User Stories

**New Members**

As a new member, I want to purchase a single sheet (first time creates new user)

As a new member, I want to sign up for a subscription (first time creates new user and subscription)

**Current Members**

As a current member (already made one or more single purchases), I want to purchase a single sheet (charges the existing customer on record at Stripe)

As a current member (already made one or more single purchases), I want to sign up for a subscription (updates existing customer to the subscription plan)

As a new member, I want to purchase a single sheet (first time creates new user)

**Account.php**

**This will build the Stripe checkout form with pre-filled data (email, charge cost, etc.)**

**User onlyl needs to supply their form of payment (credit card details)**

<form action="user\_charge.php" method="post">

<?php $isStripeCustomer = fetchStripeId($loggedInUser->user\_id); //get stripe id for this user if it exists

foreach ($isStripeCustomer as $stripeid){

if($stripeid['stripeid'] !== '0'){

$stripe\_customer = Stripe\_Customer::retrieve($stripeid['stripeid']);?>

<input type="hidden" name="customer\_id" value="<?php echo $stripeid['stripeid'] ?>" />

<?php } } ?>

<p align="center">

<script src="https://checkout.stripe.com/checkout.js" class="stripe-button"

data-key="<?php echo $stripe['publishable\_key']; ?>"

data-email="<?php echo $loggedInUser->email; ?>"

data-amount="250"

data-description="Single Purchase ($2.50)"

data-image="img/charge-logo.png"

data-name="Thoroughwiz Sheet"

data-panel-label="Purchase of "

data-label="Single Sheet - $2.50">

</script>

</p>

</form>

**Note**: Checkout must be loaded from https://checkout.stripe.com/checkout.js. Using a local copy of Checkout is unsupported, and may result in user-visible errors.

The *simple* integration uses a <script> tag inside your payment <form> to render the blue Checkout button. When a user clicks the button and completes payment, we will submit your form with a stripeToken along with any other<input>s in your form.

It's worth noting that Checkout doesn't actually create charges—it only creates tokens. You can use those tokens to create the actual [charge on your server](https://stripe.com/docs/charges). Alternatively, you can [save the card](https://stripe.com/docs/charges#saving-credit-card-details-for-later) for charging later, or sign the user up for [recurring charges](https://stripe.com/docs/subscriptions/tutorial).

**Received Parameters**

These parameters will be submitted to your form's action endpoint, along with any other <input> fields in the form, once the checkout is complete.

| **Parameter** | **Description** |
| --- | --- |
| stripeToken | The ID of the token you need to create a charge or a customer. |
| stripeEmail | The email address the user entered during the Checkout process. |
| stripeBillingName stripeBillingAddressLine1 stripeBillingAddressZip stripeBillingAddressState stripeBillingAddressCity stripeBillingAddressCountry | Billing address details (if [enabled](https://stripe.com/docs/checkout#integration-options-billing-address)). |
| stripeShippingName stripeShippingAddressLine1 stripeShippingAddressZip stripeShippingAddressState stripeShippingAddressCity stripeShippingAddressCountry | Shipping address details (if [enabled](https://stripe.com/docs/checkout#integration-options-shipping-address)). |

## Configuration Options

### Required

| **Option** | **Description** |
| --- | --- |
| data-key | Your publishable key (test or live). |
| tokenOnly available with the custom integration | The callback to invoke when the Checkout process is complete.  **function(token)**  **token** is the [token object](https://stripe.com/docs/api#tokens) created.  **token.id** can be used to create a charge or customer.  **token.email** contains the email address entered by the user.  **args** is an object containing [billing](https://stripe.com/docs/checkout#integration-options-billing-address) and [shipping](https://stripe.com/docs/checkout#integration-options-shipping-address) addresses if enabled.[Show example.](https://stripe.com/docs/checkout) |

### Highly recommended

| **Option** | **Description** |
| --- | --- |
| data-image | A relative or absolute URL pointing to a square image of your brand or product. The recommended minimum size is 128x128px. The recommended image types are .gif, .jpeg, and .png. |
| data-name | The name of your company or website. |
| data-description | A description of the product or service being purchased. |
| data-amount | The amount (in cents) that's shown to the user. Note that you will still have to explicitly include it when you create a charge using the API. |
| data-locale | Specify **auto** to display Checkout in the [user's preferred language](https://support.stripe.com/questions/what-languages-does-stripe-checkout-support), if available. English will be used by default. |

### Optional

| **Option** | **Description** |
| --- | --- |
| data-currency | The currency of the [amount](https://stripe.com/docs/checkout#integration-options-amount) (3-letter [ISO code](https://support.stripe.com/questions/which-currencies-does-stripe-support)). The default is USD. |
| data-panel-label | The label of the payment button in the Checkout form (e.g. “Subscribe”, “Pay {{amount}}”, etc.). If you include **{{amount}}**, it will be replaced by the provided amount. Otherwise, the amount will be appended to the end of your label. Checkout does not translate custom labels to the user's preferred language. |
| data-zip-code | Specify whether Checkout should validate the billing ZIP code (true or false). The default is false. |
| data-billing-address | Specify whether Checkout should collect the user's billing address (true or false). The default is false. |
| data-shipping-address | Specify whether Checkout should collect the user's shipping address (true or false). The default is false. |
| data-email | If you already know the email address of your user, you can provide it to Checkout to be pre-filled. |
| data-label | The text to be shown on the default blue button. Checkout does not translate custom labels to the user's preferred language. |
| data-allow-remember-me | Specify whether to include the option to "Remember Me" for future purchases (true or false). The default is true. |
| data-bitcoin | Specify whether to accept [Bitcoin](https://stripe.com/docs/bitcoin) (true or false). The default is false. |
| data-alipay | Specify whether to accept [Alipay](https://stripe.com/docs/alipay) ('auto', true, or false). The default is false. |
| data-alipay-reusable | Specify if you need [reusable access to the customer's Alipay account](https://stripe.com/docs/alipay#reusable-and-non-reusable-account-access)(true or false). The default is false. |
| openedOnly available with the custom integration | **function()**  The callback to invoke when Checkout is opened (not supported in IE6 and IE7). |
| closedOnly available with the custom integration | **function()**  The callback to invoke when Checkout is closed (not supported in IE6 and IE7). |

Once you've securely collected and tokenized your customer's credit card using [Checkout](https://stripe.com/docs/checkout/tutorial) or [Stripe.js](https://stripe.com/docs/custom-form), you can charge the card. Unlike collection, which occurs in the browser, charge attempts are made from your server, normally using one of our [client libraries](https://stripe.com/docs/libraries). If you haven't already, install the library for your favorite language now. This tutorial shows code for Ruby, PHP, Python, and Node, but we also have libraries for Java and Go.

On your server, grab the Stripe token in the POST parameters submitted by your form. From there, it's one simple API call to charge the card:

The following collects information from the form scope. Token, email, and current stripe customer id if it already exists. If it already exists we simply charge the current customer using their saved information both on our servers and on Stripes.   
  
if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {require\_once('scripts/stripe/stripe-config.php');  
 $errors = array();  
 $token = $\_POST['stripeToken'];  
 $email = $\_POST['stripeEmail'];  
 if(isset($\_POST['customer\_id'])){ // if this is set, then the customer already exists...  
 try {  
 $charge = Stripe\_Charge::create(array(  
 'amount' => 250,  
 'currency' => 'usd',  
 'customer' => $\_POST['customer\_id'],  
 'description' => "Single Sheet purchase"  
 ));  
 } catch (Stripe\_CardError $e) {  
 // Card was declined.  
 $e\_json = $e->getJsonBody();  
 $err = $e\_json['error'];  
 $errors['stripe'] = $err['message'];  
 } catch (Stripe\_ApiConnectionError $e) {  
 // Network problem, perhaps try again.  
 } catch (Stripe\_InvalidRequestError $e) {  
 // You screwed up in your programming. Shouldn't happen!  
 } catch (Stripe\_ApiError $e) {  
 // Stripe's servers are down!  
 } catch (Stripe\_CardError $e) {  
 // Something else that's not the customer's fault.  
 }

If the customer does not exists in our servers (i.e. no stripe id), then we must create a new customer and then charge them…  
  
 } else { // customer does not exist, create one then charge the new customer  
 try {  
 // create customer  
 $customer = Stripe\_Customer::create(array(  
 'email' => $email,  
 'card' => $token  
 ));  
 // charge the customer  
 $charge = Stripe\_Charge::create(array(  
 'customer' => $customer->id,  
 'amount' => 250,  
 'currency' => 'usd',  
 'description' => 'Single Sheet Purchase'  
 ));  
 } catch (Stripe\_CardError $e) {  
 // Card was declined.  
 $e\_json = $e->getJsonBody();  
 $err = $e\_json['error'];  
 $errors['stripe'] = $err['message'];  
 } catch (Stripe\_ApiConnectionError $e) {  
 // Network problem, perhaps try again.  
 } catch (Stripe\_InvalidRequestError $e) {  
 // You screwed up in your programming. Shouldn't happen!  
 } catch (Stripe\_ApiError $e) {  
 // Stripe's servers are down!  
 } catch (Stripe\_CardError $e) {  
 // Something else that's not the customer's fault.  
 }  
 }  
}

if ($charge->paid == true) {  
$isStripeCustomer = fetchStripeId($loggedInUser->user\_id); //get stripe id for this user if it exists  
foreach ($isStripeCustomer as $stripeid){  
 if($stripeid['stripeid'] === '0'){ // if it equals 0 user is not a stripe customer yet // update member record with stripe customer id createCustomer($customer->id, $loggedInUser->user\_id); }  
}

// add user to quickie role unless it already exists  
if ($loggedInUser->checkPermission(array(4))){ // this is in class.user.php // return true;  
}else{  
addPermission(4, $loggedInUser->user\_id);  
}

// add one credit to this users credit tally, to be used at their discretion  
addCredit(1, $loggedInUser->user\_id);  
  
// echo transaction details out to the user. Prompt to email receipt if user desires

} else { // Charge was not paid!

if (isset($errors) && !empty($errors) && is\_array($errors)) {  
echo '<div class="col-md-8 col-md-offset-2 text-center" style="border:1px solid red;"><h3 class="text-center" style="color:#303030;">Error Log:</h2><ul>';

foreach ($errors as $e) {  
 echo "<li>$e</li>";  
}  
echo '</ul></div>';  
}  
}

Here's the whole workflow:

1. The customer arrives at your payment page, loaded over HTTPs, that includes the Checkout code.
2. The customer clicks the payment button (e.g., "Pay with Card"), completes the payment form, and clicks "Pay $xx.xx" within the Checkout window.
3. Checkout sends the credit card details directly to Stripe from the customer's browser, assuming the details pass basic validation.
4. Stripe returns a token to Checkout, or an error message if real validation fails.
5. Checkout takes the returned token and stores it in the page's primary form—the one surrounding the SCRIPT tag above, in a hidden element named stripeToken.
6. Checkout submits the form to your server.
7. Your server uses the posted token to [charge the card](https://stripe.com/docs/charges).

To be absolutely clear on the process, the client-side code—Checkout—securely accepts the card information but does not initiate a payment attempt. The actual charge request is triggered from your server.

As a current member (already made one or more single purchases), I want to purchase a single sheet (charges the existing customer on record at Stripe)

**Account.php**

**This will build the Stripe checkout form with pre-filled data (email, charge cost, etc.)**

**User only needs to supply their form of payment (credit card details)**

<form action="user\_charge.php" method="post">

<?php $isStripeCustomer = fetchStripeId($loggedInUser->user\_id); //get stripe id for this user if it exists

foreach ($isStripeCustomer as $stripeid){

if($stripeid['stripeid'] !== '0'){ //we have a current Stripe customer

$stripe\_customer = Stripe\_Customer::retrieve($stripeid['stripeid']);?>

<input type="hidden" name="customer\_id" value="<?php echo $stripeid['stripeid'] ?>" />

<?php } } ?>

<p align="center">

<script src="https://checkout.stripe.com/checkout.js" class="stripe-button"

data-key="<?php echo $stripe['publishable\_key']; ?>"

data-email="<?php echo $loggedInUser->email; ?>"

data-amount="250"

data-description="Single Purchase ($2.50)"

data-image="img/charge-logo.png"

data-name="Thoroughwiz Sheet"

data-panel-label="Purchase of "

data-label="Single Sheet - $2.50">

</script>

</p>

</form>

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {require\_once('scripts/stripe/stripe-config.php');  
 $errors = array();  
 $token = $\_POST['stripeToken'];  
 $email = $\_POST['stripeEmail'];  
 if(isset($\_POST['customer\_id'])){ // if this is set, then the customer already exists...  
 try {  
 $charge = Stripe\_Charge::create(array(  
 'amount' => 250,  
 'currency' => 'usd',  
 'customer' => $\_POST['customer\_id'],  
 'description' => "Single Sheet purchase"  
 ));  
 } catch (Stripe\_CardError $e) {  
 // Card was declined.  
 $e\_json = $e->getJsonBody();  
 $err = $e\_json['error'];  
 $errors['stripe'] = $err['message'];  
 } catch (Stripe\_ApiConnectionError $e) {  
 // Network problem, perhaps try again.  
 } catch (Stripe\_InvalidRequestError $e) {  
 // You screwed up in your programming. Shouldn't happen!  
 } catch (Stripe\_ApiError $e) {  
 // Stripe's servers are down!  
 } catch (Stripe\_CardError $e) {  
 // Something else that's not the customer's fault.  
 }

If the customer does not exists in our servers (i.e. no stripe id), then we must create a new customer and then charge them…  
  
 } else { // customer does not exist, create one then charge the new customer  
 try {  
 // create customer  
 $customer = Stripe\_Customer::create(array(  
 'email' => $email,  
 'card' => $token  
 ));  
 // charge the customer  
 $charge = Stripe\_Charge::create(array(  
 'customer' => $customer->id,  
 'amount' => 250,  
 'currency' => 'usd',  
 'description' => 'Single Sheet Purchase'  
 ));  
 } catch (Stripe\_CardError $e) {  
 // Card was declined.  
 $e\_json = $e->getJsonBody();  
 $err = $e\_json['error'];  
 $errors['stripe'] = $err['message'];  
 } catch (Stripe\_ApiConnectionError $e) {  
 // Network problem, perhaps try again.  
 } catch (Stripe\_InvalidRequestError $e) {  
 // You screwed up in your programming. Shouldn't happen!  
 } catch (Stripe\_ApiError $e) {  
 // Stripe's servers are down!  
 } catch (Stripe\_CardError $e) {  
 // Something else that's not the customer's fault.  
 }  
 }  
}

if ($charge->paid == true) {  
$isStripeCustomer = fetchStripeId($loggedInUser->user\_id); //get stripe id for this user if it exists  
foreach ($isStripeCustomer as $stripeid){  
 if($stripeid['stripeid'] === '0'){ // if it equals 0 user is not a stripe customer yet // update member record with stripe customer id createCustomer($customer->id, $loggedInUser->user\_id); }  
}

// add user to quickie role unless it already exists  
if ($loggedInUser->checkPermission(array(4))){ // this is in class.user.php // return true;  
}else{  
addPermission(4, $loggedInUser->user\_id);  
}

// add one credit to this users credit tally, to be used at their discretion  
addCredit(1, $loggedInUser->user\_id);  
  
// echo transaction details out to the user. Prompt to email receipt if user desires

} else { // Charge was not paid!

if (isset($errors) && !empty($errors) && is\_array($errors)) {  
echo '<div class="col-md-8 col-md-offset-2 text-center" style="border:1px solid red;"><h3 class="text-center" style="color:#303030;">Error Log:</h2><ul>';

foreach ($errors as $e) {  
 echo "<li>$e</li>";  
}  
echo '</ul></div>';  
}  
}