# Braintrust Interactive - Abandon Cart App for Shopify

This document outlines a Shopify app that will leverage the ChatFuel Facebook Messenger platform to perform abandoned cart reclamation tasks. A proof of concept has proven that this is feasible and can be created with a simple app.

***Development is only required for the Shopify app as the ChatFuel portion has been designed and created.***

## How the App Works

### Syncing a Shopify Visitor with a Chat Bot Subscription

Using Facebook Messenger Checkbox code generated and provided by ChatFuel, the app will insert the checkbox code beneath a product page’s Add to Cart button.

Clientside JavaScript included in the inserted code would also attach an event listener to the Add to Cart button that when triggered, would create a delimited string containing the Shopify store’s ID (or domain), along with a unique identifier for the user that would allow tying the current user to their Cart (cart token, session, or other cookie – TBD).

Shopify’s Cart AJAX code could be used to fetch unique identifiers for use with Messenger.

The delimited string would then be base64 encoded and used as the REF parameter when calling Messenger’s window.confirmOptIn function (this code has already been written and will be provided).

When Messenger’s window.confirmOptIn function is called, this subscribes the user to the store owner’s chat bot, which is powered by the ChatFuel platform.

The dynamically generated, delimited REF parameter is passed into the chat bot. Logic built into the Chat Bot would then trigger a POST back to the Shopify app with the REF parameter, the bot subscriber’s name, and unique Messenger ID.

The Shopify app would receive the POST, parse the data and store it in a database table. The data stored would be the bot subscriber’s Messenger ID, name, Shopify ID and unique cart identifier. That record would also have an “active” tag and date stamp added.

The app would then respond with strictly formatted JSON that sets several user attributes inside the ChatFuel platform. *This code has already been developed as a PHP class that makes it easy to respond to ChatFuel with the properly formatted JSON.*

Should the user check out and complete an order, the record stored in the table would be marked as inactive.

### Triggering Cart Abandonment Messages

When the user is subscribed to the Chat Bot, they are subscribed to timed sequences inside the bot that trigger at 1 hour, 21 hours and 48 hours after the user has added an item to their cart.

In the initial sequence, the bot would perform another POST to the Shopify App, again sending the Shopify store ID, the cart identifier, and the Messenger ID.

The app would then perform a lookup based on the posted data and retrieve the cart contents should the cart still exist (abandon, not checked out).

Based on app settings, the cart would again set user attributes in the Chatfuel platform and generate a preformatted, JSON based message that included a call to action, and an image of the least expensive item left in the cart linked with several buttons that take the user to the store’s cart page or the related items page.

The app may also include a call to action with a coupon code that was previously generated by the store owner.

Again, all the JSON data sent back to ChatFuel is easily generated and rendered using the already created framework.

If the cart doesn’t exist (user checked out), then the app would simply set a user attribute in the ChatFuel platform that would unsubscribe the user from future cart abandonment checks.

Subsequent cart abandonment checks would use content and calls to action based on several basic settings inside the app.

### Order notifications

It would be preferable that the app be able to notify the user of order updates/shipping when orders are updated in the initial phase of development, but could be pushed to a second phase of development.

For order notifications, the app would need to either use a cron job or Shopify webhooks to trigger a message to bot users that their order has changed status.

Using the ChatFuel broadcast API, the app could set basic user attributes in the bot platform and trigger a block inside the bot to message the user with order updates. Code for performing a broadcast to ChatFuel users has already been developed – the app would simply need to format the URL and perform a single CURL operation.

## App Options

The app would have limited options:

* ChatFuel Bot ID
* Chatfuel CheckBox Code
* Order Receipt Message
* Fulfillment/Shipping Notification Message

The following options would be duplicated for message 1, 2 and 3 so you could have more insistent CTAs:

* Abandon Cart Message: "Hey {{first name}}, it looks like you forgot something!"
* How Many items to Display: (1-3)
* Item Title: Product Title (or other?)
* Sub Title: price/description/other?
* Display Order (price asc/desc)
* Button 1 Text: Buy Now, Checkout Now, Other
* Button 1 Link: Defaults to Cart Link
* Button 2 Text: Related Items, Other
* Button 2 Link: Related Items, Store Link, Other
* Offer Coupon Code? Y/N
* Coupon Code Message "If you complete your order, use coupon code {{coupon code}} at checkout for {{coupon effect}}!"
* Coupon Code (select from existing coupon codes)

## Database tables

The app would need a limited number of tables:

* Store table – store app authentication data and all store details provided by Shopify, in addition to an active/inactive flag for management
* Options table – store app options
* User table – store messenger IDs with store IDs for cart lookups and order notifications
* Cart table – store cart data with store ID on a per store, per user basis for cart abandonment

## App Platform

The app would need to be designed as an embedded app for Shopify using the resources listed at the end of this document.

The app should be developed in PHP (Laravel, Cake, etc) and will leverage a MySQL database.

## App Installation

Initially, the app would not be published on the app marketplace, so installation would be manually performed on stores. Shopify has recently updated methods for this, so would need to be easily implemented.

## Resources for Development

Shopify’s Embedded App SDK  
<https://help.shopify.com/api/sdks/shopify-apps/embedded-app-sdk>

Shopify’s AJAX API  
<https://help.shopify.com/themes/development/getting-started/using-ajax-api>

Shopify’s Webhooks  
<https://help.shopify.com/manual/sell-online/notifications/webhooks>

MicroApps EASDK Embedded App Frontend Framework  
<http://seaff.microapps.com/>

ChatFuel  
<http://docs.chatfuel.com/>  
<http://docs.chatfuel.com/plugins/plugin-documentation/json-api>  
<http://docs.chatfuel.com/broadcasting/broadcasting-documentation/broadcasting-api>