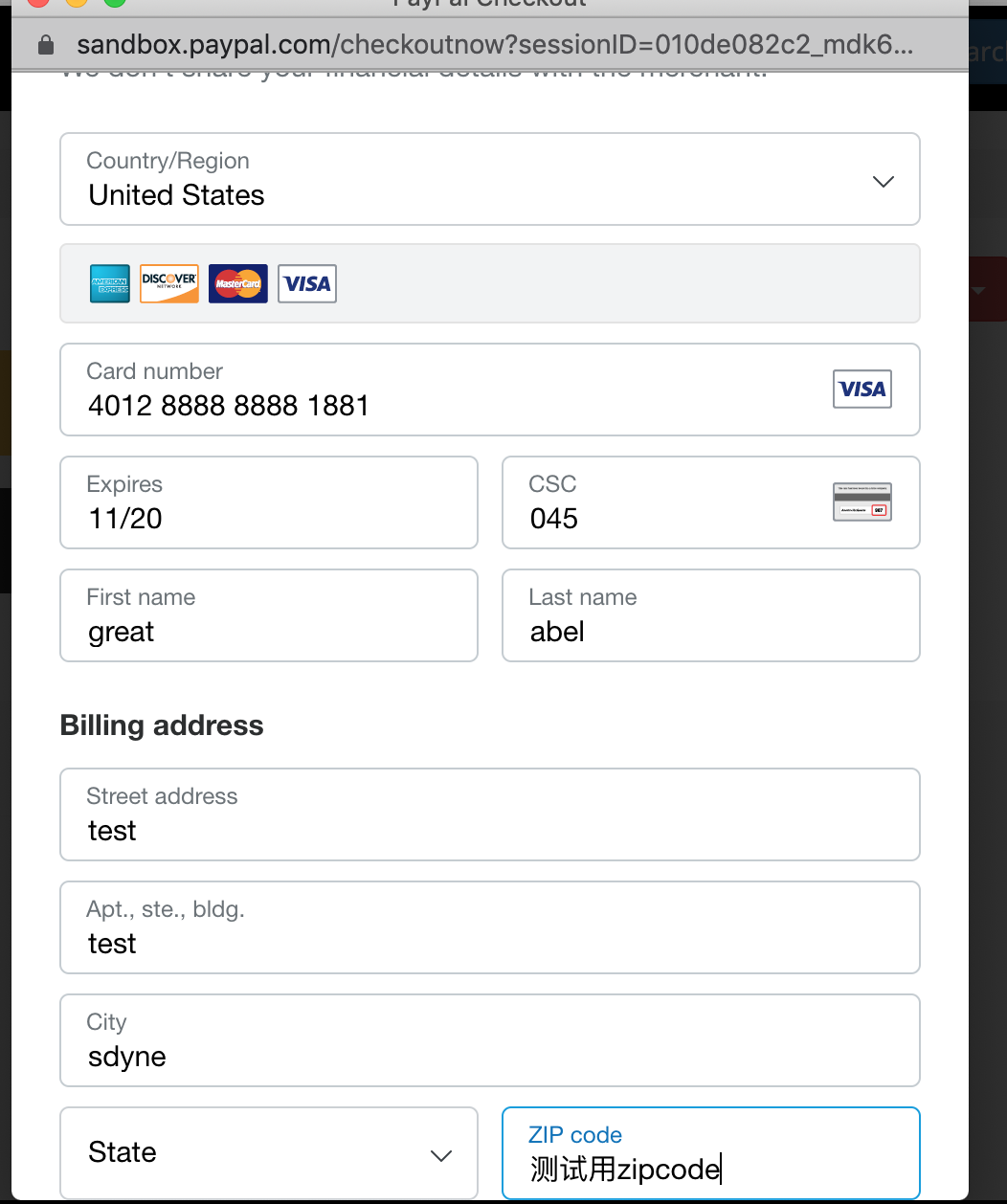
<!-- /.row -->

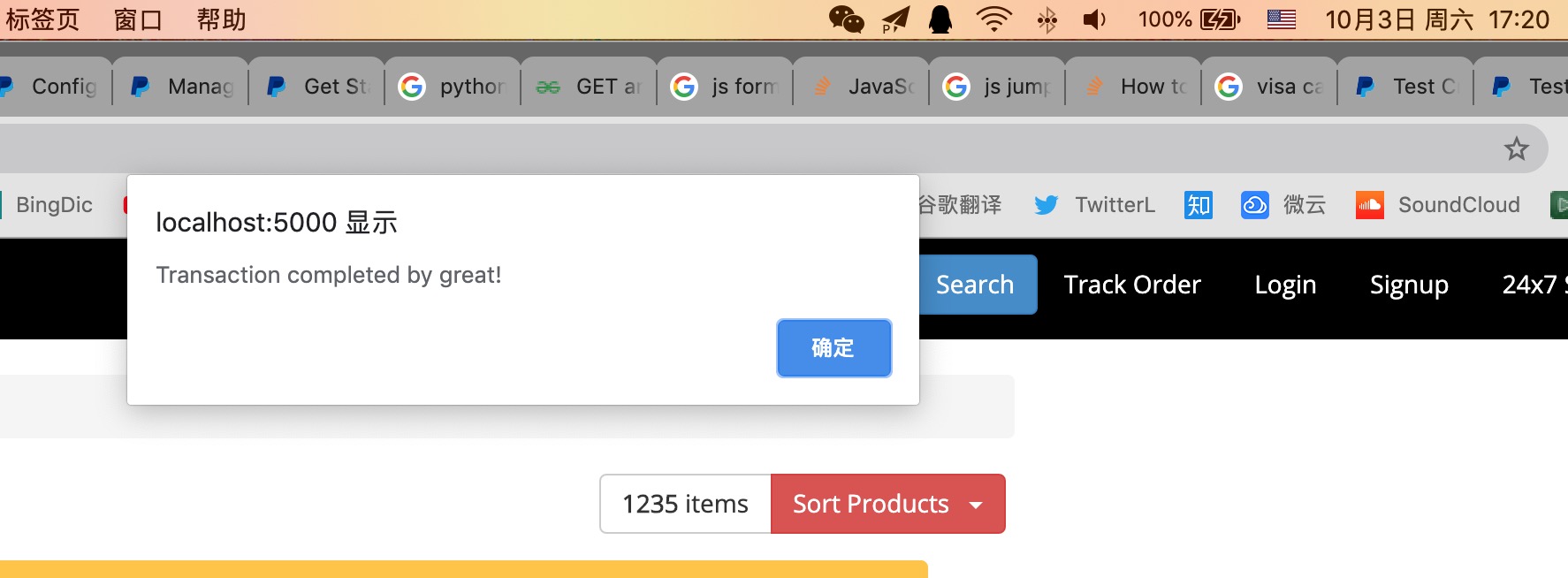
</div>

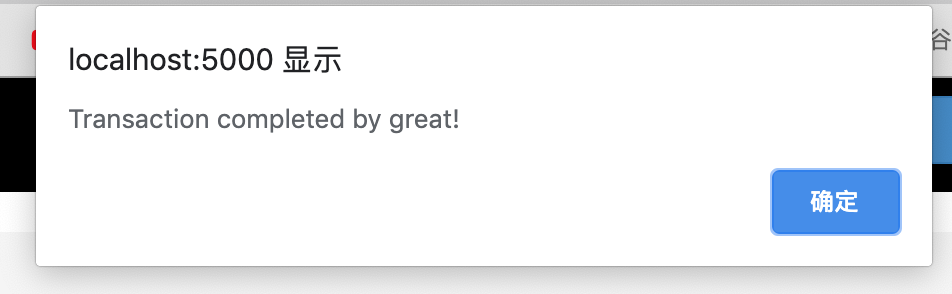
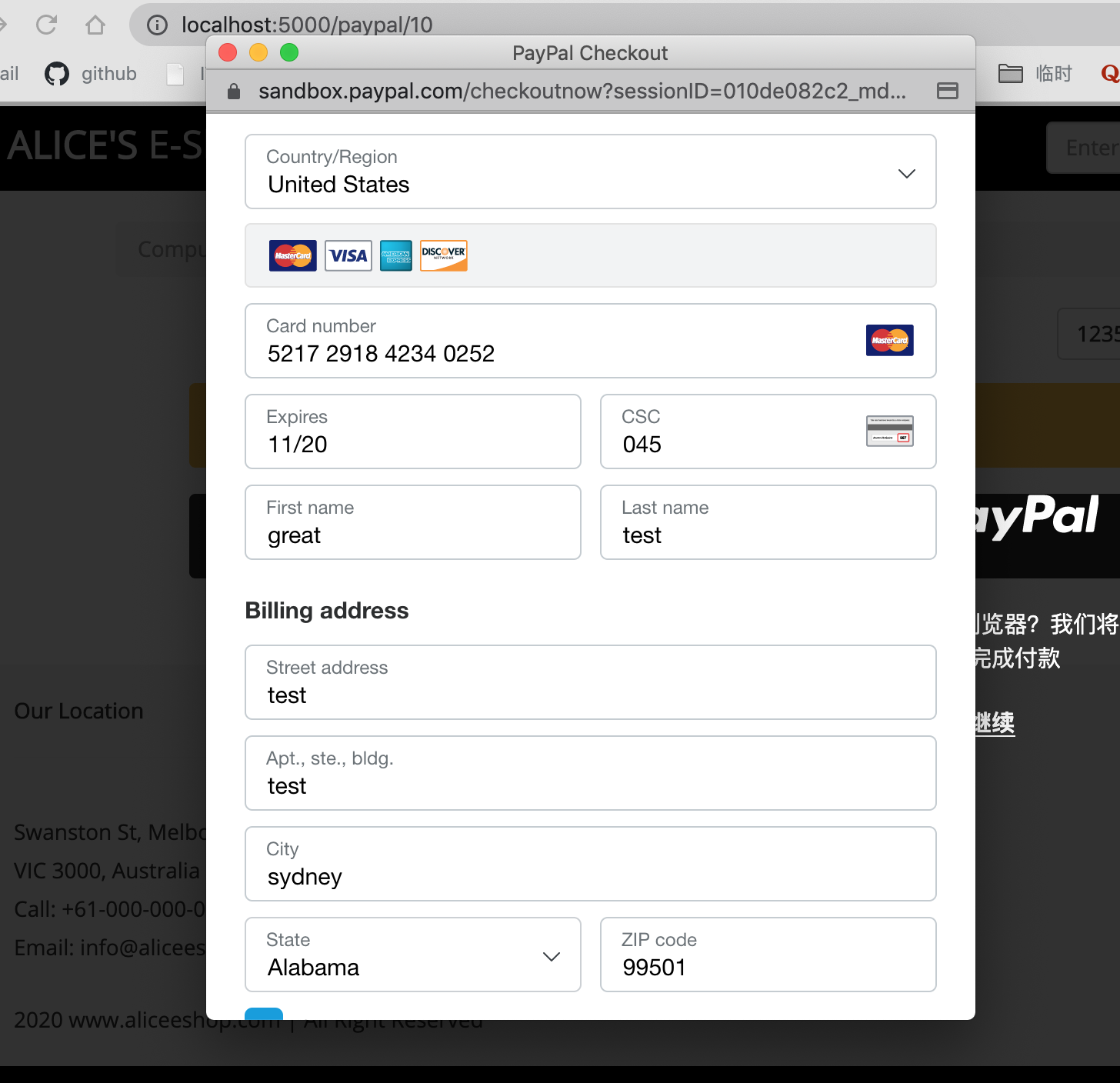
5. then we can pay with : papal , or other integrated 3 credit card:

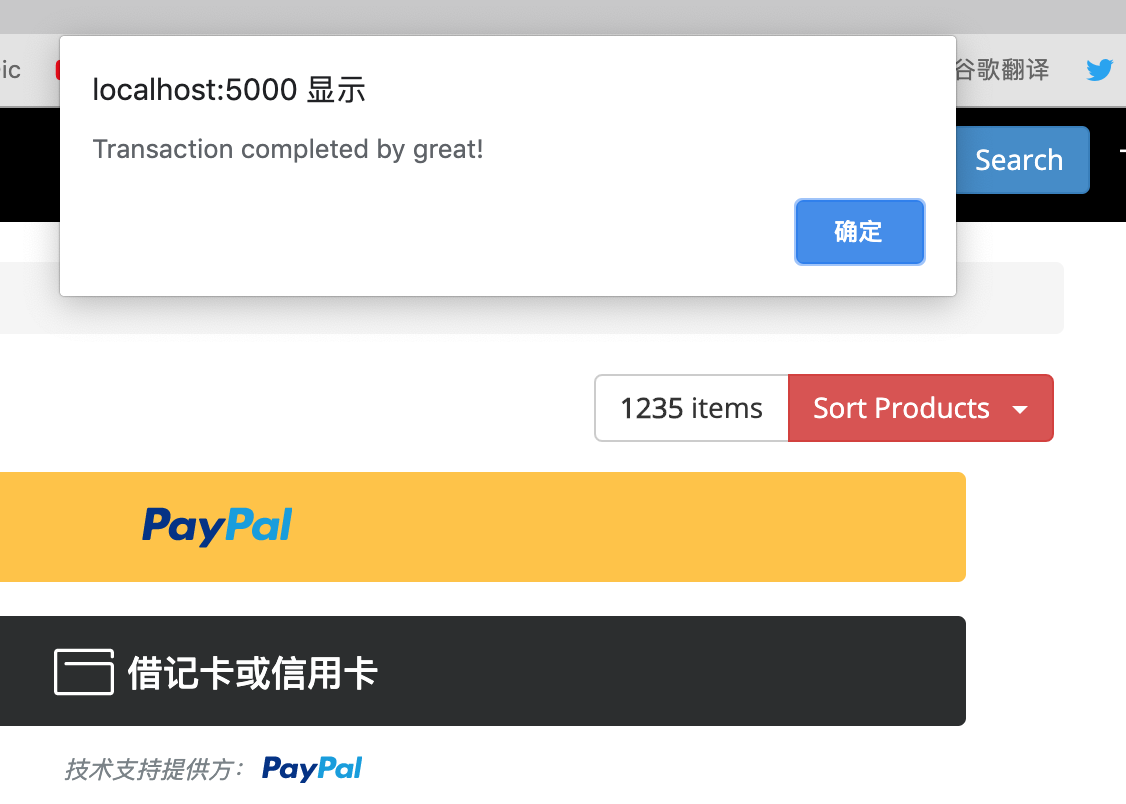
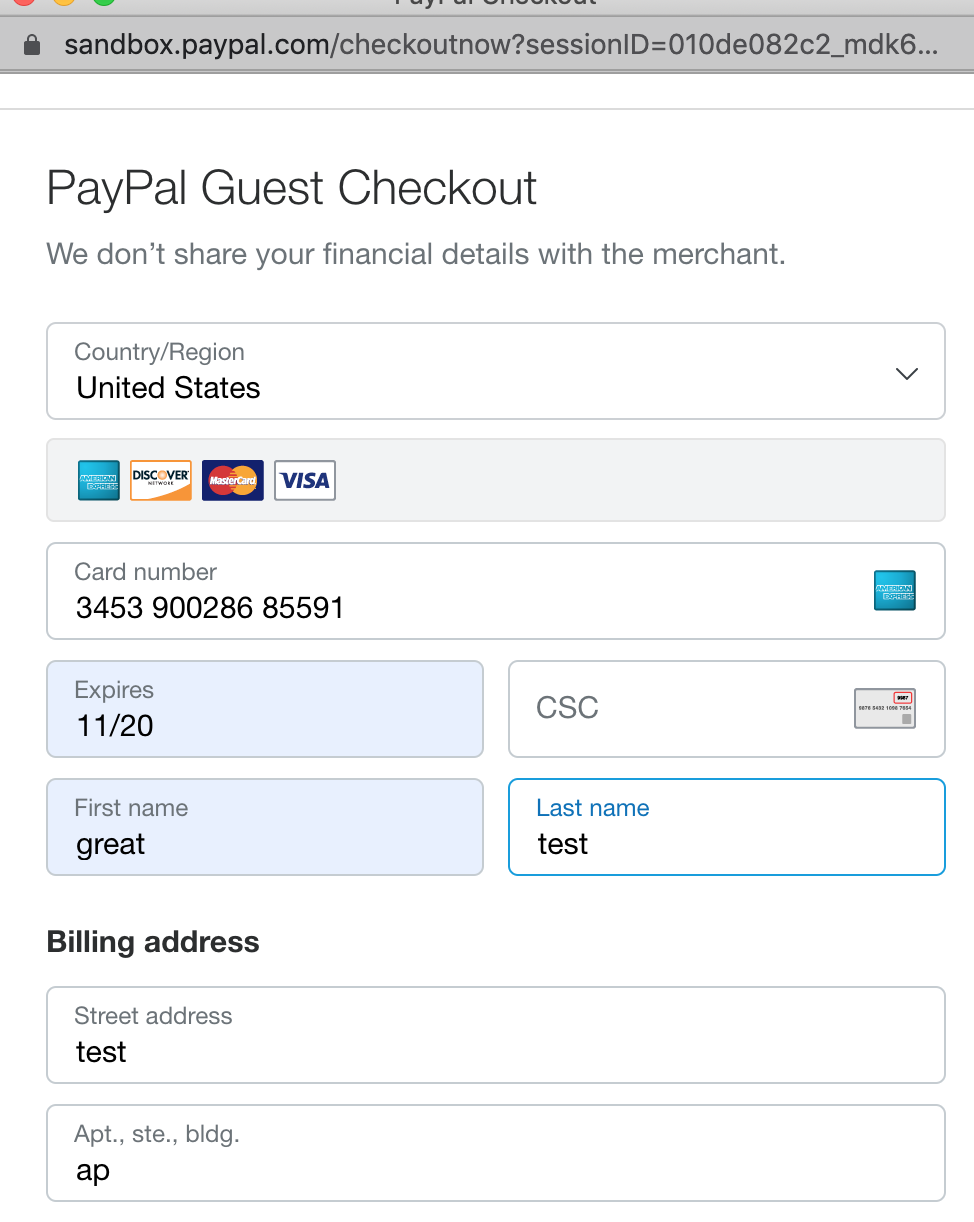












**Q2(a):**

steps:

import ast

import math

from Crypto.Random import random

from Crypto.PublicKey import DSA

from Crypto.Hash import SHA

# s1 = 'cf80cf5858c00cf654cd39644ec66873'

# s2 = '789041040361e1bf4dc3a120e1e1397'

# s3 = '674a611a52d39e462b05acd95f2aab1'

# i1 = int(s1, 16)

# i2 = int(s2, 16)

# i3 = int(s3, 16)

'''

# print(i1, i2, i3)

# print('n=', 151 \* 157)

# print(150 \* 156)

# print('e=', 20981)

# print('-'\*20)

# x = 0

# y = 0

# 20981\*x + 23400\*y = 1

# x = (1 - 23400\*y)/20981

for i in range(-100000, 100000):

x = (1 - 23400\*i)/20981

if x.is\_integer():

print('possilbe private d in consideration:', x, 'i=', i)

print('-'\*20)

# 97421.0 i= -87350

# 74021.0 i= -66369

# 50621.0 i= -45388

# 27221.0 i= -24407

# 3821.0 i= -3426

for i in range(-100000, 100000):

x = (1 - 23400\*i)/20983

if x.is\_integer():

print('possilbe private d in consideration:', x, 'i=', i)

print('-'\*20)

for i in range(-100000, 100000):

x = (1 - 23400\*i)/21067

if x.is\_integer():

print('possilbe private d in consideration:', x, 'i=', i)

print('-end of rsa(n,e,d) design consideration.\n')

print('#'\*20)

# print('d=', 3821)

# message = 88

# private\_d = 23

# common\_n = 187

# public\_e = 7

# private\_d = 2753

# common\_n = 3233

# public\_e = 17

message = 10000

print('message is:', message)

# private\_d = 50621

common\_n = 23707

public\_e = 20981

true\_count = 0

for name, private\_d, public\_e in \

[('Alice', 50621 , 20981), ('Bob',12247, 20983), ('Karen',1003, 21067)]:

print('At client side[' + name + '], encrypt with private\_d:',private\_d)

print('\nencrypt process is: pow(message, private\_d) % common\_n')

s0 = pow(message, private\_d) % common\_n

print('encrypt output=', s0)

print('\nAt bank side, decrypt with common\_n, public\_e:', common\_n, public\_e)

print('decrypt process is: pow(s0, public\_e) % common\_n')

t0 = pow(s0, public\_e) % common\_n

print('decrypt message=', t0)

if t0 == message:

print('The bank believes this signature is authentic\n')

true\_count+= 1

else:

print('The signature is not passed!')

if true\_count == 3:

print("All 3 signature is authentic, now the pay check can be paid!")

else:

print('BanK: you need all 3 signature authentic')

output is:

python3 Q2\_a.py

possilbe private d in consideration: 97421.0 i= -87350

possilbe private d in consideration: 74021.0 i= -66369

possilbe private d in consideration: 50621.0 i= -45388

possilbe private d in consideration: 27221.0 i= -24407

possilbe private d in consideration: 3821.0 i= -3426

possilbe private d in consideration: -19579.0 i= 17555

possilbe private d in consideration: -42979.0 i= 38536

possilbe private d in consideration: -66379.0 i= 59517

possilbe private d in consideration: -89779.0 i= 80498

--------------------

possilbe private d in consideration: 105847.0 i= -94914

possilbe private d in consideration: 82447.0 i= -73931

possilbe private d in consideration: 59047.0 i= -52948

possilbe private d in consideration: 35647.0 i= -31965

possilbe private d in consideration: 12247.0 i= -10982

possilbe private d in consideration: -11153.0 i= 10001

possilbe private d in consideration: -34553.0 i= 30984

possilbe private d in consideration: -57953.0 i= 51967

possilbe private d in consideration: -81353.0 i= 72950

possilbe private d in consideration: -104753.0 i= 93933

--------------------

possilbe private d in consideration: 94603.0 i= -85171

possilbe private d in consideration: 71203.0 i= -64104

possilbe private d in consideration: 47803.0 i= -43037

possilbe private d in consideration: 24403.0 i= -21970

possilbe private d in consideration: 1003.0 i= -903

possilbe private d in consideration: -22397.0 i= 20164

possilbe private d in consideration: -45797.0 i= 41231

possilbe private d in consideration: -69197.0 i= 62298

possilbe private d in consideration: -92597.0 i= 83365

-end of rsa(n,e,d) design consideration.

####################

message is: 10000

At client side[Alice], encrypt with private\_d: 50621

encrypt process is: pow(message, private\_d) % common\_n

encrypt output= 23200

At bank side, decrypt with common\_n, public\_e: 23707 20981

decrypt process is: pow(s0, public\_e) % common\_n

decrypt message= 10000

The bank believes this signature is authentic

At client side[Bob], encrypt with private\_d: 12247

encrypt process is: pow(message, private\_d) % common\_n

encrypt output= 4819

At bank side, decrypt with common\_n, public\_e: 23707 20983

decrypt process is: pow(s0, public\_e) % common\_n

decrypt message= 10000

The bank believes this signature is authentic

At client side[Karen], encrypt with private\_d: 1003

encrypt process is: pow(message, private\_d) % common\_n

encrypt output= 16206

At bank side, decrypt with common\_n, public\_e: 23707 21067

decrypt process is: pow(s0, public\_e) % common\_n

decrypt message= 10000

The bank believes this signature is authentic

All 3 signature is authentic, now the pay check can be paid!

abeltekiMacBook-Pro:b5793pay\_integration abel$

**Q2(b):**

steps:

import ast

import math

from Crypto.Random import random

from Crypto.PublicKey import DSA

from Crypto.Hash import SHA

# s1 = 'cf80cf5858c00cf654cd39644ec66873'

# s2 = '789041040361e1bf4dc3a120e1e1397'

# s3 = '674a611a52d39e462b05acd95f2aab1'

# i1 = int(s1, 16)

# i2 = int(s2, 16)

# i3 = int(s3, 16)

# print(i1, i2, i3)

# print('n=', 151 \* 157)

# print(150 \* 156)

# print('e=', 20981)

# print('-'\*20)

# x = 0

# y = 0

# 20981\*x + 23400\*y = 1

# x = (1 - 23400\*y)/20981

for i in range(-100000, 100000):

x = (1 - 23400\*i)/20981

if x.is\_integer():

print('possilbe private d in consideration:', x, 'i=', i)

print('-'\*20)

# 97421.0 i= -87350

# 74021.0 i= -66369

# 50621.0 i= -45388

# 27221.0 i= -24407

# 3821.0 i= -3426

# print('d=', 3821)

# message = 88

# private\_d = 23

# common\_n = 187

# public\_e = 7

# private\_d = 2753

# common\_n = 3233

# public\_e = 17

message = 10000

print('message is:', message)

# private\_d = 50621

common\_n = 23707

public\_e = 20981

print('Q2\_b 3 client shere public\_e, common\_n, so if Alice, Bob Karen first check\

whether their 3 signatures are same, if not re-sign it, if is the same, then\

send the same output-signature to bank, bank only have to check the comon-signature')

signatures = []

for name, private\_d in [('Alice', 50621), ('Bob',27221), ('Karen',3821)]:

print('At client side[' + name + '], encrypt with private\_d:',private\_d)

print('\nencrypt process is: pow(message, private\_d) % common\_n')

s = pow(message, private\_d) % common\_n

print('encrypt output=', s)

signatures.append(s)

if signatures[0] == signatures[1] == signatures[2]:

print('Alice, Bob Karen send multi-signature-in-one to book', signatures[0])

print('At bank side, decrypt with public\_e:', public\_e)

print('\ndecrypt process is: pow(s0, public\_e) % common\_n')

t0 = pow(signatures[0], public\_e) % common\_n

print('decrypt message=', t0)

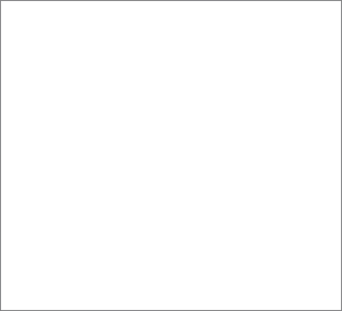
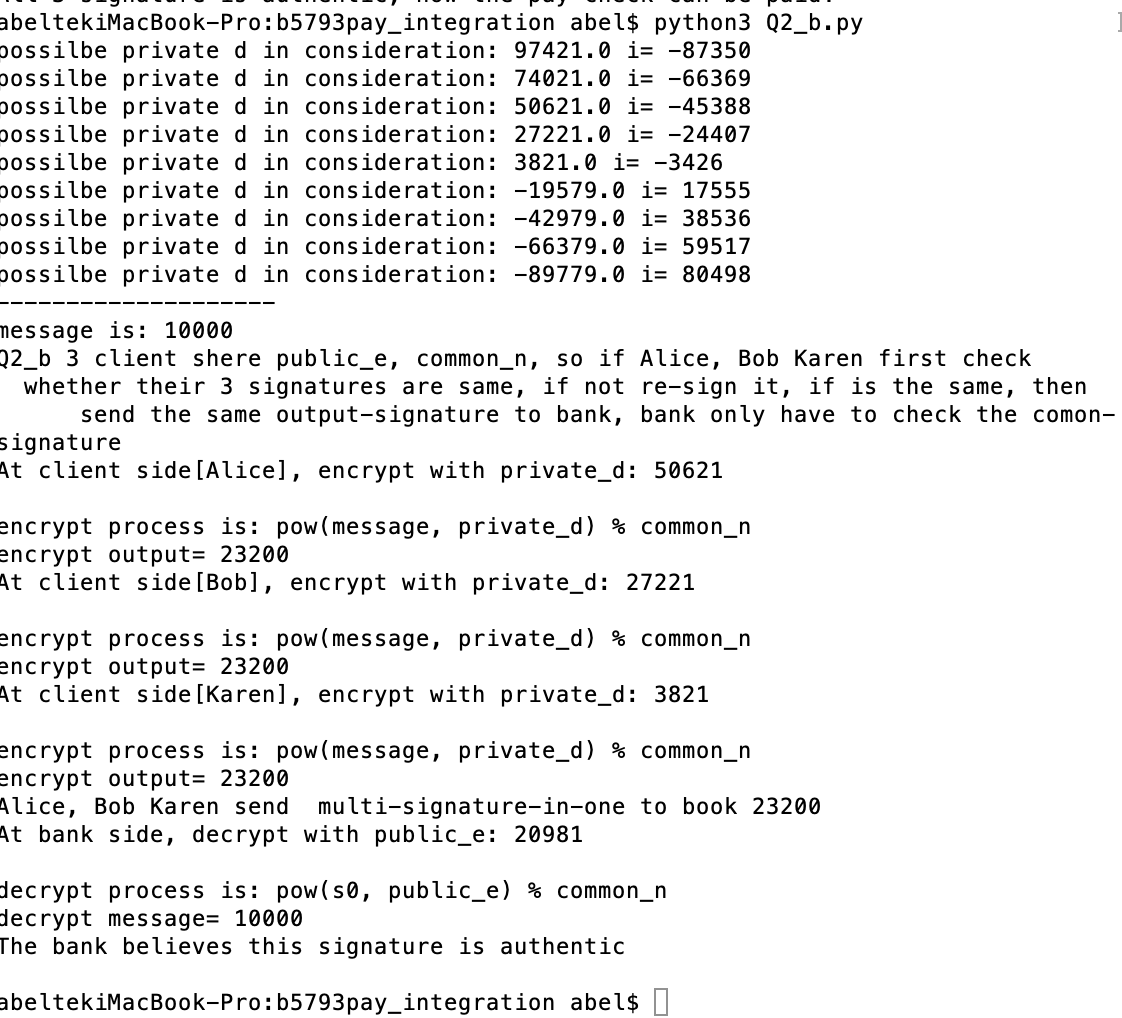
if t0 == message:

print('The bank believes this signature is authentic\n')

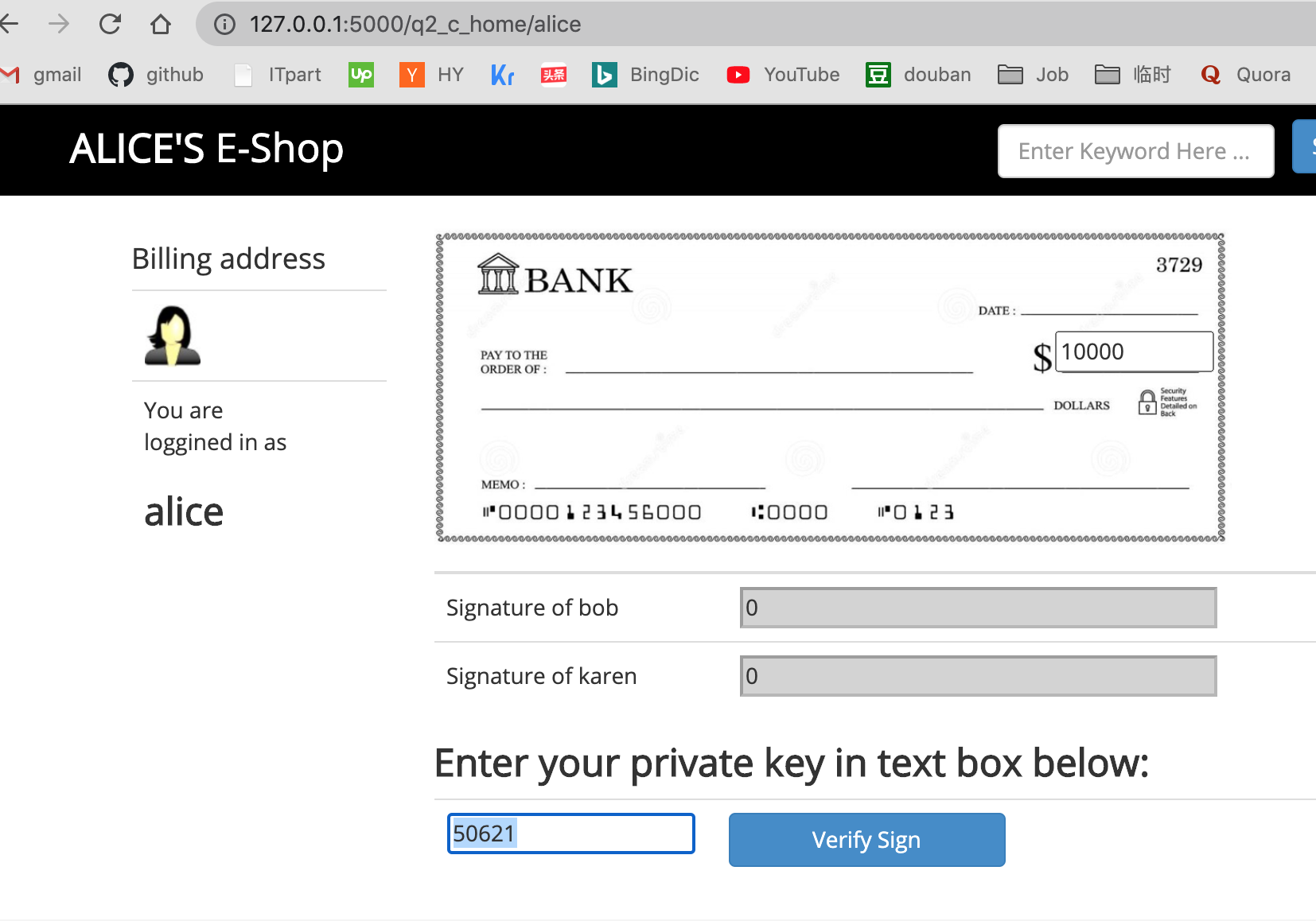
else:

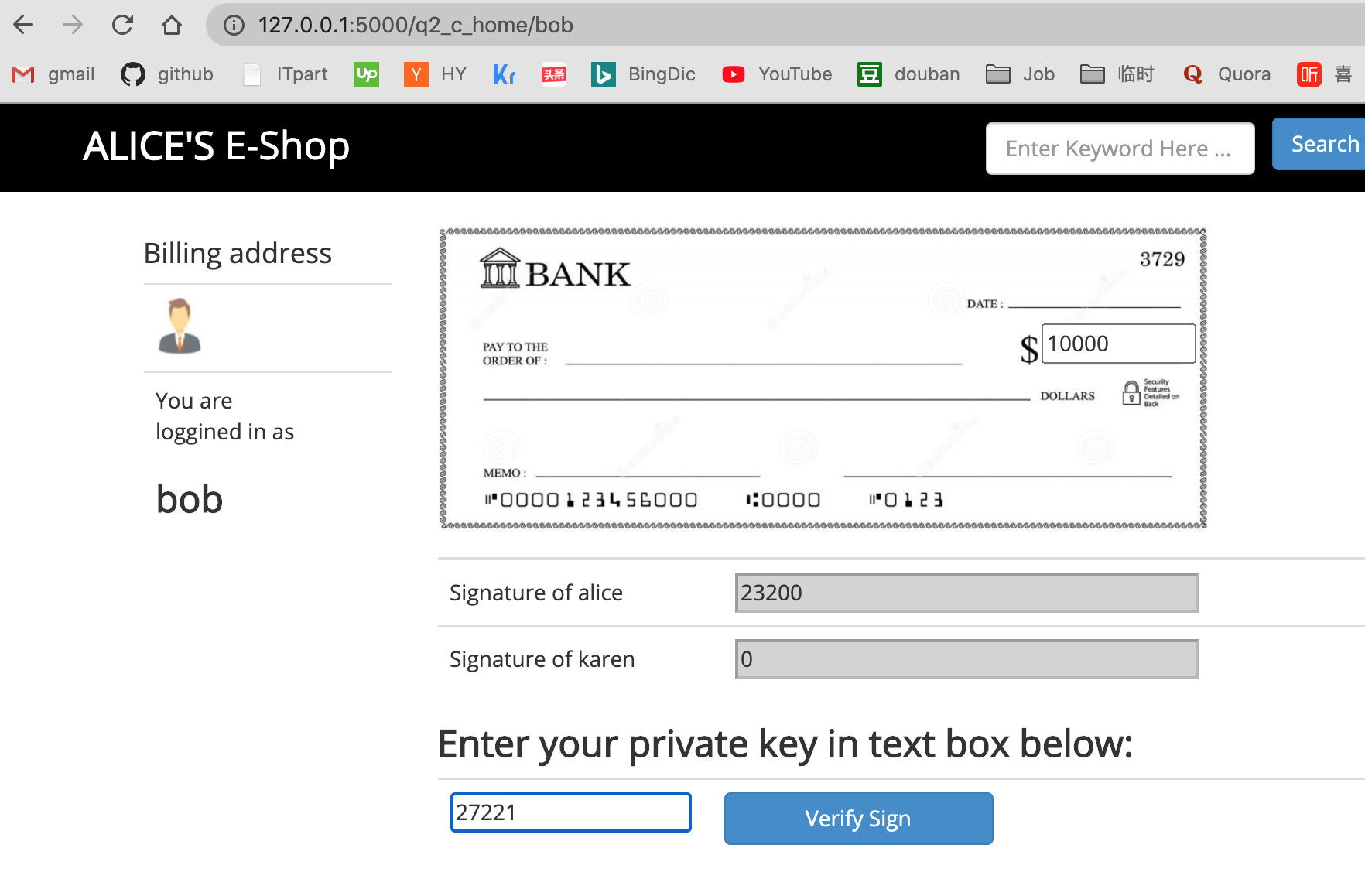
print('The signature is not passed!')

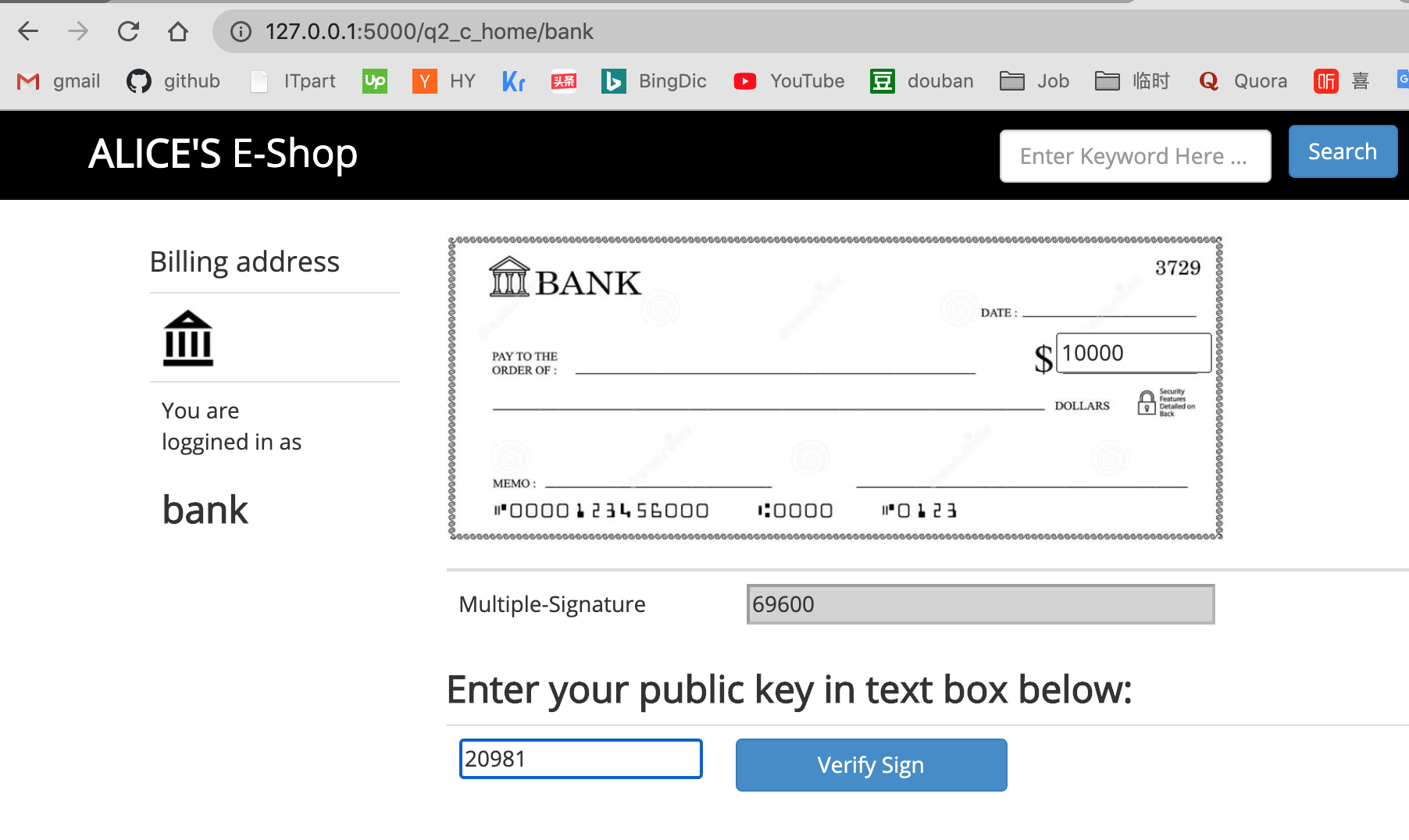
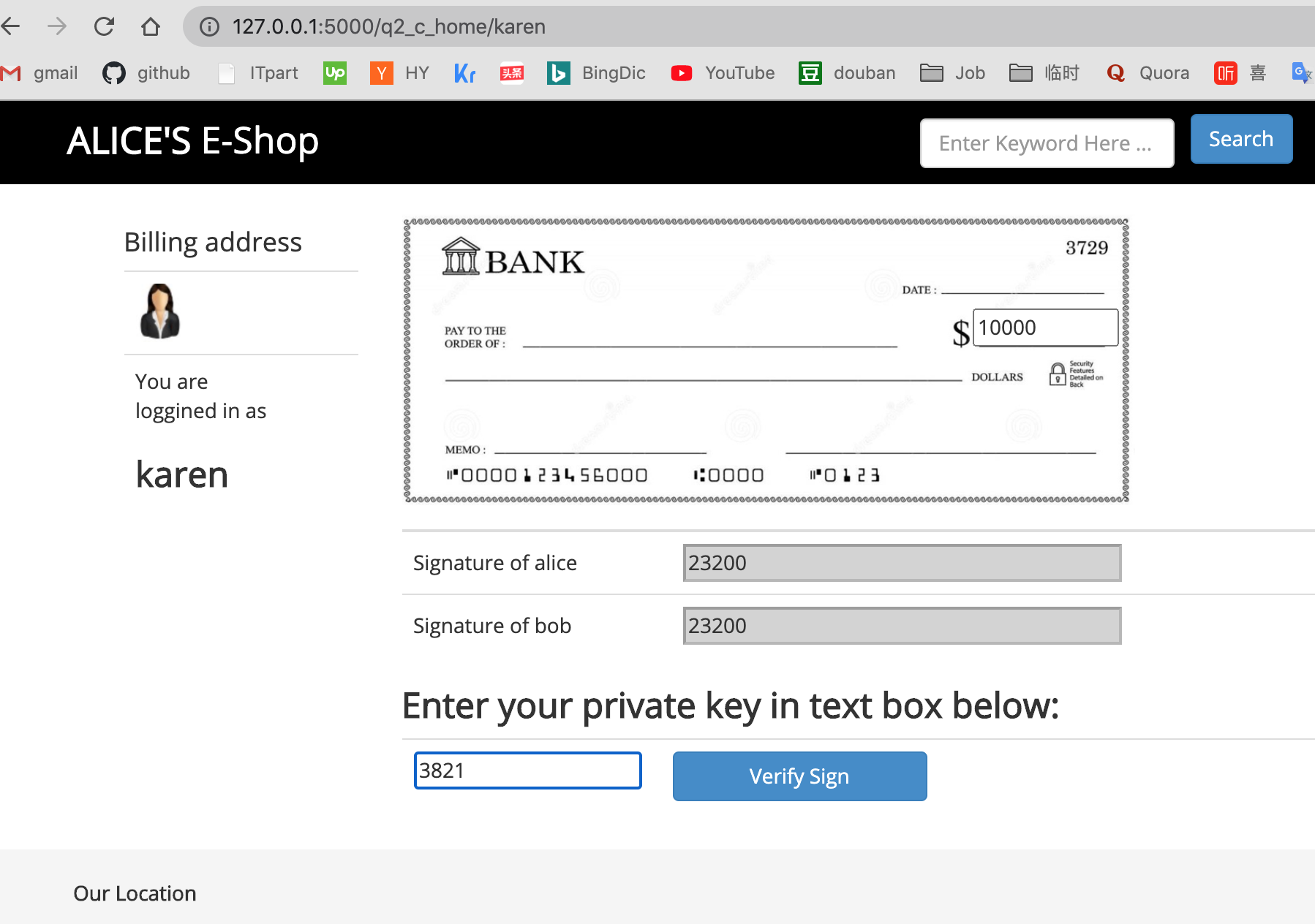
outputs:

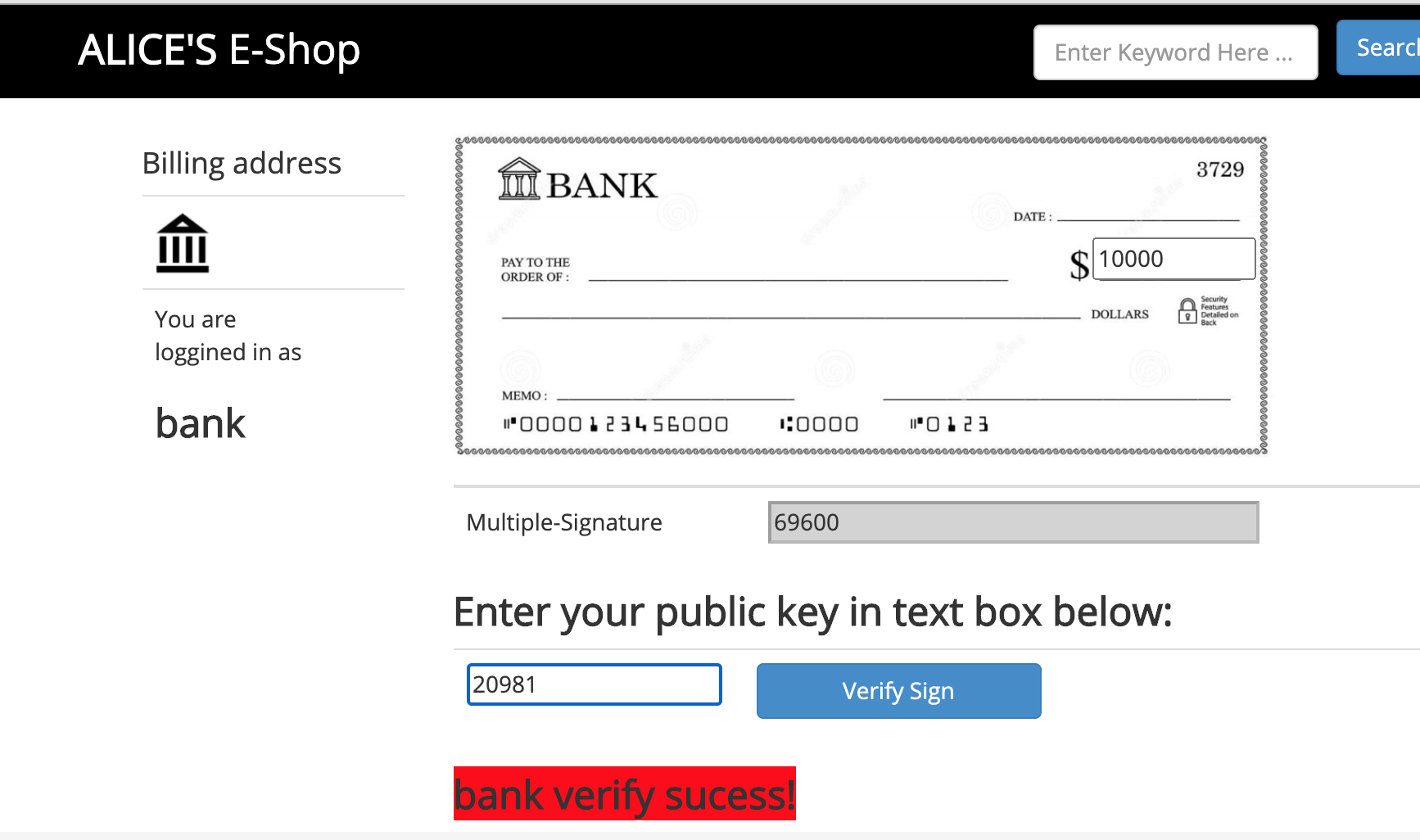


**Q2(c):**

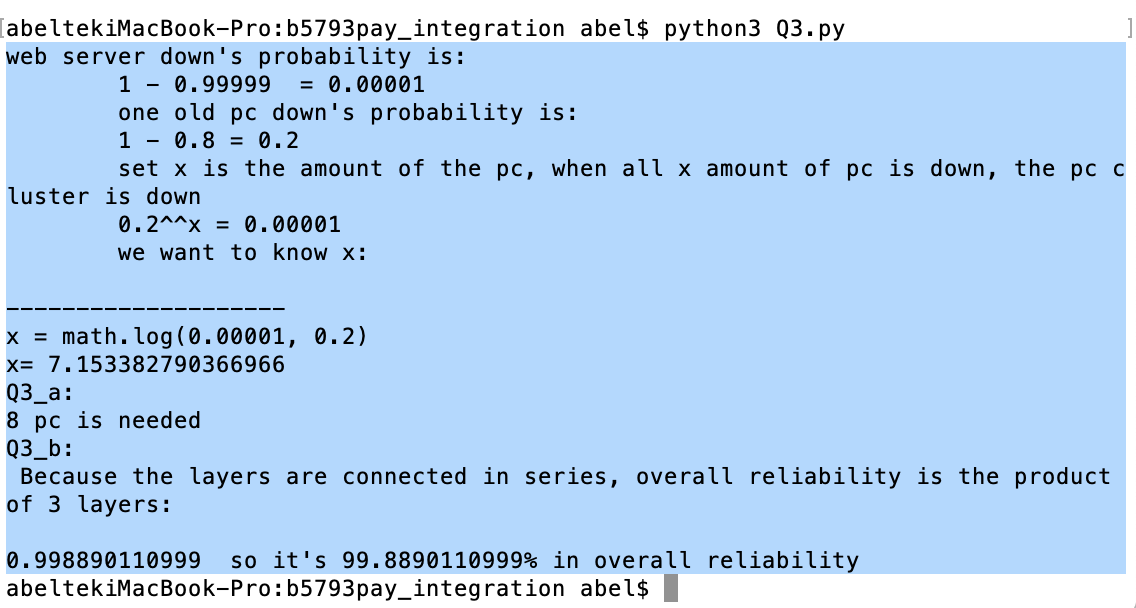








**Q3**



**Q4:**

1. sequence steps:

GQ invloves 3 message betwen Alice(cliament) and verifier(bob).

Bank generate paremters for protol to work.

Alice (the Prover) generates 3 values:

x(secret)= 3

N= 101

X= 27

Alice generates a random value (y):

y= 20

Alice computes Y = y^e (mod N) and passes to Office-Bob:

Y= 21

Office-Bob generates a random value (c) and passes to Alice:

c= 85

Alice calculates z = y.x^c (mod N) and send to Office-Bob (the Verifier):

Office-Bob now computes val=z^e (mod N) and (y x^c (mod N)) and determines if they are the same

val1= 48 val2= 48

Alice has proven that he knows x

2. also illustion sequence charts in Q4.png

Let me add,in parenthesis: in fact , in Q2\_b I alreay using some of 'Guillou-Quisquater (GQ) Identification',

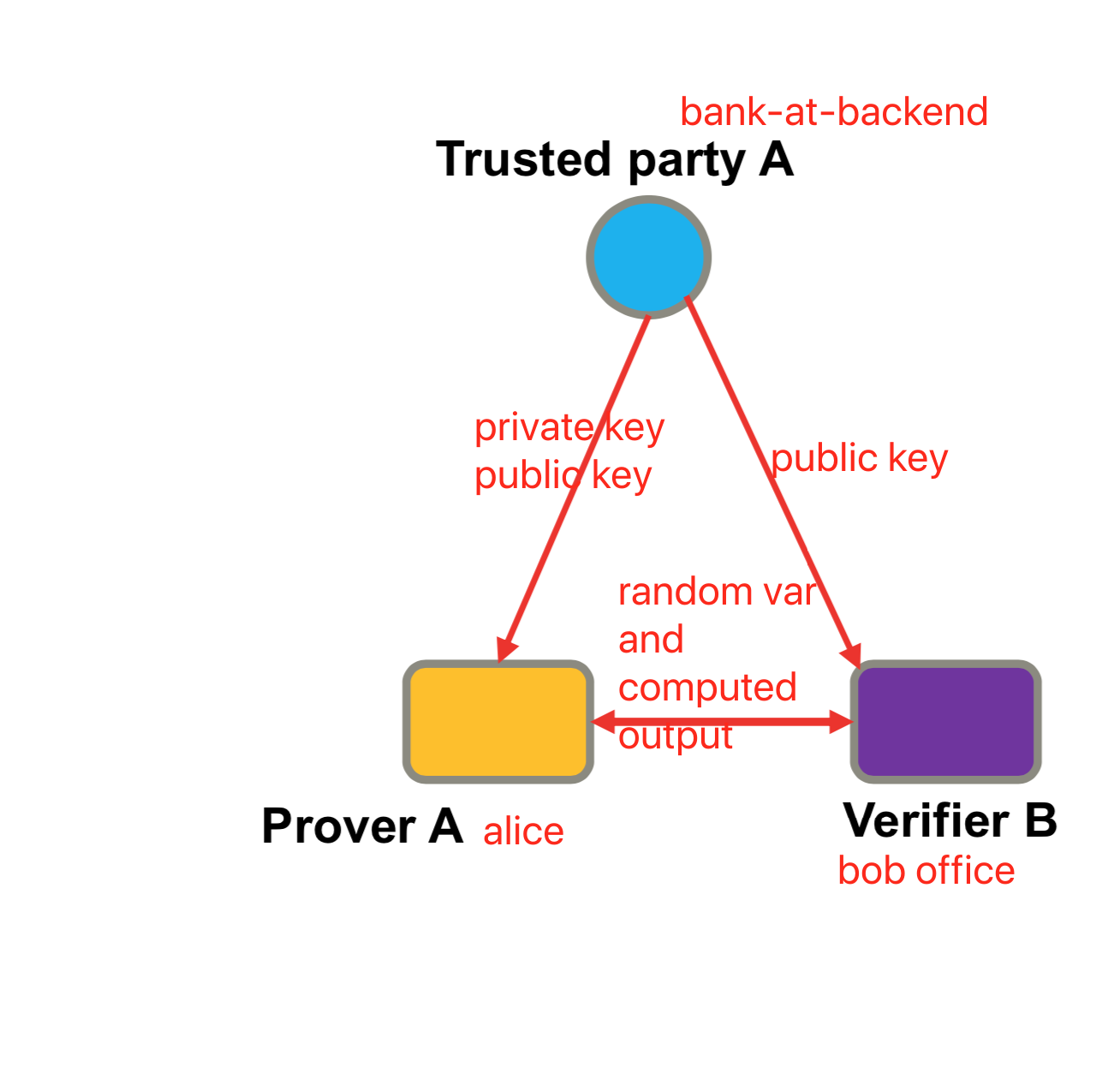
and made a code implementation.

3.what the bank must prepare in advance to facilitate this?

Bank must generate paremters for protol to work: generete public key and private key. Like save questions and answers before this happened.

For example: bank should know some questions and asked alice in advanced ,and save it for future's difficult

situation. When alice lost all cards , now she goto bob's brank office, now answered some pre-saved questions correctly,

now bob can trust her, gave her new private key to get her assets or trust she have rights to acess certain level of assets.