# CHAI

**NET412 Capstone Project**

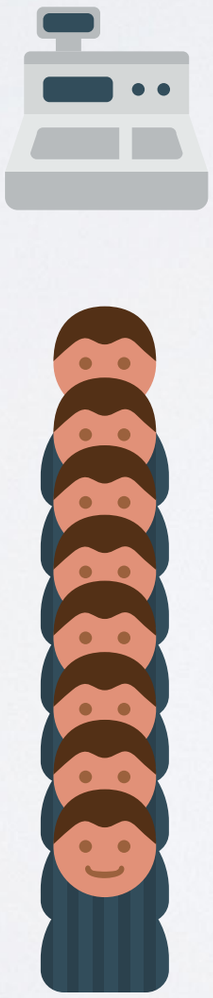
Danny Yiu
Marthe Nsaba
Kenneth Sullivan

https://github.com/dannyyiu/410coffee

# CURRENT PROBLEM

## PEAK HOUR

### CASHIER
(100%)



### KITCHEN
(70%)

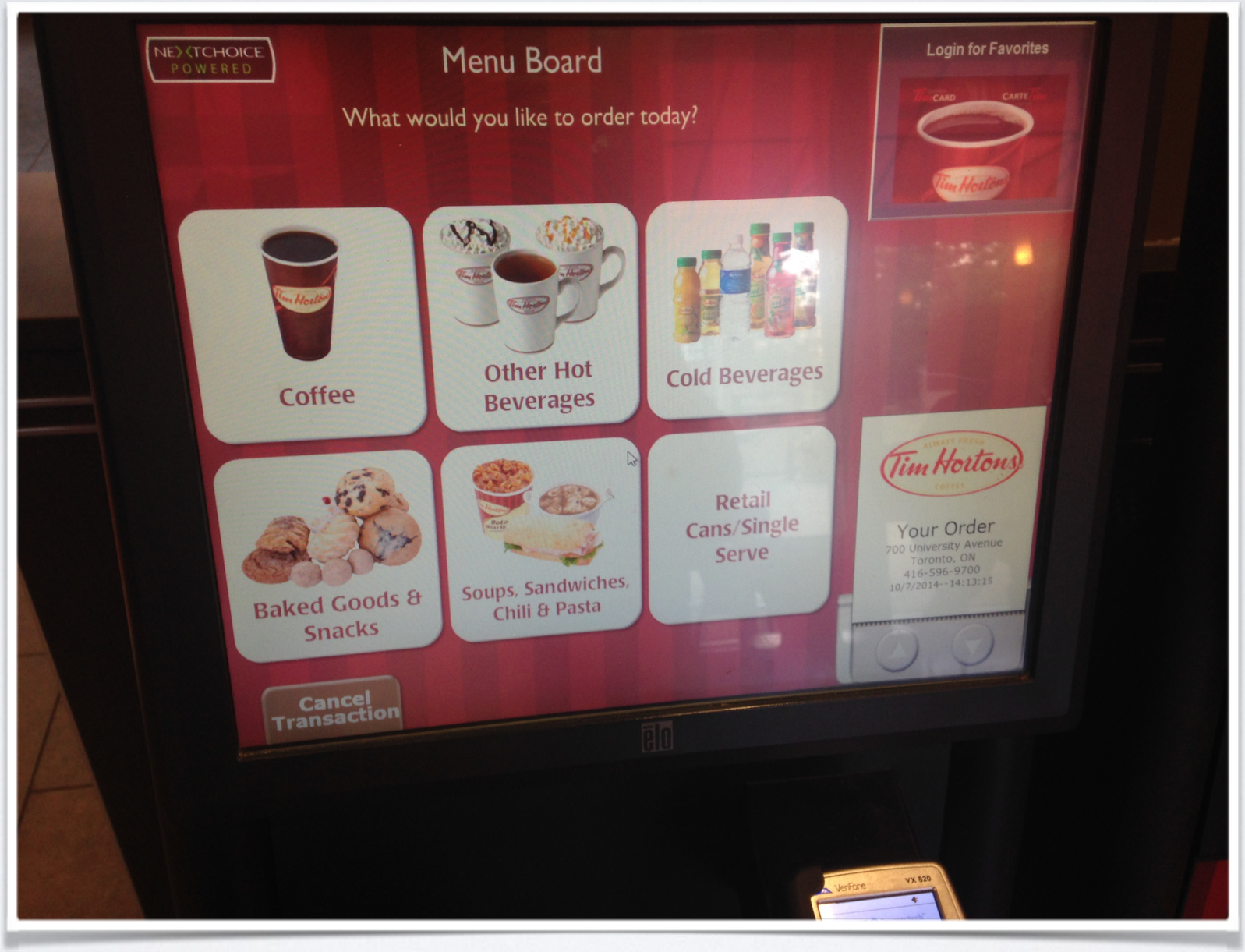
# SOLUTION

## PEAK HOUR

### CASHIER
(60%)

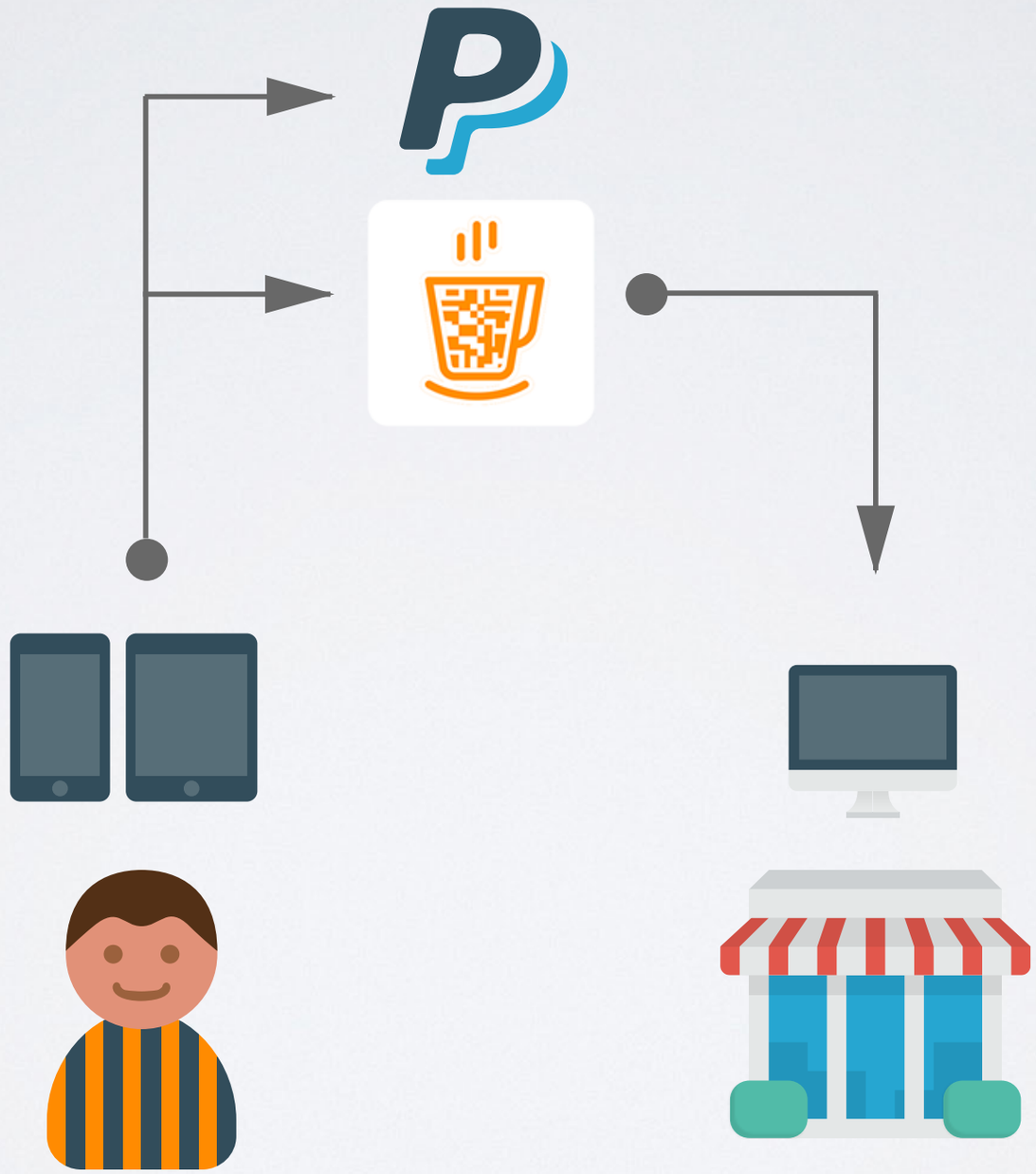
### KITCHEN
(100%)

# CURRENT ATTEMPTS

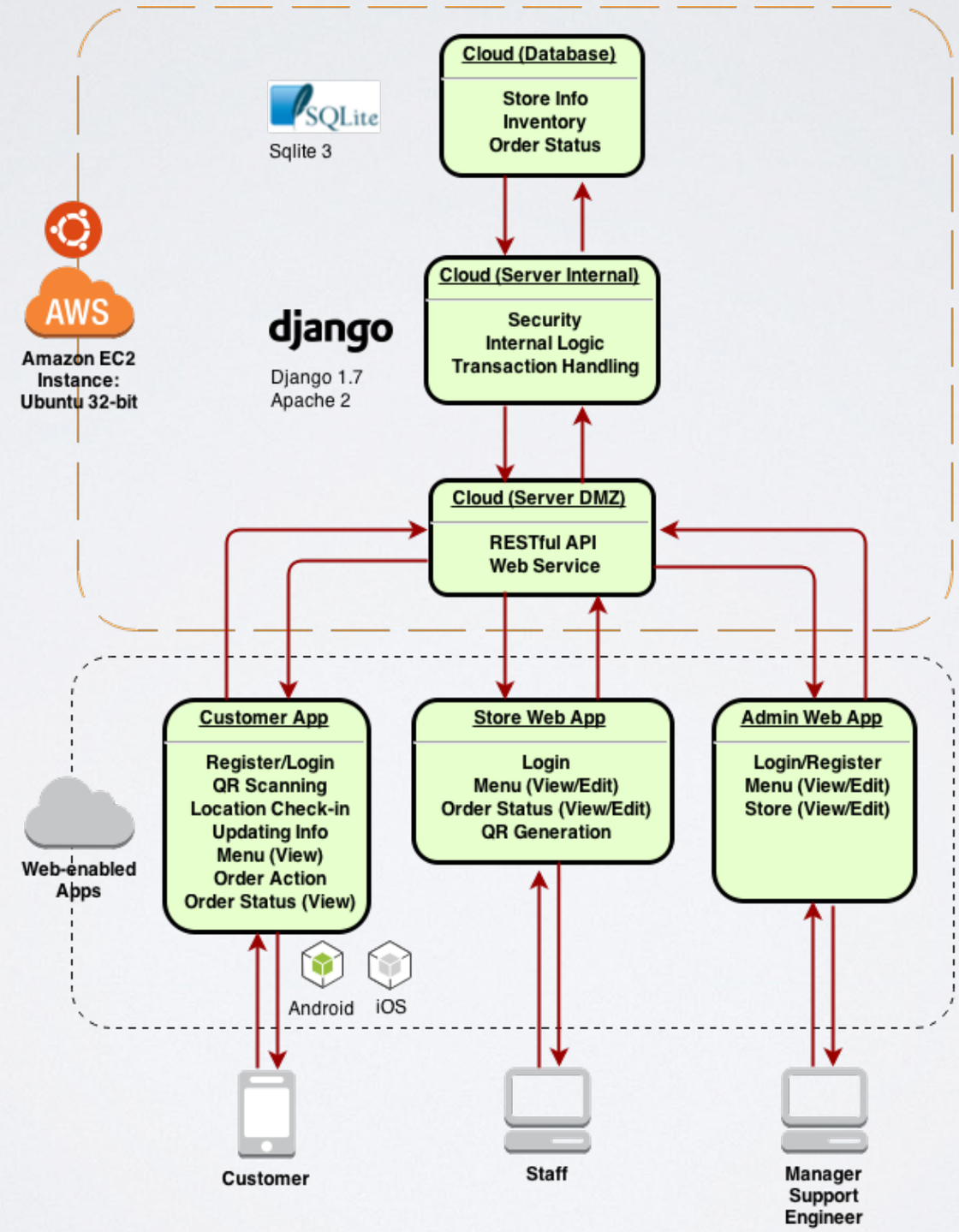



Tim Hortons ticket machine, at College/University

# SOLUTION OVERVIEW



# LOGIC FLOWCHART



# COMPONENTS

① ②

# CHAI APP

Interface

# DEMO

Store landing (no password): https://chaiapp.tk/store-api
Store landing (password): https://chaiapp.tk/store-danny_store
Manager page: https://chaiapp.tk/admin

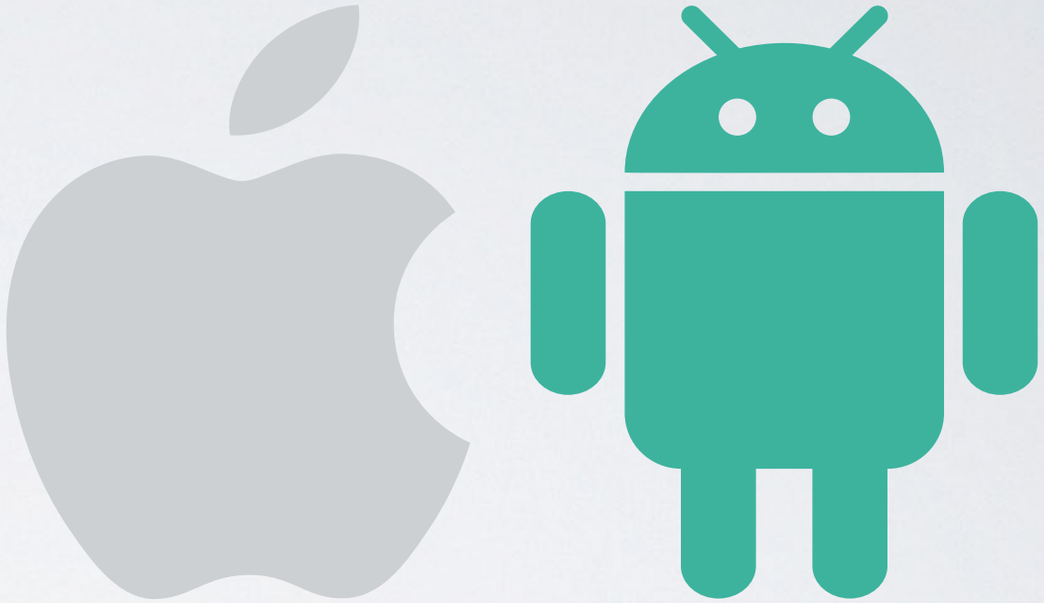
Customer PayPal accounts:
chai.buyer.1@gmail.com (Pass: 00000000)
chai.buyer.2@gmail.com (Pass: 00000000)

Store/manager accounts:
chaiadmin (pass: 00000000)
storemanager (pass: 00000000)

# CHAI APP

Logic

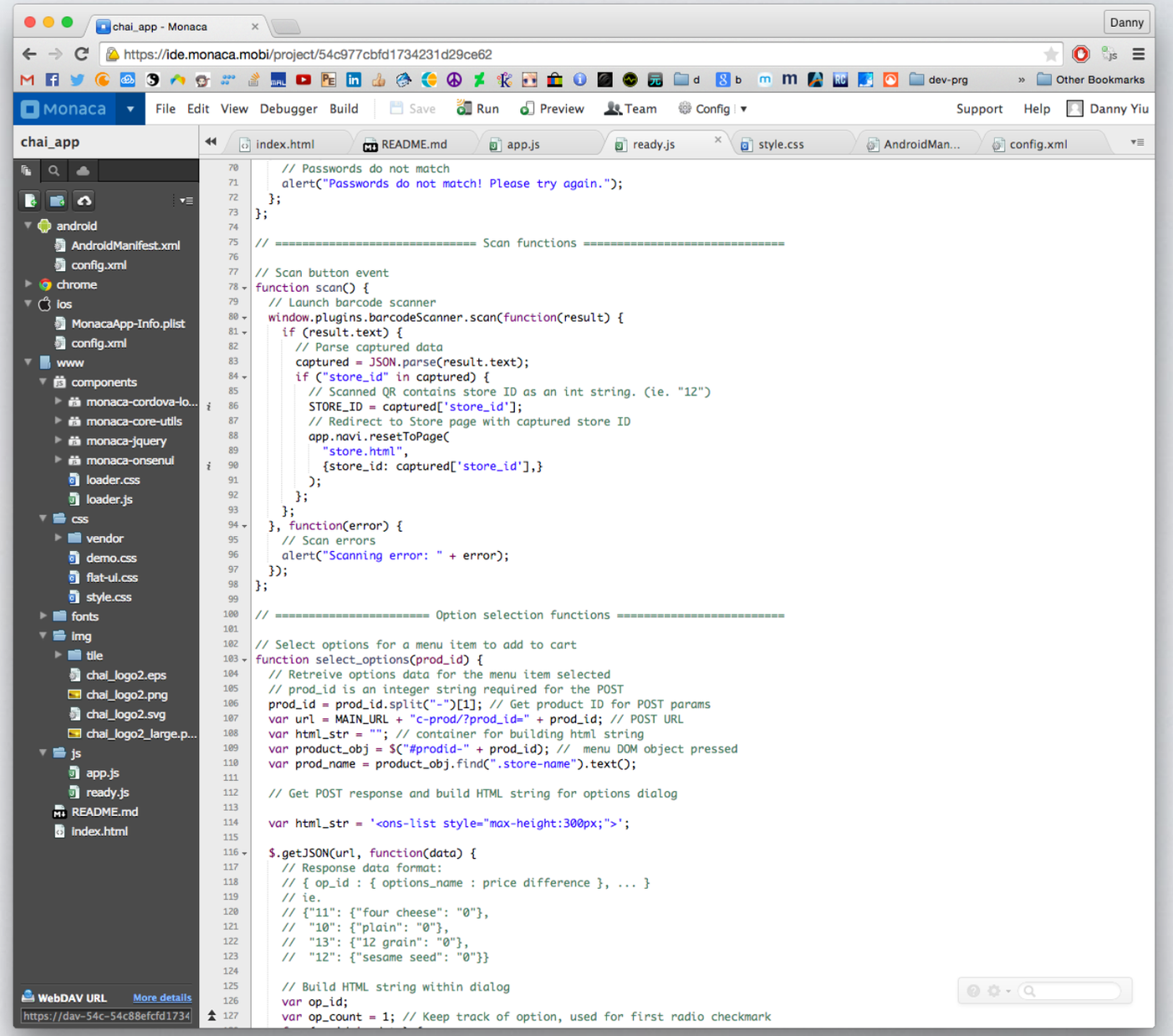
# SPECIFICATIONS



- Android (Target API: 19)
- iOS (Minimum 8.0)
- Cordova 4.1.0

# DEVELOPMENT

- Monaca.io
- Cross-platform
- HTML/CSS, JavaScript/jQuery, AngularJS

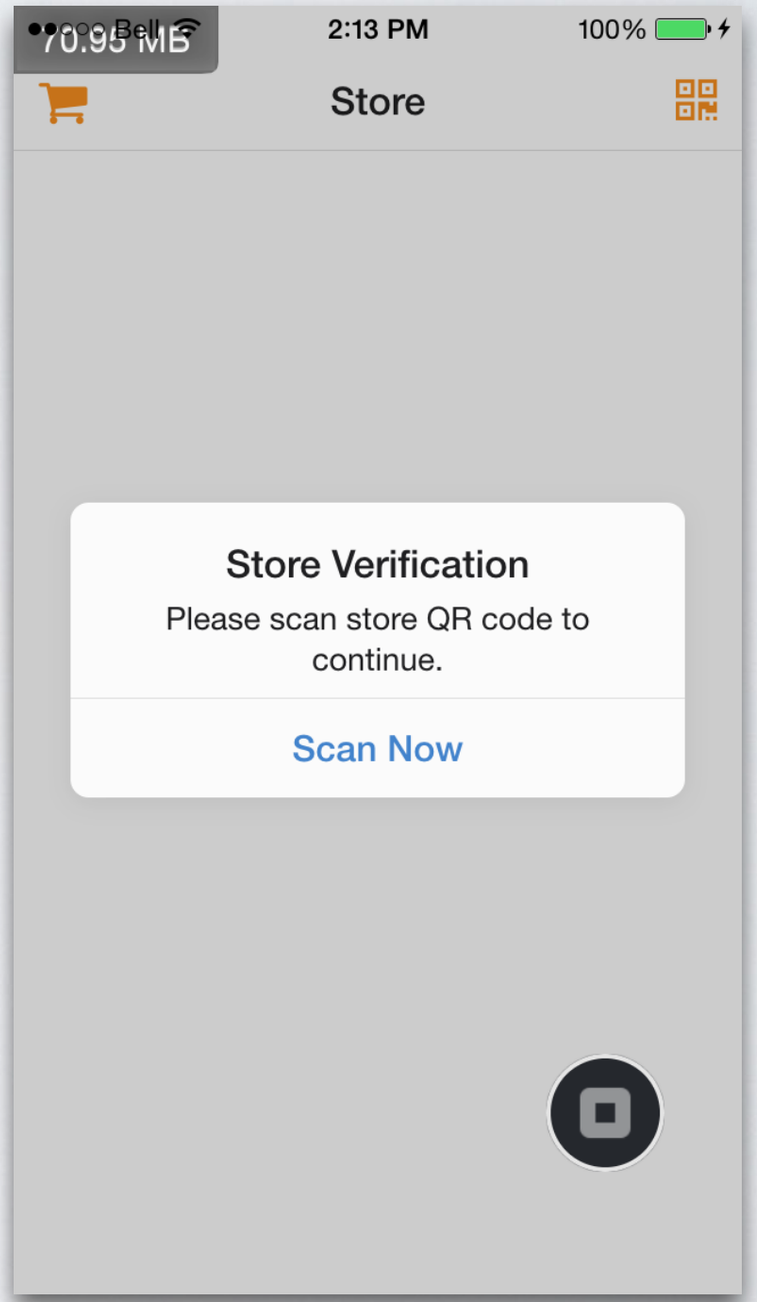



Store
Store
Scan
Favorites
Coffee
Cold Drinks
Snacks
Breakfast
Meals
Product
$4.75
Product
$4.75
QR Scanner
QR Scanner
Payment
Payment
Full Name
Card Number
Expiration
Month
Year
CVC
Credit
Debit
or
PayPal
PURCHASE
Order Status
Order Status
Order Number:
#501
Queue

# STORE: BEFORE SCAN

- User must verify presence in store.
- Logic: If no store ID, prompt scan.

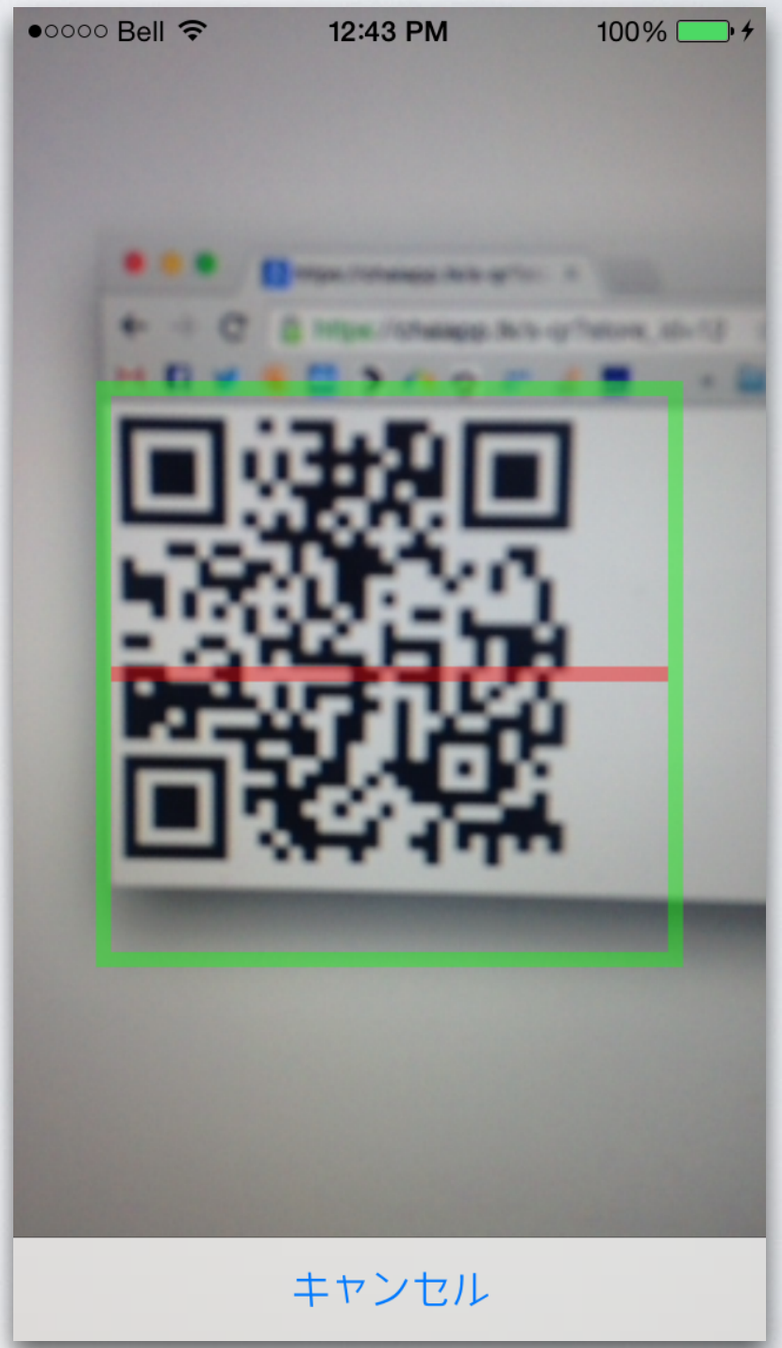
```
// ============================== Store page ====================
$(document).on('pageinit', '#store-page', function() {
  // Get options
  var options = app.navi.getCurrentPage().options;

  // Check for store_id
  // store_id is only passed if QR code for store is scanned
  if ("store_id" in options) {⟺
  } else {
    // Not inside store, or not scanned QR yet
    ons.notification.alert({
      message: 'Please scan store QR code to continue.',
      title: 'Store Verification',
      buttonLabel: 'Scan Now',
      animation: 'default', // or 'none'
      // modifier: 'optional-modifier'
      callback: function() {
        scan();
      }
    });
  }
});
```



# QR SCANNER

- BarcodeScanner Cordova plugin.
- Logic: Return scanned result, or return error message.

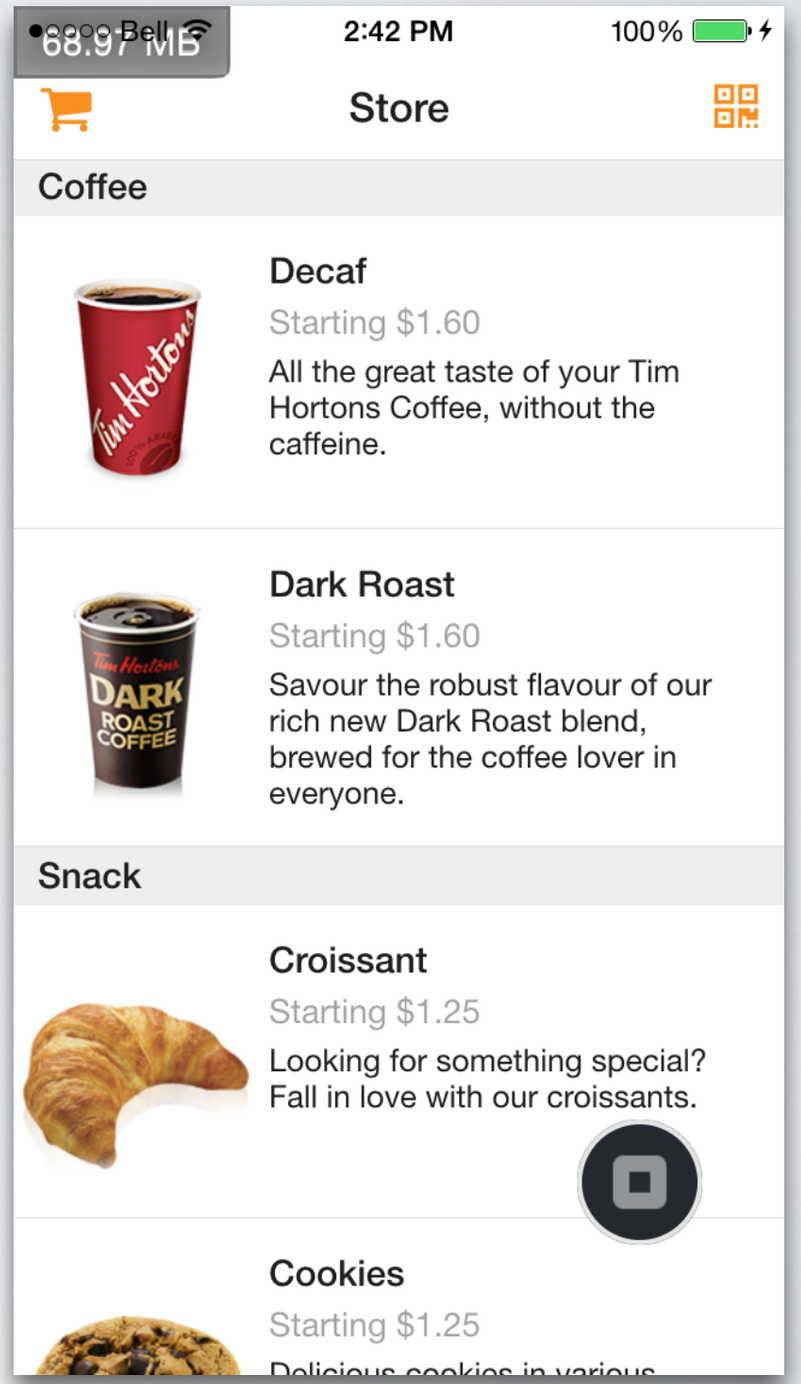
```
// Scan button event
function scan() {
  // Launch barcode scanner
  window.plugins.barcodeScanner.scan(function(result) {
    if (result.text) {
      // Parse captured data
      captured = JSON.parse(result.text);
      if ("store_id" in captured) {
        // Scanned QR contains store ID as an int string. (ie. "12")
        STORE_ID = captured['store_id'];
        // Redirect to Store page with captured store ID
        app.navi.resetToPage(
          "store.html",
          {store_id: captured['store_id'],}
        );
      };
    };
  }, function(error) {
    // Scan errors
    alert("Scanning error: " + error);
  });
};
```



# STORE: MENU

- Logic: Call c-menu API using store ID from QR. Display store menu with response JSON data.

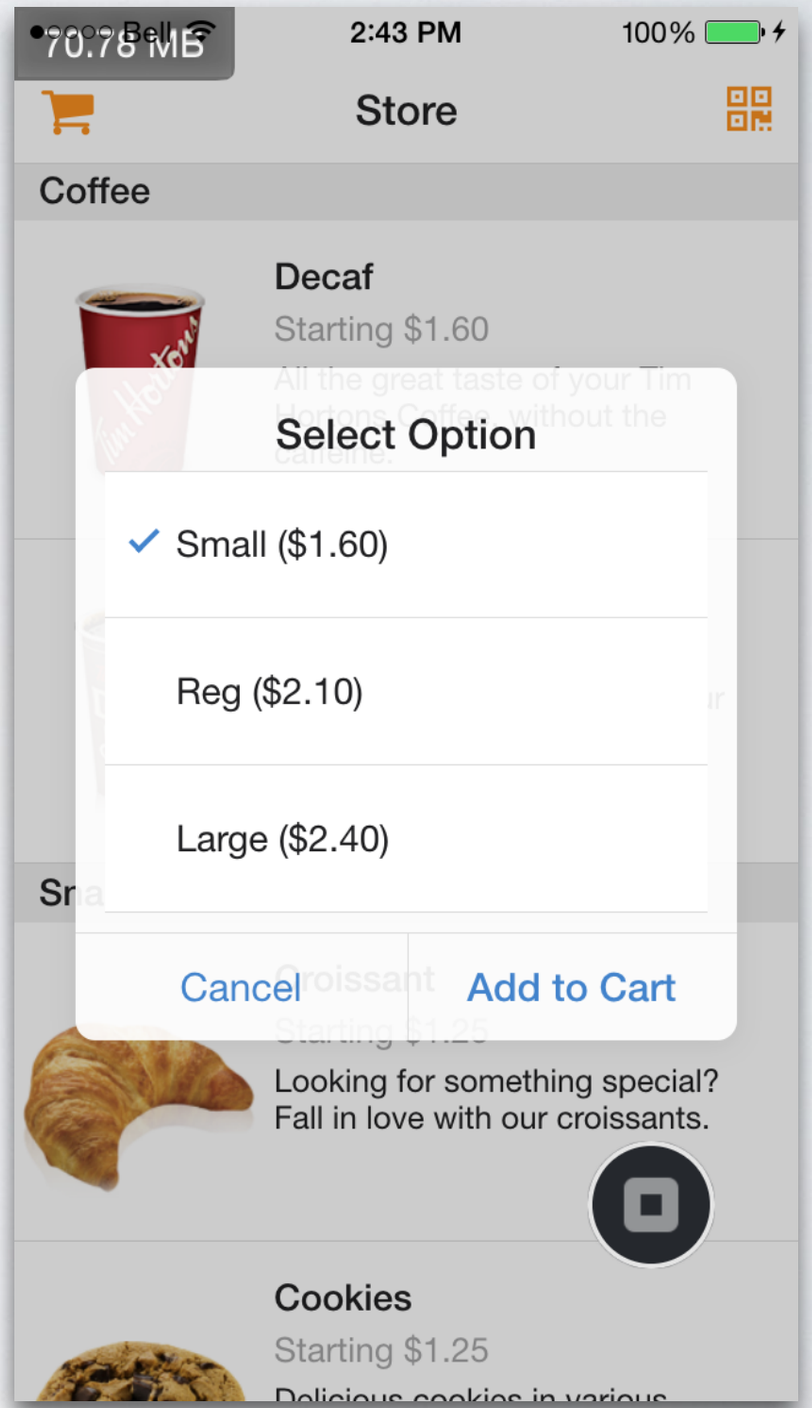
```
// ============================== Store page ==============================
$(document).on('pageinit', '#store-page', function() {
  // Get options
  var options = app.navi.getCurrentPage().options;

  // Check for store_id
  // store_id is only passed if QR code for store is scanned
  if ("store_id" in options) {
    // Inside valid store

    // Send request for store menu
    var url = MAIN_URL + "c-menu/?store_id=" + options.store_id;
    $.getJSON(url, function(data) {
      // Store data as json string in hidden div
      $("#inventory-json").html(JSON.stringify(data));
      var html = ""; // empty inner HTML container to store new content
      var category; // string of category name from JSON response
      var content = document.getElementById("store-list");

      for (category in data) {
        // HTML for each category
        // Note: category may be lower case, so capitalize CSS is used
        if (!(jQuery.isEmptyObject(data[category]))) {⟸};
      };
      // Update store menu list
      content.innerHTML = html;
      ons.compile(content);
    });
```



# STORE: OPTIONS

- Logic: Call c-prod API using product ID from menu. Display options with response JSON data.

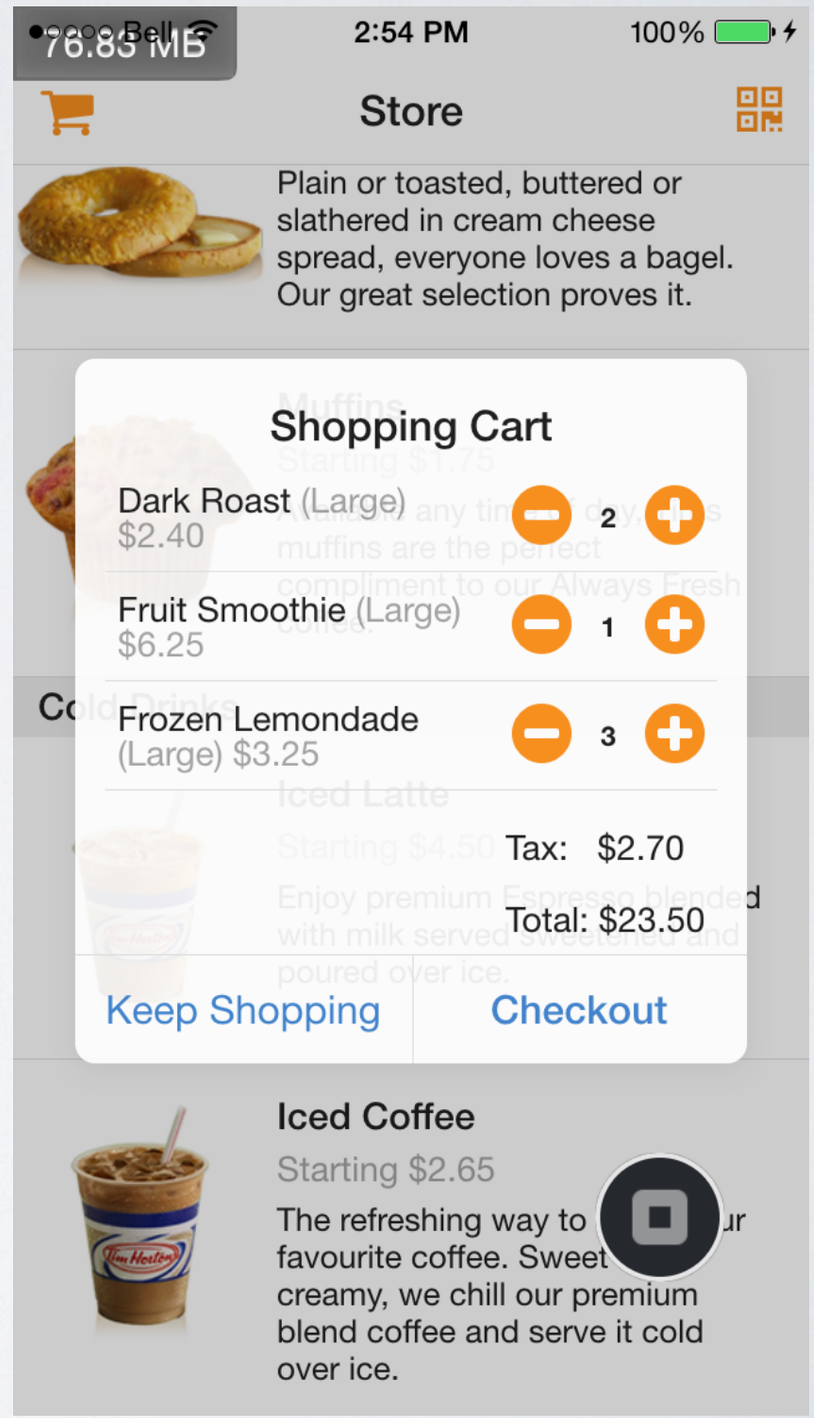
```
$.getJSON(url, function(data) {
  // Response data format:
  // { op_id : { options_name : price difference }, ... }
  // ie.
  // {"11": {"four cheese": "0"},
  //  "10": {"plain": "0"},
  //  "13": {"12 grain": "0"},
  //  "12": {"sesame seed": "0"}}

  // Build HTML string within dialog
  var op_id;
  var op_count = 1; // Keep track of option, used for first radio checkmark
  for (op_id in data) {

    var op_name;
    for (op_name in data[op_id]) {⟷};
    if (op_count == 1) {
      op_count = 2; // Only first count required for radio checklist default
    }
  };
  html_str += "</ons-list>";

  // Put HTML string into a dialog for option selection
  ons.notification.confirm({⟷});
```



# STORE: SHOPPING CART

- Logic: Append product and option to cart HTML. Hide product from the menu. Show HTML in dialog.

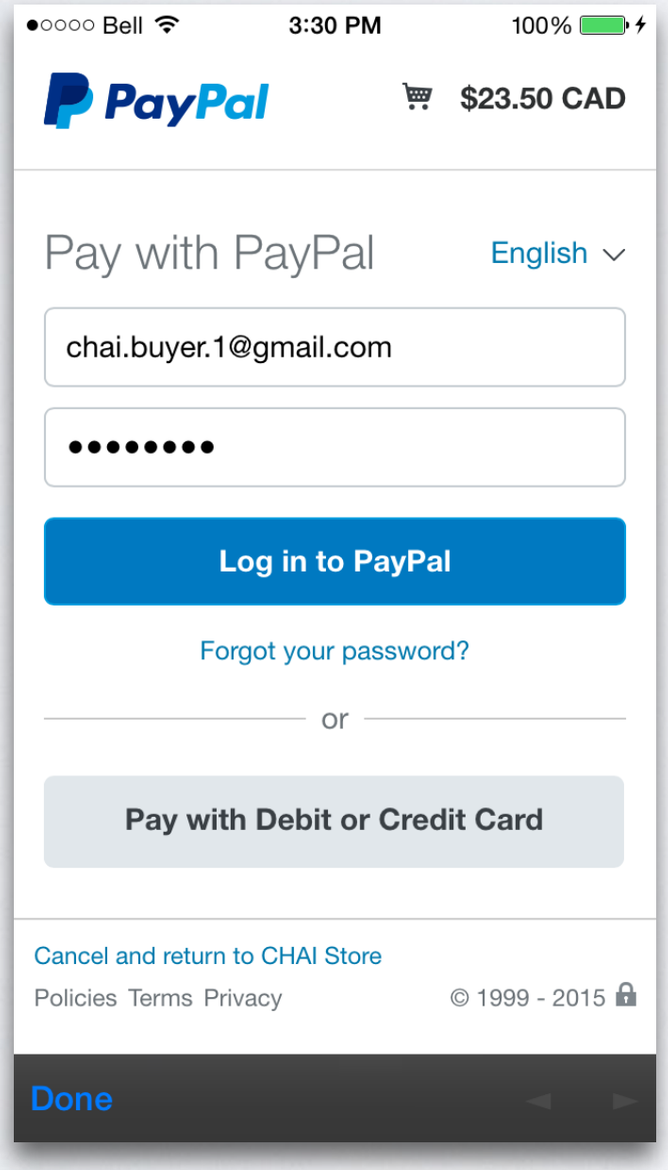
```
// Update cart
cart_append_html(html_str);

// Disable item in store menu
var prod_id = "prodid-" + get_prod_id(product);
$("#" + prod_id).hide(); // hide menu item

// Show cart
cart();
```

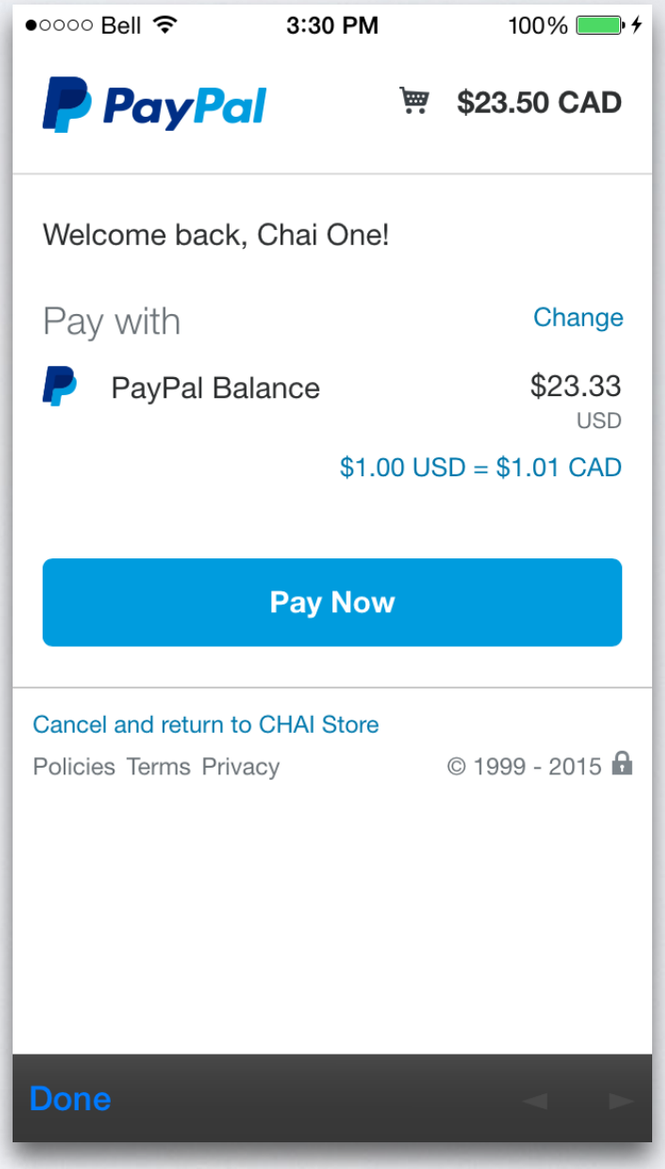


# PAYMENT

- c-token API call (SetExpressCheckout).
  Note: Why not call directly to PayPal?

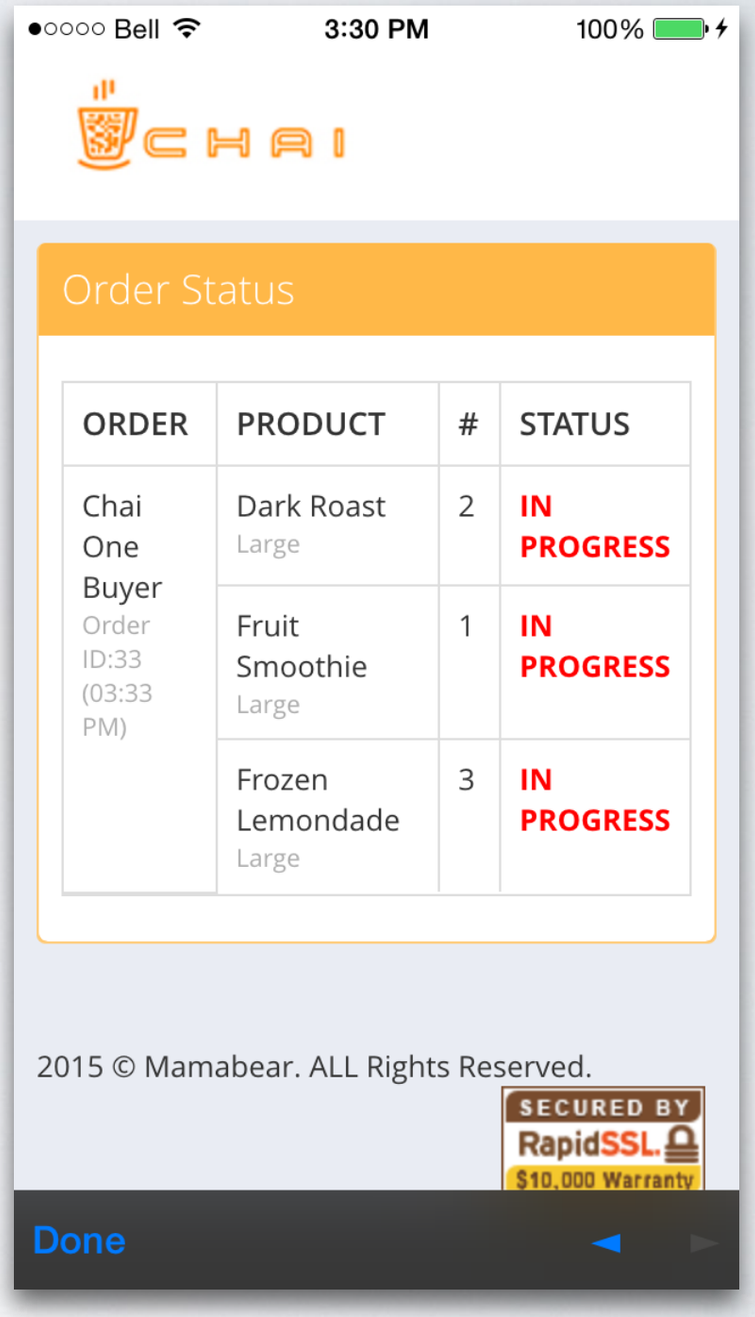
- PayPal In-context Express Checkout URL + token

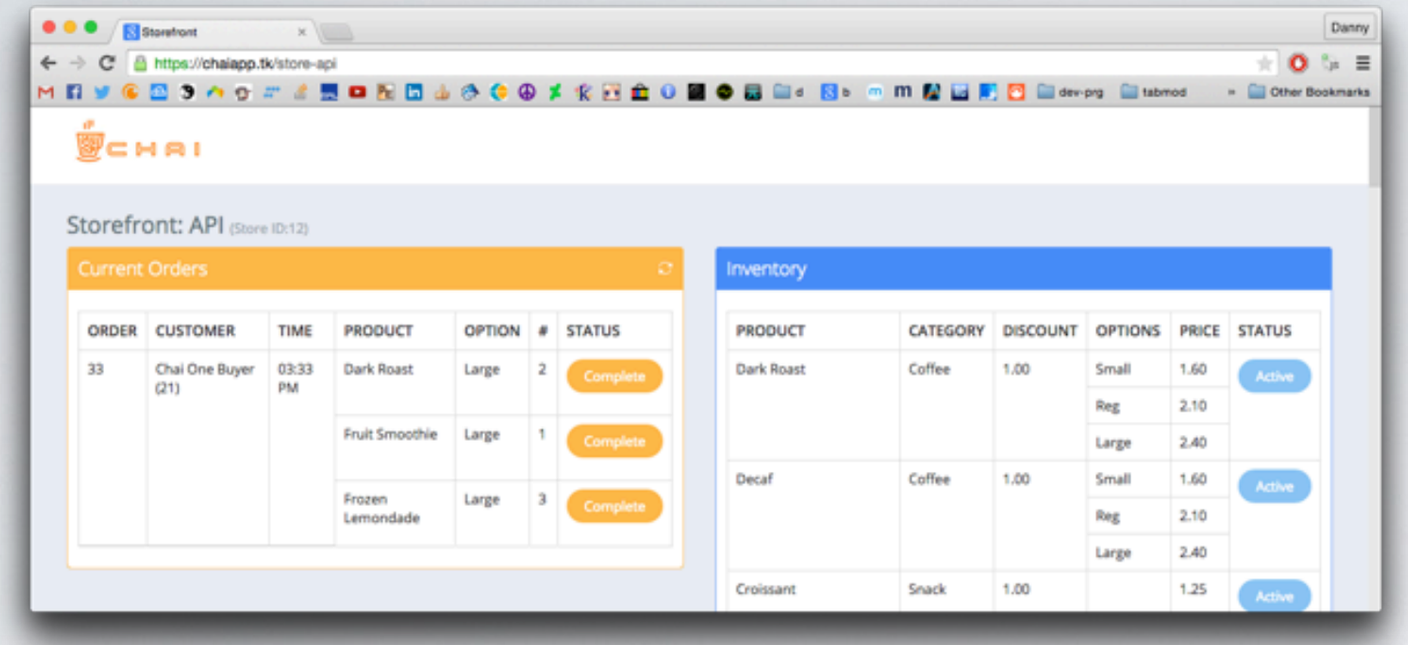


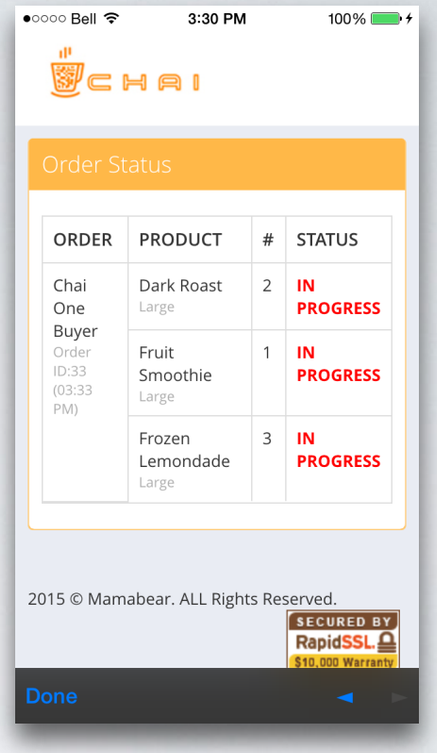



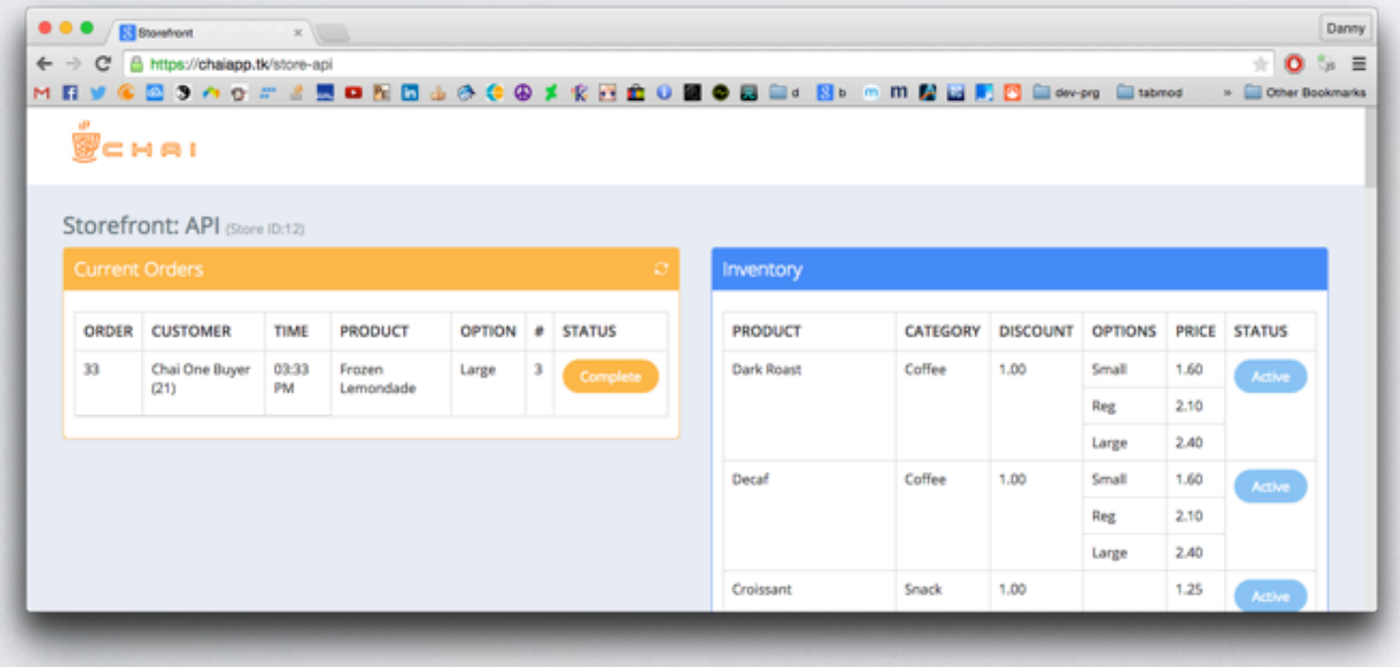
# ORDER STATUS

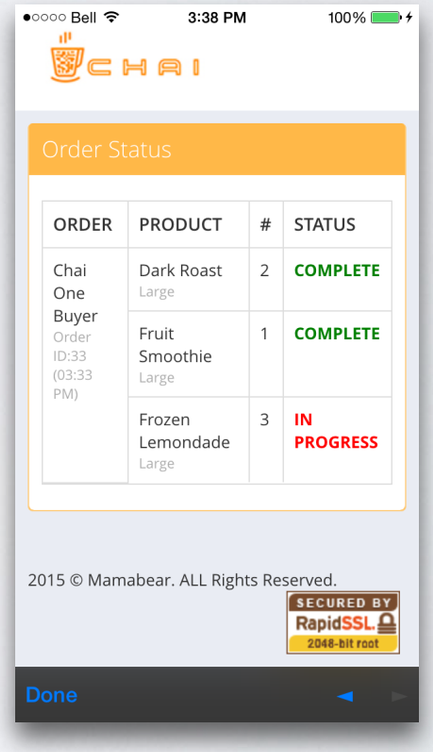
- c-status-<store name> API
- GetExpressCheckoutDetails
- DoExpressCheckoutPayment




Storefront
Danny
https://chaiapp.tk/store-api
dev-prg
tabmod
Other Bookmarks
CHAI
Storefront: API (Store ID:12)
Current Orders
ORDER CUSTOMER TIME PRODUCT OPTION # STATUS
33 Chai One Buyer (21) 03:33 PM Dark Roast Large 2 Complete
Fruit Smoothie Large 1 Complete
Frozen Lemondade Large 3 Complete
Inventory
PRODUCT CATEGORY DISCOUNT OPTIONS PRICE STATUS
Dark Roast Coffee 1.00 Small 1.60 Active
Reg 2.10
Large 2.40
Decaf Coffee 1.00 Small 1.60 Active
Reg 2.10
Large 2.40
Croissant Snack 1.00 1.25 Active


●○○○○ Bell
3:30 PM
100%
CHAI
Order Status
ORDER PRODUCT # STATUS
Chai One Buyer Order ID:33 (03:33 PM)
Dark Roast Large 2 IN PROGRESS
Fruit Smoothie Large 1 IN PROGRESS
Frozen Lemondade Large 3 IN PROGRESS
2015 © Mamabear. ALL Rights Reserved.
SECURED BY RapidSSL $10,000 Warranty
Done


Storefront
Danny
https://chaiapp.tk/store-api
dev-prg
tabmod
Other Bookmarks
CHAI
Storefront: API (Store ID:12)
Current Orders
ORDER CUSTOMER TIME PRODUCT OPTION # STATUS
33 Chai One Buyer (21) 03:33 PM Frozen Lemondade Large 3 Complete
Inventory
PRODUCT CATEGORY DISCOUNT OPTIONS PRICE STATUS
Dark Roast Coffee 1.00 Small 1.60 Active
Reg 2.10
Large 2.40
Decaf Coffee 1.00 Small 1.60 Active
Reg 2.10
Large 2.40
Croissant Snack 1.00 1.25 Active

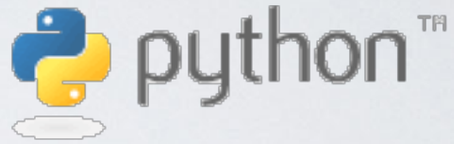

●○○○○ Bell
3:38 PM
100%
CHAI
Order Status
ORDER PRODUCT # STATUS
Chai One Buyer Order ID:33 (03:33 PM)
Dark Roast Large 2 COMPLETE
Fruit Smoothie Large 1 COMPLETE
Frozen Lemondade Large 3 IN PROGRESS
2015 © Mamabear. ALL Rights Reserved.
SECURED BY RapidSSL 2048-bit root
Done

# CLOUD

Logic

# CLOUD SPECIFICATIONS

- SQLite3
- Django 1.7
- Apache 2
- Ubuntu (32-bit)
- Amazon EC2

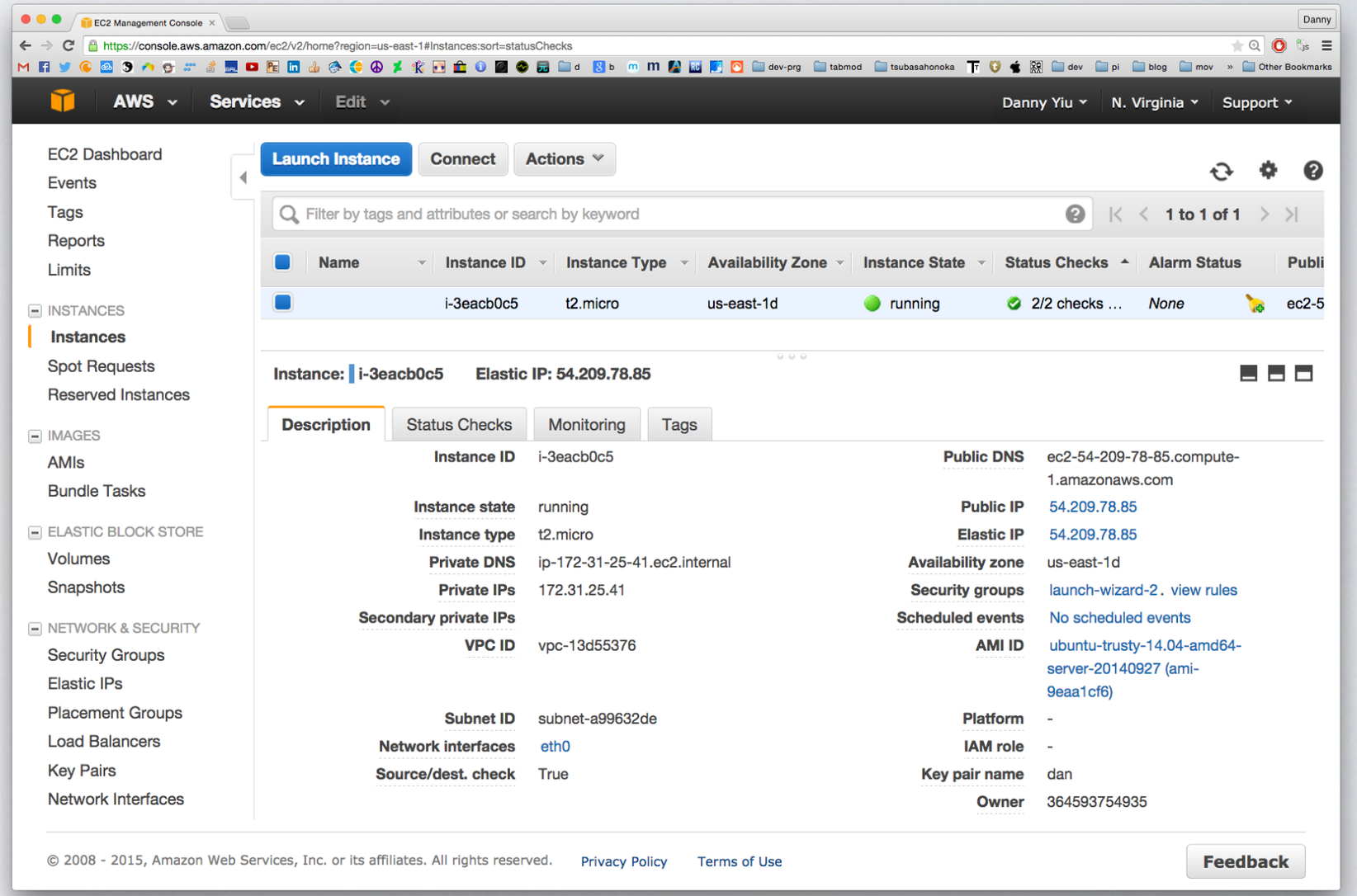



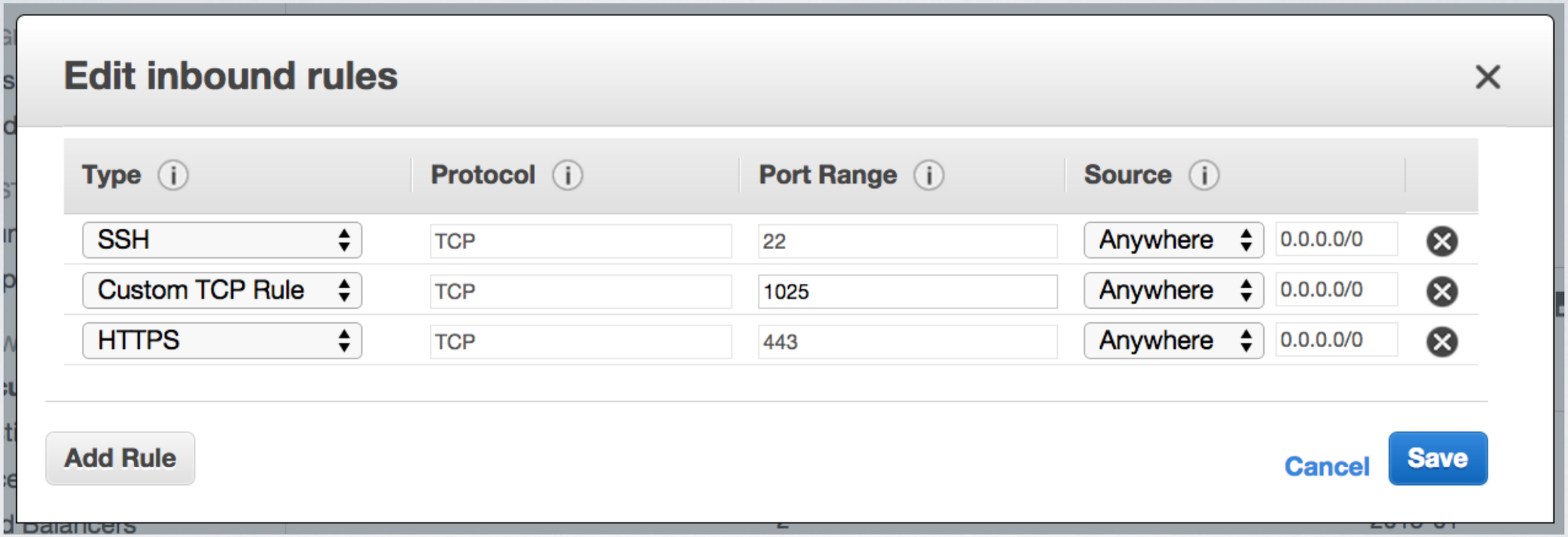




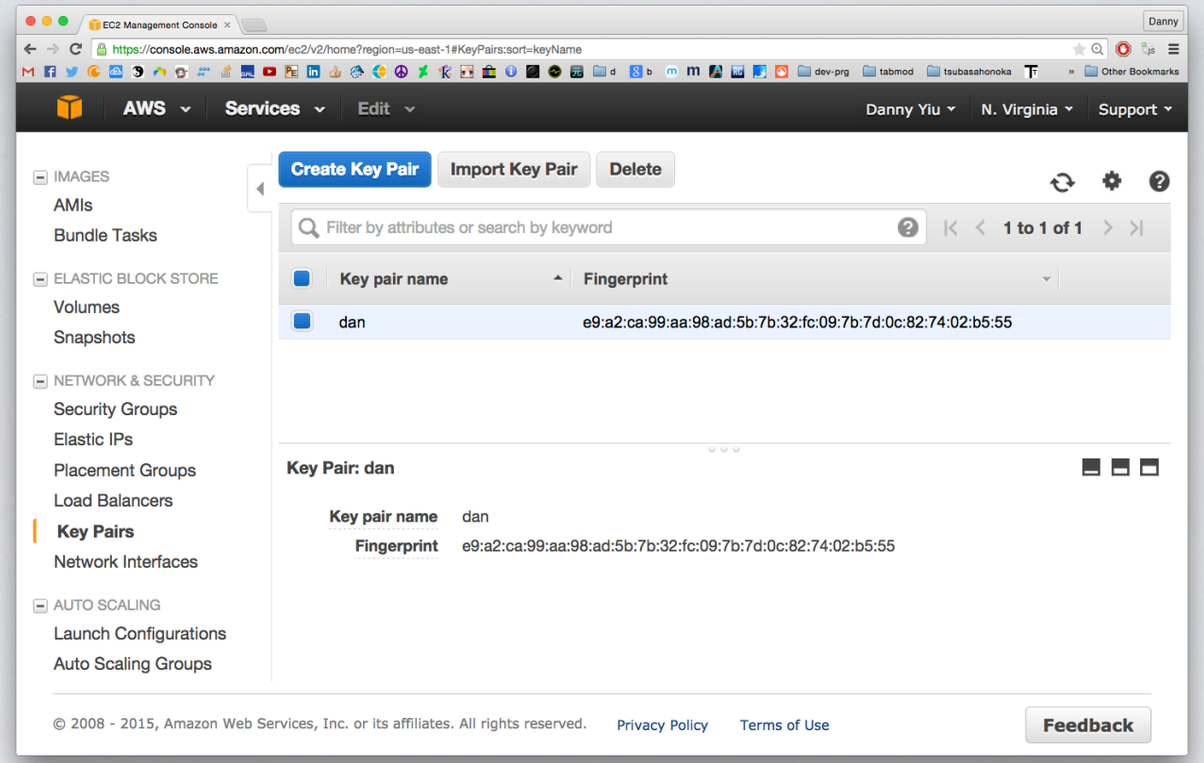
# EC2 INSTANCE

- t1.micro / t2.micro
- Elastic IP
- Security Policies
- Key Pair
- DNS

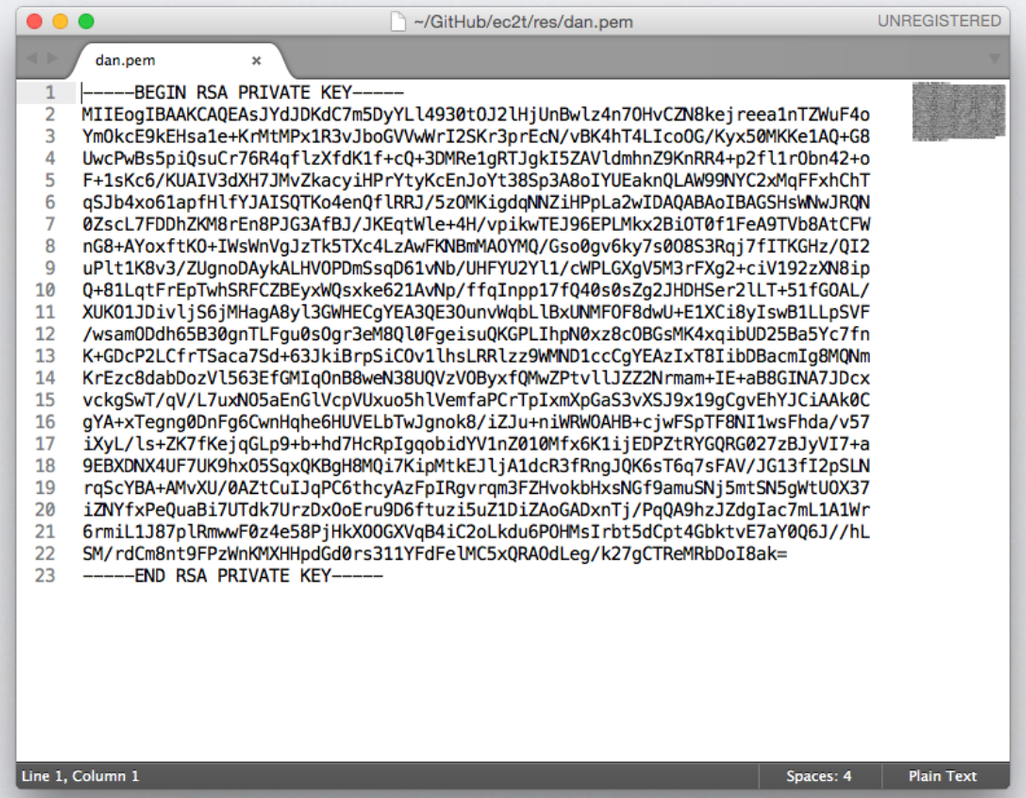


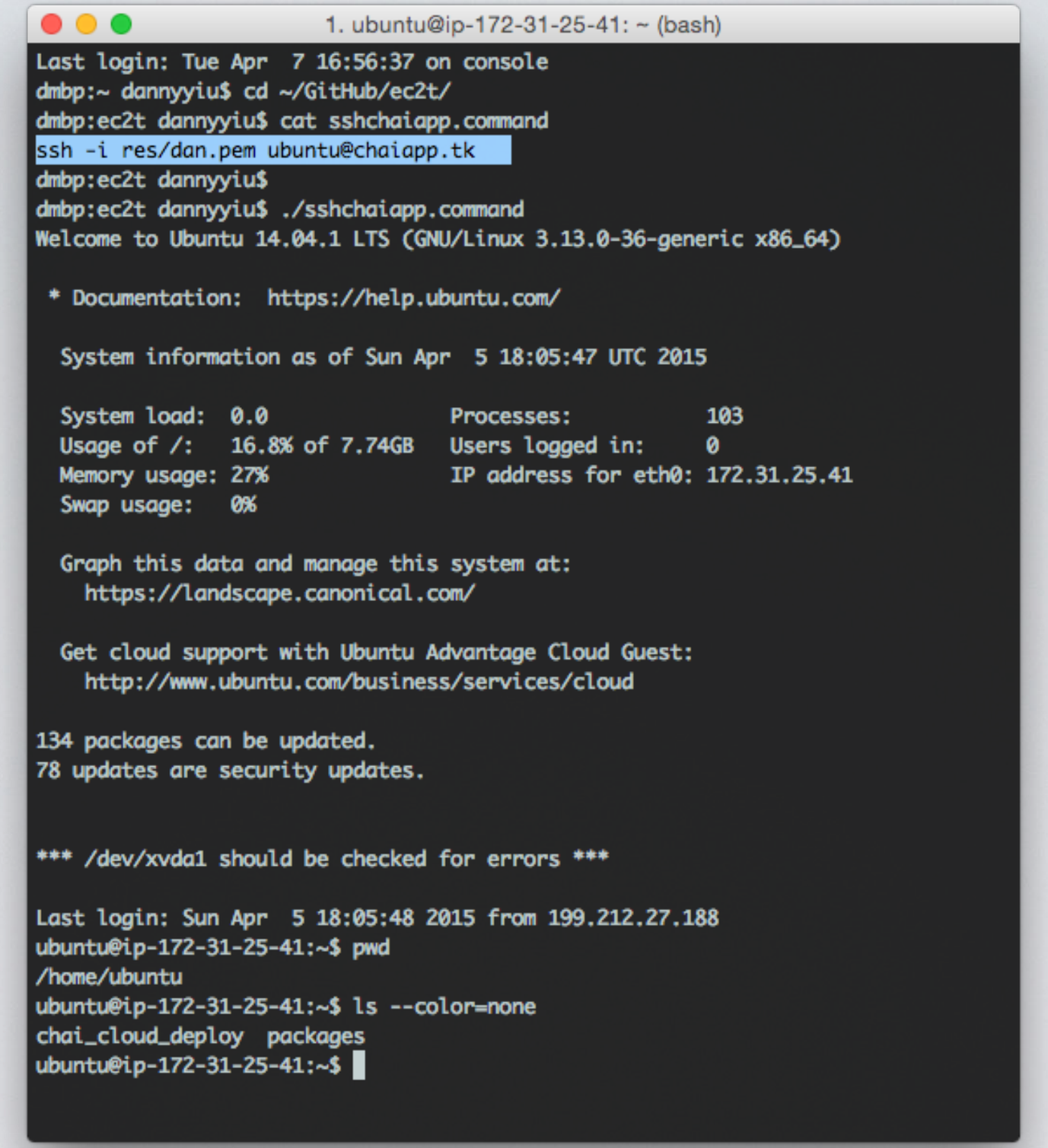



Inbound rules example




# Key pair generation

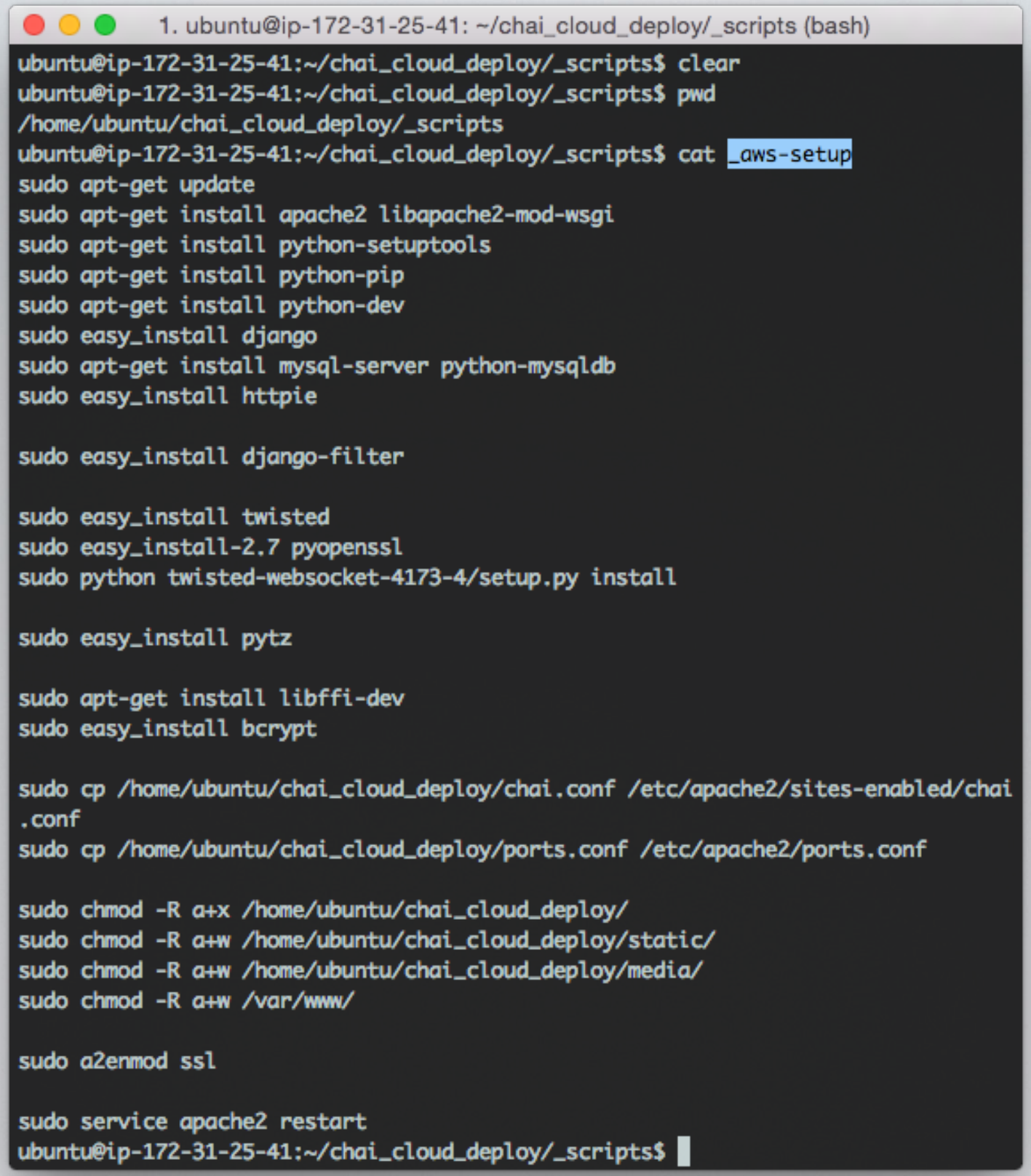





SSH connection with EC2 instance (instead of Elastic Beanstalk).

# SOFTWARE STACK

- Ubuntu (OS and application dependencies)
- Apache + Twisted (Web servers)
- Python Libraries (Django and library dependencies)




Installation script for entire software stack.

# APACHE CONFIG

1. ubuntu@ip-172-31-25-41: /etc/apache2/sites-enabled (bash)

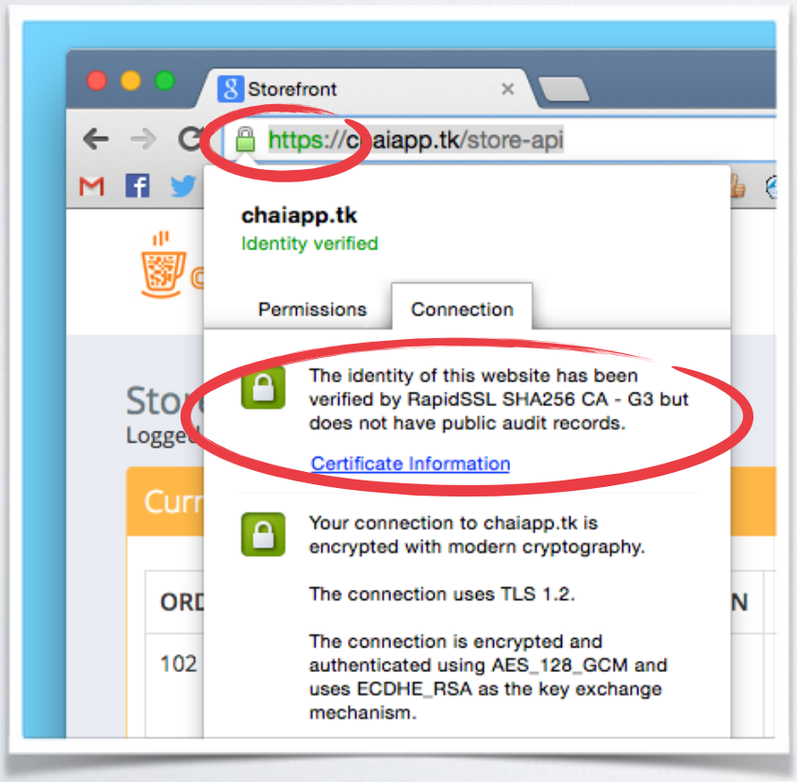
```
ubuntu@ip-172-31-25-41:/etc/apache2/sites-enabled$ pwd
/etc/apache2/sites-enabled
ubuntu@ip-172-31-25-41:/etc/apache2/sites-enabled$ cat chai.conf
WSGIScriptAlias / /home/ubuntu/chai_cloud_deploy/chai_cloud/wsgi.py
WSGIPythonPath /home/ubuntu/chai_cloud_deploy/
<Directory /home/ubuntu/chai_cloud_deploy/chai_cloud>
    <Files wsgi.py>
        Order deny,allow
        Require all granted
    </Files>
</Directory>

<Directory /var/www/static>
    Require all granted
</Directory>

<Directory /home/ubuntu/chai_cloud_deploy/media>
    Require all granted
</Directory>

<VirtualHost *:443>
    SSLEngine On
    SSLCertificateFile /etc/apache2/ssl-certs/rapidssl_publickey_2015.crt
    SSLCertificateKeyFile /etc/apache2/ssl-certs/rapidssl_privatekey_2015.key
    SSLCertificateChainFile /etc/apache2/ssl-certs/rapidssl_intermediateca_2015.crt

</VirtualHost>

ubuntu@ip-172-31-25-41:/etc/apache2/sites-enabled$
```

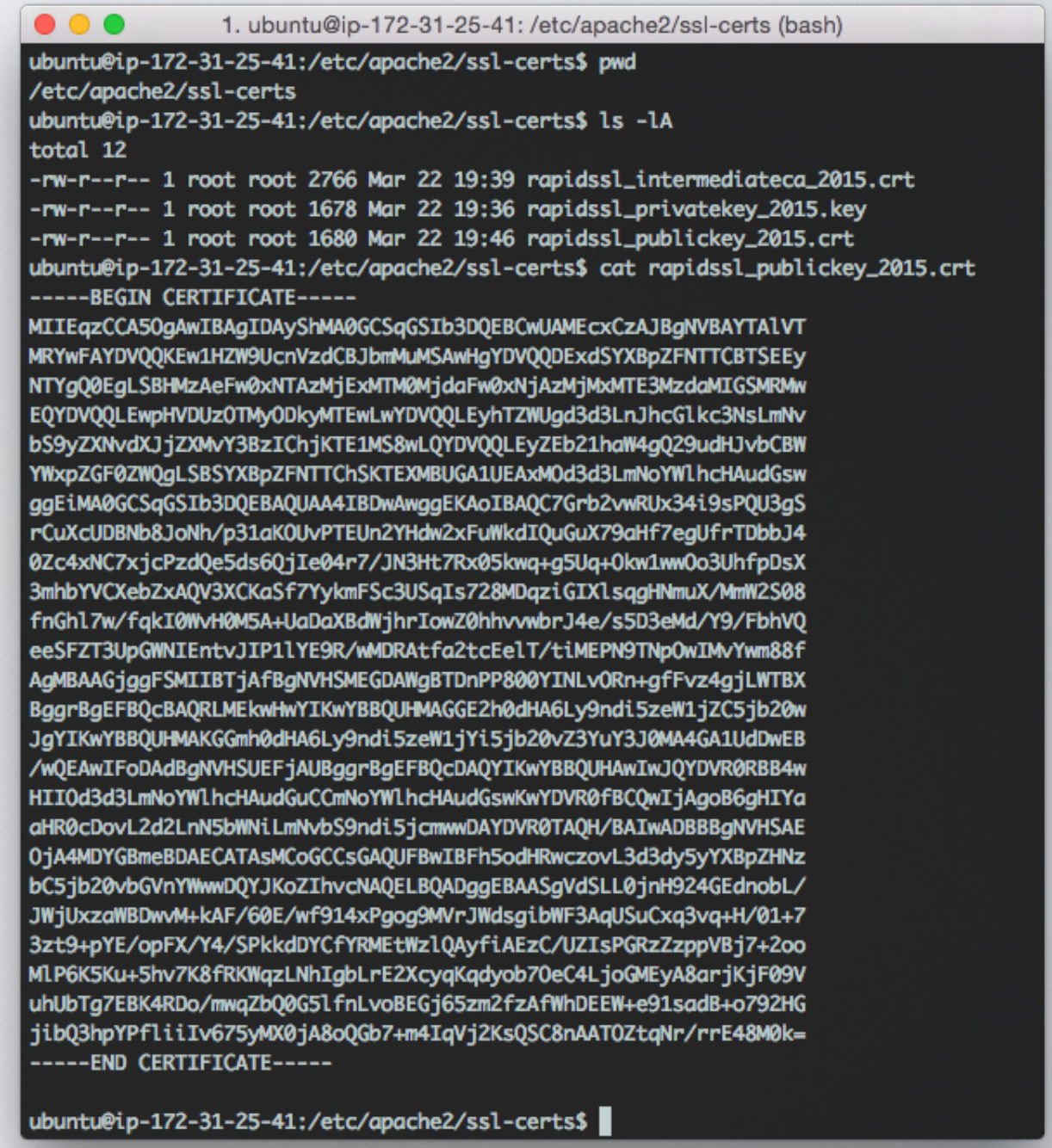
1. ubuntu@ip-172-31-25-41: /etc/apache2 (bash)

```
ubuntu@ip-172-31-25-41:/etc/apache2$ pwd
/etc/apache2
ubuntu@ip-172-31-25-41:/etc/apache2$ cat ports.conf
# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

# Disabled HTTP connections, only allowing SSL
Listen 80

<IfModule ssl_module>
        Listen 443
</IfModule>

<IfModule mod_gnutls.c>
        Listen 443
</IfModule>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
ubuntu@ip-172-31-25-41:/etc/apache2$
```

# SSL

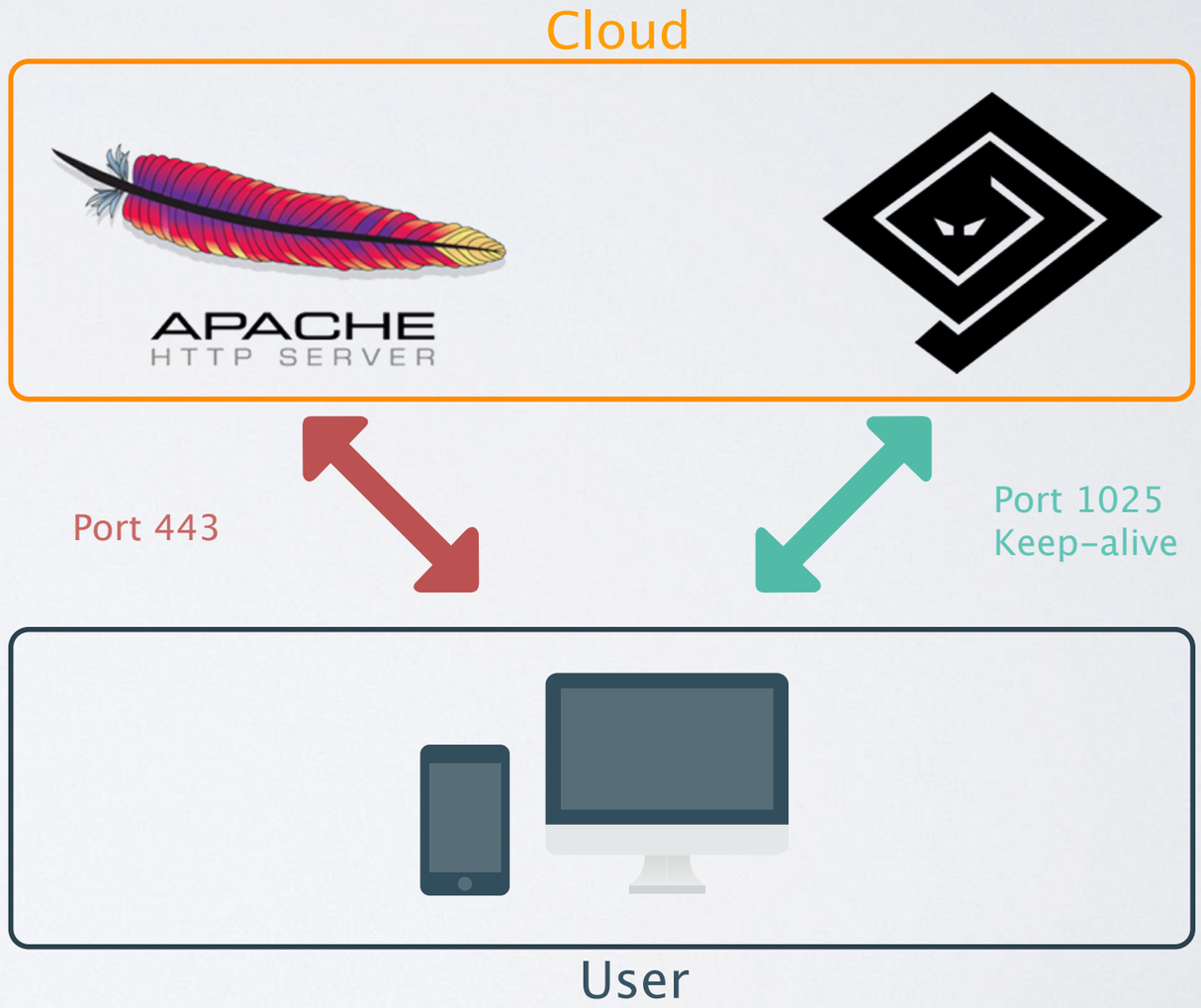
- Network data encryption
- Three keys required: Private, public, intermediate CA.



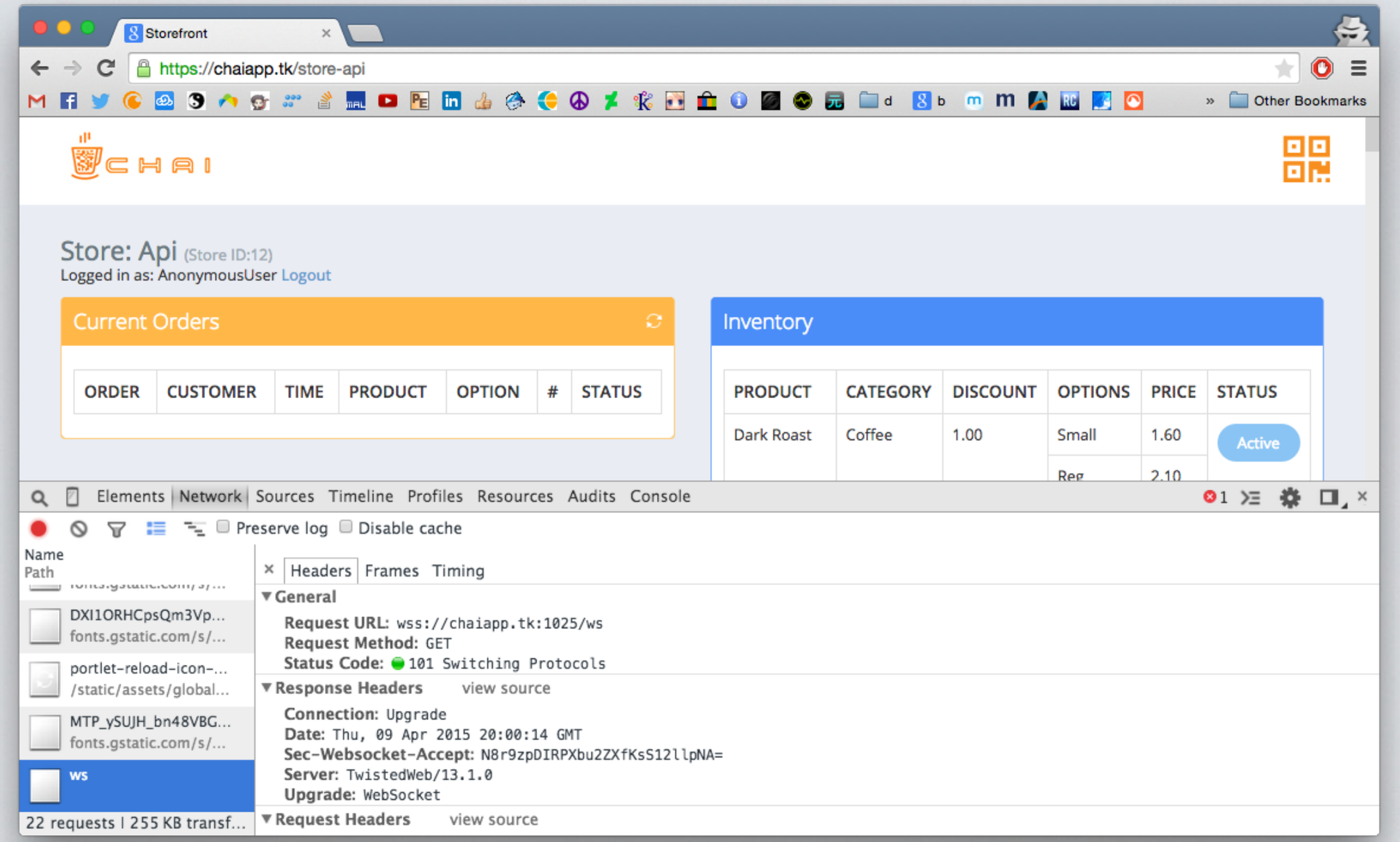



# WEBSOCKET SERVER

- Twisted 13.1.0 (Python)
- Real-time components

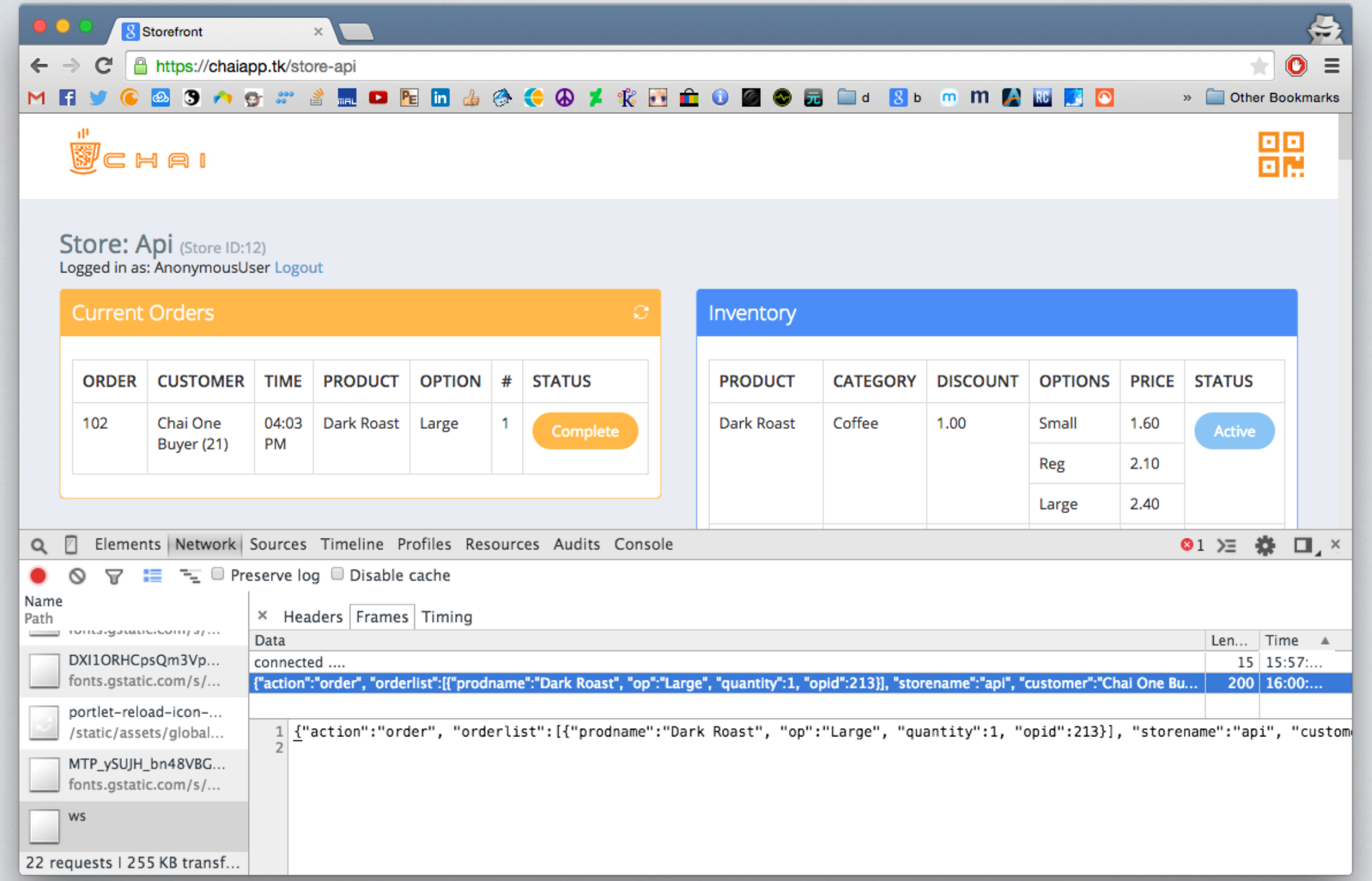


# WEBSOCKET EXAMPLE




Listen to websocket server on page load.

# WEBSOCKET EXAMPLE




On server message, update page in real-time without refreshing.

# DJANGO OVERVIEW

- MVC framework
- Main files: Settings, URLs, models, views, templates

2. ubuntu@ip-172-31-25-41: ~/chai_cloud_deploy (ssh)

```
ubuntu@ip-172-31-25-41:~/chai_cloud_deploy$ tree -n -I '*pyc|_scripts|twisted*|s
tatic|ssl|admin|media|twistd.*|ws_serv*|_ws*|*.conf'
.
├── api
|   ├── add_groups.py
|   ├── admin.py
|   ├── customer_simulate.py
|   ├── forms.py
|   ├── __init__.py
|   ├── management.py
|   ├── models.py
|   ├── permissions.py
|   ├── serializers.py
|   ├── templatetags
|   |   ├── custom_filters.py
|   |   └── __init__.py
|   ├── tests.py
|   ├── urls.py
|   └── views.py
├── chai_cloud
|   ├── __init__.py
|   ├── settings.py
|   ├── urls.py
|   └── wsgi.py
├── db.sqlite3
├── manage.py
└── templates
    └── api
        ├── e9b14fld.htm
        ├── order-status.html
        ├── qr.html
        ├── store-login.html
        └── stores.html

5 directories, 25 files
ubuntu@ip-172-31-25-41:~/chai_cloud_deploy$
```

# DJANGO: SETTINGS

- PayPal merchant credentials
- SSL and CSRF: Header and cookie settings
- Static files location

~/GitHub/410coffee/src/chai_cloud_deploy/chai_cloud/settings.py — UNREGISTERED

settings.py

```
25  HTTP_URL = "http://localhost/" # DEPLOY (HTTP is okay here since used locally)
26  WS_URL = "wss://chaiapp.tk:1025/ws" # SSL DEPLOY
27
28  # PAYPAL GLOBALS
29  PP_CURRENCY = "CAD"
30  PP_TOKEN_URL = "https://api-3t.sandbox.paypal.com/nvp"
31  PP_EMAIL = "chai-facilitator_api1.gmail.com"
32  PP_MERC_ID = "BZ8PCS9ZDYTTG"
33  PP_PSS = "TSHBEMMJVLWXBHHU" # note: this is not login password
34  PP_SIGNATURE = "AFcWxV21C7fd0v3bYYYRCpSSRl31AXEh9liEDPaCcNurPvdeesaTwyYs"
35  PP_TEST_EMAIL = "chai-buyer@gmail.com" # used to autofill checkout email (testi
36  PP_STORE_NAME = "CHAI+Store"
37
38  ALLOWED_HOSTS = []
39
40
41  # Application definition
42
43  INSTALLED_APPS = (
44      'django.contrib.auth',
45      'django.contrib.contenttypes',
```

Line 84, Column 16 — Spaces: 4 — Python

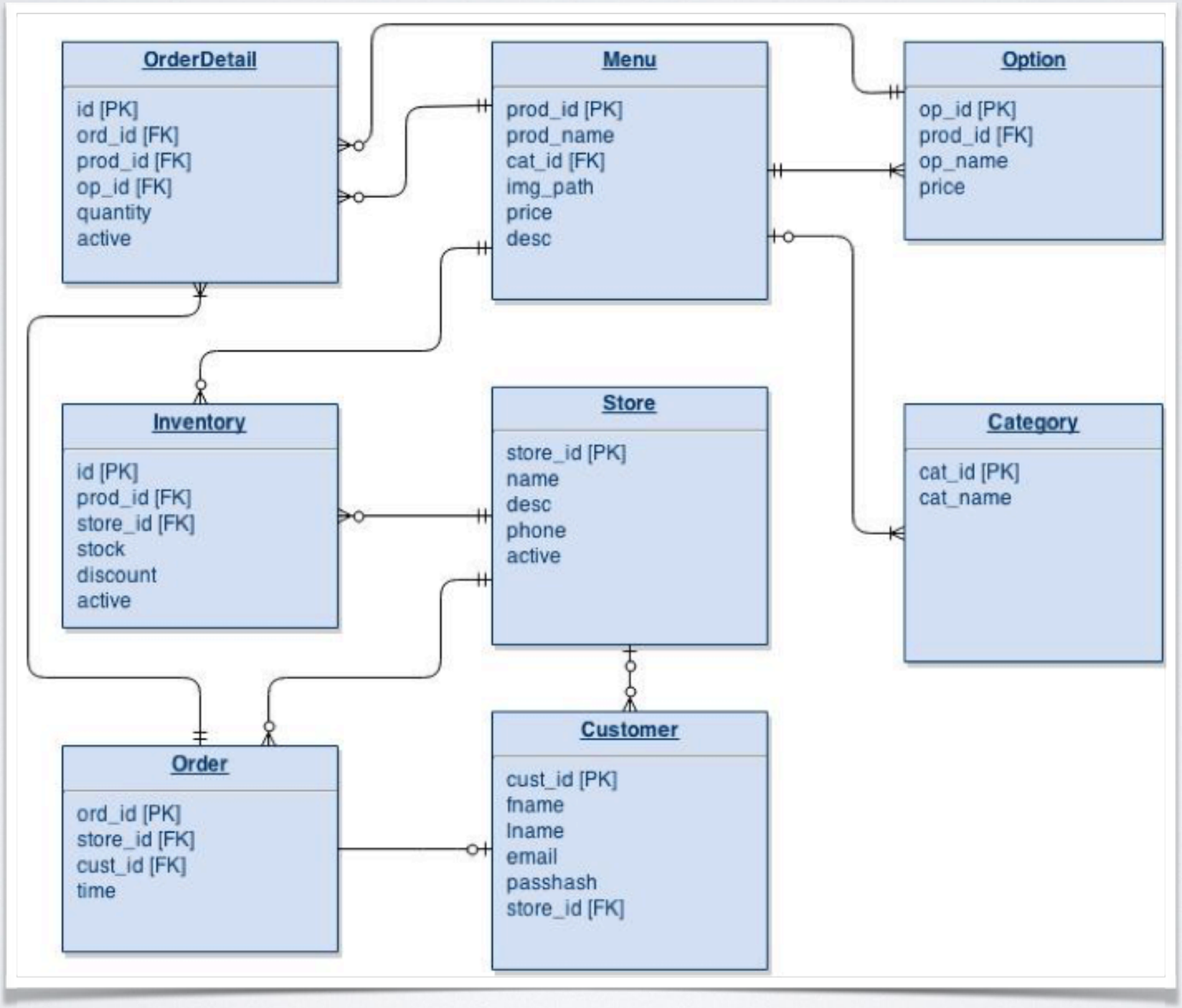
~/GitHub/410coffee/src/chai_cloud_deploy/chai_cloud/settings.py — UNREGISTERED

settings.py

```
92
93  # secure proxy SSL header and secure cookies
94  SECURE_PROXY_SSL_HEADER = ('HTTP_X_FORWARDED_PROTO', 'https')
95  SESSION_COOKIE_SECURE = True
96  CSRF_COOKIE_SECURE = True
97
98  # session expire at browser close
99  SESSION_EXPIRE_AT_BROWSER_CLOSE = True
100
101 # wsgi scheme
102 os.environ['wsgi.url_scheme'] = 'https'
103
104 # Static files (CSS, JavaScript, Images)
105
106 STATIC_ROOT = '/var/www/static/'
107 STATIC_URL = '/static/'
108 STATICFILES_DIRS = (
109     #os.path.join(BASE_DIR, "static"),
110     '/home/ubuntu/chai_cloud_deploy/static',
111 )
112 STATICFILES_FINDERS = (
113     'django.contrib.staticfiles.finders.FileSystemFinder',
114     'django.contrib.staticfiles.finders.AppDirectoriesFinder',
115 )
116
```

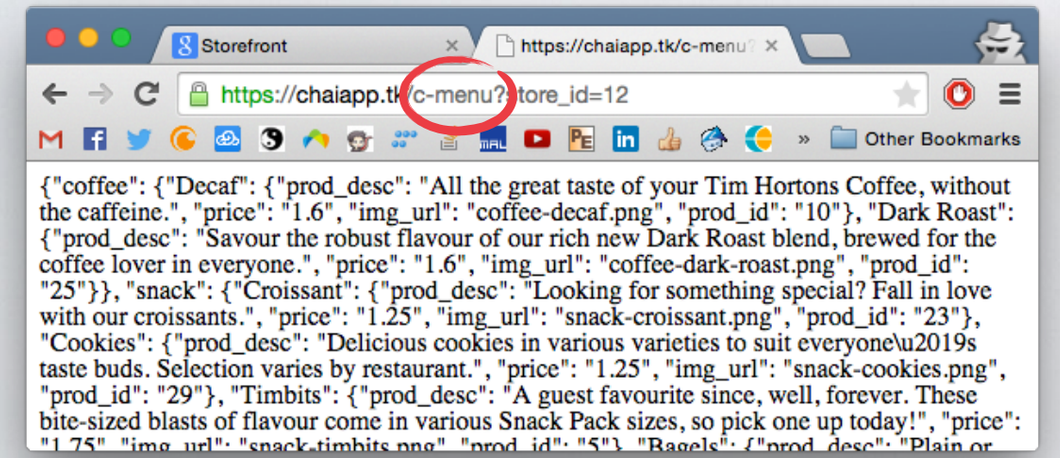
Line 84, Column 16 — Spaces: 4 — Python

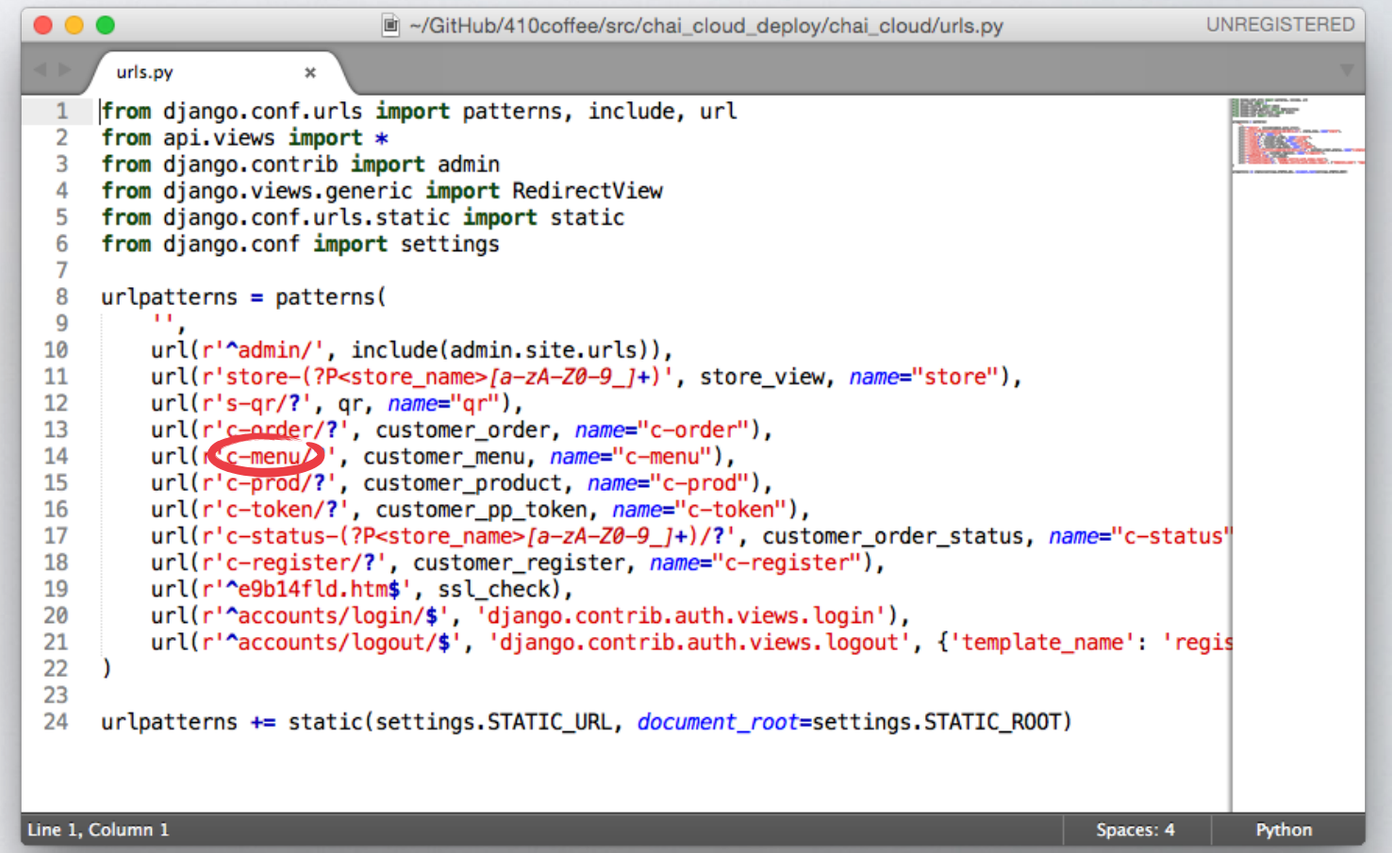
# DJANGO: MODELS



# DJANGO: URL ROUTING

- Regex URL patterns matching views



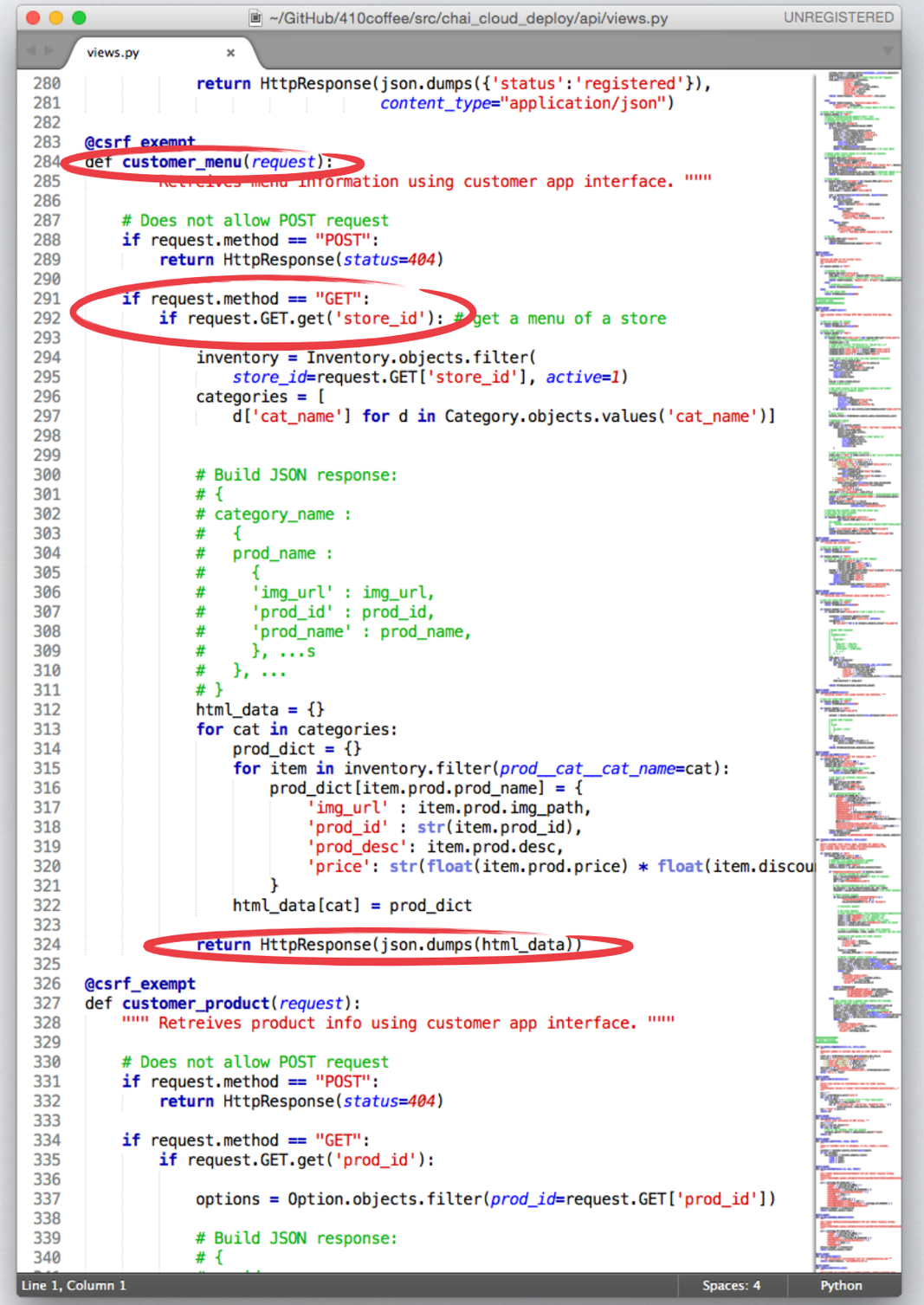
# DJANGO:VIEWS

- Combination of Controller and View in MVC

- Handles HTTP requests

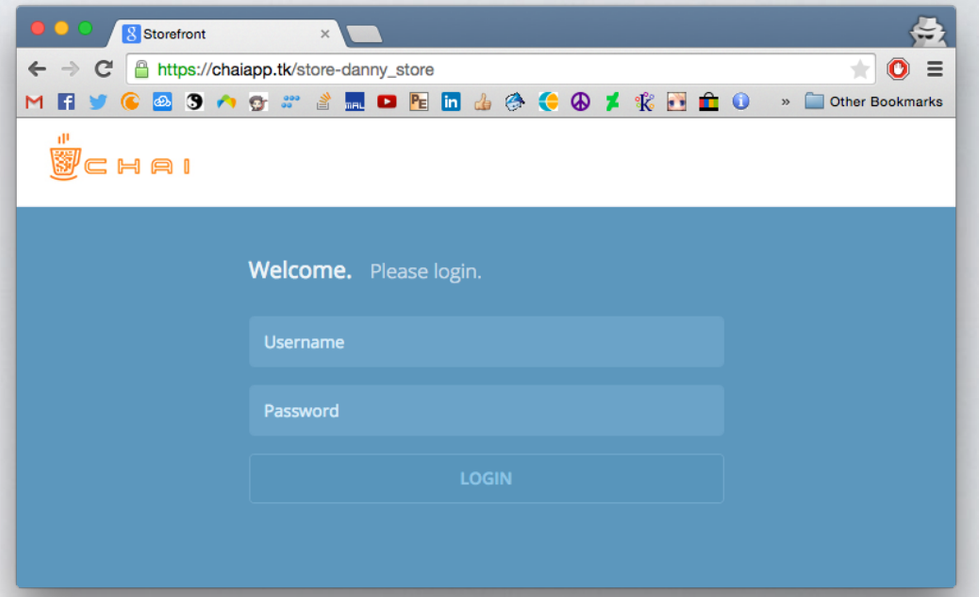
- HTTP response: Either JSON or rendered HTML.

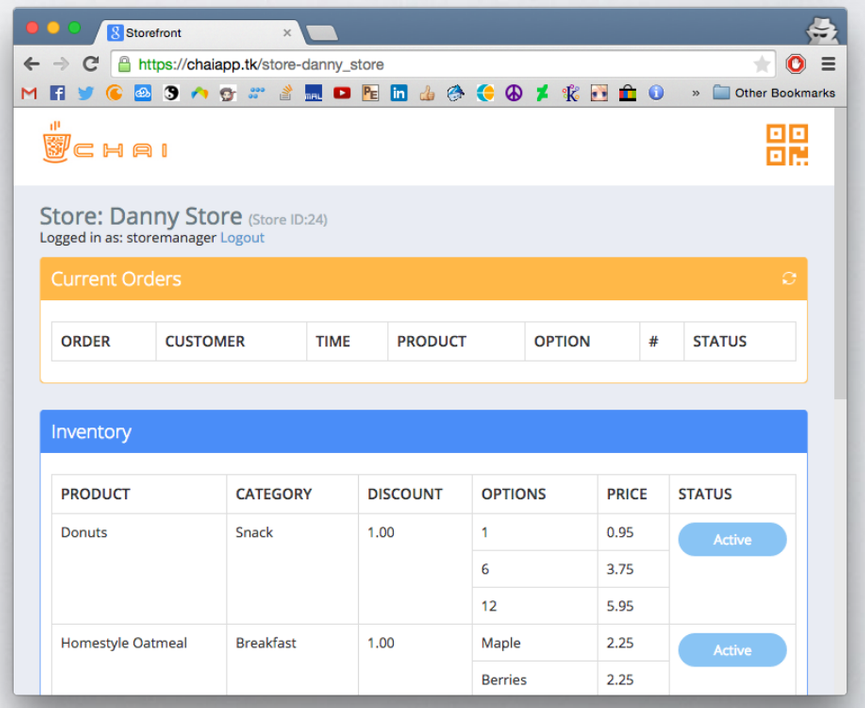
- RESTful API: Store and Customer



# API: STORE

- URL: store-<store name>/
- GET: Store login page, Store landing page
- HTML rendered response



# API: STORE

- POST: Inventory active/ inactive, Order completion, Login action, Logout action.
- JSON response.
- Logic variation depends on form data passed in POST
- Intended for internal calls

~/GitHub/410coffee/src/chai_cloud_deploy/api/views.py — UNREGISTERED

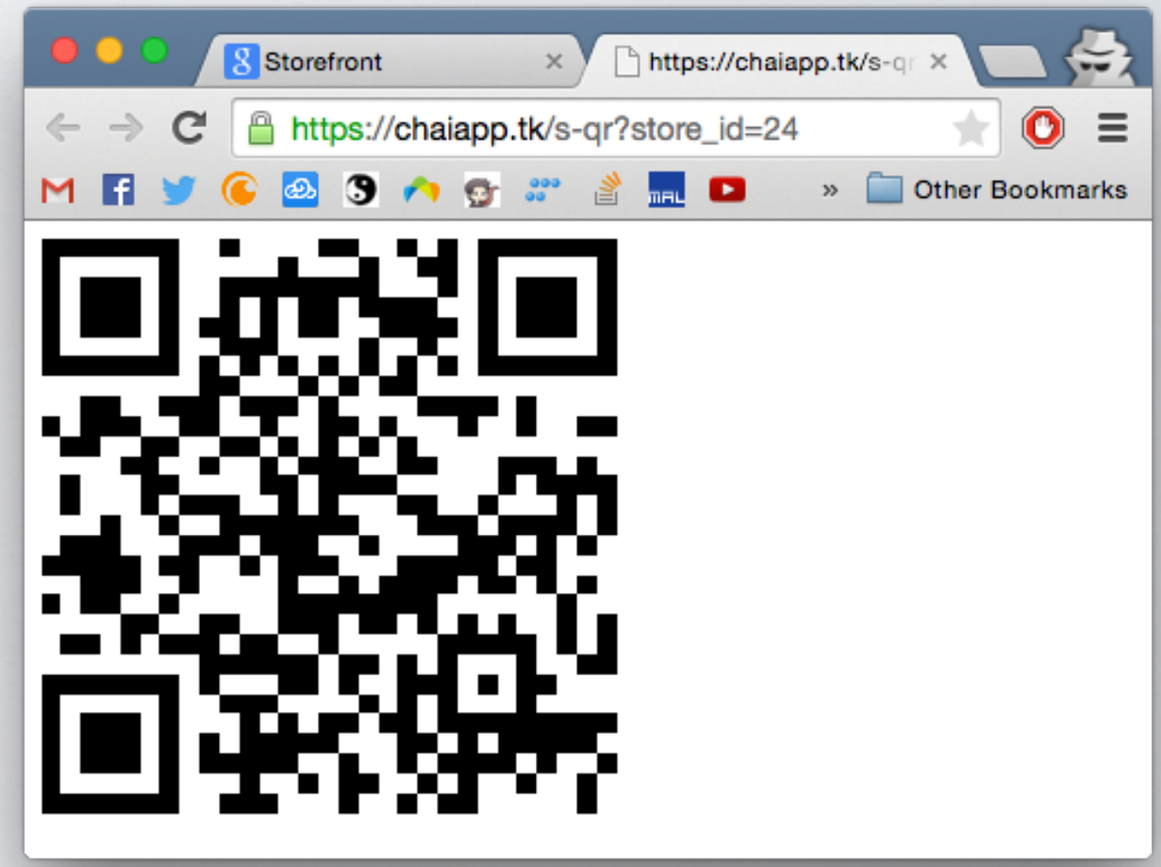
views.py

```
75      # Handle POST requests (forms)
76      if request.method == 'POST':
77          #return HttpResponse(json.dumps({'here': 1}))
78          # Updates active/inactive status of inventory item
79          # Intended for AJAX calls
80          if request.POST.get('active'):
81              form = InventoryActiveForm(request.POST)
82              if form.is_valid():
83                  inventory = Inventory.objects.all()
84                  prod_id = form.cleaned_data['prod_id']
85                  store_id = form.cleaned_data['store_id']
86                  active = form.cleaned_data['active']
87                  selected = Inventory.objects.filter(
88                      prod_id=prod_id).filter(
89                      store_id=store_id)
90                  selected.update(active=active)
91                  return HttpResponse(json.dumps(active)) # for ajax calls
92
93          # Update order active status to 0 when order is complete
94          # Intended for AJAX calls
95          if request.POST.get('complete_order'):
96              form = CompleteOrder(request.POST)
97              details_id = request.POST['details_id']
98              print "[DEBUG] Updating status to 0 for Order Detail ID:", details_
99              selected = OrderDetail.objects.filter(id=details_id)
100             selected.update(active=0)
101             ws_detail_complete(details_id, store_name) # websocket update to cu
102             return HttpResponse(json.dumps(details_id)) # for ajax calls
103
104         # Store login
105         if request.POST.get('username') and request.POST.get('passw'):
106             form = StoreLogin(request.POST)
107             username = request.POST['username']
108             passw = request.POST['passw']
109             store_name = request.POST['store_name']
110
111             user = authenticate(username=username, password=passw)
112             if user is not None:
113                 if user.is_active:
114                     login(request, user)
115                     return redirect('/store-' + store_name)
116                 else:
117                     return render(
118                         request,
119                         'api/store-login.html',
120                         {'store_name': store_name,
121                          'alert': "This account is disabled."})
```

Line 1, Column 1 — Spaces: 4 — Python

# API: STORE

- URL: s-qr/
- GET: Generate QR code for store.
- HTML Response
- Returns an image, easily embed with other HTML applications.



# API: CUSTOMER

- URL: c-order/
- POST: Place new order.
- Response: JSON (confirmation)
- Internal API called after PayPal transaction.

~/GitHub/410coffee/src/chai_cloud_deploy/api/views.py — UNREGISTERED

views.py

```python
@csrf_exempt
def customer_order(request):
    """
    Take customer orders through HTTP POST requests from customer app.
    """

    # Does not allow GET request
    if request.method == "GET":
        return HttpResponse(status=404)

    # Handle POST requests
    if request.method == "POST":
        # New orders
        if request.POST.get('order_list') and request.POST.get('store_name'):
            #order_list = form.cleaned_data['order_list']
            response_data = {}
            # order_list format: "[{'prod_id':2, 'op_id':1},...]"
            # POST is just a string representation
            response_data['order_list'] = request.POST['order_list']
            response_data['store_name'] = request.POST['store_name']
            response_data['email'] = request.POST['email']

            # Add order to DB (one order can have serveral products)
            store_id = Store.objects.get(
                name=response_data['store_name']).store_id
            cust_id = Customer.objects.get(
                email=response_data['email']).cust_id
            order_create = Order.objects.create(
                store_id=store_id,
                cust_id=cust_id,
                time=timezone.now()
            )
            ord_id = order_create.ord_id
            #order_create.save()

            # Add order details to DB (individual products per order)
            # Create list of products models
            details_list = [
                OrderDetail(
                    ord_id=ord_id,
                    prod_id=int(details['prod_id']),
                    op_id=int(details['op_id']),
                    quantity=int(details['quantity']),
                    active=1
                ) for details in ast.literal_eval(response_data['order_list'])
            ]
            # Bulk insert
            details_create = OrderDetail.objects.bulk_create(details_list)

            # Websocket update
            order_list = ''
            for detail in details_create:
                order_list += '{"prodname":"%s", "op":"%s", "quantity":%d, "opi
                    detail.prod.prod_name,
                    detail.op.op_name.title(),
                    detail.quantity,
                    OrderDetail.objects.get( # order detail id
                        ord_id=detail.ord_id,
                        prod_id=detail.prod_id,
                        op_id=detail.op_id,
                        active=1).id,
                )

            # list of orders formatted for output
            order_list = "[%s]" % order_list[:-2] # get rid of trailing comma/s
            # send websocket POST
            post_str = "{\"action\":\"order\"," + \
               " \"orderlist\":%s," % order_list + \
               " \"storename\":\"%s" % request.POST['store_name'] + \
               "\", \"customer\":\"%s %s\"," % (
                    Customer.objects.get(
                        email=response_data['email']).fname,
                    Customer.objects.get(
                        email=response_data['email']).lname) + \
               " \"custid\":%d, " % cust_id + \
               "\"ordertime\":\"%s\"," % \
                    Order.objects.get(ord_id=ord_id).time.astimezone(
                        pytz.timezone('US/Eastern')).strftime(
                        "%I:%M %p") + \
               " \"orderid\":%d}" % ord_id
            post_data = [('new_message', post_str),]
            #result = urlopen('http://localhost:1025', urlencode(post_data))
            result = urlopen('https://localhost:1025', urlencode(post_data))
            print "WS:::", result
            response_data['order_id'] = ord_id
            return HttpResponse(json.dumps(response_data),
                                content_type="application/json")
```

Line 191, Column 14 — Spaces: 4 — Python

# API: CUSTOMER

- URL: c-register/
- POST: Register new customer account.
- Response: JSON (confirmation)
- Intended for registration form submission event.

~/GitHub/410coffee/src/chai_cloud_deploy/api/views.py — UNREGISTERED

views.py

```
260 @csrf_exempt
261 def customer_register(request):
262     """ Creates new customer account. """
263
264     # Does not allow GET request
265     if request.method == "GET":
266         return HttpResponse(status=404)
267
268     if request.method == "POST":
269         # check all required info is in the POST request
270         if request.POST.get('email') and \
271                     request.POST.get('pass') and \
272                     request.POST.get('fname') and \
273                     request.POST.get('lname'):
274             hashed = bcrypt.hashpw(
275                 request.POST['pass'].encode('utf-8'),
276                 bcrypt.gensalt())
277             register = Customer.objects.create(
278                 fname=request.POST['fname'],
279                 lname=request.POST['lname'],
280                 email=request.POST['email'],
281                 passhash=hashed)
282             return HttpResponse(json.dumps({'status':'registered'}),
283                                 content_type="application/json")
```

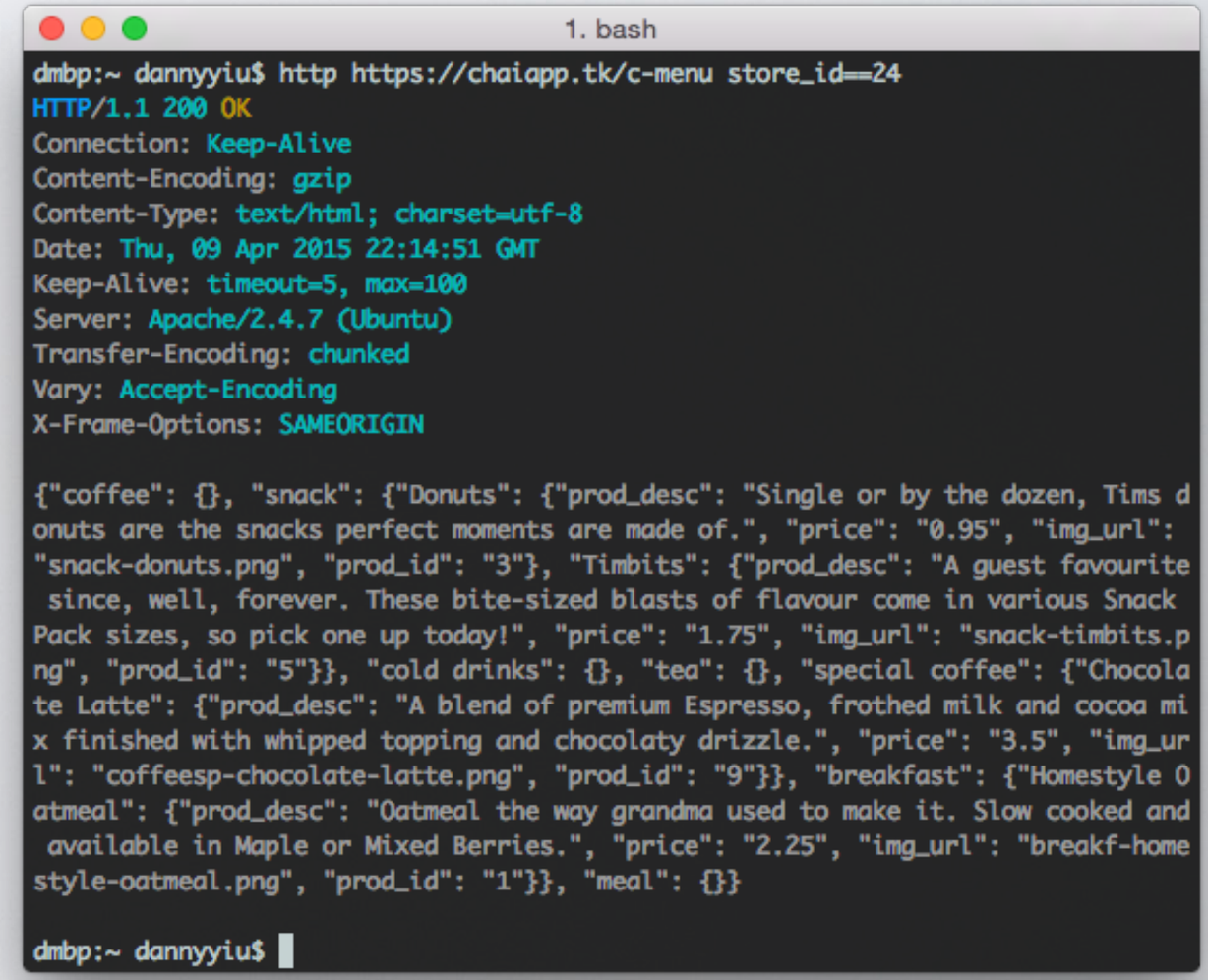
Line 276, Column 17 — Spaces: 4 — Python

1. bash

```
dmbp:~ dannyyiu$ http -f POST https://chaiapp.tk/c-register email='dandan@gmail.
com' fname='dandan' lname='Dan' pass='asdf'
HTTP/1.1 200 OK
Connection: Keep-Alive
Content-Type: application/json
Date: Thu, 09 Apr 2015 22:05:00 GMT
Keep-Alive: timeout=5, max=100
Server: Apache/2.4.7 (Ubuntu)
Transfer-Encoding: chunked
X-Frame-Options: SAMEORIGIN

{
    "status": "registered"
}

dmbp:~ dannyyiu$
```

# API: CUSTOMER

- URL: c-menu/
- GET: Store menu (active inventory).
- Response: JSON
- Intended for customer app to load store menu after QR scan



# API: CUSTOMER

- URL: c-token/
- GET: PayPal token.
- Response: JSON
- Intended for customer app to initiate payment.

~/GitHub/410coffee/src/chai_cloud_deploy/api/views.py — UNREGISTERED

views.py

```
354 @csrf_exempt
355 def customer_pp_token(request):
356     """ Retrieves a PayPal token for checkout page. """
357     if request.method == "GET":
358         if request.GET.get('amt') and \
359         request.GET.get('orderdetails') and \
360         request.GET.get('store_id'):
361             # Get store name (required for order)
362             store_name = Store.objects.get(
363                 store_id=request.GET['store_id']).name
364
365             # Get email for autofill (optional)
366             email_str = ""
367             email = settings.PP_TEST_EMAIL
368             if request.GET.get('email'):
369                 email = request.GET['email']
370                 email_str = "&EMAIL=" + email
371
372             # Call SetExpressCheckout API
373             url = settings.PP_TOKEN_URL + \
374                   "?USER=" + settings.PP_EMAIL + \
375                   "&PWD=" + settings.PP_PSS + \
376                   "&SIGNATURE=" + settings.PP_SIGNATURE + \
377                   "&METHOD=SetExpressCheckout" + \
378                   "&VERSION=78" + \
379                   "&NOSHIPPING=1" + \
380                   "&BRANDNAME=" + settings.PP_STORE_NAME + \
381                   "&PAYMENTREQUEST_0_PAYMENTACTION=SALE" + \
382                   "&PAYMENTREQUEST_0_AMT=" + request.GET['amt'] + \
383                   "&PAYMENTREQUEST_0_CURRENCYCODE=" + settings.PP_CURRENCY + \
384                   email_str + \
385                   "&cancelUrl=http://www.google.com" + \
386                   "&returnUrl=https://chaiapp.tk/c-status-" + store_name + \
387                   "?orderdetails=" + request.GET['orderdetails']
388             token_request = urlopen(url)
389             return HttpResponse(
390                 json.dumps({'PP_SETCHECKOUT_RESPONSE': token_request.read()}))
```

2 lines, 35 characters selected — Spaces: 4 — Python

1. bash

```
dmbp:~ dannyyiu$ http https://chaiapp.tk/c-token amt==6.38 orderdetails==cart-25-27-1cart-27-68-1
store_id==12
HTTP/1.1 200 OK
Connection: Keep-Alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Thu, 09 Apr 2015 22:20:22 GMT
Keep-Alive: timeout=5, max=100
Server: Apache/2.4.7 (Ubuntu)
Transfer-Encoding: chunked
Vary: Accept-Encoding
X-Frame-Options: SAMEORIGIN

{"PP_SETCHECKOUT_RESPONSE": "TOKEN=EC%2d9BN73043AJ300005E&TIMESTAMP=2015%2d04%2d09T22%3a17%3a51Z&C
ORRELATIONID=fe671eaadeaa5&ACK=Success&VERSION=78&BUILD=16139311"}

dmbp:~ dannyyiu$
```

# API: CUSTOMER

- URL: c-prod/
- GET: Product options.
- Response: JSON
- Intended for customer app to show product options.

~/GitHub/410coffee/src/chai_cloud_deploy/api/views.py — UNREGISTERED

views.py

```
327 @csrf_exempt
328 def customer_product(request):
329     """ Retreives product info using customer app interface. """
330
331     # Does not allow POST request
332     if request.method == "POST":
333         return HttpResponse(status=404)
334
335     if request.method == "GET":
336         if request.GET.get('prod_id'):
337
338             options = Option.objects.filter(prod_id=request.GET['prod_id'])
339
340             # Build JSON response:
341             # {
342             # op_id
343             #   {
344             #   op_name : price
345             #   }
346             # }
347             html_data = {}
348             for option in options:
349                 html_data[str(option.op_id)] = {
350                     option.op_name: str(option.price)
351                 }
352             return HttpResponse(json.dumps(html_data))
353
```

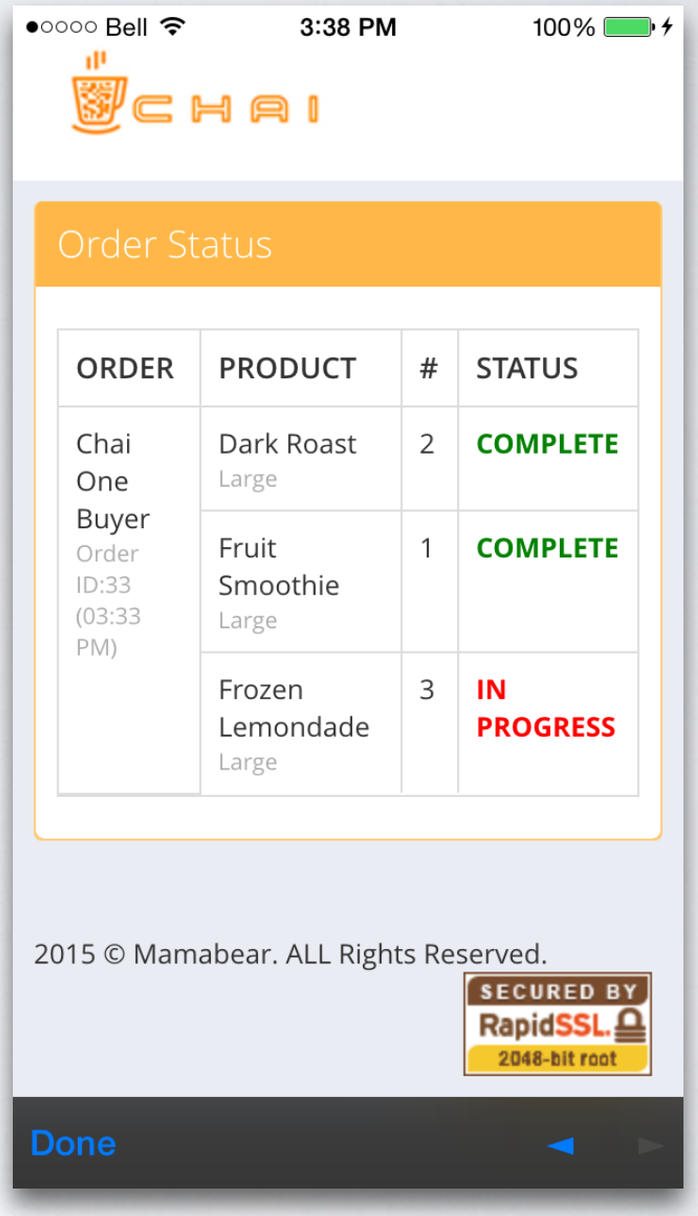
2 lines, 35 characters selected — Spaces: 4 — Python

1. bash

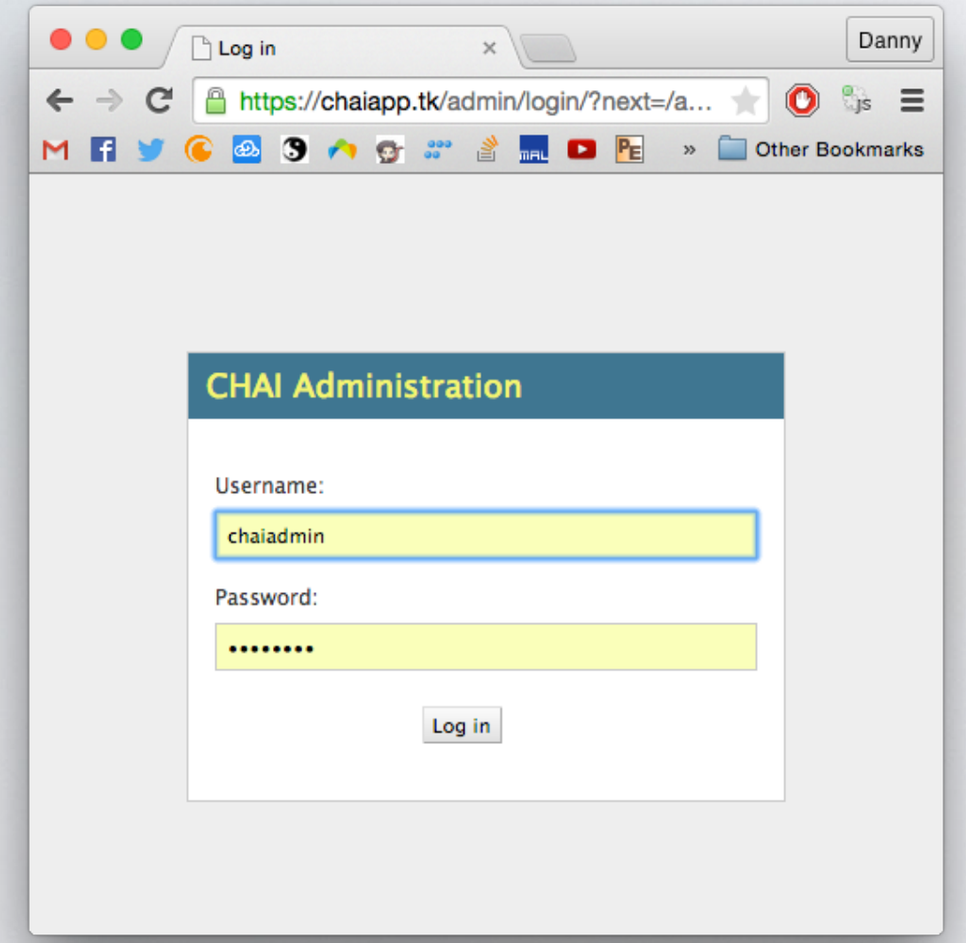
```
dmbp:~ dannyyiu$ http https://chaiapp.tk/c-prod prod_id==25
HTTP/1.1 200 OK
Connection: Keep-Alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Thu, 09 Apr 2015 22:22:57 GMT
Keep-Alive: timeout=5, max=100
Server: Apache/2.4.7 (Ubuntu)
Transfer-Encoding: chunked
Vary: Accept-Encoding
X-Frame-Options: SAMEORIGIN

{"63": {"small": "0"}, "64": {"reg": "0.5"}, "65": {"large": "0.8"}}

dmbp:~ dannyyiu$
```

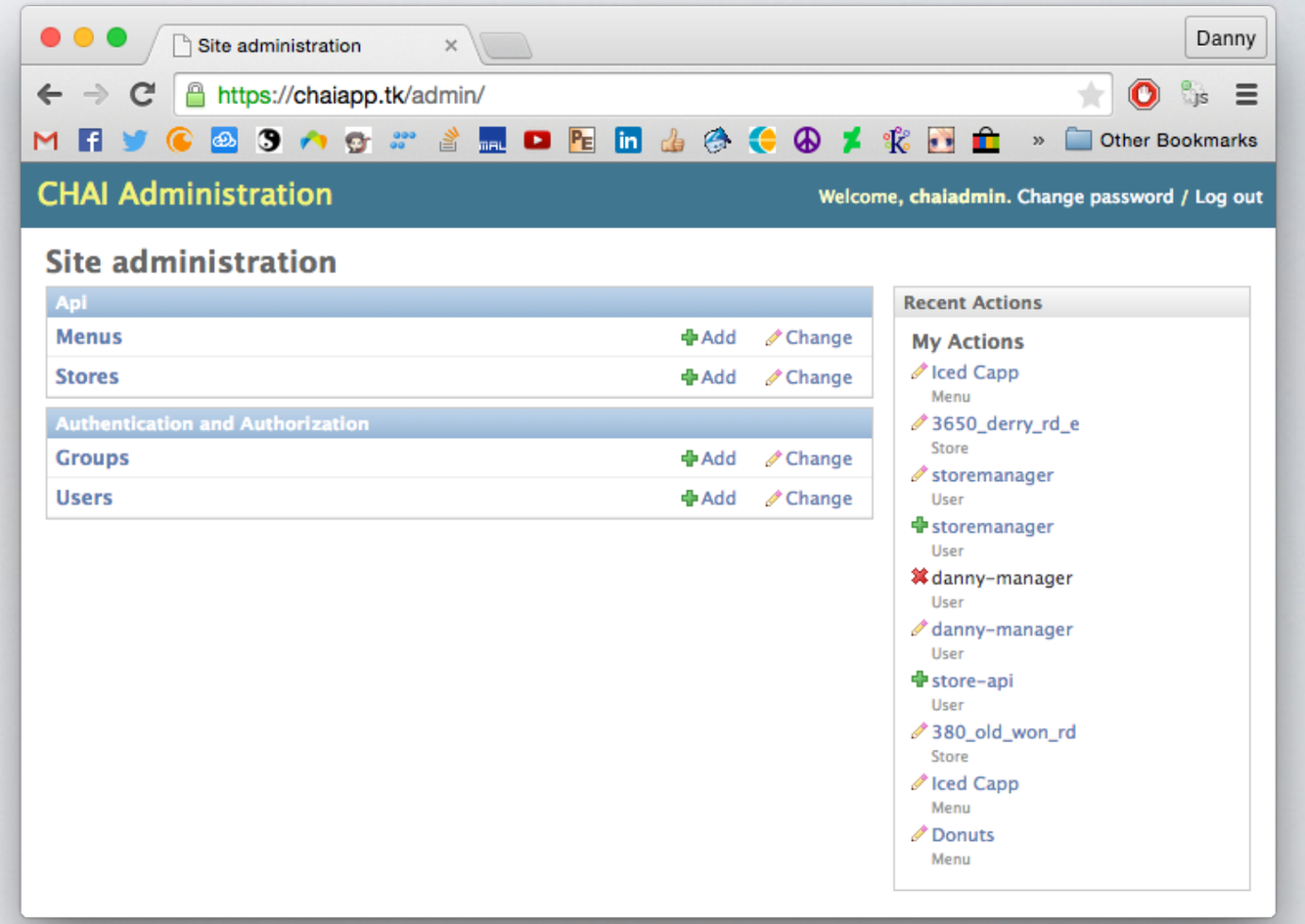
# API: CUSTOMER

- URL: c-status-<store name>/
- GET: Order status page for customer.
- Response: HTML
- Intended for customer app after payment.

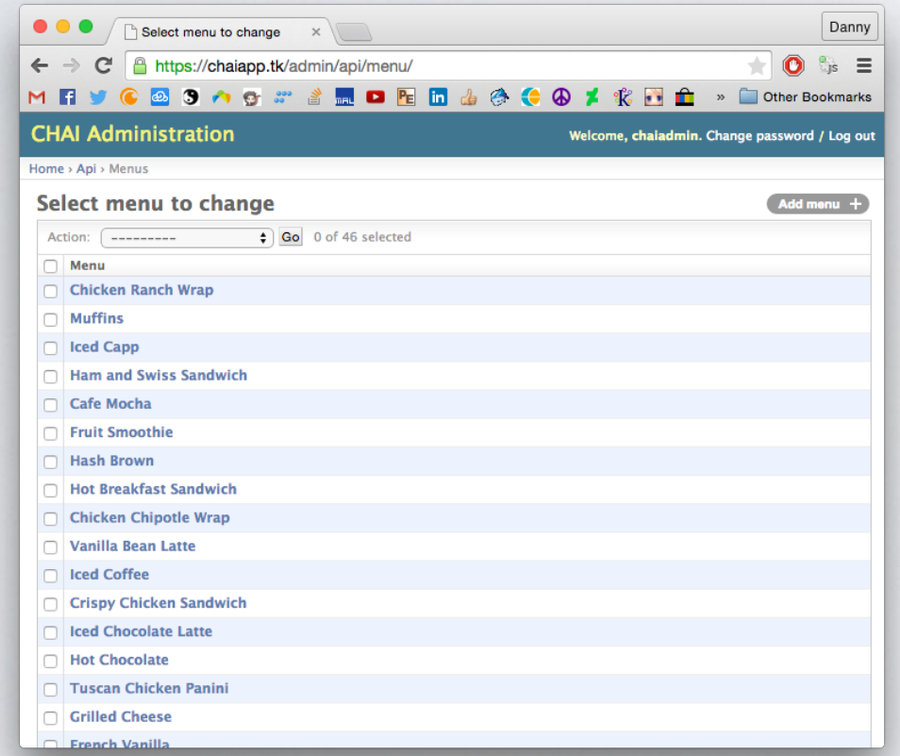


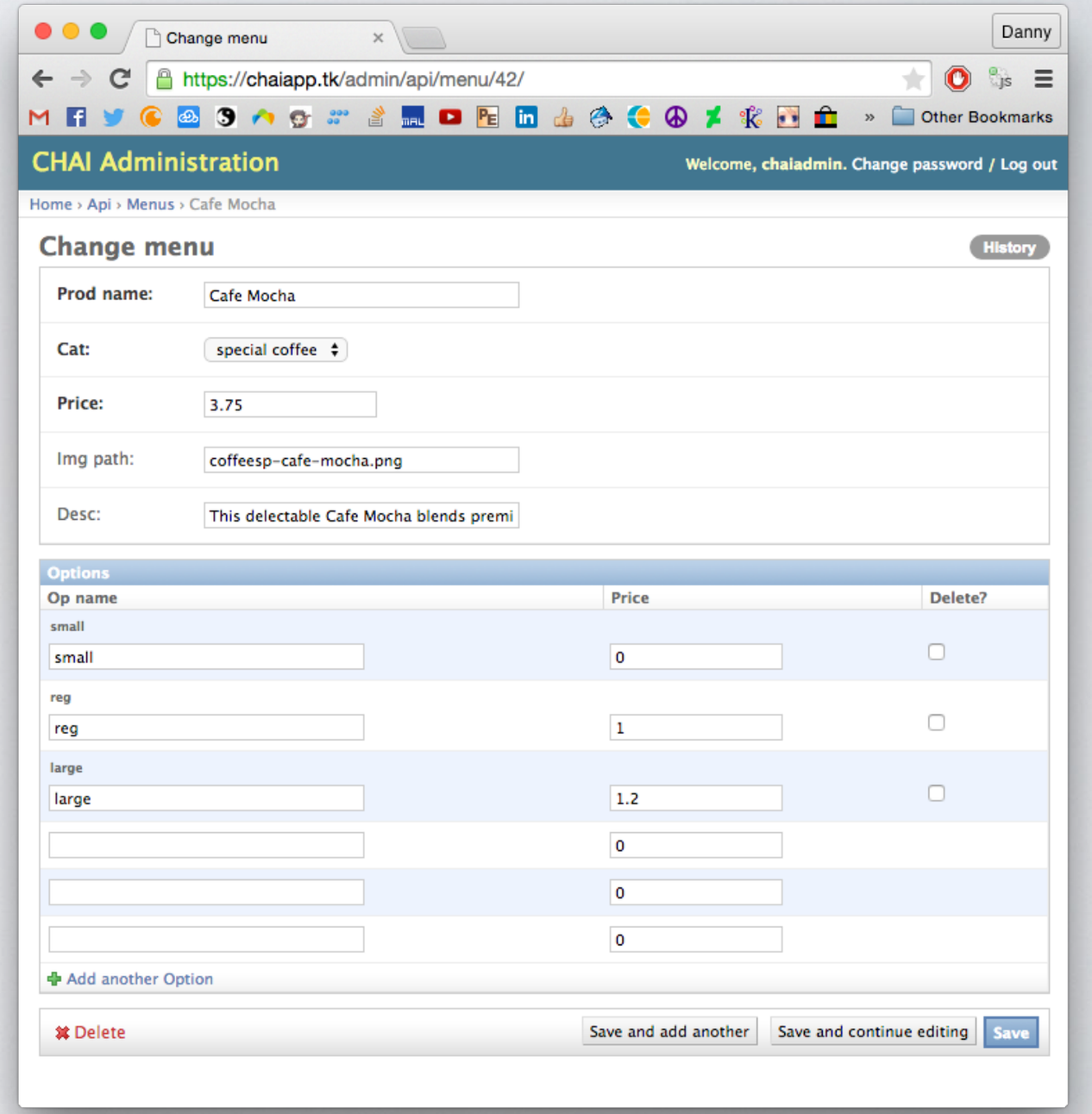
# ADMIN: LANDING



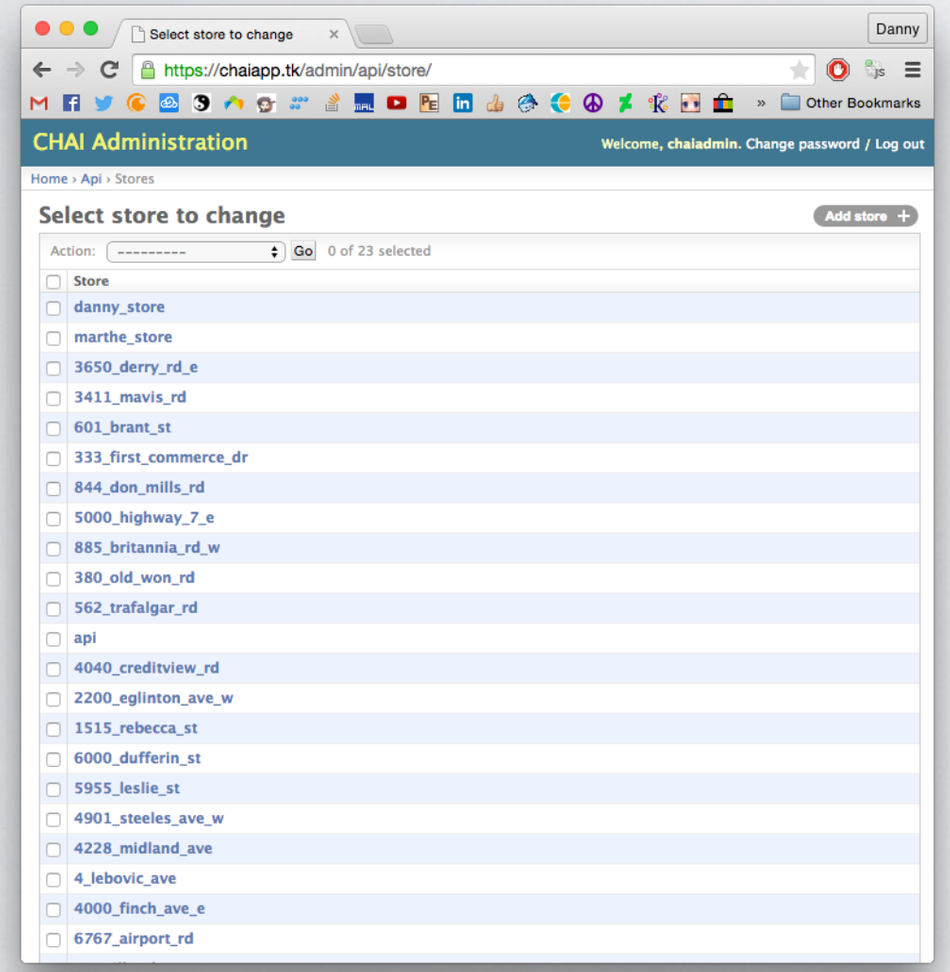



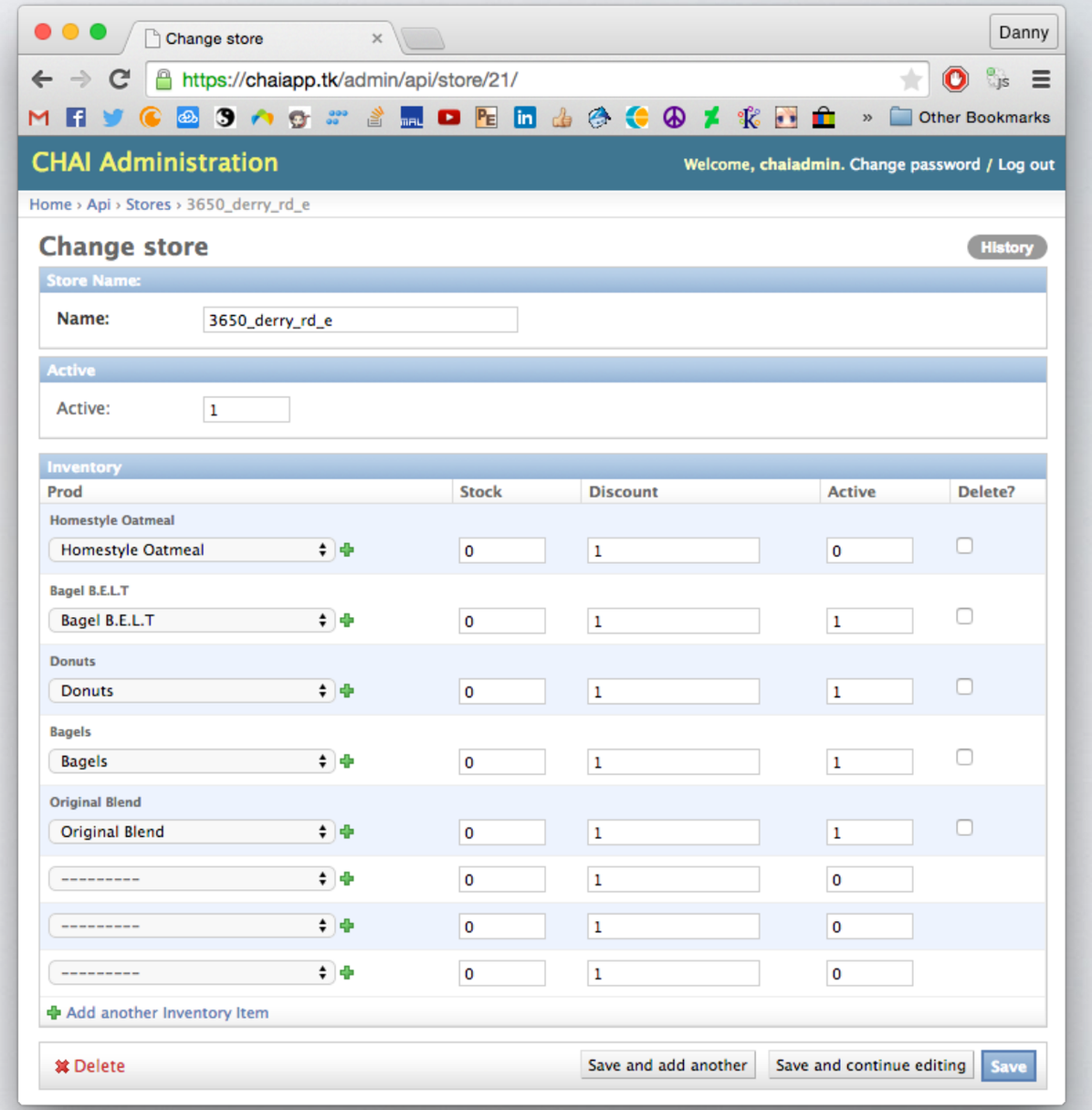
# ADMIN: MENU


Select menu to change
Danny
https://chaiapp.tk/admin/api/menu/
Other Bookmarks
CHAI Administration
Welcome, chaiadmin. Change password / Log out
Home › Api › Menus
Select menu to change
Add menu
Action: --------- Go 0 of 46 selected
Menu
Chicken Ranch Wrap
Muffins
Iced Capp
Ham and Swiss Sandwich
Cafe Mocha
Fruit Smoothie
Hash Brown
Hot Breakfast Sandwich
Chicken Chipotle Wrap
Vanilla Bean Latte
Iced Coffee
Crispy Chicken Sandwich
Iced Chocolate Latte
Hot Chocolate
Tuscan Chicken Panini
Grilled Cheese
French Vanilla

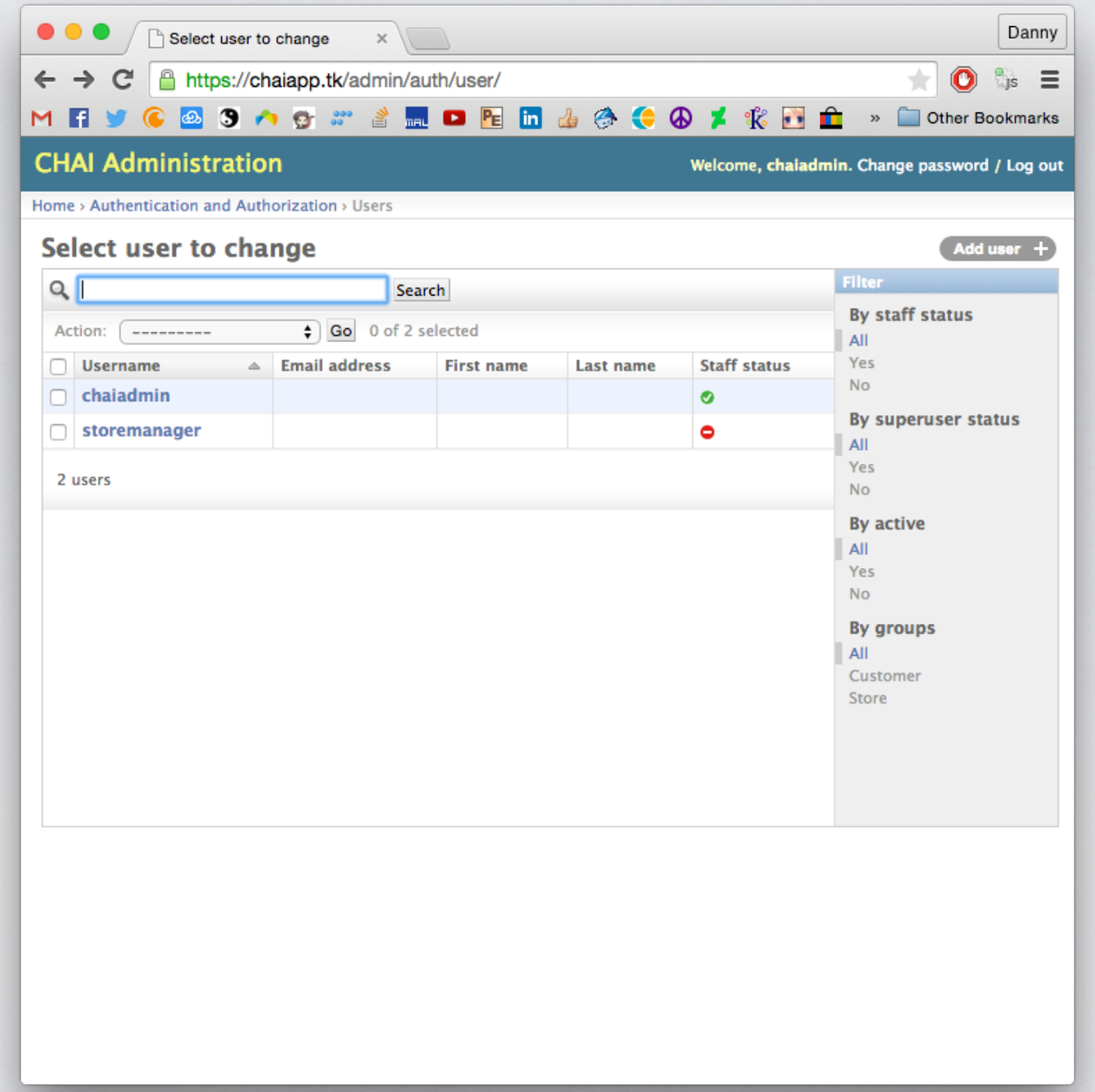

Change menu
Danny
https://chaiapp.tk/admin/api/menu/42/
Other Bookmarks
CHAI Administration
Welcome, chaiadmin. Change password / Log out
Home › Api › Menus › Cafe Mocha
Change menu
History
Prod name: Cafe Mocha
Cat: special coffee
Price: 3.75
Img path: coffeesp-cafe-mocha.png
Desc: This delectable Cafe Mocha blends premi
Options
Op name
Price
Delete?
small
small
0
reg
reg
1
large
large
1.2
0
0
0
Add another Option
Delete
Save and add another
Save and continue editing
Save

# ADMIN: STORE

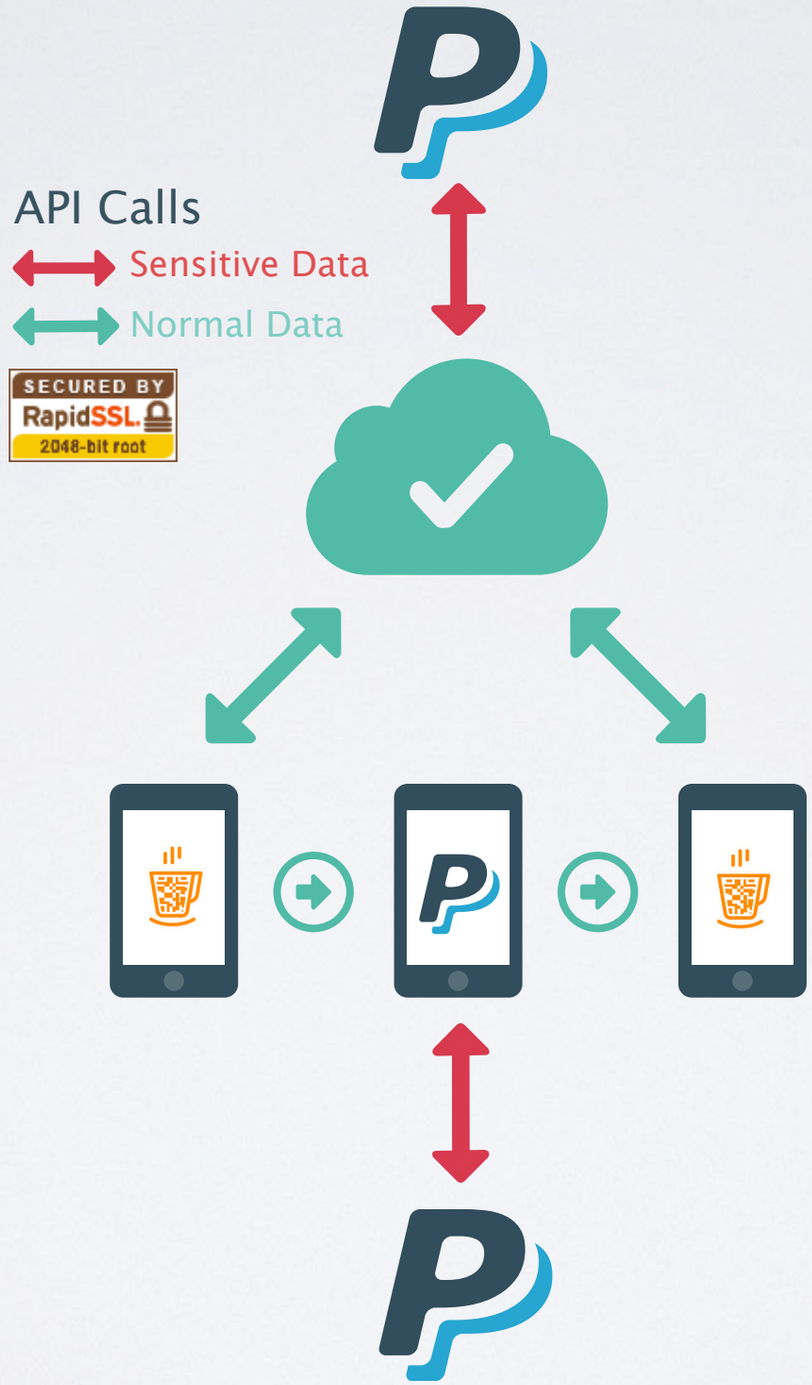

Select store to change
Danny
https://chaiapp.tk/admin/api/store/
Other Bookmarks
CHAI Administration
Welcome, chaiadmin. Change password / Log out
Home › Api › Stores
Select store to change
Add store
Action: ---------- Go 0 of 23 selected
Store
danny_store
marthe_store
3650_derry_rd_e
3411_mavis_rd
601_brant_st
333_first_commerce_dr
844_don_mills_rd
5000_highway_7_e
885_britannia_rd_w
380_old_won_rd
562_trafalgar_rd
api
4040_creditview_rd
2200_eglinton_ave_w
1515_rebecca_st
6000_dufferin_st
5955_leslie_st
4901_steeles_ave_w
4228_midland_ave
4_lebovic_ave
4000_finch_ave_e
6767_airport_rd


Change store
Danny
https://chaiapp.tk/admin/api/store/21/
Other Bookmarks
CHAI Administration
Welcome, chaiadmin. Change password / Log out
Home › Api › Stores › 3650_derry_rd_e
Change store
History
Store Name:
Name: 3650_derry_rd_e
Active
Active: 1
Inventory
Prod
Stock
Discount
Active
Delete?
Homestyle Oatmeal
Homestyle Oatmeal 0 1 0
Bagel B.E.L.T
Bagel B.E.L.T 0 1 1
Donuts
Donuts 0 1 1
Bagels
Bagels 0 1 1
Original Blend
Original Blend 0 1 1
--------- 0 1 0
--------- 0 1 0
--------- 0 1 0
Add another Inventory Item
Delete
Save and add another
Save and continue editing
Save

# ADMIN: USERS


Select user to change
Danny
https://chaiapp.tk/admin/auth/user/
Other Bookmarks
CHAI Administration
Welcome, chaiadmin. Change password / Log out
Home › Authentication and Authorization › Users
Select user to change
Add user
Search
Action: --------- Go 0 of 2 selected
Username
Email address
First name
Last name
Staff status
chaiadmin
storemanager
2 users
Filter
By staff status
All
Yes
No
By superuser status
All
Yes
No
By active
All
Yes
No
By groups
All
Customer
Store

# PCI COMPLIANCE



# FUTURE IMPROVEMENTS

- Admin page customization
- Store landing page width bug (992px-1156px, and under 565px)
- Recurring "Uncaught SyntaxError: Unexpected token c" (but no known effects)
- CSS, Javascript/jQuery cleanup
- Android InAppBrowser plugin won't show mobile site. Probably user-agent header.
- Recurring "Uncaught SyntaxError: Unexpected token c", but doesn't affect anything
- Option selection dialog doesn't have transparency.
- Merge port 1025 with reverse proxy to 443.

Questions?