Time	~ 2 - 3 hours
Learning Goals	• Learn how to use an external API • Learn about Paypal Braintree payment system

What's the use of PairBnB if you can't make money? Let's include a payment system into our PairBnB. But before we integrate the payment system into our PairBnB app, let's first work on a fresh app without any user login or listings and understand how the system works.

Objectives

Understanding Braintree Payments

Before we delve into code, let's understand why we need a payment gateway by doing some research on the following:

1 What is PCI Compliance?
2 What is a payment gateway?
3 How do you validate a credit card number?
4 Why can't we just charge a credit card ourselves?
5 Read up on how Braintree processes your payments (tokens, payloads, payment method nonce, etc).

Sign up for a Braintree Sandbox Account

Please proceed to Braintree Payments and learn to sign up for an account. Make sure you sign up for the sandbox account.

Coding it up

For this challenge, we will work on a new Rails app without any user login/logout or models. Let's keep it simple first. Don't forget to bundle install and bundle exec rake db:create after creating your app.

The following tutorial will be based on the JavaScript v3 SDK provided by the Braintree documentation which helps us collect customer payment information and sends it to Braintree in exchange for a payment method nonce. We strongly encourage you to **read the documentation** first before following the tutorial below. Try to understand what is going on and not just copy-and-paste. Don't be a code monkey or you'll risk not being able to build anything without a tutorial :) When you have read the documentation, proceed as follow:

1. Protect your keys - include gem figaro in the Gemfile

2. Include gem braintree - in your Gemfile, includegem 'braintree'.

3. Run bundle install.

4. List down your keys in application.yml.

From your Braintree sandbox account, go to Account > My User. Scroll down until you see "View Authorizations". The necessary keys will be there. Copy and paste the keys into application.yml:

```
BRAINTREE_MERCHANT_ID: 'your_merchant_id'
```

```
BRAINTREE_PUBLIC_KEY: 'your_public_key'

BRAINTREE_PRIVATE_KEY: 'your_private_key'
```

5. Set up Braintree configuration environment

In config/initializers, create a new file called braintree.rb and do the following:

```
Braintree::Configuration.environment = :sandbox

Braintree::Configuration.logger = Logger.new('log/braintree.log')

Braintree::Configuration.merchant_id = ENV['BRAINTREE_MERCHANT_ID']

Braintree::Configuration.public_key = ENV['BRAINTREE_PUBLIC_KEY']

Braintree::Configuration.private_key = ENV['BRAINTREE_PRIVATE_KEY']
```

6. Set up a controller and view page as root

```
rails g controller Welcome index
```

Remember to go to routes.rband set welcome#index as your root path. We will send the user to the payment form and generate a client token at the same time later in step 8 from here. Let's set up a path:

```
<%= link_to "Make Payment", braintree_new_path %>
```

Note that if you load the page, there will be an error message because we have not yet set up the braintree_new_path. So let's set it up now:

```
#In the terminal:

  rails g controller Braintree new
```

7. Set up a payment form

In this example, we will set up the payment form in braintree/new.html.erb. In your PairBnB app, you may want to set it up in another view file. We will leave it to you to decide on what works best for your use case.
The form that we will be using is from Braintree's Hosted Fields solution which is PCI-compliant. Let's now set up the form.
a. Go to this link and look for the example form. Click on "CSS" and copy the code into a **new file** in your app. You can name the file as payment.scss or braintree.scss or anything that helps you remember that this code is for your payment form. Remember to include Bootstrap in your application.html.erb file as well.

NOTE: **DO NOT** copy the HTML and JS code yet!
b. We will now write HTML code for the form. We will be using the **form_for helper** instead of the form tag as shown in the documentation. Why do you think we have to use the **form_for** helper? (Hint: authenticity token)

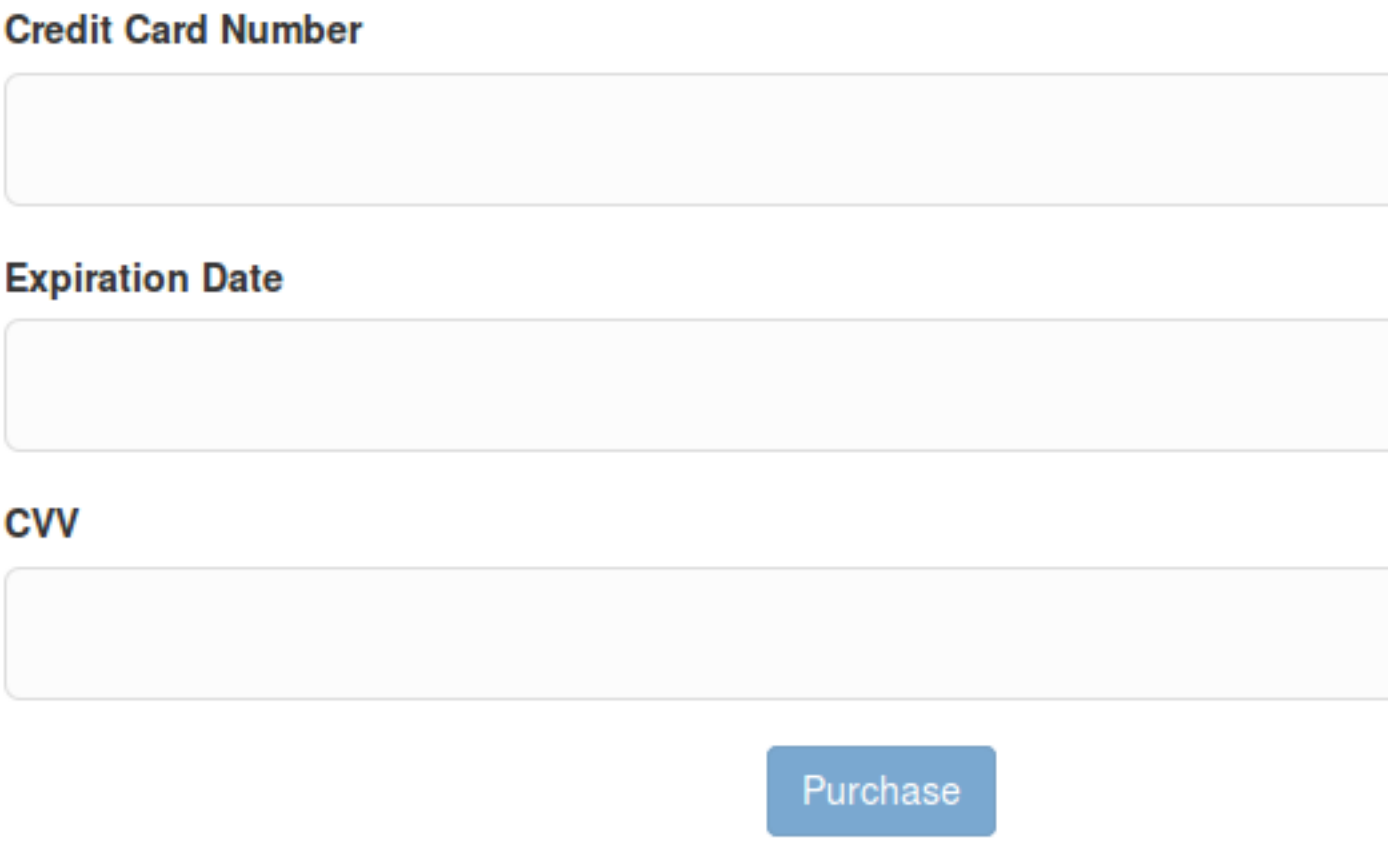
In braintree/new.html.erb, set up the form as follows. As you type, **take note of the class**

and ID names.

```
  <div class="demo-frame">

    <%= form_for :checkout_form, html: { id: "cardForm"} do |form| %>

      <%= form.label :card_number, "Credit Card Number", html: {class: "hosted-fields--label"} %>
      <div id="card-number" class="hosted-field"></div>

      <%= form.label :expiration_date, "Expiration Date", html: {class: "hosted-fields--label"} %>
      <div id="expiration-date" class="hosted-field"></div>

      <%= form.label :cvv, "CVV", html: {class: "hosted-fields--label"} %>
      <div id="cvv" class="hosted-field"></div>

      <%= form.hidden_field "payment_method_nonce" %>

      <div class="button-container">
        <%= form.submit "Purchase", disabled: true, class: "btn btn-primary", id: "submit-payment-btn" %>
      </div>
    <% end %>
  </div>
```

Note that in the CSS code copied previously, the styles for the submit payment button are styled under the class button and not the ID submit-payment-btn. You may want to change the name accordingly in your CSS code.
(Word-of-caution: whenever you learn from tutorials, take note of seemingly little things like this. You can have named your HTML elements as something else but the style file that you took from somewhere else might use another name for that particular element.)

Now when you run your server and load your page, you should see something like this:



Note that the button has been disabled for now, and you will not be able to submit anything yet. We will work on this later. Let's move on to the next step.

8. Generate a client token and get a payment method nonce

From the diagram in the documentation, we learn that that our server is responsible for generating a client token. We also learnt that this can be achieved by calling the Braintree:ClientToken.generate. Once the token is generated, we will then embed the client token into our JavaScript which will help us collect customer payment information and send it to Braintree in exchange for a payment method nonce.

Note that there are various ways to embed the client token. In this example, we will generate the token in the controller and embed it in JavaScript.

In braintree_controller.rb:

```
def new
  @client_token = Braintree::ClientToken.generate
end
```

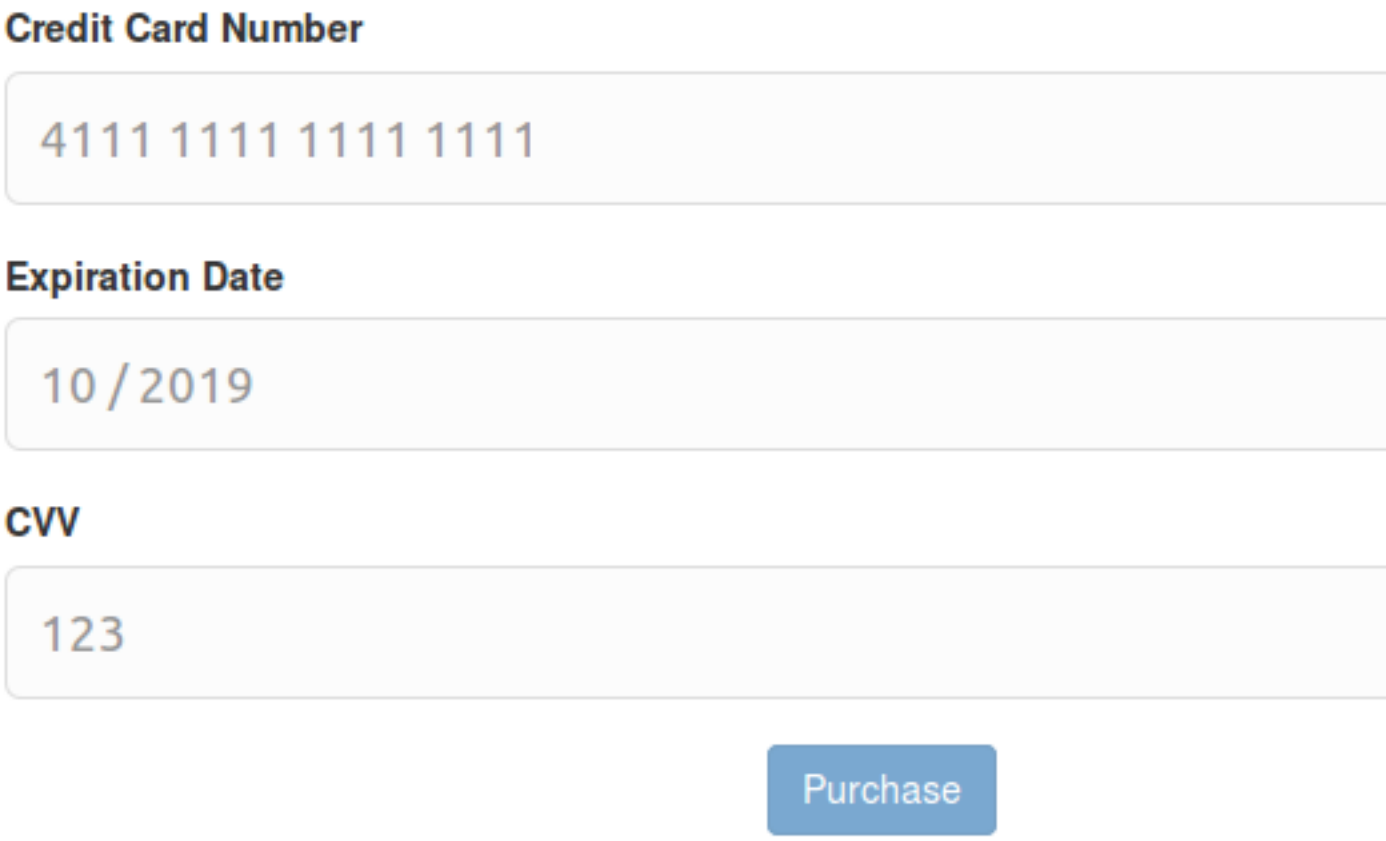
In braintree/new.html.erb, after the form, include the following:

```
<!-- Load the Client component. -->
<script src="https://js.braintreegateway.com/web/3.6.3/js/client.min.js"></script>

<!-- Load the Hosted Fields component. -->
<script src="https://js.braintreegateway.com/web/3.6.3/js/hosted-fields.min.js"></script>
```

Then, copy the JavaScript code from the documentation with the following tweaks:

- Let `var authorization = '<%= @client_token %>'` instead of the given token in the documentation.
- Let `var submit = document.querySelector("#submit-payment-btn");` instead of the selector given.
- Let `var form = document.querySelector('#cardForm');` instead of the selector given.
- Let `document.querySelector('input[name="checkout_form[payment_method_nonce]"]').value = payload.nonce;` instead of what was given.

Note that in your PairBnB, you might be naming your buttons and forms differently. Please also change the selectors accordingly.
Your payment form should now look something like this:



Your client-side working flow now is all set up. Your client gets a token which provides the necessary information to initialize the client SDK (“the JavaScript”) to communicate with Braintree in exchange for a payment method nonce, which is then sent back to your server. Once your server receives the nonce, it can then finally create a transaction.

9. Receive a payment method nonce and create a transaction

We will now refer to the documentation here. In your app, you will **need to give a URL to your form** and do the following (note that there are **three** things to do be done):

#1. In braintree/new.html.erb:

#Add braintree_checkout_path to the form_for tag, i.e.

```
<%= form_for :checkout_form, url: braintree_checkout_path, html: { id: "cardForm"} do
|form| %>
```

#2. In routes.rb:

```
post 'braintree/checkout'
```

```
#3. In braintree_controller.rb:
def checkout
  nonce_from_the_client = params[:checkout_form][:payment_method_nonce]

  result = Braintree::Transaction.sale(
   :amount => "10.00", #this is currently hardcoded
   :payment_method_nonce => nonce_from_the_client,
   :options => {
      :submit_for_settlement => true
    }
   )

  if result.success?
    redirect_to :root, :flash => { :success => "Transaction successful!" }
  else
    redirect_to :root, :flash => { :error => "Transaction failed. Please try again." }
  end
end
```

Note that in the code, we have hard-coded the amount to be charged. Figure out how to allow for different payment amounts later on for your PairBnB app. Read up more on Braintree:Transaction as well.

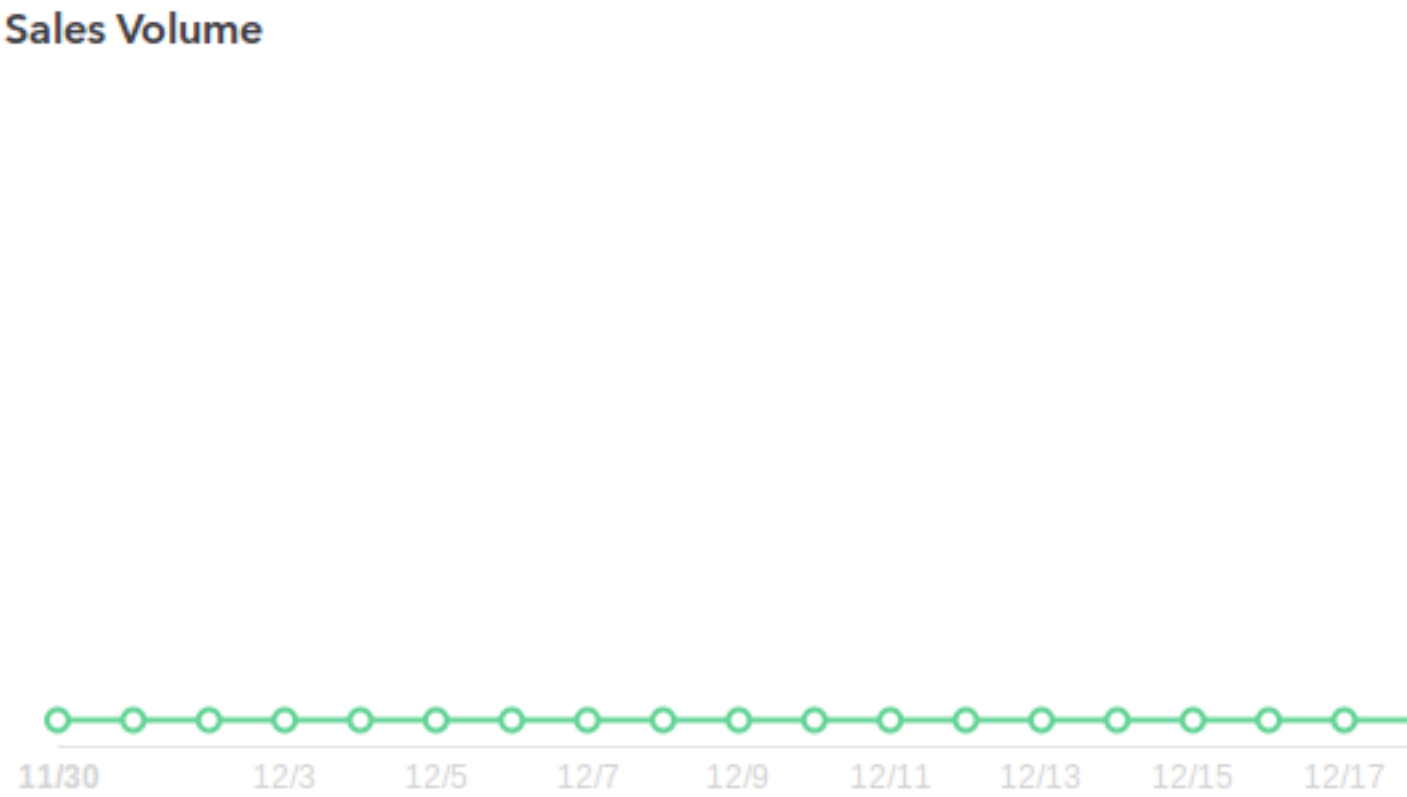
10. Set up helper methods for flash alerts

You would have also noticed that we have set up flash alerts in the controller. We would also like the alerts to appear according to the colour of the status (e.g. success, error, etc). Here is a reference to help you get the bootstrap flash alerts set up. If a flash message appears but without a coloured box, read the comment section of the blog article and make the necessary edits. Feel free to look up other resources on how to set up bootstrap flash alerts and other various options available.

11. Test the integration

We are now ready to test the integration. Key in the necessary payment details (remember to follow the credit card details given by Braintree!). A flash alert will appear to tell you if your transaction is successful or not. If you have a successful transaction, go

to the dashboard of your Sandbox account. You should see something like this:



Once you have a successful transaction, good job! Feel free to check out this reference if you want more practice.

Consolidate Learning

Let's now take a step back and revise what you have been coding. With your partner or on your own, work out the following:

1 Explain to each other or to yourself the process between the client, your server and Braintree. This might be helpful.
2 Match each step outlined in the diagram with your code.
3 Write down things that you are unclear about and do more research on it.
4 Compare and contrast the difference between this tutorial and the documentation. What was done differently? (Sometimes reading documentation might take hours or even days - but it's okay - as long as we learnt something!)

Once you are done and comfortable with the whole process, move on to the next challenge and integrate the payment system with our PairBnB app!