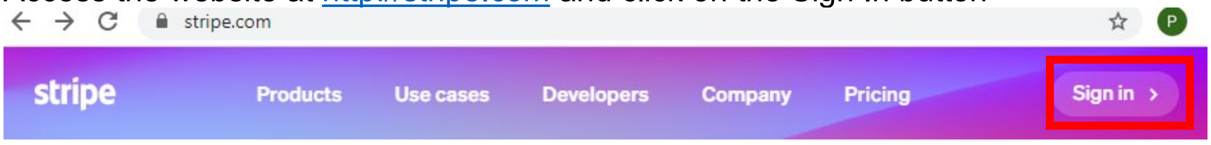
Lab 10 Part 6
Stripe Payment

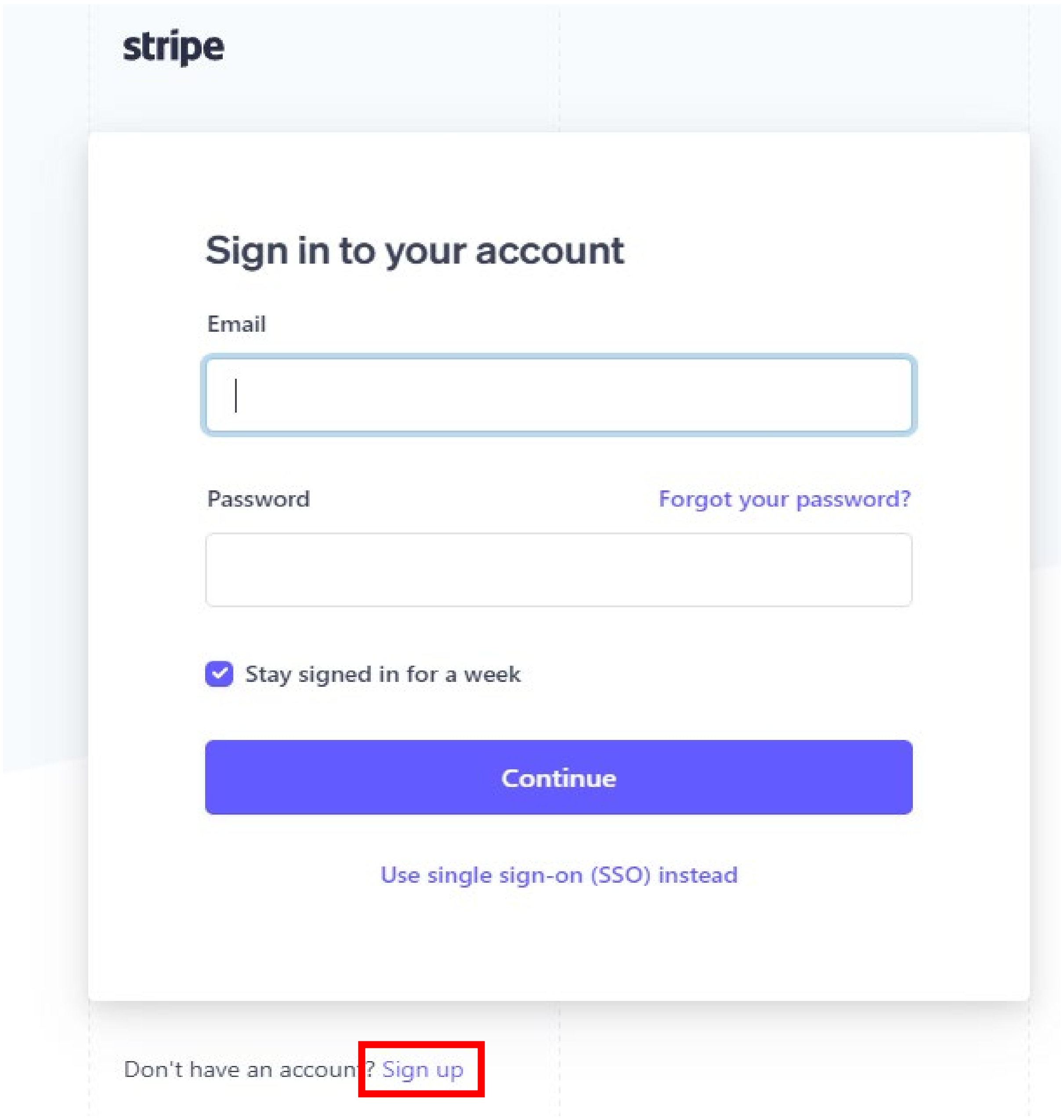
Step 1: Create a Stripe Account

We will process credit card payments using Stripe. Before you proceed with this exercise you need to create an account on stripe.com.

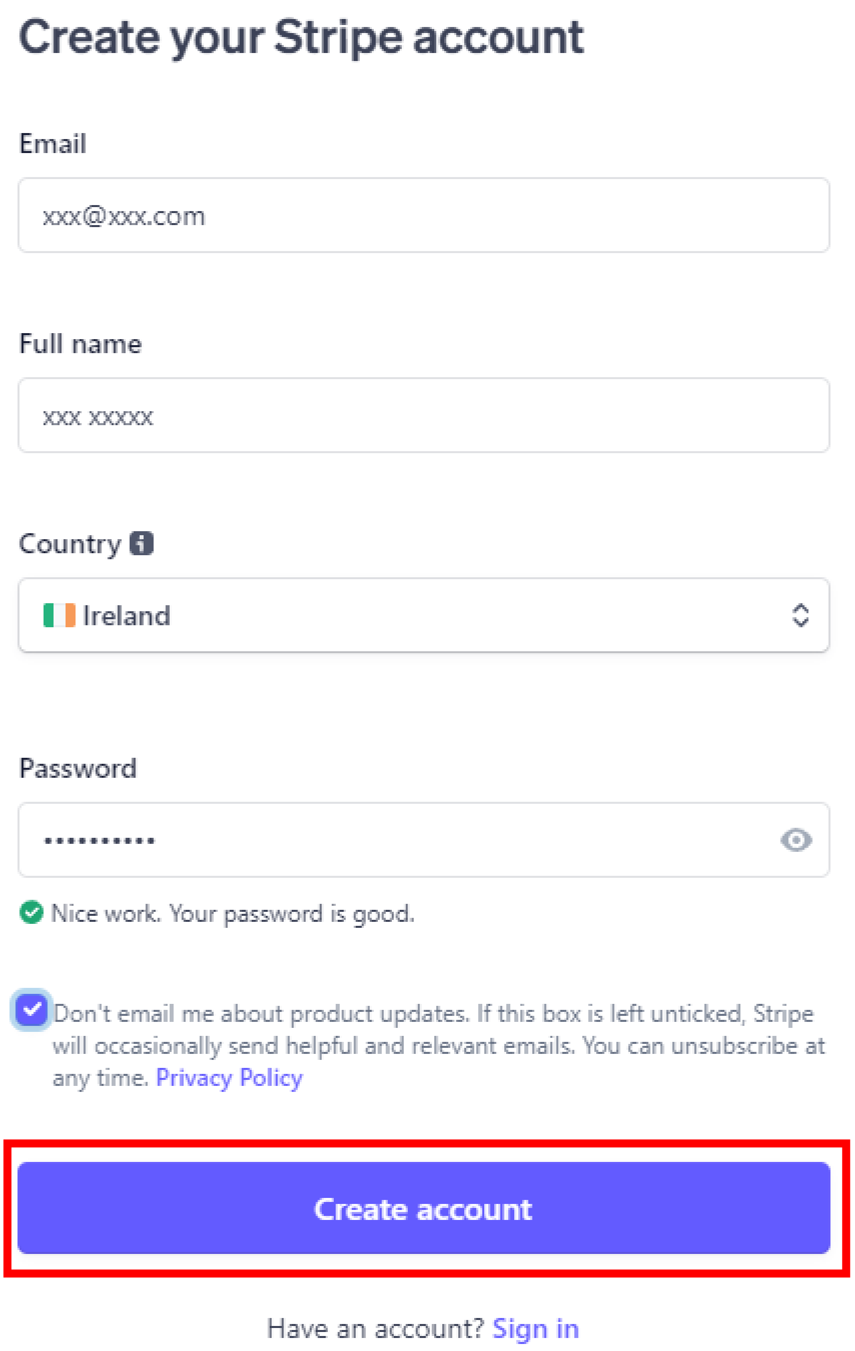
Access the website at http://stripe.com and click on the Sign In button



When you click on the Sign In menu option you will see the following screen. Click on Sign up to register a new account:


stripe
Sign in to your account
Email
Password
Forgot your password?
Stay signed in for a week
Continue
Use single sign-on (SSO) instead
Don't have an account? Sign up

When you have all the required information entered, click on the “Create Account” button highlighted below:


Create your Stripe account

Email

xxx@xxx.com

Full name

xxx xxxxx

Country

Ireland

Password

••••••••••

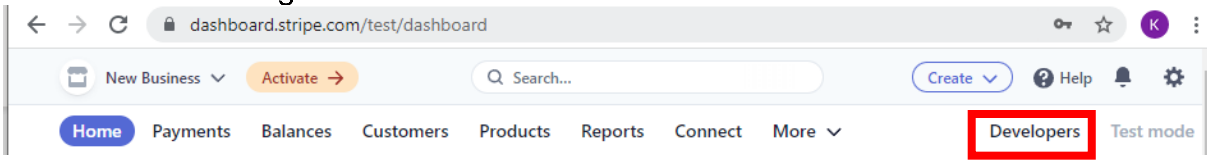
Nice work. Your password is good.

✓ Don't email me about product updates. If this box is left unticked, Stripe will occasionally send helpful and relevant emails. You can unsubscribe at any time. Privacy Policy

Have an account? Sign in

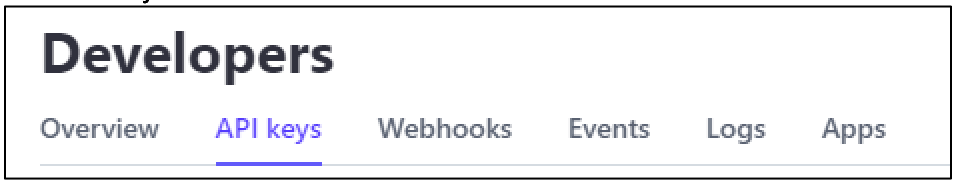
Go to your email and you should see a message from Stripe asking you to verify your email address. Click the button to verify your email address and you will then be redirected back to the Stripe website.

The menu should be displayed on the dashboard, and you will see the Developers menu item on the right-hand side as shown below:

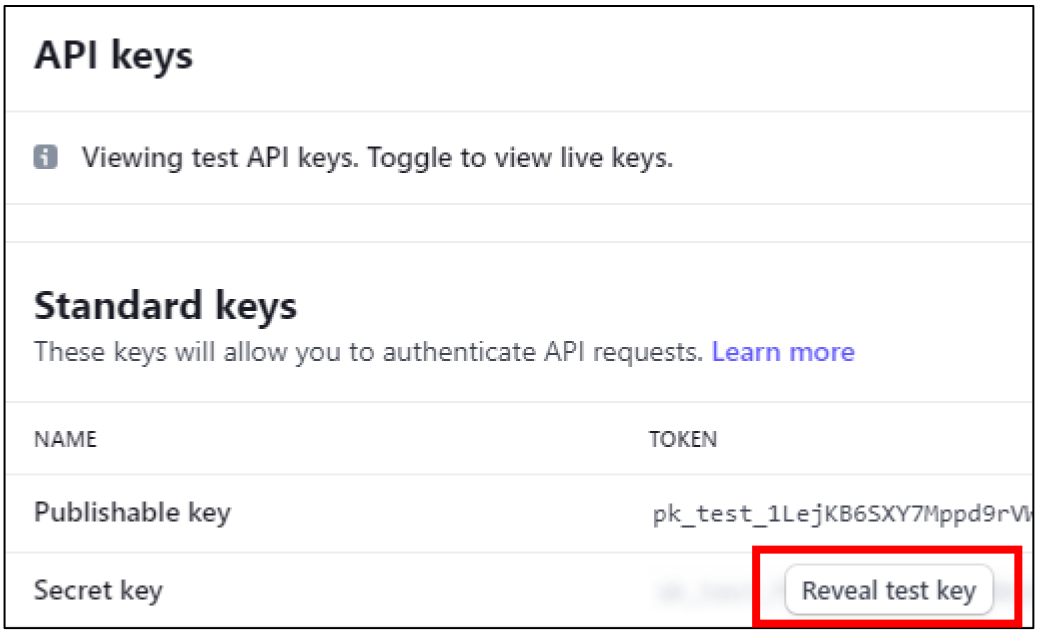


We are using the website in test mode as developers to implement test payments on our website.

When you click on the Developers menu item you will see the following menu. Click on API keys.



Hen you click on API keys, a publishable key is displayed but the secret key is not visible. Click on the “Reveal test key” button to reveal your secret key



API keys

Viewing test API keys. Toggle to view live keys.

Standard keys

These keys will allow you to authenticate API requests. Learn more

NAME	TOKEN
Publishable key	pk_test_1LejKB6SXY7Mppd9rVW
Secret key	Reveal test key

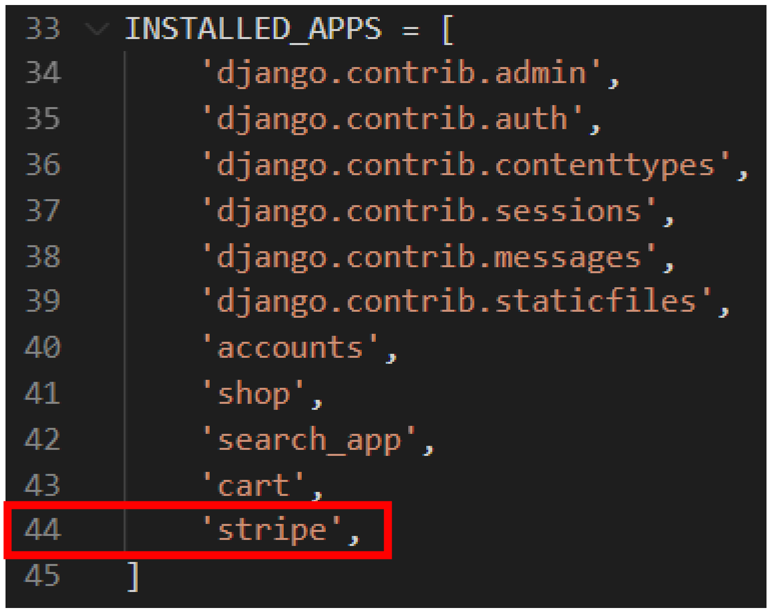
Each Stripe account has four API keys: two for testing and two for live production. Each pair has a "secret key" and a "publishable key". Do not reveal the secret key to anyone; the publishable key will be embedded in the JavaScript on the page that anyone can see. We are only working with test keys.

Step 2: Install Stripe
Open Windows Command Line, activate your virtual environment and install stripe:

---

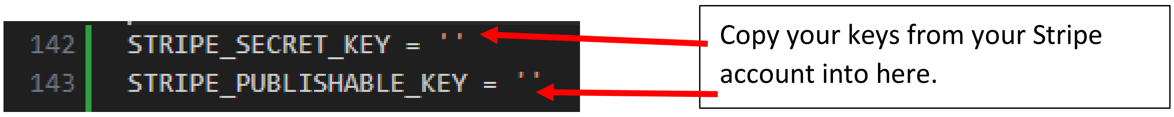
pip install stripe

Open the **settings.py** file and add the stripe app into the INSTALLED_APPS settings as shown below:



```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'accounts',
    'shop',
    'search_app',
    'cart',
    'stripe',
]
```

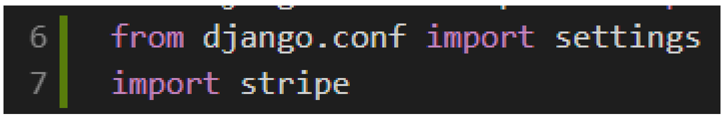
At the bottom of your **settings.py** file, add the following two lines including your own test secret and test publishable keys from your Stripe account. Make sure to include the ‘ ‘ characters around the actual keys.



```
STRIPE_SECRET_KEY = ''
STRIPE_PUBLISHABLE_KEY = ''
```

Copy your keys from your Stripe account into here.

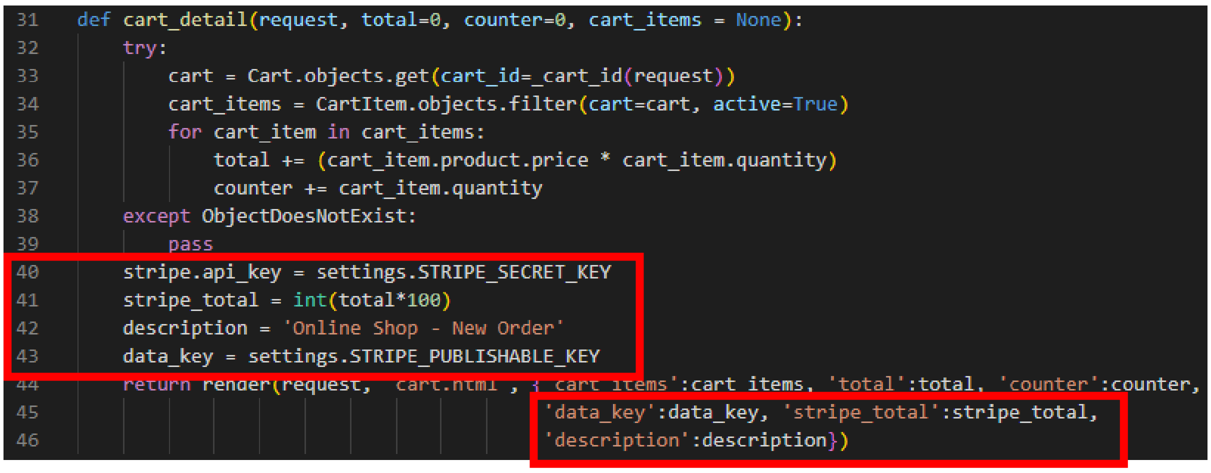
Open **cart/views.py** and add the following imports at the top of the file:



```
from django.conf import settings
import stripe
```

Step 3: Update the cart detail view to use Stripe

Add the following code to the **cart_detail** function:



```
31  def cart_detail(request, total=0, counter=0, cart_items = None):
32      try:
33          cart = Cart.objects.get(cart_id=_cart_id(request))
34          cart_items = CartItem.objects.filter(cart=cart, active=True)
35          for cart_item in cart_items:
36              total += (cart_item.product.price * cart_item.quantity)
37              counter += cart_item.quantity
38      except ObjectDoesNotExist:
39          pass
40      stripe.api_key = settings.STRIPE_SECRET_KEY
41      stripe_total = int(total*100)
42      description = 'Online Shop - New Order'
43      data_key = settings.STRIPE_PUBLISHABLE_KEY
44      return render(request, 'cart.html', {'cart_items':cart_items, 'total':total, 'counter':counter,
45                                           'data_key':data_key, 'stripe_total':stripe_total,
46                                           'description':description})
```

Line 40: We initialise the stripe.api_key variable using the Stripe private key

Line 41: We initialise a variable stripe_total by multiplying the total by 100. We multiply by 100 here because Stripe works in the smallest denomination i.e. cents

Line 42: We initialise a variable description with some text

Line 43: We initialise a variable data_key using the Stripe public ley

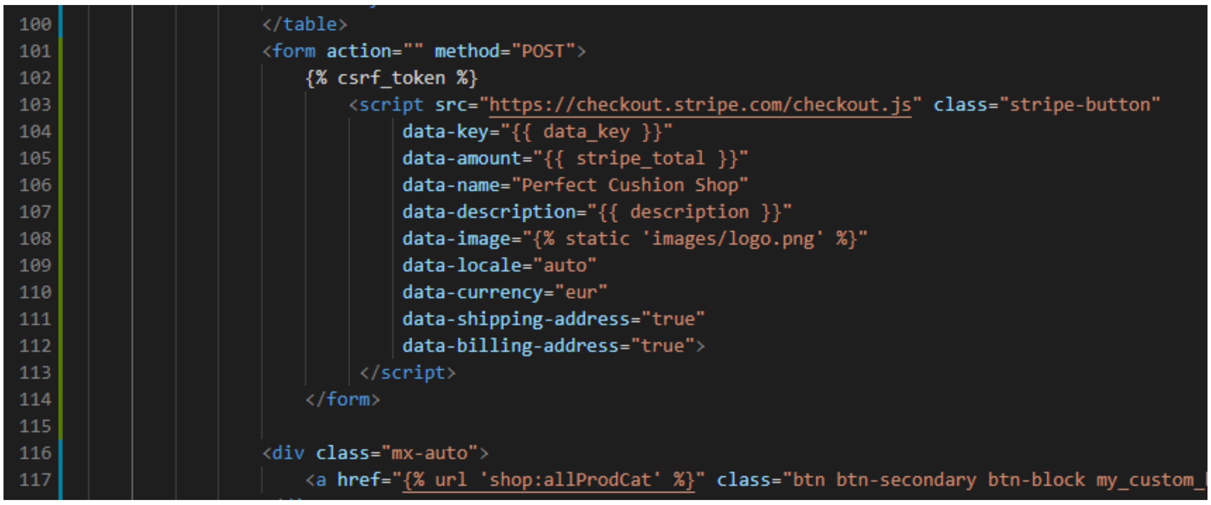
Lines 44-46: We call the render function passing in the request object, the template and the context data including the data_key, stripe_total & description.

Step 4: Add Stripe Checkout Form

Now that we have our API keys we need to add them to our website. We can use Stripe Checkout so we don't have to create the form. Stripe provides the following code which you should copy to the bottom of the **cart.html** file. See screenshot for the exact location of this code in **cart.html**:

```
<form action="" method="POST">
        {% csrf_token %}
        <script src="https://checkout.stripe.com/checkout.js" class="stripe-button"
                data-key="{{ data_key }}"
                data-amount="{{ stripe_total }}"
                data-name="Perfect Cushion Shop"
                data-description="{{ description }}"
                data-image="{% static 'images/logo.png' %}"
                data-locale="auto"
                data-currency="eur"
                data-shipping-address="true"
                data-billing-address="true"
                data-zip-code="true">
         </script>
</form>
```

The code should be at this location just below the closing table tag:

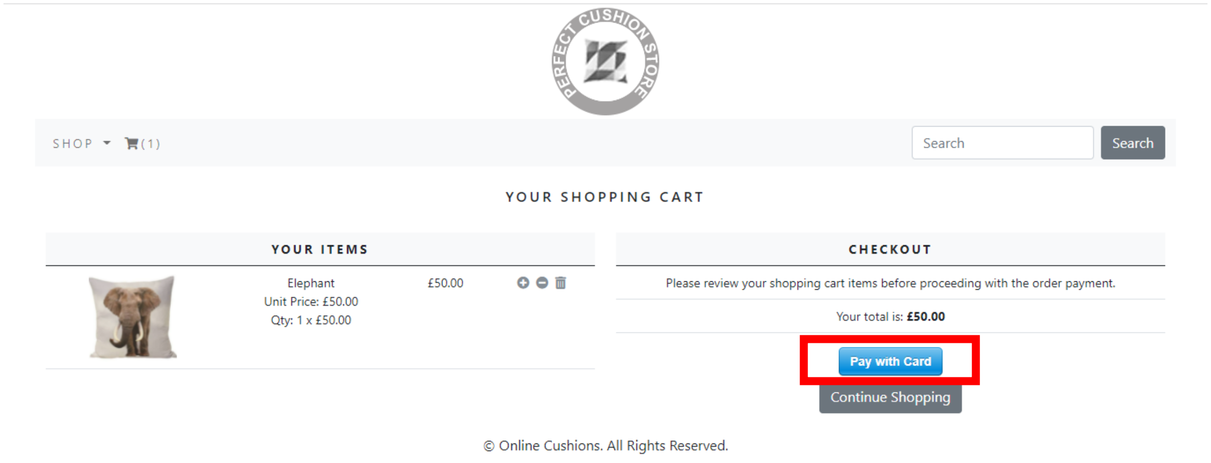


```
</table>
<form action="" method="POST">
    {% csrf_token %}
        <script src="https://checkout.stripe.com/checkout.js" class="stripe-button"
            data-key="{{ data_key }}"
            data-amount="{{ stripe_total }}"
            data-name="Perfect Cushion Shop"
            data-description="{{ description }}"
            data-image="{% static 'images/logo.png' %}"
            data-locale="auto"
            data-currency="eur"
            data-shipping-address="true"
            data-billing-address="true">
         </script>
    </form>

<div class="mx-auto">
    <a href="{% url 'shop:allProdCat' %}" class="btn btn-secondary btn-block my_custom_
```

Step 5: Try it out

Run the server and view the shopping cart. If it is empty, add an item into it and then you should see the button **Pay with Card** as shown below:



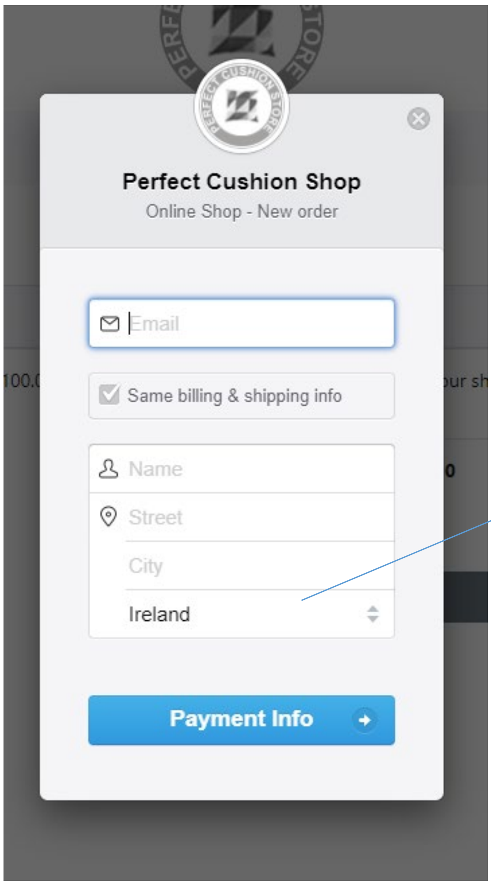
Elephant
Unit Price: £50.00
Qty: 1 x £50.00
£50.00

Please review your shopping cart items before proceeding with the order payment.

Your total is: **£50.00**

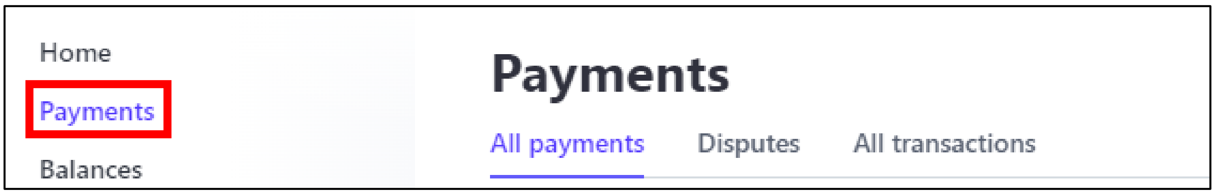
© Online Cushions. All Rights Reserved.

When you click the Pay with Card button the Stripe payment form appears. Fill in all the details and for the credit card number, Stripe provides some test credit card numbers such as 4242-4242-4242-4242, the expiry date just needs to be a date in the future and the CVC can be any three digits.



Change the country to Ireland which means that a zip code is not required

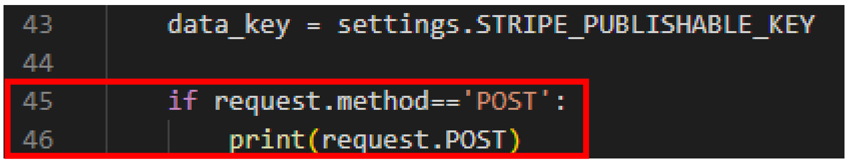
Go to stripe.com and click on "Payments" meu item – there are no payments/charges visible yet.



Step 6: Using the Stripe Token

In order to make charges visible in this section we need to use a token. In the code we wrote earlier we have used the **publishable key** to send the credit card information to Stripe, and Stripe has sent us back a token. We now need to use this token to generate the payment.

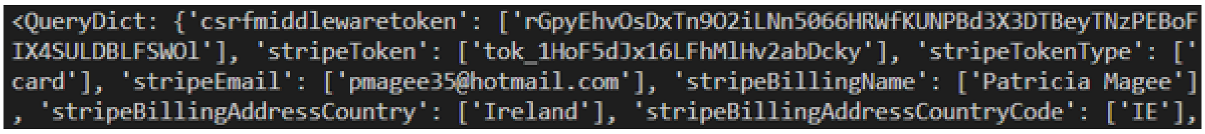
To view the **token** generated by Stripe, add the following if statement to **cart/views.py** inside the **cart_detail** function:



```
    data_key = settings.STRIPE_PUBLISHABLE_KEY

    if request.method=='POST':
        print(request.POST)
```

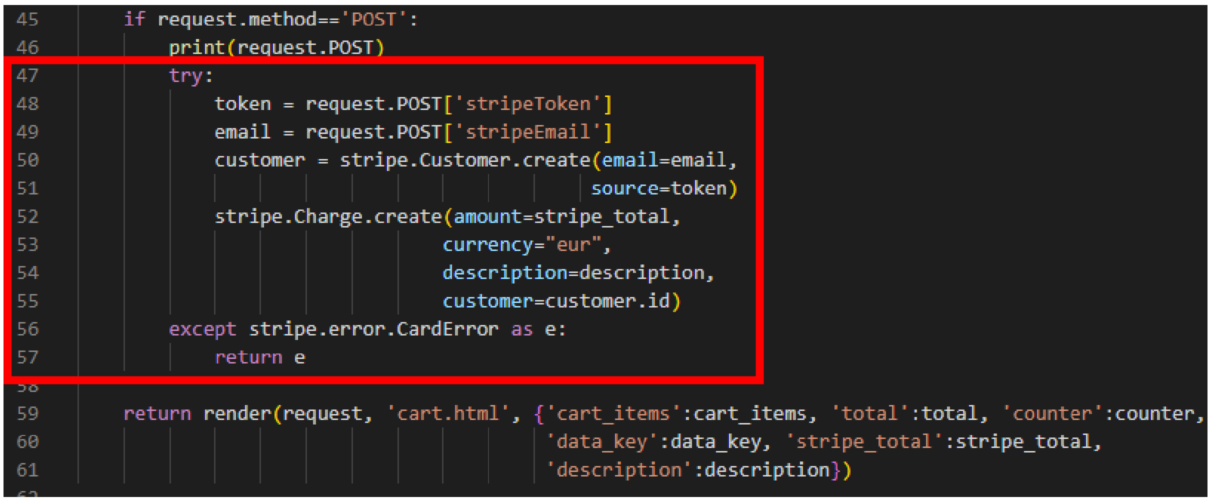
Restart the server and click the **Pay With Card** button again & complete the details in the form. When you do this check the output in Windows Command Line:



```
<QueryDict: {'csrfmiddlewaretoken': ['rGpyEhvOsDxTn9O2iLNn5066HRWfKUNPBd3X3DTBeyTNzPEBoF
IX4SULDBLFSWOl'], 'stripeToken': ['tok_1HoF5dJx16LFhMlHv2abDcky'], 'stripeTokenType': ['
card'], 'stripeEmail': ['pmagee35@hotmail.com'], 'stripeBillingName': ['Patricia Magee']
, 'stripeBillingAddressCountry': ['Ireland'], 'stripeBillingAddressCountryCode': ['IE'],
```

You will see a dictionary containing a lot of data. This is a Stripe token that has been generated which contains an id, address information etc of the information that you typed in when you made the payment. We can retrieve this information in **views.py** and use it to create an actual stripe **charge** in the Stripe account.

Open **cart/views.py** file and add in the following code:



```
45      if request.method=='POST':
46          print(request.POST)
47          try:
48              token = request.POST['stripeToken']
49              email = request.POST['stripeEmail']
50              customer = stripe.Customer.create(email=email,
51                                                source=token)
52              stripe.Charge.create(amount=stripe_total,
53                                   currency="eur",
54                                   description=description,
55                                   customer=customer.id)
56          except stripe.error.CardError as e:
57              return e
58
59      return render(request, 'cart.html', {'cart_items':cart_items, 'total':total, 'counter':counter,
60                                           'data_key':data_key, 'stripe_total':stripe_total,
61                                           'description':description})
```

Line 48: Initialise a token variable using the stripe token retrieved from request.POST

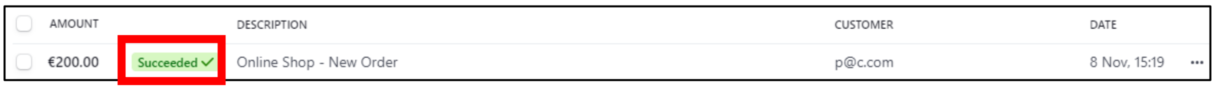
Line 49: Initialise an email variable using the stripe token retrieved from request.POST

Lines 50-51: Create a stripe Customer object supplying the email and the token

Lines 52:55: Create a stripe Charge object supplying the amount, currency, description, and customer id

Lines 56-57: Exception handling code if there is an error with the card

Restart the server and proceed to Pay with Card again and make another payment. Now go to stripe.com Payments section of the website and you should see the charge has succeeded and is shown in the account.

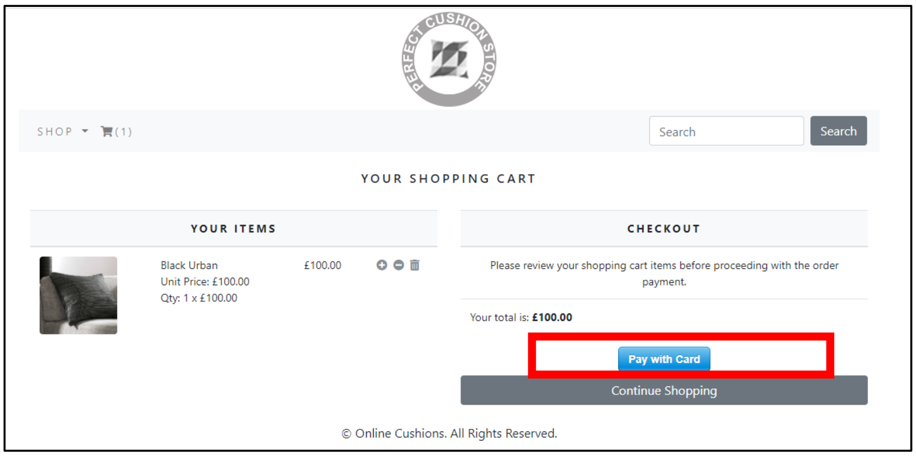


AMOUNT		DESCRIPTION	CUSTOMER	DATE
€200.00	Succeeded ✓	Online Shop - New Order	p@c.com	8 Nov, 15:19

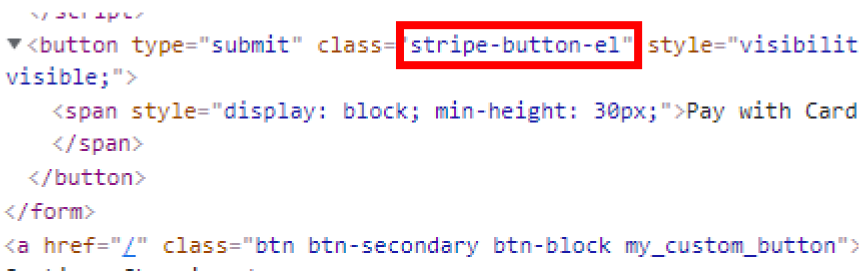
You can click on the **Succeeded** link to view the details of the payment.

Step 7: Style the Pay with Card button

The last step is to improve the look of the **Pay with Card** button and the **Continue Shopping** button using some css code.



To do this, we need to find out the name of this button. Right click on the button and choose the Inspect menu option. On the right-hand side, you should see the name as shown below:

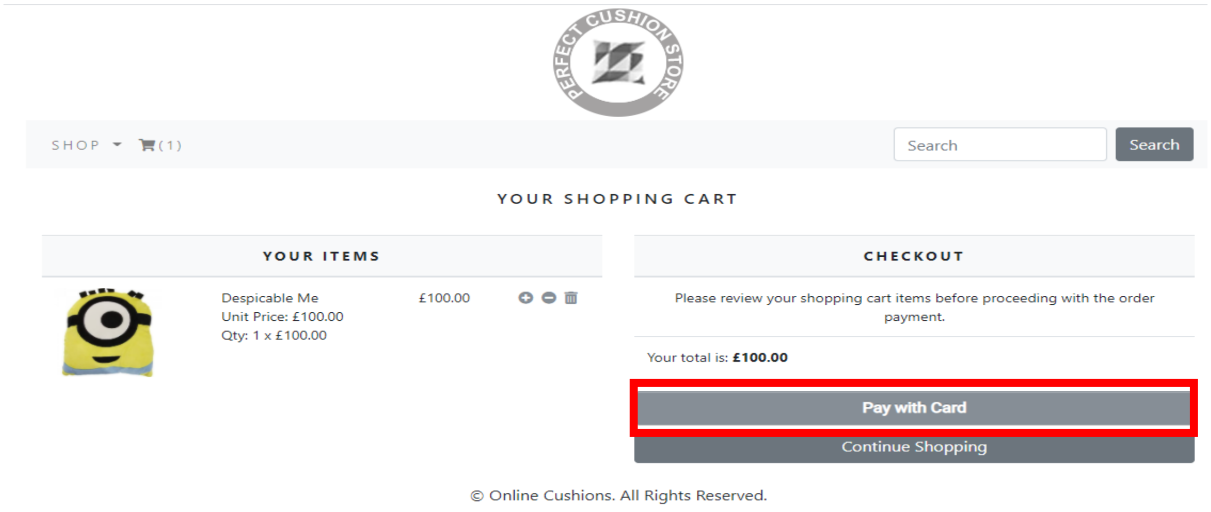


```
▼<button type="submit" class="stripe-button-el" style="visibilit
visible;">
    <span style="display: block; min-height: 30px;">Pay with Card
    </span>
  </button>
</form>
<a href="/" class="btn btn-secondary btn-block my_custom_button">
```

Copy the following css into **custom.css** to style this button:

```
/***Stripe Button***/
.stripe-button-el, .my_custom_button {
        width: 100% !important;
        display: block !important;
        background-color: #868e96 !important;
        border: 0px !important;
        background-image: none !important;
}
.stripe-button-el span {
        display: block;
        position: relative;
        padding: 0px 12px;
        height: 36px !important;
        line-height: 36px !important;
        font-family: 'Roboto', sans-serif !important;
        font-size: 16px !important;
        color: #fff !important;
        background-image: none !important;
        background: none !important;
}
```

Save your project and reload the server. If the changes are not applied, try, and clear the browser cache to view the newly styled buttons.



Step 8: Commit your changes and push your code to the lab 10 repo.

Stop the server and run the following git commands to update the local and remote repositories:

```
git add -A
git commit -m "lab 10 part 6 commit"
git push -u origin main
```