



# 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

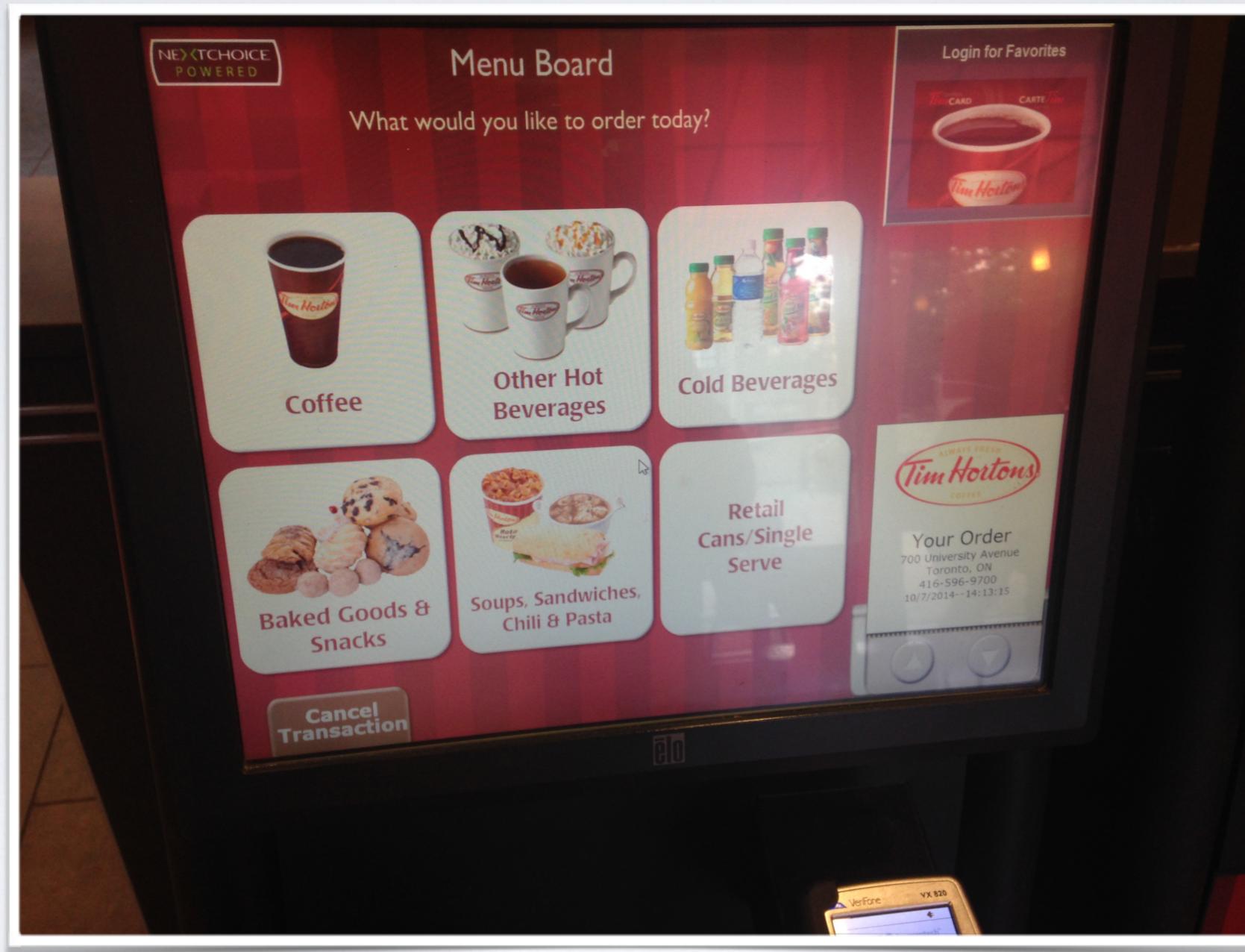
CASHIER  
(60%)



KITCHEN  
(100%)

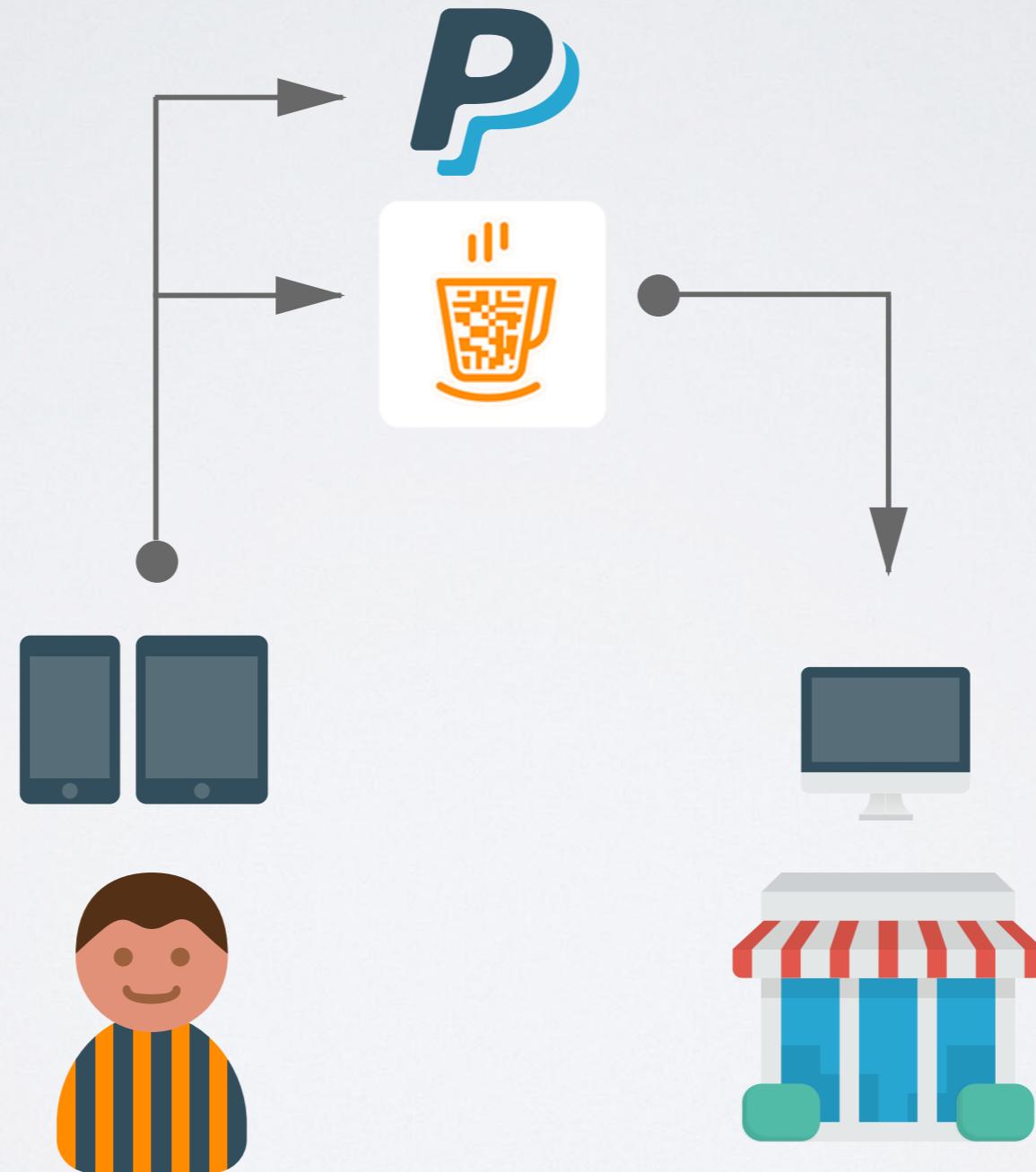


# CURRENT ATTEMPTS

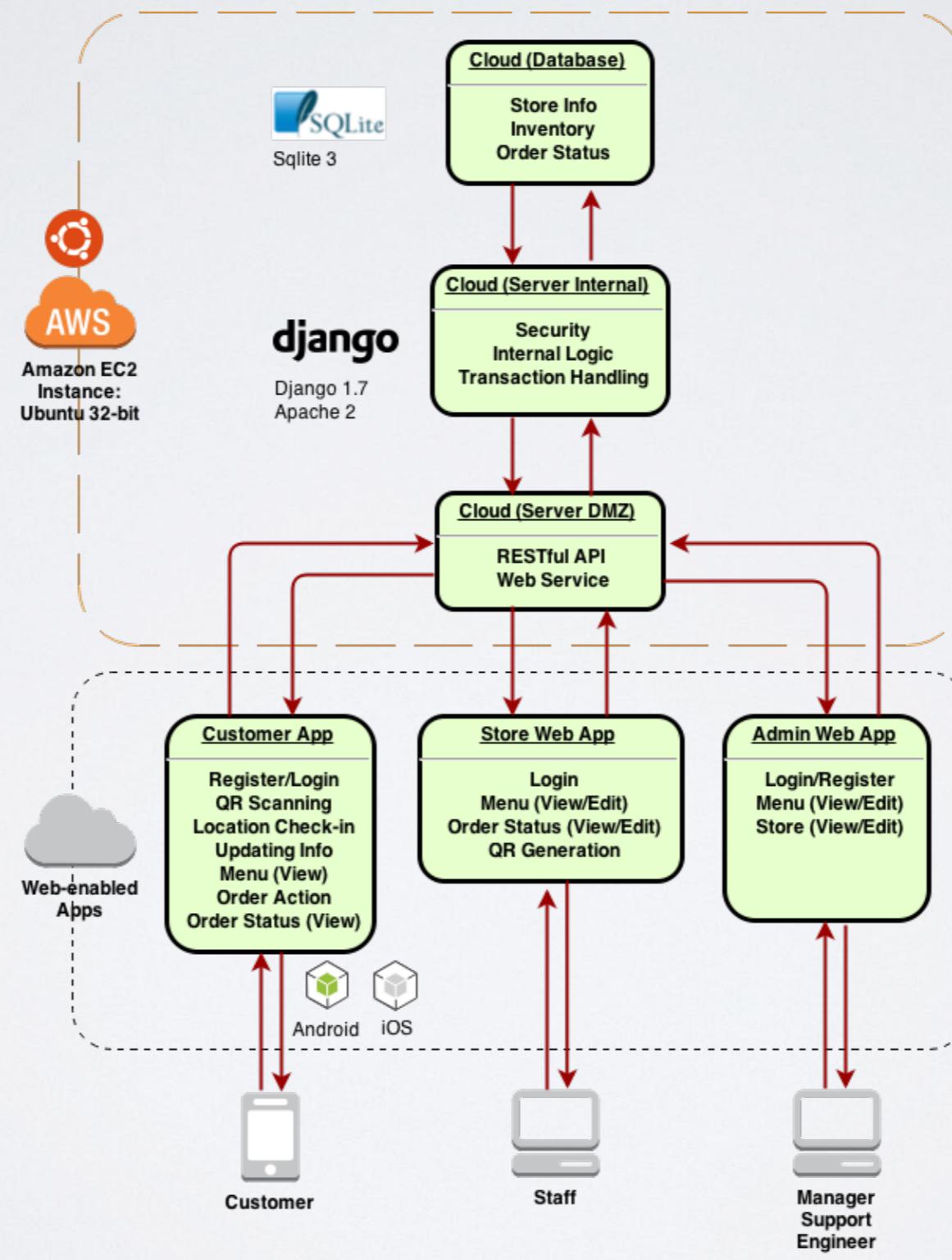


Tim Hortons ticket machine, at College/University

# SOLUTION OVERVIEW



# LOGIC FLOWCHART



# COMPONENTS



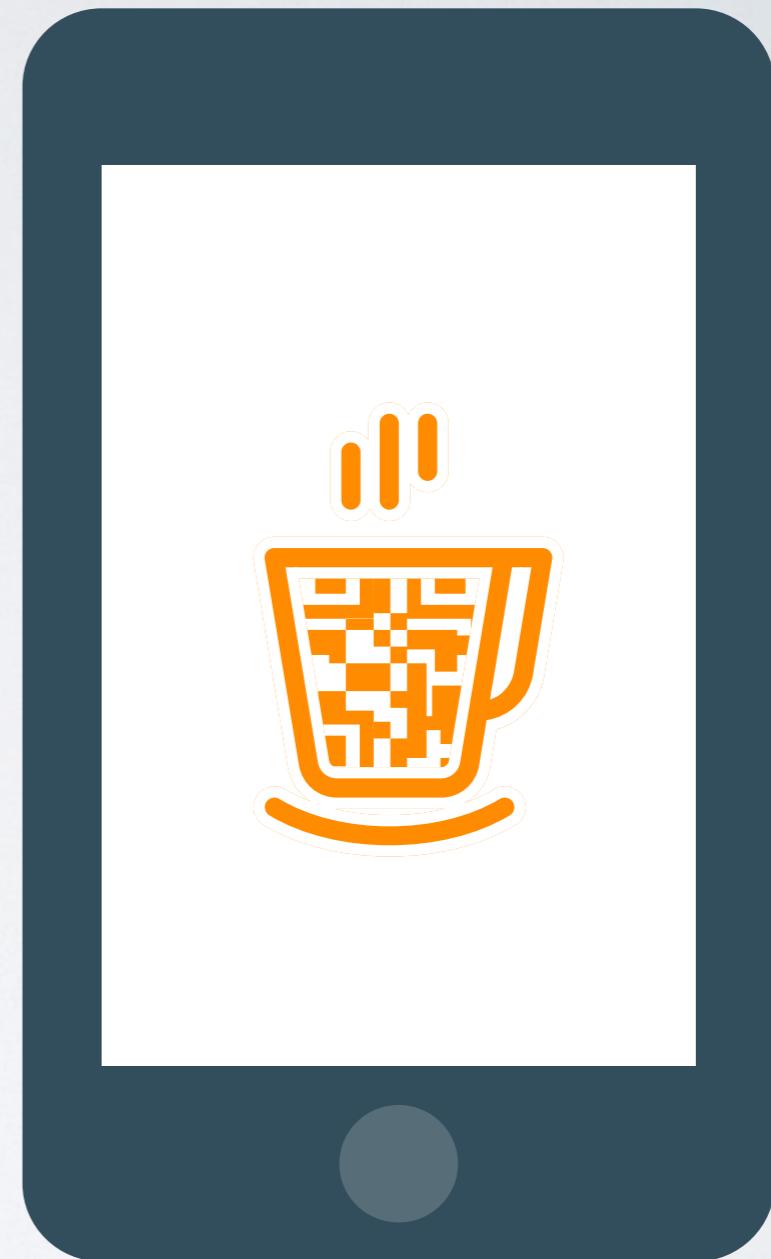
①



②

# CHAI APP

## Interface



# DEMO

Store landing (no password): <https://chaiapp.tk/store-api>

Store landing (password): [https://chaiapp.tk/store-danny\\_store](https://chaiapp.tk/store-danny_store)

Manager page: <https://chaiapp.tk/admin>

Customer PayPal accounts:

[chai.buyer.1@gmail.com](mailto:chai.buyer.1@gmail.com) (Pass: 00000000)

[chai.buyer.2@gmail.com](mailto:chai.buyer.2@gmail.com) (Pass: 00000000)

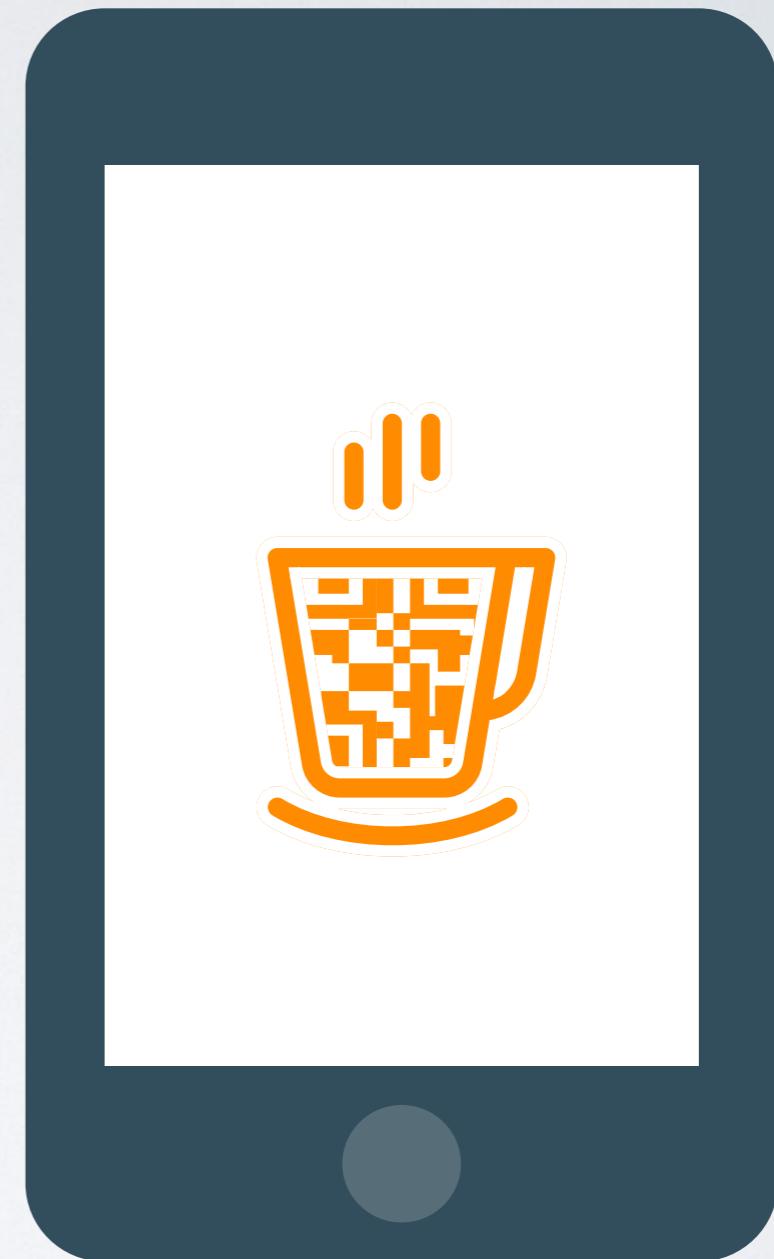
Store/manager accounts:

chaiadmin (pass: 00000000)

storemanager (pass: 00000000)

# CHAI APP

## Logic



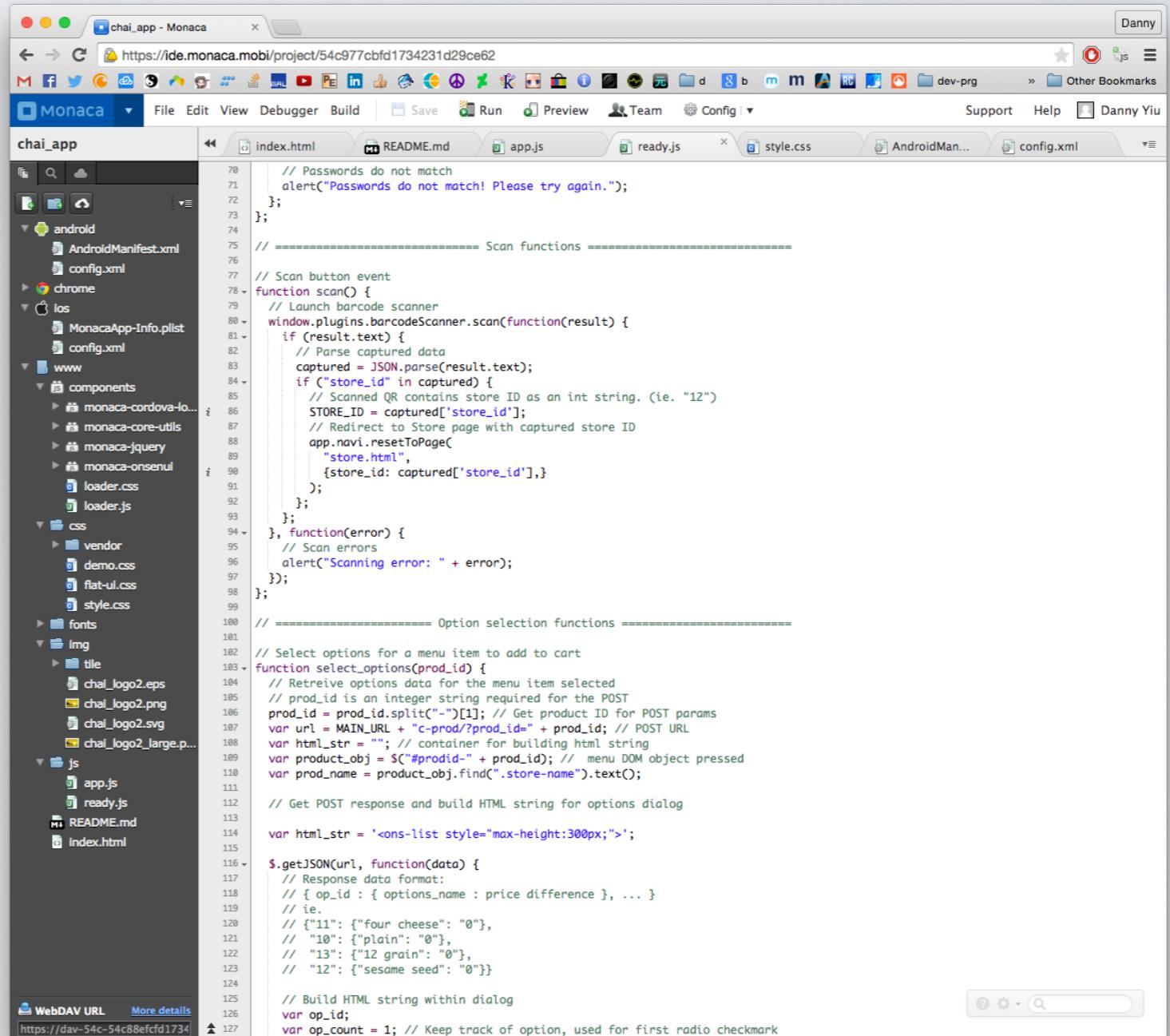
# SPECIFICATIONS

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



# DEVELOPMENT

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



The screenshot shows the Monaca IDE interface. The left sidebar displays the project structure for 'chai\_app' across three platforms: android, chrome, and ios. The 'www' folder contains components like monaca-cordova-lo..., monaca-core-utils, monaca-jquery, monaca-onsenui, loader.css, loader.js, css, vendor, demo.css, flat-ul.css, style.css, fonts, img, js, app.js, ready.js, README.md, and index.html. The right panel shows the code editor with a portion of the JavaScript file 'app.js'. The code includes functions for password matching, barcode scanning, and option selection, utilizing Cordova plugins and AngularJS.

```
// Passwords do not match
alert("Passwords do not match! Please try again.");
};

// ===== Scan functions =====

// 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);
    });
};

// ===== Option selection functions =====

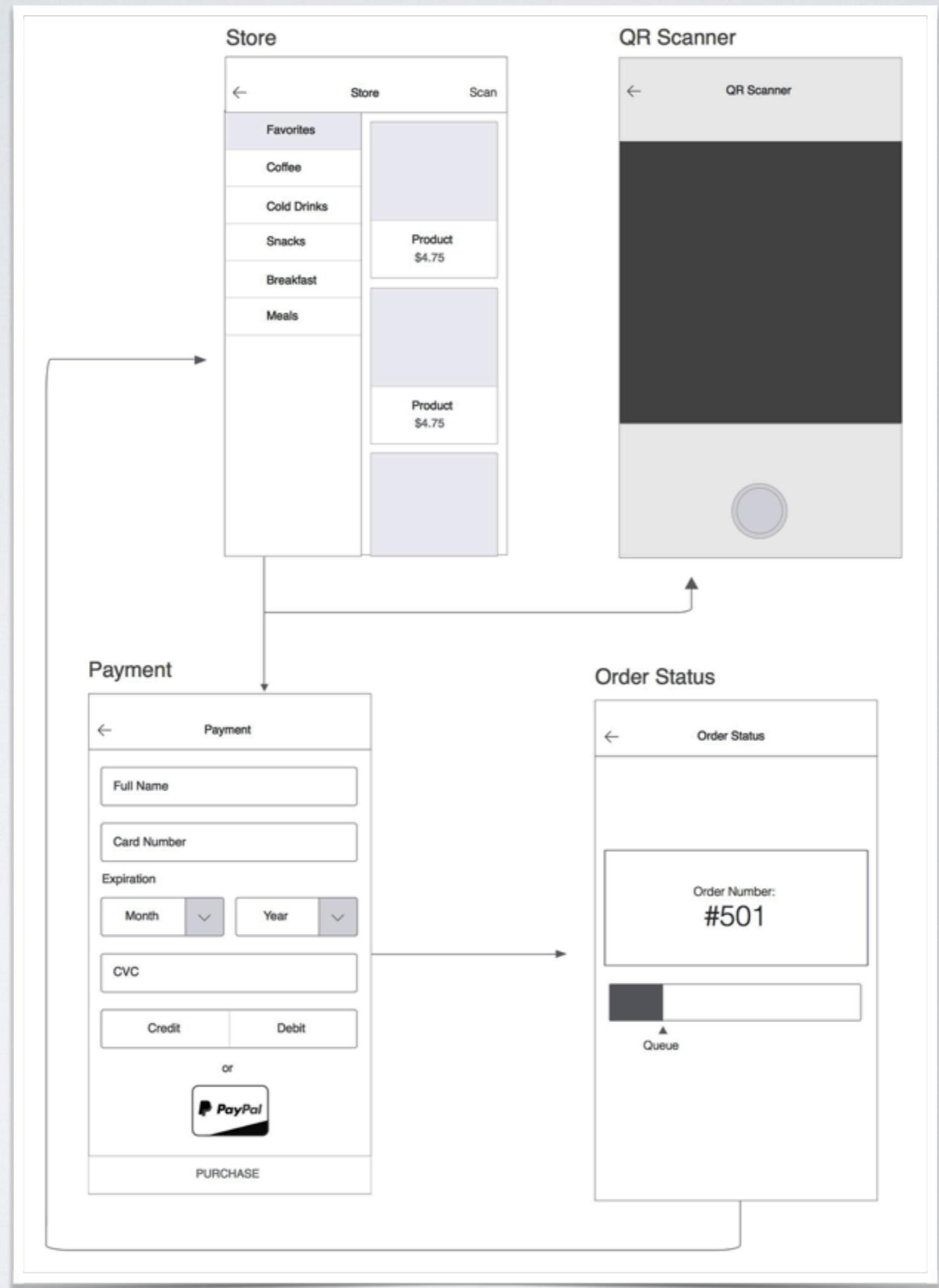
// Select options for a menu item to add to cart
function select_options(prod_id) {
    // Retrive options data for the menu item selected
    // prod_id is an integer string required for the POST
    prod_id = prod_id.split("-")[1]; // Get product ID for POST params
    var url = MAIN_URL + "c-prod/?prod_id=" + prod_id; // POST URL
    var html_str = ""; // container for building html string
    var product_obj = $("#" + prod_id); // menu DOM object pressed
    var prod_name = product_obj.find(".store-name").text();

    // Get POST response and build HTML string for options dialog
    var html_str = '<ons-list style="max-height:300px;">';

    $.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
    });
}
```

WebDAV URL: <https://dav-54c-54c88efcf1734>

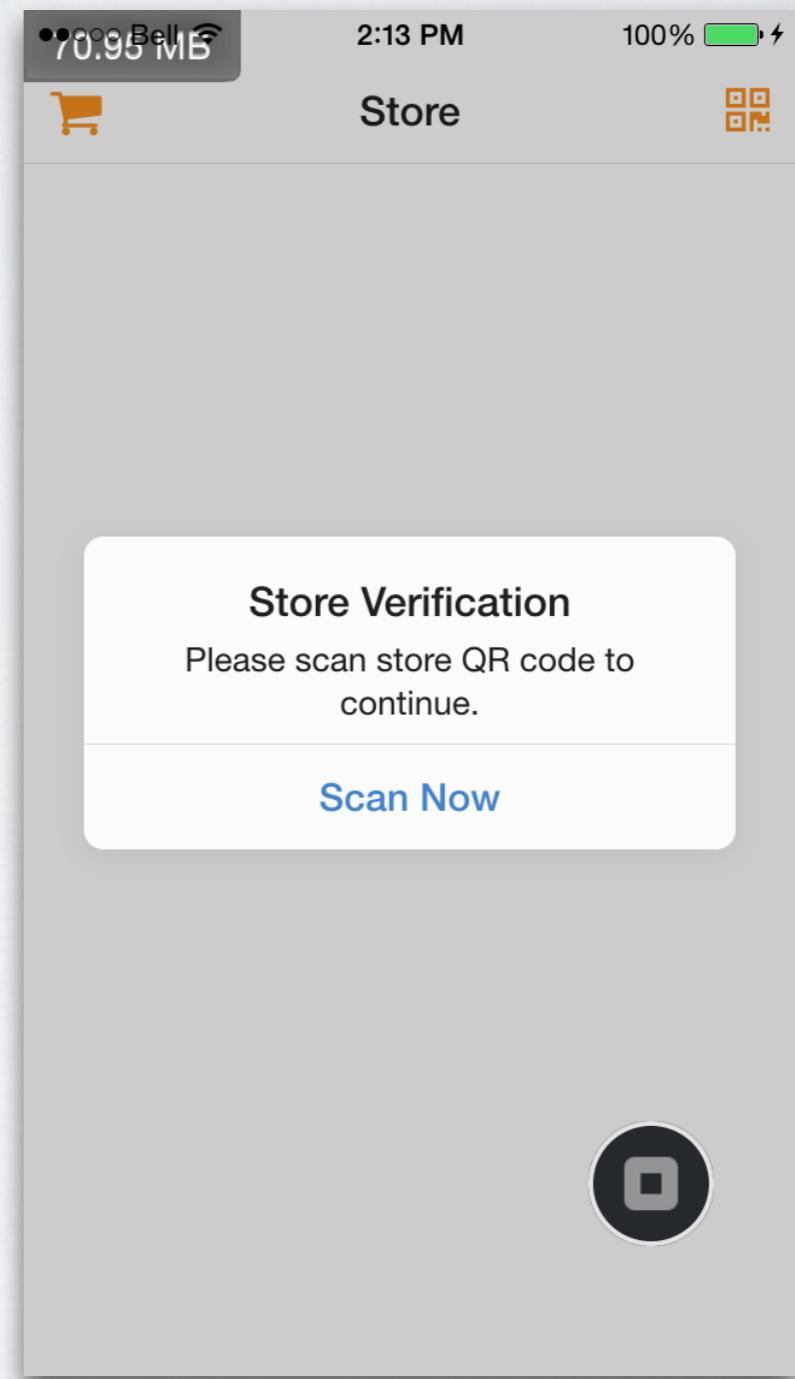


# STORE: BEFORE SCAN

- 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;

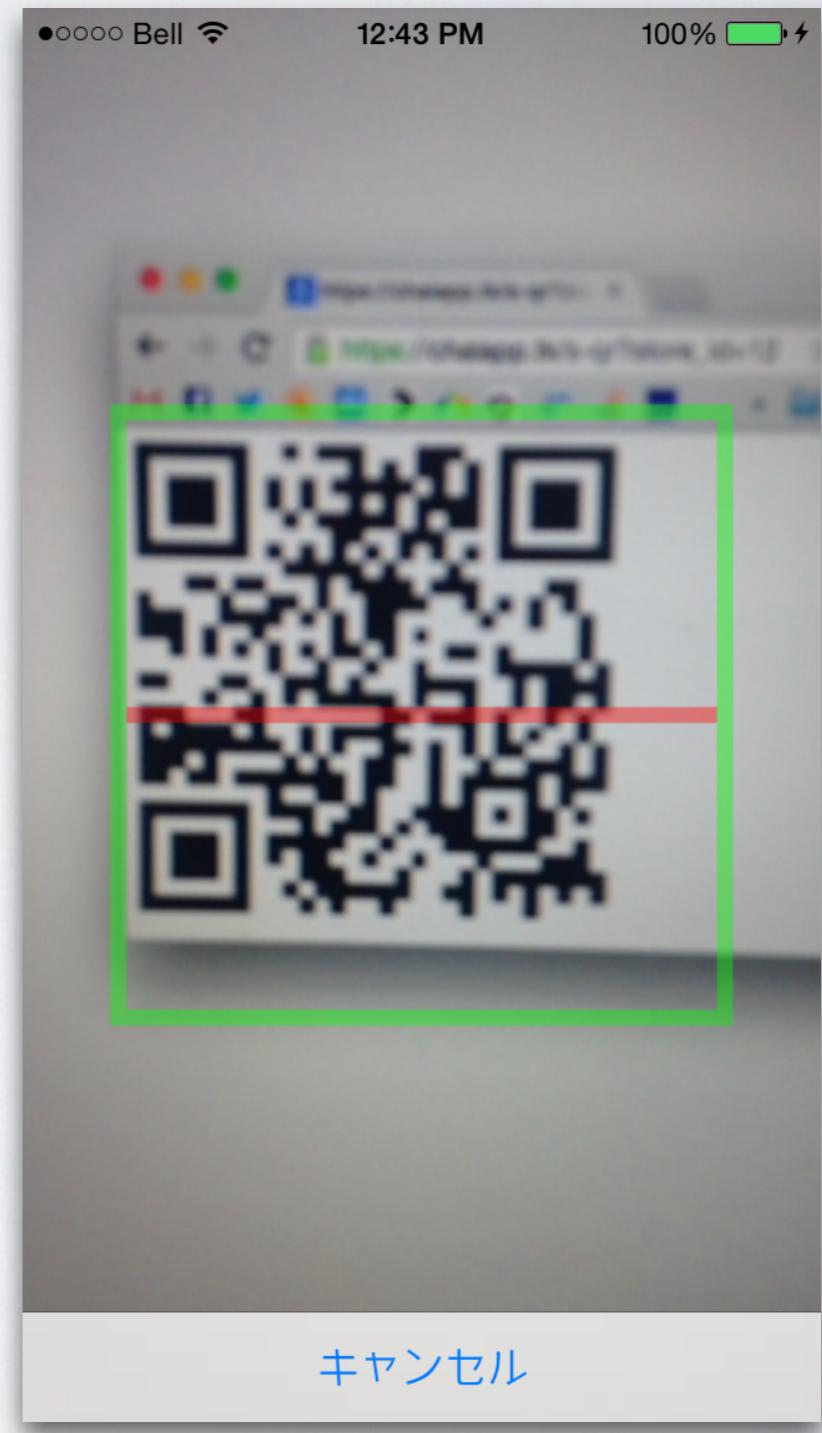
    // Check for store_id
    // store_id is only passed if QR code for store is scanned
    if ("store_id" in options) {} else {
        // Not inside store, or not scanned QR yet
        ons.notification.alert({
            message: 'Please scan store QR code to continue.',
            title: 'Store Verification',
            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

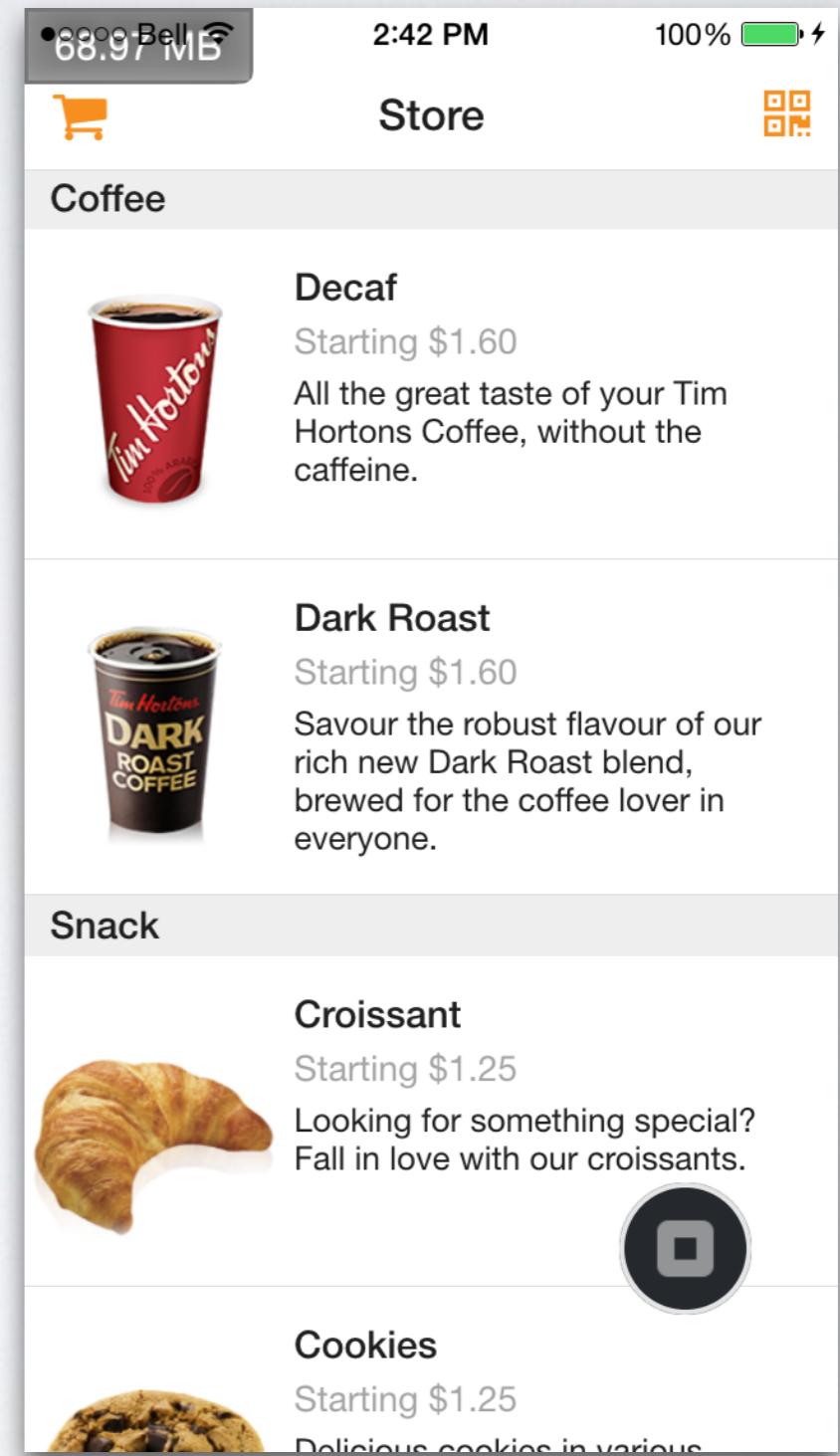
- Logic: Call c-menu API using store ID from QR. Display store menu with response JSON data.

```
// ===== 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");

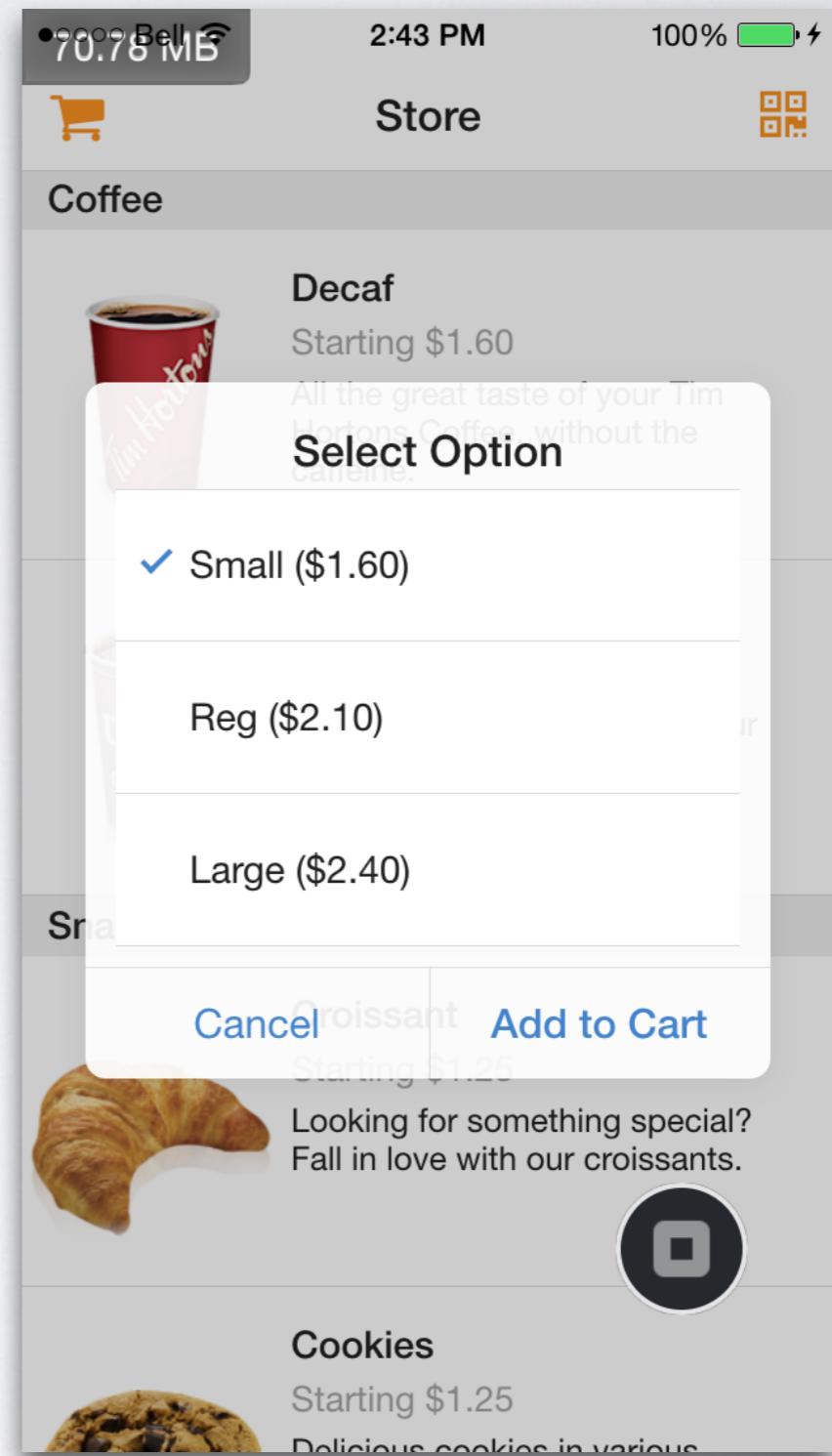
            for (category in data) {
                // HTML for each category
                // Note: category may be lower case, so capitalize CSS is used
                if (!(jQuery.isEmptyObject(data[category]))) {};
            };
            // Update store menu list
            content.innerHTML = html;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



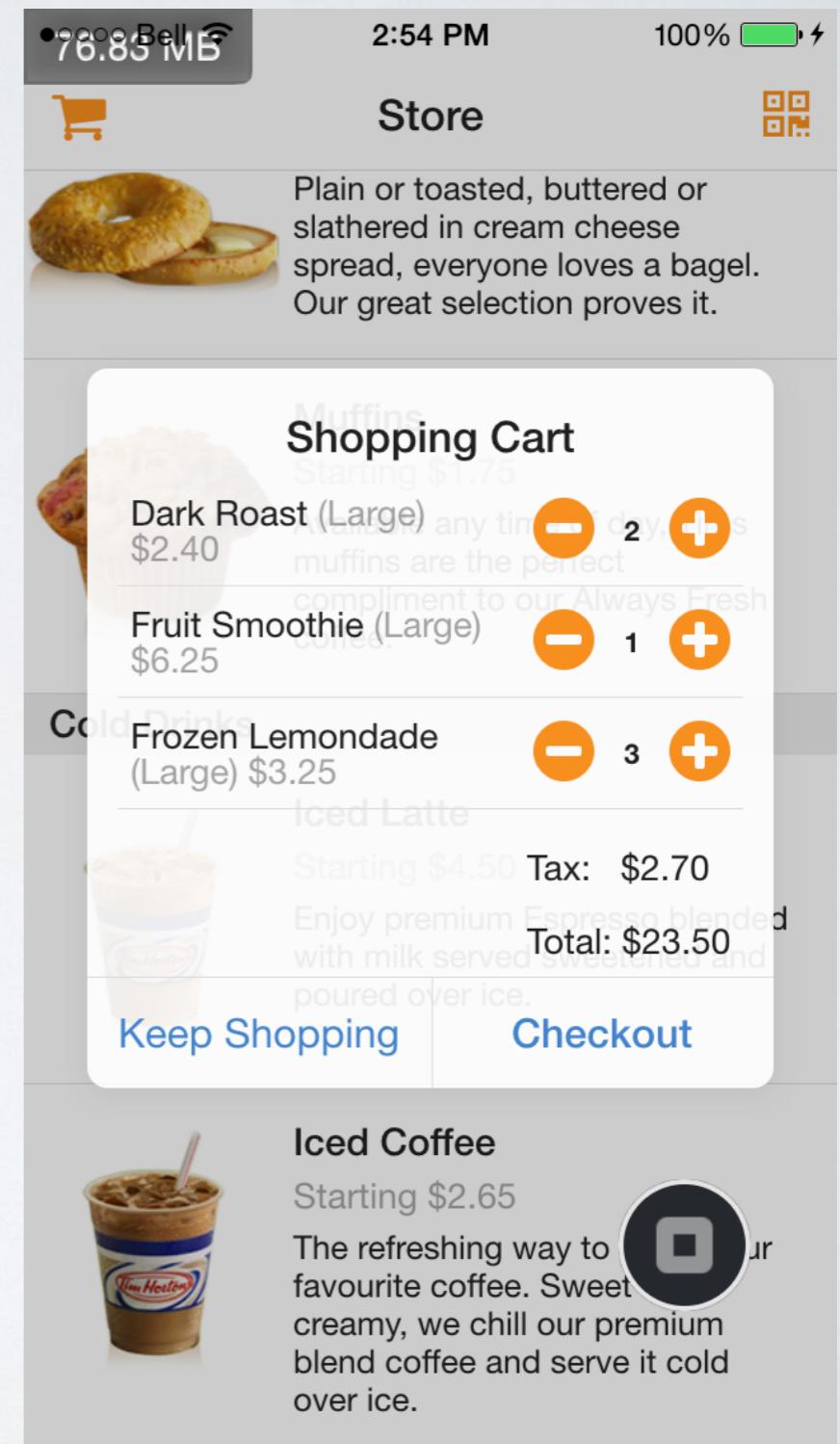
# STORE: SHOPPING CART

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

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

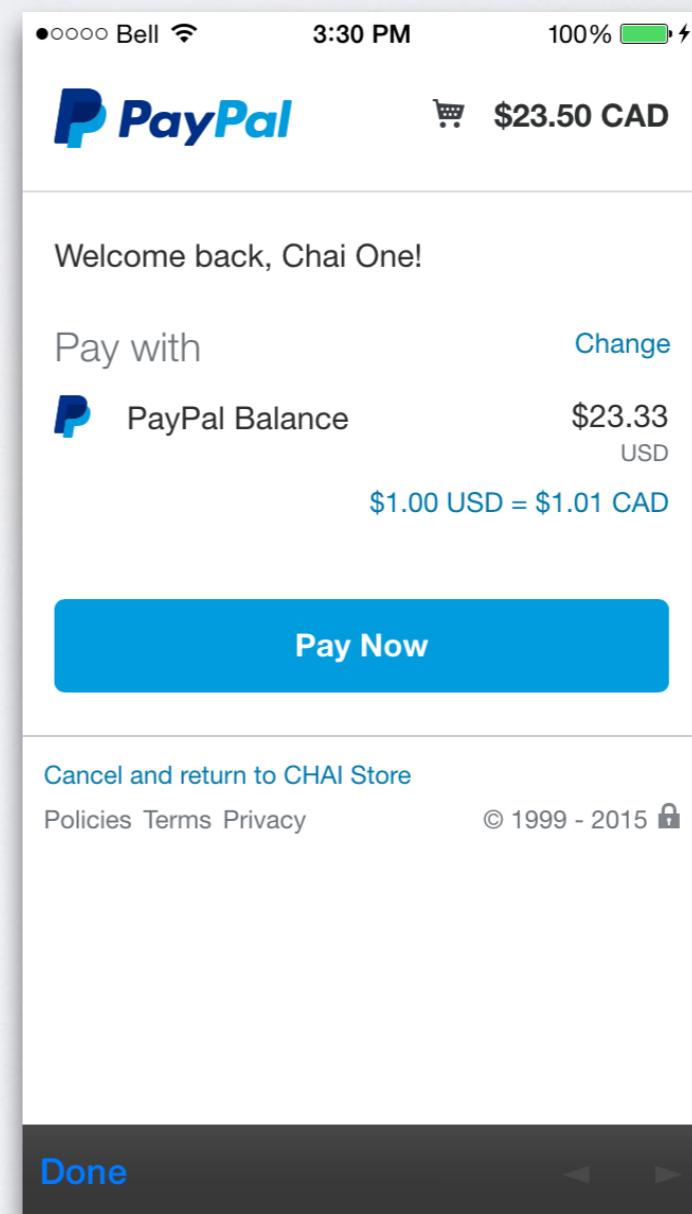
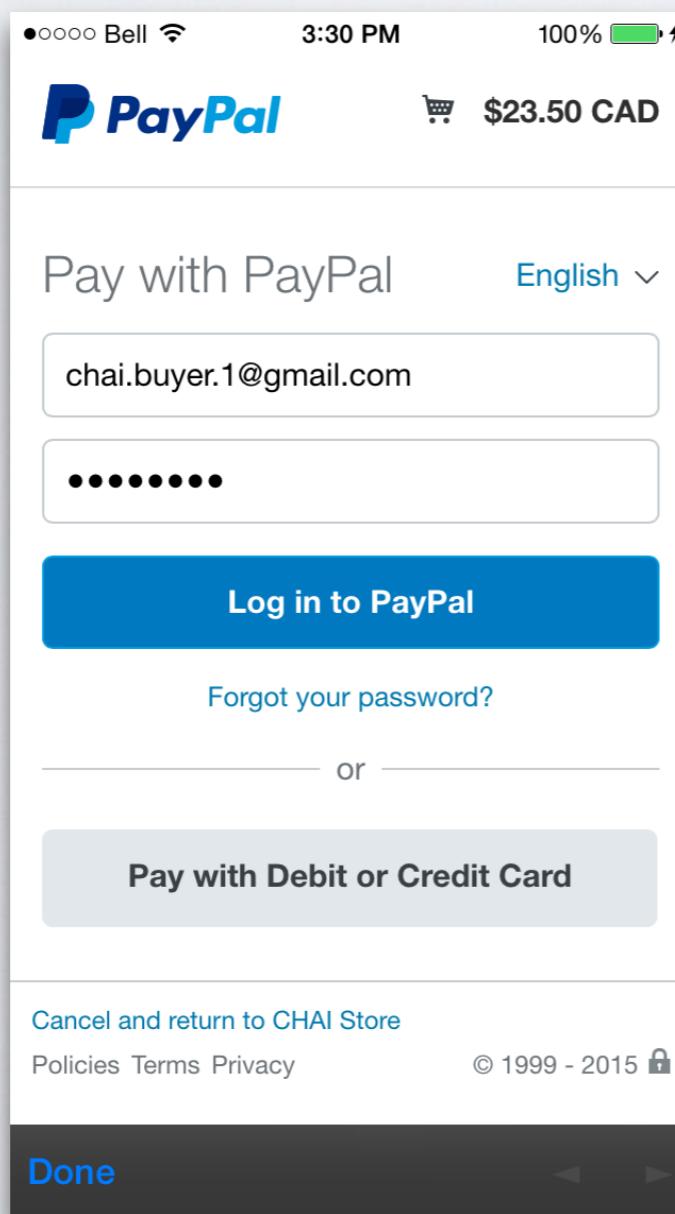
// Disable item in store menu
var prod_id = "prodid-" + get_prod_id(product);
$("#" + prod_id).hide(); // hide menu item

// Show cart
cart();
```



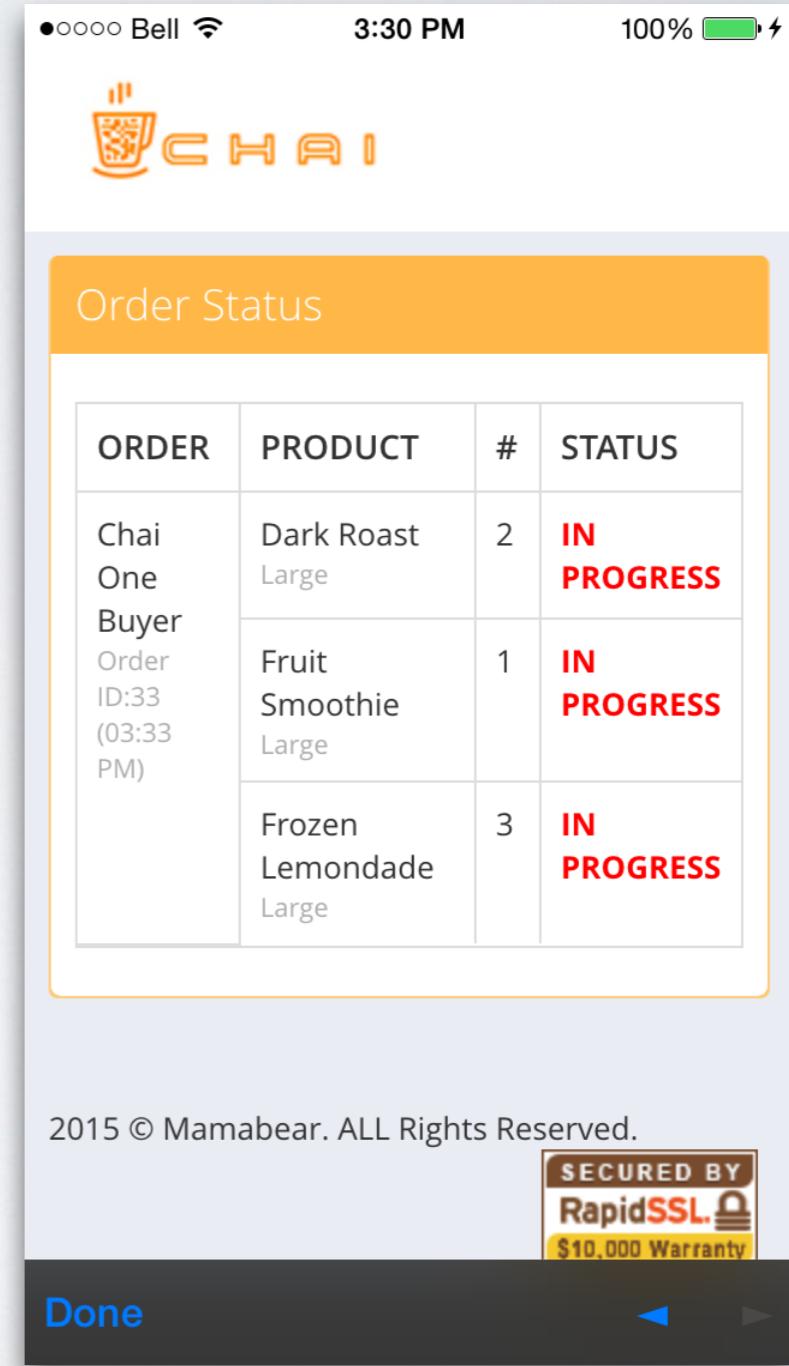
# PAYMENT

- 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: API (Store ID:12)

**Current Orders**

ORDER	CUSTOMER	TIME	PRODUCT	OPTION	#	STATUS
33	Chai One Buyer (21)	03:33 PM	Dark Roast	Large	2	<button>Complete</button>
			Fruit Smoothie	Large	1	<button>Complete</button>
			Frozen Lemonade	Large	3	<button>Complete</button>

**Inventory**

PRODUCT	CATEGORY	DISCOUNT	OPTIONS	PRICE	STATUS
Dark Roast	Coffee	1.00	Small	1.60	<button>Active</button>
			Reg	2.10	
			Large	2.40	
Decaf	Coffee	1.00	Small	1.60	<button>Active</button>
			Reg	2.10	
			Large	2.40	
Croissant	Snack	1.00		1.25	<button>Active</button>

•oooo Bell 3:30 PM 100%

**CHAI**

### Order Status

ORDER	PRODUCT	#	STATUS
Chai One Buyer Order ID:33 (03:33 PM)	Dark Roast Large	2	<b>IN PROGRESS</b>
	Fruit Smoothie Large	1	<b>IN PROGRESS</b>
	Frozen Lemonade Large	3	<b>IN PROGRESS</b>

2015 © Mamabear. ALL Rights Reserved.

**SECURED BY RapidSSL \$10,000 Warranty**

**Done**

Storefront: API (Store ID:12)

**Current Orders**

ORDER	CUSTOMER	TIME	PRODUCT	OPTION	#	STATUS
33	Chai One Buyer (21)	03:33 PM	Frozen Lemonade	Large	3	<button>Complete</button>

**Inventory**

PRODUCT	CATEGORY	DISCOUNT	OPTIONS	PRICE	STATUS
Dark Roast	Coffee	1.00	Small	1.60	<button>Active</button>
			Reg	2.10	
			Large	2.40	
Decaf	Coffee	1.00	Small	1.60	<button>Active</button>
			Reg	2.10	
			Large	2.40	
Croissant	Snack	1.00		1.25	<button>Active</button>

•oooo Bell 3:38 PM 100%

**CHAI**

### Order Status

ORDER	PRODUCT	#	STATUS
Chai One Buyer Order ID:33 (03:33 PM)	Dark Roast Large	2	<b>COMPLETE</b>
	Fruit Smoothie Large	1	<b>COMPLETE</b>
	Frozen Lemonade Large	3	<b>IN PROGRESS</b>

2015 © Mamabear. ALL Rights Reserved.

**SECURED BY RapidSSL 2048-bit root**

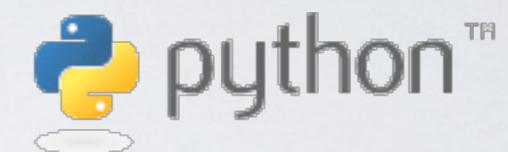
**Done**

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

The screenshot shows the AWS EC2 Management Console interface. The left sidebar navigation includes EC2 Dashboard, Events, Tags, Reports, Limits, INSTANCES (selected), Instances, Spot Requests, Reserved Instances, IMAGES, AMIs, Bundle Tasks, ELASTIC BLOCK STORE, Volumes, Snapshots, and NETWORK & SECURITY (Security Groups, Elastic IPs, Placement Groups, Load Balancers, Key Pairs, Network Interfaces). The main content area displays a single instance: i-3eacb0c5, t2.micro, us-east-1d, running, 2/2 checks, None, ec2-5. Below this, detailed instance information is shown in a table:

Description	Value	Description	Value
Instance ID	i-3eacb0c5	Public DNS	ec2-54-209-78-85.compute-1.amazonaws.com
Instance state	running	Public IP	54.209.78.85
Instance type	t2.micro	Elastic IP	54.209.78.85
Private DNS	ip-172-31-25-41.ec2.internal	Availability zone	us-east-1d
Private IPs	172.31.25.41	Security groups	launch-wizard-2, view rules
Secondary private IPs		Scheduled events	No scheduled events
VPC ID	vpc-13d55376	AMI ID	ubuntu-trusty-14.04-amd64-server-20140927 (ami-9ea1cf6)
Subnet ID	subnet-a99632de	Platform	-
Network interfaces	eth0	IAM role	-
Source/dest. check	True	Key pair name	dan
		Owner	364593754935

At the bottom, there are links for © 2008 - 2015, Amazon Web Services, Inc. or its affiliates. All rights reserved., Privacy Policy, Terms of Use, and Feedback.

## Edit inbound rules

X

Type <small>i</small>	Protocol <small>i</small>	Port Range <small>i</small>	Source <small>i</small>	
SSH	TCP	22	Anywhere <small>▼</small> 0.0.0.0/0	<small>X</small>
Custom TCP Rule	TCP	1025	Anywhere <small>▼</small> 0.0.0.0/0	<small>X</small>
HTTPS	TCP	443	Anywhere <small>▼</small> 0.0.0.0/0	<small>X</small>

Add Rule

Cancel

Save

Inbound rules example

The screenshot shows the AWS EC2 Management Console interface. The left sidebar includes sections for IMAGES, AMIs, Bundle Tasks; ELASTIC BLOCK STORE, Volumes, Snapshots; NETWORK & SECURITY, Security Groups, Elastic IPs, Placement Groups, Load Balancers; and AUTO SCALING, Launch Configurations, Auto Scaling Groups. The 'Key Pairs' section is currently selected. The main content area displays a table with one row for the key pair 'dan'. The table columns are 'Key pair name' and 'Fingerprint'. The 'Key pair name' column shows 'dan' and the 'Fingerprint' column shows 'e9:a2:ca:99:aa:98:ad:5b:7b:32:fc:09:7b:7d:0c:82:74:02:b5:55'. Below the table, a detailed view of the 'Key Pair: dan' is shown, repeating the key pair name and fingerprint. At the bottom of the page, there is a copyright notice for Amazon Web Services, Inc. and links for Privacy Policy and Terms of Use.

## Key pair generation

The screenshot shows a terminal window with the title bar 'dan.pem' and the status bar 'UNREGISTERED'. The window displays a large block of text representing an RSA private key. The text starts with '-----BEGIN RSA PRIVATE KEY-----' and ends with '-----END RSA PRIVATE KEY-----'. The key itself is a long string of characters, including letters, numbers, and symbols. At the bottom of the terminal window, there are status indicators: 'Line 1, Column 1', 'Spaces: 4', and 'Plain Text'.

```
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAsJYdJDKdC7m5YLL4930t0J2lHjUnBwlz4n70HvCZN8kejreea1tZwuF4o
Ym0kcE9kEHsa1e+KrMtMPx1R3vJboGVVwWrI2SKr3prEcN/vBK4hT4LicoOG/Kyx50MKKe1AQ+G8
UwcPwBs5piQsuCr76R4qflzXfdkLf+c0+3DMRe1gRTJgkISZAVldmhznZKnRR4+p2f1r0bn42+o
F+1sKc6/KUAIV3dXH7JMvZkacyiHPrYtyKcEnJoYt38Sp3A8oIYUEaknQLAW99NYC2xMqFFxhChT
qSJb4xo61apfHlfYJAISQTko4enQfLRRJ/5z0MKigdqNNZiHPpLa2wIDAQABoIBAGSHshNwJRQN
0ZscL7FDhZKM8rEn8PJG3afBJ/JKEqtWle+4H/vpikwTEJ96EPLMkx2Bi0Tf1Fea9TVb8AtCFW
nG8+AYOxfTK0+IWslwnVgJzTk5Tx4LzAwFKNbma0YMQ/Gso0gv6ky7s008S3Rqj7fITKGHz/QI2
uPltK8v3tZUgnoDAyKALHV0PDmSs61vNb/UHFYU2Yl1/cwPLGXgV5M3rFXg2+ciV192zXN8ip
Q+81LqtFrEpTwhSRFCZBEyxwQsxke621AvNp/ffqInpp17f040s0sZg2JHDHSer2LLT+51fG0AL/
XUK01JDivljS6jMHagA8yl3GWHECgYEAE3QEZ0unvWqbLlBxUNMF0F8dwU+E1XCi8yIswB1LLpSVF
/wsam0Ddh65B30gnTLFgu0s0gr3eM80l0FgeisuQKGPLIhpN0xz8c0BGsMK4xqibUD25Ba5Yc7fn
K+GDcP2LCfrTSaca75d+63JkiBrpSiCoV1lhsLRRlzz9MND1ccCgYEAzIxT81ibDBacmIg8MQNm
KrEzc8adDozVL563EfGMIq0nB8weN38ByVz0ByxfQMwZPtvl1JZZ2Nrnam+IE+aB8GINA7JDcx
vckgSwT/qV/L7uxN05aEnGlVcpVUxuo5hLVemfaPCrTpIxmxpGaS3vXSJ9x19gCgvEhYJCiAAk0C
gYA+xTegng0Dnf6CwnHqhe6HUVELbTwJgnok8/iZlu+niiRW0AHB+cjwF5pTF8N1IwsFhda/v57
iXYl/ls+ZK7fKejqGLp9+b+hd7HcRpIgqobidYV1nZ010Mfx6K1ijEDPztRYGQRG027zBjyVI7+a
9EBXDNX4UF7UK9hx05SqxQKBgH8Mqi7KipMtkEJ1jA1dcR3fRngJQK6sT6q7sFAV/JG13fI2pSLN
rqScYBA+AMvXU/0AZtCuIJqPC6thcyAzFpIRgvrqm3FZhvkobHxsNGf9amuSNj5mtSN5gWtU0X37
iZNYfxPeQuaBi7UTdk7UrzdXo0Eru9D6ftuzi5uZ1diZAoGADxnTj/PqQA9hzJZdgIac7mL1A1Wr
6rmil1J87plRmwF0z4e58PjhkX00GXVqb4iC2oLkdu6P0HMsIrbt5dCpt4GbktvE7aY0Q6J//hl
SM/rdCm8nt9FPzWnKMKHpdGd0rs311YFdFelMC5xQRA0dLeg/k27gCTReMRbDoI8ak=
-----END RSA PRIVATE KEY-----
```

```
1. ubuntu@ip-172-31-25-41: ~ (bash)
Last login: Tue Apr  7 16:56:37 on console
dmbp:~ dannyiu$ cd ~/GitHub/ec2t/
dmbp:ec2t dannyiu$ cat sshchaiapp.command
ssh -i res/dan.pem ubuntu@chaiapp.tk
dmbp:ec2t dannyiu$ ./sshchaiapp.command
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-36-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

System information as of Sun Apr  5 18:05:47 UTC 2015

System load:  0.0          Processes:      103
Usage of /:   16.8% of 7.74GB  Users logged in:    0
Memory usage: 27%          IP address for eth0: 172.31.25.41
Swap usage:   0%

Graph this data and manage this system at:
https://landscape.canonical.com/

Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud

134 packages can be updated.
78 updates are security updates.

*** /dev/xvda1 should be checked for errors ***

Last login: Sun Apr  5 18:05:48 2015 from 199.212.27.188
ubuntu@ip-172-31-25-41:~$ pwd
/home/ubuntu
ubuntu@ip-172-31-25-41:~$ ls --color=none
chai_cloud_deploy  packages
ubuntu@ip-172-31-25-41:~$
```

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)

```
1. ubuntu@ip-172-31-25-41: ~/chai_cloud_deploy/_scripts (bash)
ubuntu@ip-172-31-25-41:~/chai_cloud_deploy/_scripts$ clear
ubuntu@ip-172-31-25-41:~/chai_cloud_deploy/_scripts$ pwd
/home/ubuntu/chai_cloud_deploy/_scripts
ubuntu@ip-172-31-25-41:~/chai_cloud_deploy/_scripts$ cat _aws-setup
sudo apt-get update
sudo apt-get install apache2 libapache2-mod-wsgi
sudo apt-get install python-setuptools
sudo apt-get install python-pip
sudo apt-get install python-dev
sudo easy_install django
sudo apt-get install mysql-server python-mysqldb
sudo easy_install httpie

sudo easy_install django-filter

sudo easy_install twisted
sudo easy_install-2.7 pyopenssl
sudo python twisted-websocket-4173-4/setup.py install

sudo easy_install pytz

sudo apt-get install libffi-dev
sudo easy_install bcrypt

sudo cp /home/ubuntu/chai_cloud_deploy/chai.conf /etc/apache2/sites-enabled/chai.conf
sudo cp /home/ubuntu/chai_cloud_deploy/ports.conf /etc/apache2/ports.conf

sudo chmod -R a+x /home/ubuntu/chai_cloud_deploy/
sudo chmod -R a+w /home/ubuntu/chai_cloud_deploy/static/
sudo chmod -R a+w /home/ubuntu/chai_cloud_deploy/media/
sudo chmod -R a+w /var/www/

sudo a2enmod ssl

sudo service apache2 restart
ubuntu@ip-172-31-25-41:~/chai_cloud_deploy/_scripts$
```

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

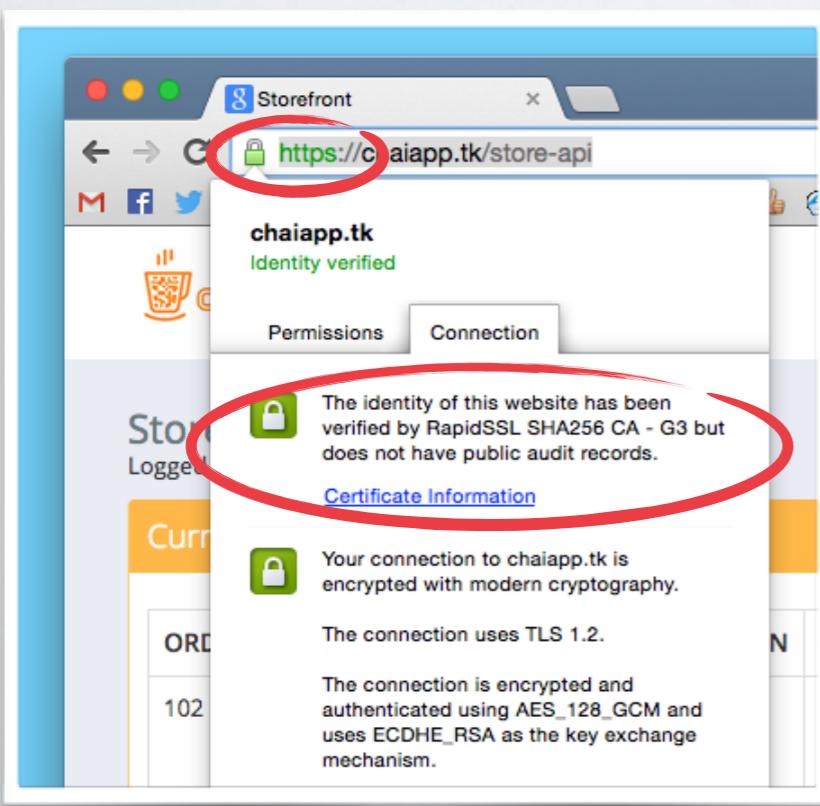
<IfModule ssl_module>
    Listen 443
</IfModule>

<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

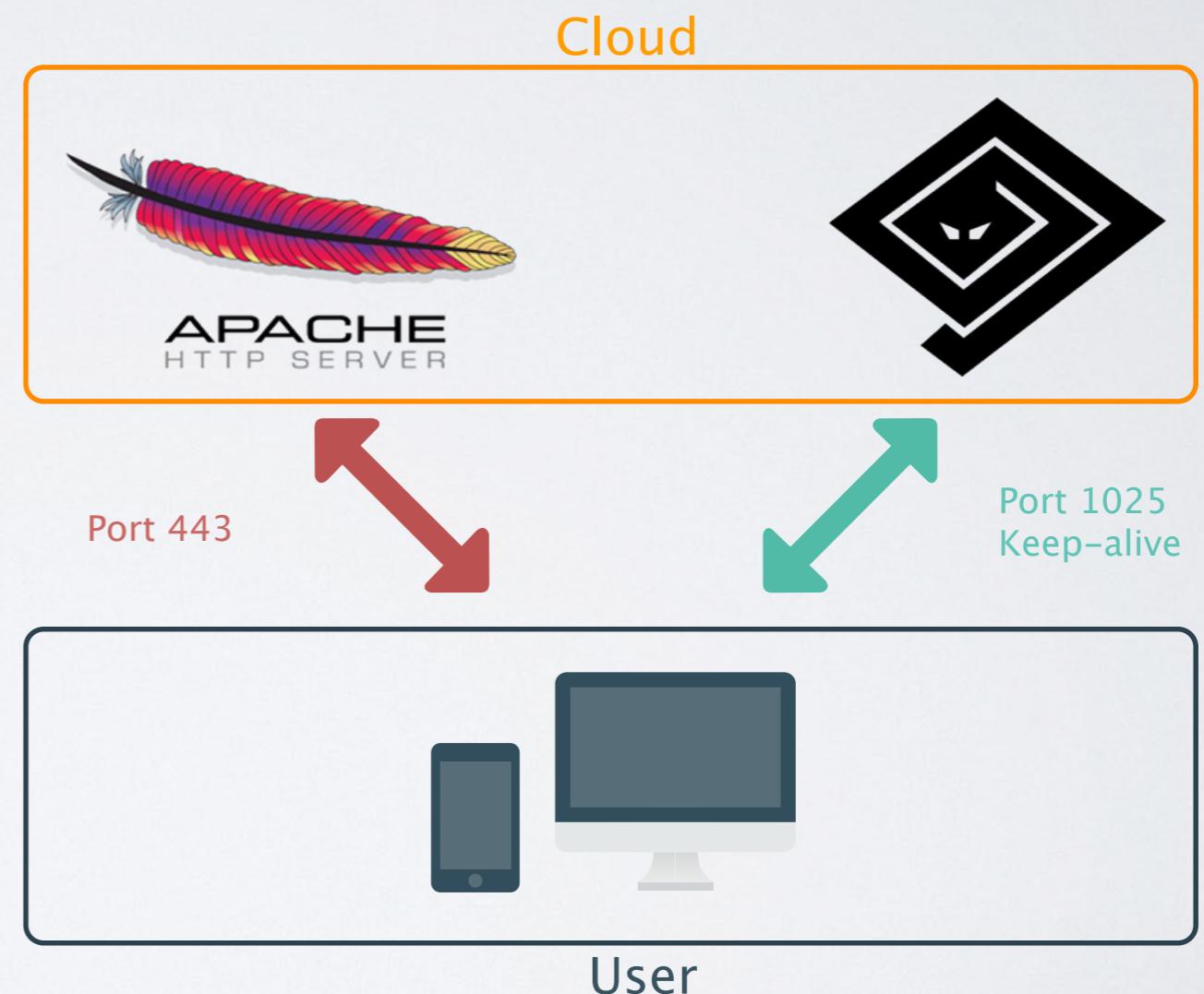
- Network data encryption
- Three keys required: Private, public, intermediate CA.



```
1. ubuntu@ip-172-31-25-41: /etc/apache2/ssl-certs (bash)
ubuntu@ip-172-31-25-41:/etc/apache2/ssl-certs$ pwd
/etc/apache2/ssl-certs
ubuntu@ip-172-31-25-41:/etc/apache2/ssl-certs$ ls -lA
total 12
-rw-r--r-- 1 root root 2766 Mar 22 19:39 rapidssl_intermediateca_2015.crt
-rw-r--r-- 1 root root 1678 Mar 22 19:36 rapidssl_privatekey_2015.key
-rw-r--r-- 1 root root 1680 Mar 22 19:46 rapidssl_publickey_2015.crt
ubuntu@ip-172-31-25-41:/etc/apache2/ssl-certs$ cat rapidssl_publickey_2015.crt
-----BEGIN CERTIFICATE-----
MIIEeqzCCA50gAwIBAgIDAyShMA0GCSqGSIb3DQEBCwUAMEcxCzAJBgNVBAYTA1VT
MRYwFAYDVQQKEw1HZW9UcnVzdCBjbmMuMSAwHgYDVQQLDExdSYXBpZFNTTCBTSEEy
NTYgQ0EgLSBHMsAeFw0xNTAzMjExMTM0MjdaFw0xNjAzMjMxMTE3MzdaMIGSMRMw
EQYDVQQLEwpHVDUz0TMy0DkyMTEwLwYDVQQLEyhTZWUgd3d3LnJhcG1kc3NsLmNv
bS9yZXNvdXJjZXMuY3BzIChjKTE1MS8wLQYDVQQLEyZeb21haW4gQ29udHJvbCBW
YWhpZGF0ZWQgLSBSYXBpZFNTTChSKTEXMBUGA1UEAxM0d3d3LmNoYWlhchHAudGsw
ggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCTGrb2vwRUx34i9sPQU3gS
rCuXcUDBNb8JoNh/p31aKOUvPTEUn2YHdw2xFuWkdIQuGuX79aHf7egUfrTDbbJ4
0Zc4xNC7xjcPzdQe5ds6QjIe04r7/JN3Ht7Rx05kwq+g5Uq+0kw1ww0o3UhfpDsX
3mhbYVCXebZxAQV3XCKaSf7YkmFSc3USqIs728MDqziGIXlsqgHNmuX/MmW2S08
fnGh17w/fqkI0WvH0M5A+UaDaXBdWjhrIowZ0hhvvwbrJ4e/s5D3eMd/Y9/FbhVQ
eeSFZT3UpGNNIEntvJIP11YE9R/wMDRAtfaf2tcEelT/tiMEPN9TNp0wIMvYwm88f
AgMBAAGjggFSMIIIBTjAfBgNVHSMEGDAwBTDnPP800YINLw0Rn+gfFvz4gjLWTBX
BggrBgfEFBQcBAQRRLMEkwHwYIKwYBBQUHMAGGE2h0dHA6Ly9ndi5zeW1jZC5jb20w
JgYIKwYBBQUHMAKGGmh0dHA6Ly9ndi5zeW1jYi5jb20vZ3YuY3J0MA4GA1UdDwEB
/wQEAWIFoDAdBgNVHSUEfjAUBggrBgfEFBQcDAQYIKwYBBQUHAvIwJQYDVR0RBB4w
HII0d3d3LmNoYWlhchHAudGuCCmNoYWlhchHAudGswKwYDVR0fBCQwIjAgoB6gHIYa
aHR0cDovL2d2LnN5bWWNiLmNvbS9ndi5jcmwwDAYDVR0TAQH/BAIwADBBBgNVHSAE
OjA4MDYGBmeBDAECATAsMC0GCCsGAQUFBwIBFh5odHRwczovL3d3dy5yYXBpZHNz
bC5jb20vbGVnYWhwDQYJKoZIhvcNAQELBQADggEBAASgVdSLL0jnH924GEdnobL/
JWjUxzalWBDwvM+kAF/60E/wf914xPgog9MWrJWdsgibWF3AqUSuCxq3vq+H/01+7
3zt9+pYE/opFX/Y4/SPkkdDYCFYRMEtWz1QAyfiAEzC/UZIsPGRzZzppVBj7+2oo
M1P6K5Ku+5hv7K8fRKWqzLNhIgbLrE2XcyqKqdyob70eC4LjoGMEyA8arjKjF09V
uhUbTg7EBK4RDo/mwqZbQ0G51fnLvoBEGj65zm2fzAfWhDEEW+e91sadB+o792HG
jibQ3hpYPfliiIv675yMX0jA8oQGb7+m4IqVj2KsQSC8nAAT0ZtqNr/rrE48M0k=
-----END CERTIFICATE-----
ubuntu@ip-172-31-25-41:/etc/apache2/ssl-certs$
```

# WEBSOCKET SERVER

- Twisted 13.1.0 (Python)
- Real-time components



# WEBSOCKET EXAMPLE

The screenshot shows a web browser window titled "Storefront" at the URL <https://chaiapp.tk/store-api>. The page displays a "Store: Api" interface with "Store ID:12" and a user logged in as "AnonymousUser". Two main sections are visible: "Current Orders" (orange header) and "Inventory" (blue header). The "Current Orders" section has a table with columns: ORDER, CUSTOMER, TIME, PRODUCT, OPTION, #, and STATUS. The "Inventory" section has a table with columns: PRODUCT, CATEGORY, DISCOUNT, OPTIONS, PRICE, and STATUS, showing entries for "Dark Roast" and "Reg". Below the main content is a developer tools Network tab showing a request to "ws://chaiapp.tk:1025/ws" with a status code of 101 (Switching Protocols). The request headers include "Connection: Upgrade", "Upgrade: WebSocket", and "Sec-WebSocket-Accept: N8r9zpDIRPXbu2ZXfKsS12llpNA=". The response headers show "Status Code: 101 Switching Protocols".

Listen to websocket server on page load.

# WEBSOCKET EXAMPLE

The screenshot shows a web browser window titled "Storefront" at <https://chaiapp.tk/store-api>. The page displays a "Current Orders" section with one item: Order 102 for Chai One Buyer (21) at 04:03 PM, a Dark Roast Large. A "Complete" button is visible. To the right is an "Inventory" section for Dark Roast, categorized under Coffee, with sizes Small, Reg, and Large. An "Active" status indicator is present. Below the browser window is the browser's developer tools Network tab, specifically the Headers tab. It shows a list of requests, with the last two entries being websocket connections. The second-to-last entry is a POST request to "ws" with the following JSON payload:

```
{"action": "order", "orderlist": [{"prodname": "Dark Roast", "op": "Large", "quantity": 1, "oid": 213}], "storename": "api", "customer": "Chai One Bu..."}
```

The Network tab also shows other requests from "fonts.gstatic.com" and "static/assets/global...".

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*|static|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
├── 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

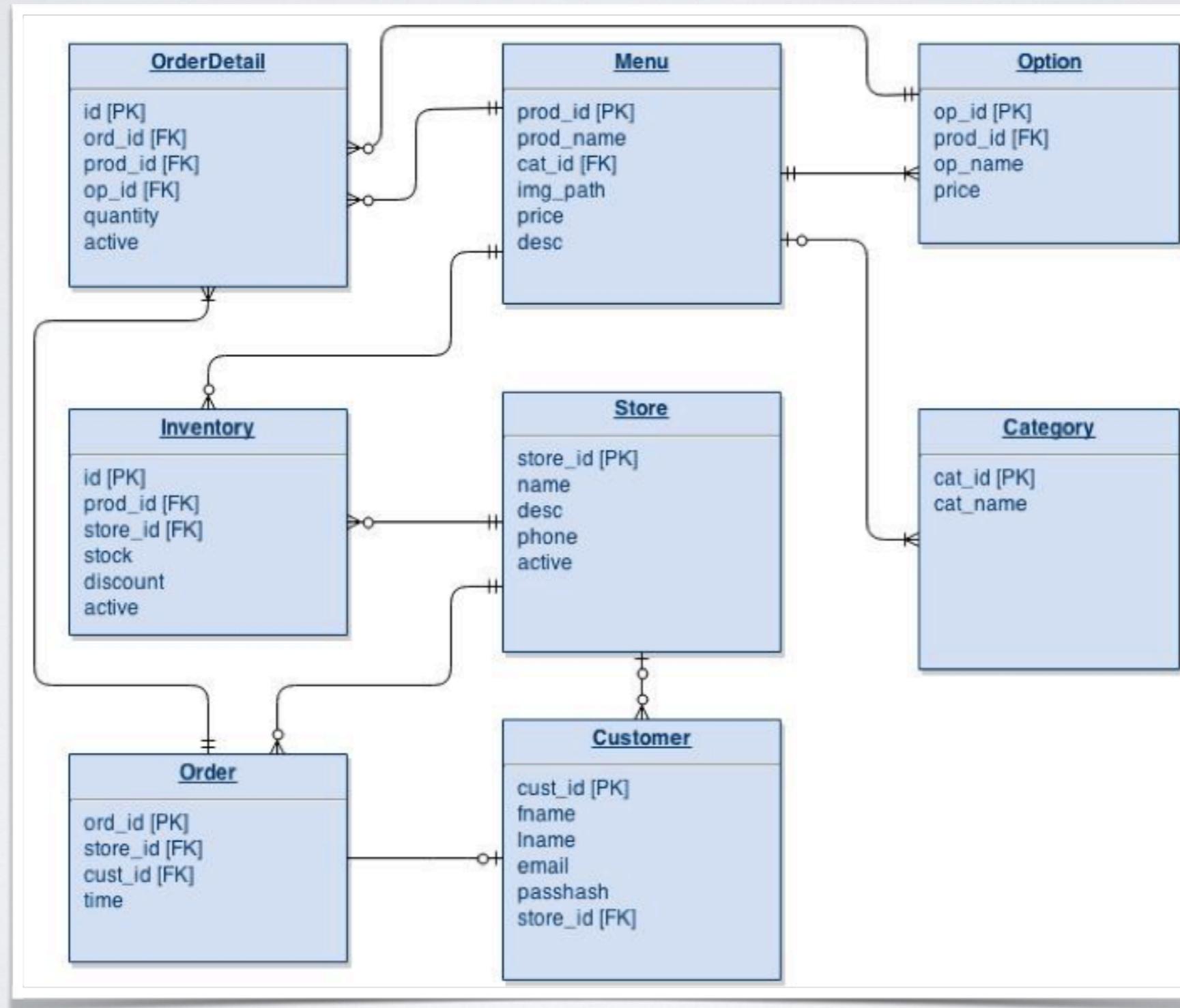
```
25 HTTP_URL = "http://localhost/" # DEPLOY (HTTP is okay here since used locally)
26 WS_URL = "wss://chaiapp.tk:1025/ws" # SSL DEPLOY
27
28 # PAYPAL GLOBALS
29 PP_CURRENCY = "CAD"
30 PP_TOKEN_URL = "https://api-3t.sandbox.paypal.com/nvp"
31 PP_EMAIL = "chai-facilitator_api1.gmail.com"
32 PP_MERC_ID = "BZ8PCS9ZDYTTG"
33 PP_PSS = "TSHBEMMJVLWBHHU" # note: this is not login password
34 PP_SIGNATURE = "AFcWxV21C7fd0v3bYYRCpSSRl31AXEh9liEDPaCcNurPvdeesaTwyYs"
35 PP_TEST_EMAIL = "chai-buyer@gmail.com" # used to autofill checkout email (testing)
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

```
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
```

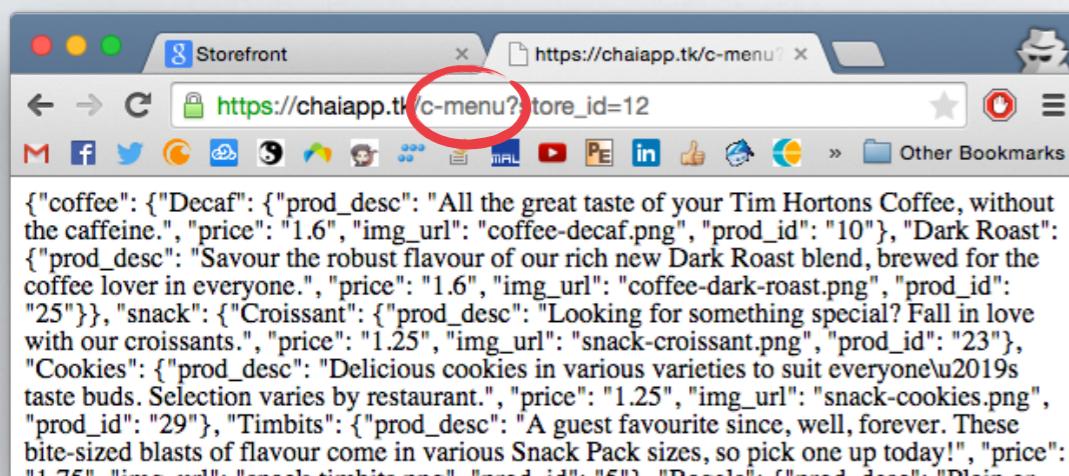
Line 84, Column 16      Spaces: 4      Python

# DJANGO: MODELS



# DJANGO: URL ROUTING

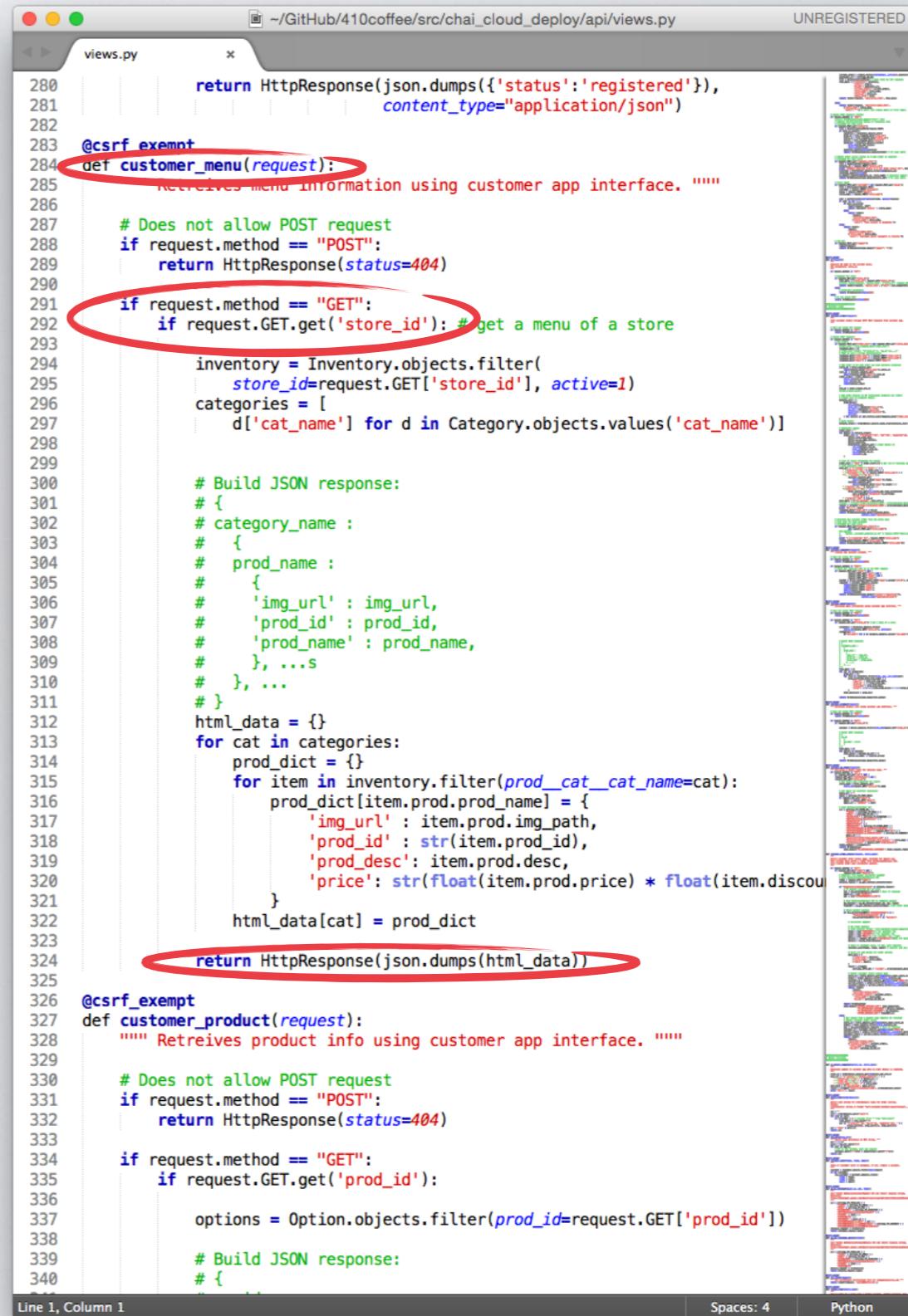
- Regex URL patterns matching views



```
urls.py ~/GitHub/410coffee/src/chai_cloud_deploy/chai_cloud/urls.py UNREGISTERED
1 from django.conf.urls import patterns, include, url
2 from api.views import *
3 from django.contrib import admin
4 from django.views.generic import RedirectView
5 from django.conf.urls.static import static
6 from django.conf import settings
7
8 urlpatterns = patterns(
9     '',
10    url(r'^admin/', include(admin.site.urls)),
11    url(r'store-(?P<store_name>[a-zA-Z0-9_]+)', store_view, name="store"),
12    url(r's-qr/?', qr, name="qr"),
13    url(r'c-order/?', customer_order, name="c-order"),
14    url(r'c-menu/?', customer_menu, name="c-menu"),
15    url(r'c-prod/?', customer_product, name="c-prod"),
16    url(r'c-token/?', customer_pp_token, name="c-token"),
17    url(r'c-status-(?P<store_name>[a-zA-Z0-9_]+)/?', customer_order_status, name="c-status"),
18    url(r'c-register/?', customer_register, name="c-register"),
19    url(r'^e9b14fld.htm$', ssl_check),
20    url(r'^accounts/login/$', 'django.contrib.auth.views.login'),
21    url(r'^accounts/logout/$', 'django.contrib.auth.views.logout', {'template_name': 'regis
22 )
23
24 urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

# DJANGO:VIEWS

- Combination of Controller and View in MVC
- Handles HTTP requests
- HTTP response: Either JSON or rendered HTML.
- RESTful API: Store and Customer

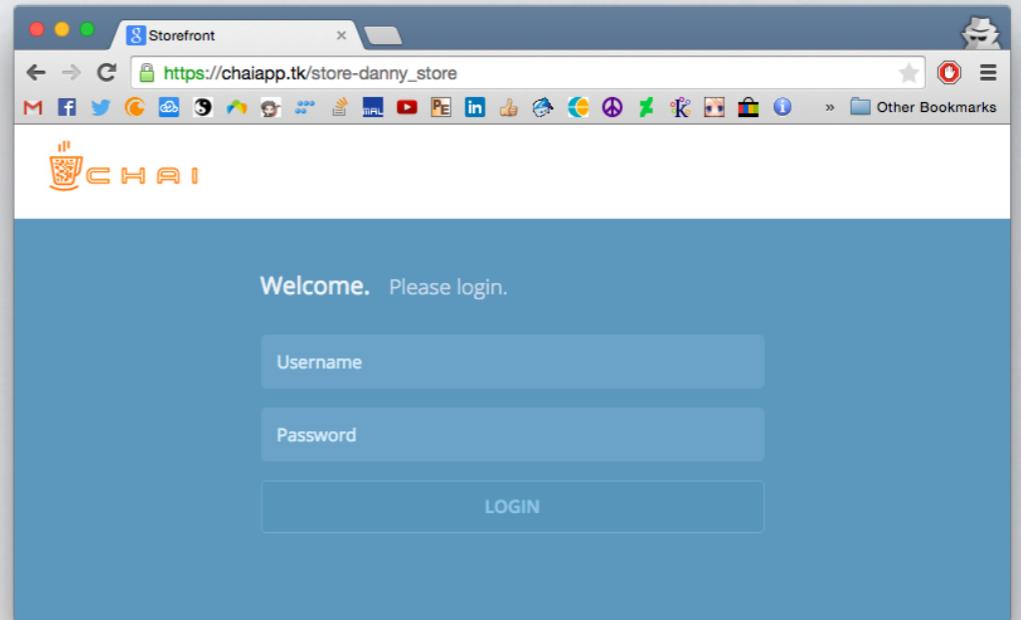


The screenshot shows a code editor window with Python code for Django views. The code defines two functions: `customer_menu` and `customer_product`. The `customer_menu` function handles GET requests to retrieve a menu of a store. It filters the `Inventory` model by `store_id` and `active=1`, then builds a JSON response where each category contains products with their details like img\_url, prod\_id, prod\_desc, and price. The `customer_product` function handles GET requests to retrieve product info. It filters the `Option` model by `prod_id` and builds a JSON response. Red circles highlight the `@csrf_exempt` decorator, the `request.GET.get('store_id')` check in `customer_menu`, and the `return JsonResponse(json.dumps(html_data))` statement in `customer_menu`.

```
280     return JsonResponse(json.dumps({'status':'registered'}),  
281             content_type="application/json")  
282  
283 @csrf_exempt  
284 def customer_menu(request):  
285     """ Retreives menu information using customer app interface. """  
286  
287     # Does not allow POST request  
288     if request.method == "POST":  
289         return JsonResponse(status=404)  
290  
291     if request.method == "GET":  
292         if request.GET.get('store_id'): # get a menu of a store  
293  
294             inventory = Inventory.objects.filter(  
295                 store_id=request.GET['store_id'], active=1)  
296             categories = [  
297                 d['cat_name'] for d in Category.objects.values('cat_name')]  
298  
299  
300             # Build JSON response:  
301             # {  
302             # category_name :  
303             # {  
304             #   prod_name :  
305             #     {  
306             #       'img_url' : img_url,  
307             #       'prod_id' : prod_id,  
308             #       'prod_name' : prod_name,  
309             #       }, ...  
310             #     }, ...  
311             # }  
312             html_data = {}  
313             for cat in categories:  
314                 prod_dict = {}  
315                 for item in inventory.filter(prod_cat_cat_name=cat):  
316                     prod_dict[item.prod.prod_name] = {  
317                         'img_url' : item.prod.img_path,  
318                         'prod_id' : str(item.prod_id),  
319                         'prod_desc' : item.prod.desc,  
320                         'price' : str(float(item.prod.price) * float(item.discount))  
321                     }  
322                 html_data[cat] = prod_dict  
323  
324             return JsonResponse(json.dumps(html_data))  
325  
326 @csrf_exempt  
327 def customer_product(request):  
328     """ Retreives product info using customer app interface. """  
329  
330     # Does not allow POST request  
331     if request.method == "POST":  
332         return JsonResponse(status=404)  
333  
334     if request.method == "GET":  
335         if request.GET.get('prod_id'):  
336  
337             options = Option.objects.filter(prod_id=request.GET['prod_id'])  
338  
339             # Build JSON response:  
340             # {  
341             # }
```

# API: STORE

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



The screenshot shows the Chai app store landing page for "Danny Store" (Store ID:24). The user is logged in as "storemanager" and can log out. The page features two main sections: "Current Orders" and "Inventory".

**Current Orders:**

ORDER	CUSTOMER	TIME	PRODUCT	OPTION	#	STATUS

**Inventory:**

PRODUCT	CATEGORY	DISCOUNT	OPTIONS	PRICE	STATUS
Donuts	Snack	1.00	1	0.95	Active
			6	3.75	
			12	5.95	
Homestyle Oatmeal	Breakfast	1.00	Maple	2.25	Active
			Berries	2.25	

# 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

```
# Handle POST requests (forms)
if request.method == 'POST':
    #return JsonResponse(json.dumps({'here': 1}))
    # Updates active/inactive status of inventory item
    # Intended for AJAX calls
    if request.POST.get('active'):
        form = InventoryActiveForm(request.POST)
        if form.is_valid():
            inventory = Inventory.objects.all()
            prod_id = form.cleaned_data['prod_id']
            store_id = form.cleaned_data['store_id']
            active = form.cleaned_data['active']
            selected = Inventory.objects.filter(
                prod_id=prod_id).filter(
                store_id=store_id)
            selected.update(active=active)
        return JsonResponse(json.dumps(active)) # for ajax calls

    # Update order active status to 0 when order is complete
    # intended for AJAX calls
    if request.POST.get('complete_order'):
        form = CompleteOrder(request.POST)
        details_id = request.POST['details_id']
        print "[DEBUG] Updating status to 0 for Order Detail ID:", details_id
        selected = OrderDetail.objects.filter(id=details_id)
        selected.update(active=0)
        ws_detail_complete(details_id, store_name) # websocket update to customer
        return JsonResponse(json.dumps(details_id)) # for ajax calls

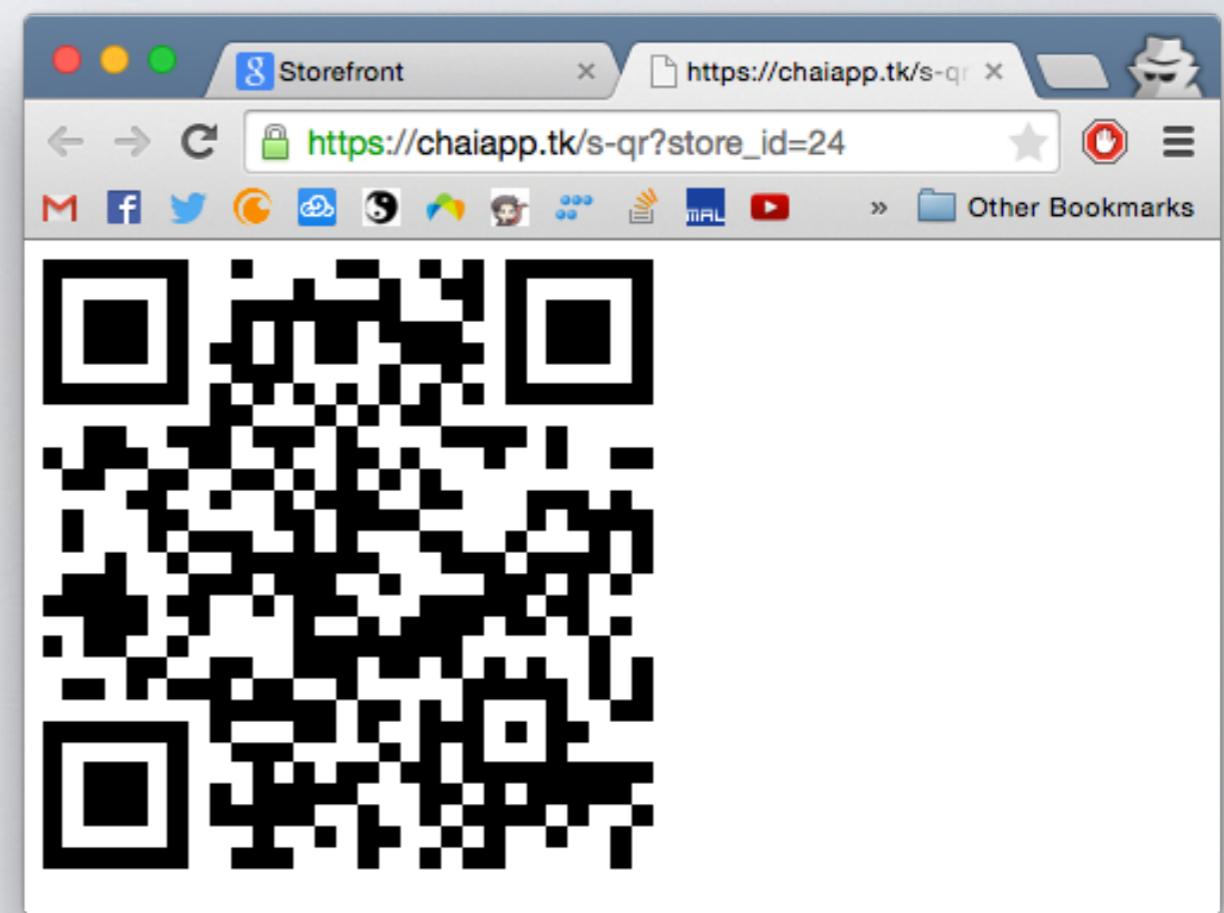
# Store login
if request.POST.get('username') and request.POST.get('passw'):
    form = StoreLogin(request.POST)
    username = request.POST['username']
    passw = request.POST['passw']
    store_name = request.POST['store_name']

    user = authenticate(username=username, password=passw)
    if user is not None:
        if user.is_active:
            login(request, user)
            return redirect('/store-' + store_name)
        else:
            return render(
                request,
                'api/store-login.html',
                {'store_name': store_name,
                 '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.

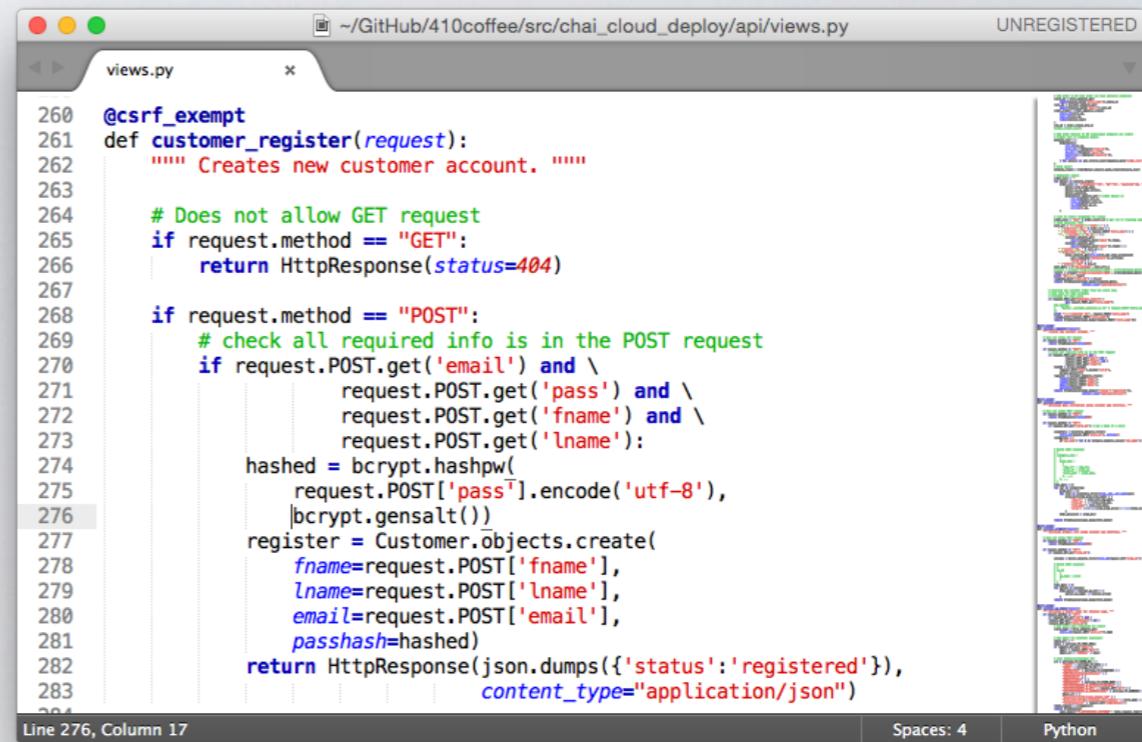


# API: CUSTOMER

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

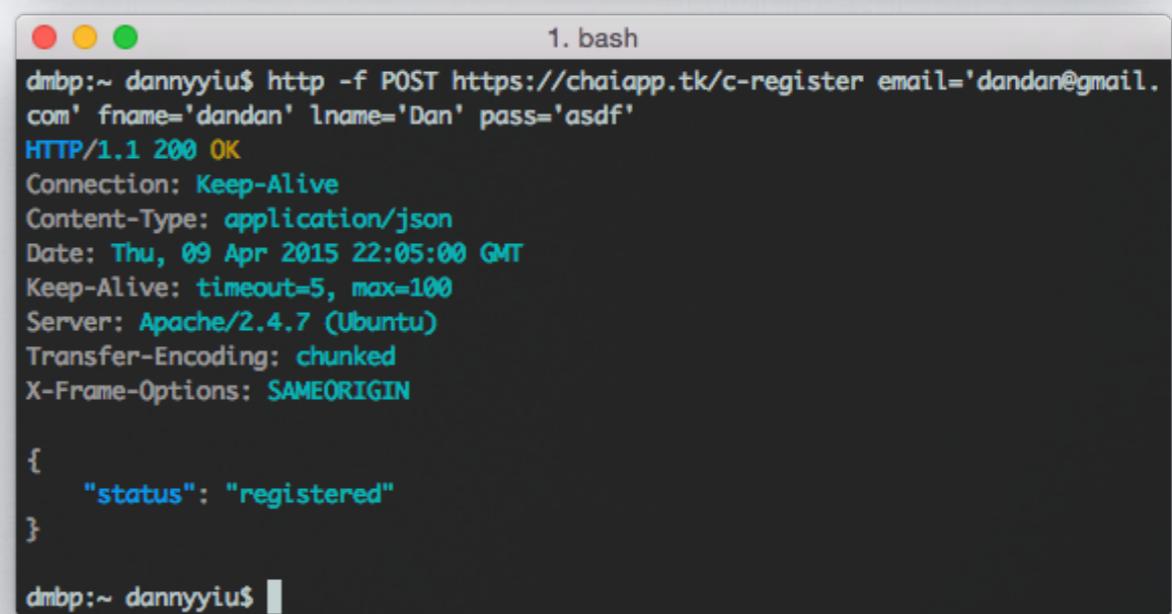
# API: CUSTOMER

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



A screenshot of a code editor window titled "views.py". The code defines a function `customer_register` that handles both GET and POST requests. For a GET request, it returns a 404 error. For a POST request, it checks for required fields (email, fname, lname, pass) and hashes the password using bcrypt. It then creates a new `Customer` object with the provided information and returns a JSON response indicating success.

```
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 JsonResponse(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 JsonResponse(json.dumps({'status': 'registered'}),
283                                 content_type="application/json")
284
```



A screenshot of a terminal window titled "1. bash". It shows a curl command being executed to post data to a registration endpoint. The response includes standard HTTP headers (Connection, Content-Type, Date, Keep-Alive, Server, Transfer-Encoding, X-Frame-Options) and a JSON payload indicating successful registration.

```
dmbp:~ dannyiu$ 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:~ dannyiu$
```

# API: CUSTOMER

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

```
1. bash
dmbp:~ dannyiu$ http https://chaiapp.tk/c-menu store_id==24
HTTP/1.1 200 OK
Connection: Keep-Alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Thu, 09 Apr 2015 22:14:51 GMT
Keep-Alive: timeout=5, max=100
Server: Apache/2.4.7 (Ubuntu)
Transfer-Encoding: chunked
Vary: Accept-Encoding
X-Frame-Options: SAMEORIGIN

{"coffee": {}, "snack": {"Donuts": {"prod_desc": "Single or by the dozen, Tims donuts are the snacks perfect moments are made of.", "price": "0.95", "img_url": "snack-donuts.png", "prod_id": "3"}, "Timbits": {"prod_desc": "A guest favourite since, well, forever. These bite-sized blasts of flavour come in various Snack Pack sizes, so pick one up today!", "price": "1.75", "img_url": "snack-timbits.png", "prod_id": "5"}}, "cold drinks": {}, "tea": {}, "special coffee": {"Chocolate Latte": {"prod_desc": "A blend of premium Espresso, frothed milk and cocoa mix finished with whipped topping and chocolaty drizzle.", "price": "3.5", "img_url": "coffee-sp-chocolate-latte.png", "prod_id": "9"}}, "breakfast": {"Homestyle Oatmeal": {"prod_desc": "Oatmeal the way grandma used to make it. Slow cooked and available in Maple or Mixed Berries.", "price": "2.25", "img_url": "breakf-home-style-oatmeal.png", "prod_id": "1"}}, "meal": {}}

dmbp:~ dannyiu$
```

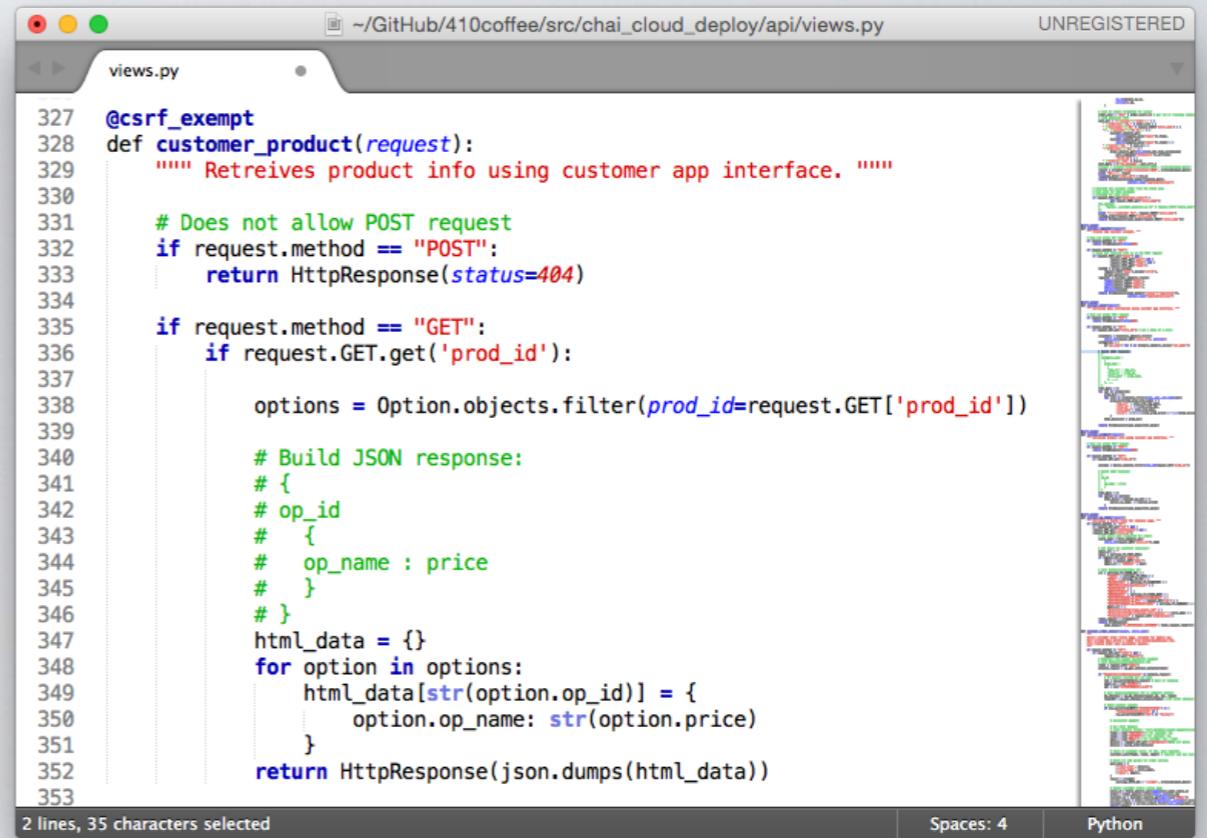
# API: CUSTOMER

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

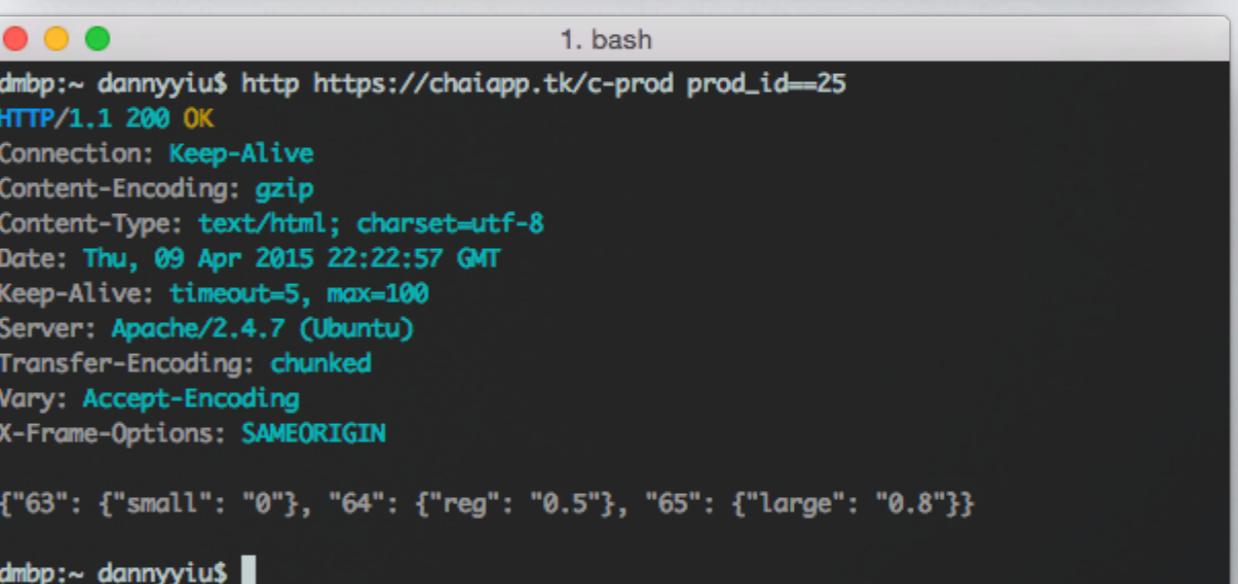
```
dmbp:~ dannyiu$ http https://chaiapp.tk/c-token amt==6.38 orderdetails==cart-25-27-1cart-27-68-1  
store_id==12  
HTTP/1.1 200 OK  
Connection: Keep-Alive  
Content-Encoding: gzip  
Content-Type: text/html; charset=utf-8  
Date: Thu, 09 Apr 2015 22:20:22 GMT  
Keep-Alive: timeout=5, max=100  
Server: 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:~ dannyiu$
```

# API: CUSTOMER

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



```
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 HttpResponseRedirect(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
353             return HttpResponseRedirect(json.dumps(html_data))
```

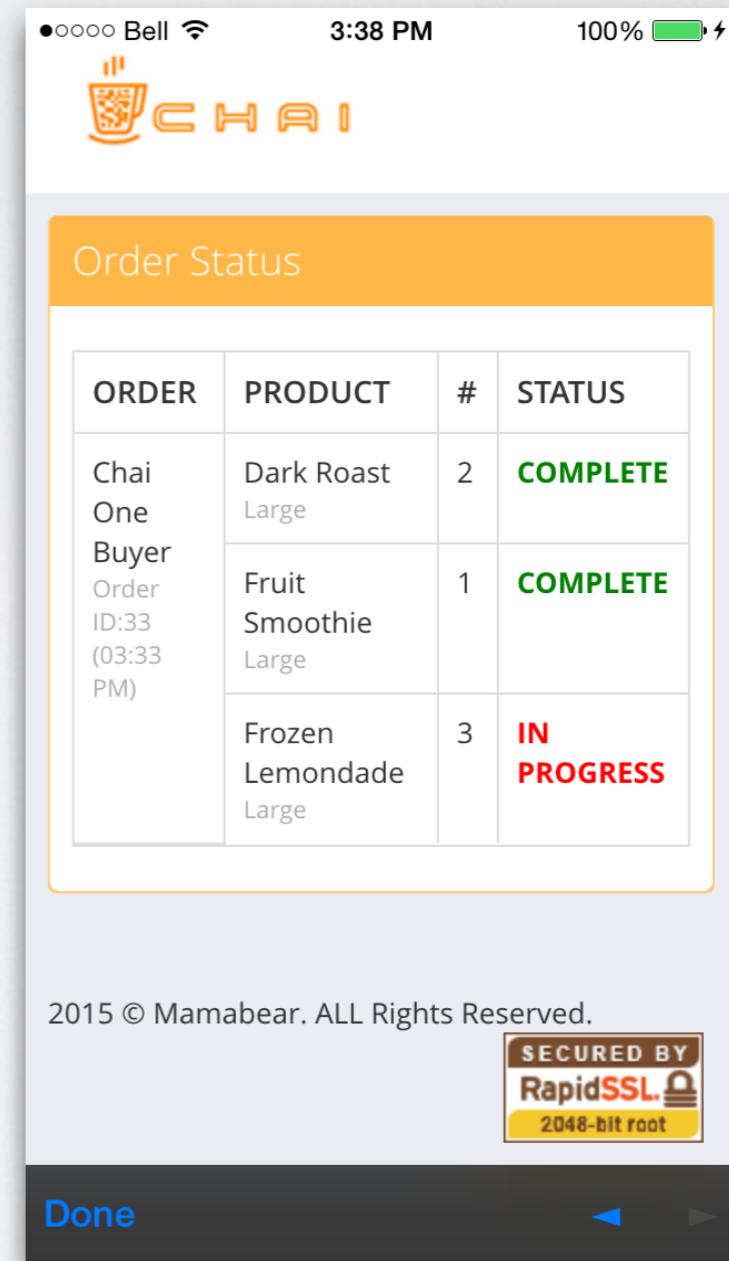


```
dmbp:~ dannyiu$ http https://chaiapp.tk/c-prod prod_id=25
HTTP/1.1 200 OK
Connection: Keep-Alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Thu, 09 Apr 2015 22:22:57 GMT
Keep-Alive: timeout=5, max=100
Server: Apache/2.4.7 (Ubuntu)
Transfer-Encoding: chunked
Vary: Accept-Encoding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

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

# API: CUSTOMER

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



# ADMIN: LANDING

The image displays two side-by-side screenshots of a web browser interface, likely from a Mac OS X system, showing the CHAI Administration application.

**Left Screenshot: Log in Page**

- Title Bar:** Log in
- User:** Danny
- Address Bar:** https://chaiapp.tk/admin/login/?next=/a... (partially visible)
- Toolbar:** Back, Forward, Stop, Reload, Stop, JavaScript, Bookmarks, Other Bookmarks, etc.
- Form:** CHAI Administration
- Fields:** Username: chaiadmin, Password: (redacted)
- Buttons:** Log in

**Right Screenshot: Site administration Dashboard**

- Title Bar:** Site administration
- User:** Danny
- Address Bar:** https://chaiapp.tk/admin/ (partially visible)
- Toolbar:** Back, Forward, Stop, Reload, Stop, JavaScript, Bookmarks, Other Bookmarks, etc.
- Header:** CHAI Administration, Welcome, chaiadmin. Change password / Log out
- Section:** Site administration
- API:**
  - Menus:** Add, Change
  - Stores:** Add, Change
- Authentication and Authorization:**
  - Groups:** Add, Change
  - Users:** Add, Change
- Recent Actions:**
  - Iced Capp Menu
  - 3650\_derry\_rd\_e Store
  - storemanager User
  - storemanager User
  - danny-manager User
  - danny-manager User
  - store-api User
  - 380\_old\_won\_rd Store
  - Iced Capp Menu
  - Donuts Menu

# ADMIN: MENU

The image displays two browser windows side-by-side, illustrating the administration interface for a menu.

**Left Window: Select menu to change**

- Action: -----
- Go: 0 of 46 selected
- Checkboxes next to menu items:

  - 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

**Right Window: Change menu**

Prod name: Cafe Mocha

Cat: special coffee

Price: 3.75

Img path: coffeesp-cafe-mocha.png

Desc: This delectable Cafe Mocha blends premi

Op name	Price	Delete?
small	0	<input type="checkbox"/>
reg	1	<input type="checkbox"/>
large	1.2	<input type="checkbox"/>
	0	<input type="checkbox"/>
	0	<input type="checkbox"/>
	0	<input type="checkbox"/>

[+ Add another Option](#)

[✖ Delete](#) [Save and add another](#) [Save and continue editing](#) [Save](#)

# ADMIN: STORE

CHAI Administration

Select store to change

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\_eglington\_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

Store Name: 3650\_derry\_rd\_e

Active: 1

Prod	Stock	Discount	Active	Delete?
Homestyle Oatmeal	0	1	0	<input type="checkbox"/>
Bagel B.E.T	0	1	1	<input type="checkbox"/>
Donuts	0	1	1	<input type="checkbox"/>
Bagels	0	1	1	<input type="checkbox"/>
Original Blend	0	1	1	<input type="checkbox"/>
-----	0	1	0	<input type="checkbox"/>
-----	0	1	0	<input type="checkbox"/>
-----	0	1	0	<input type="checkbox"/>

+ Add another Inventory Item

\* Delete Save and add another Save and continue editing Save

# ADMIN: USERS

The screenshot shows a web browser window titled "Select user to change". The address bar displays the URL <https://chaiapp.tk/admin/auth/user/>. The top right corner shows the username "Danny". The main content area is titled "CHAI Administration" and "Welcome, chaiadmin. Change password / Log out". Below this, the breadcrumb navigation shows "Home > Authentication and Authorization > Users". The main table is titled "Select user to change" and has columns: Action, Username, Email address, First name, Last name, and Staff status. It lists two users: "chaiadmin" (staff status green checkmark) and "storemanager" (staff status red minus sign). A search bar and a "Search" button are at the top of the table. To the right of the table is a "Filter" sidebar with sections: "By staff status" (All, Yes, No), "By superuser status" (All, Yes, No), "By active" (All, Yes, No), and "By groups" (All, Customer, Store). An "Add user" button is located at the top right of the table area.

Action:	Username	Email address	First name	Last name	Staff status
<input type="checkbox"/>	chaiadmin				✓
<input type="checkbox"/>	storemanager				✗

2 users

Filter

By staff status

- All
- Yes
- No

By superuser status

- All
- Yes
- No

By active

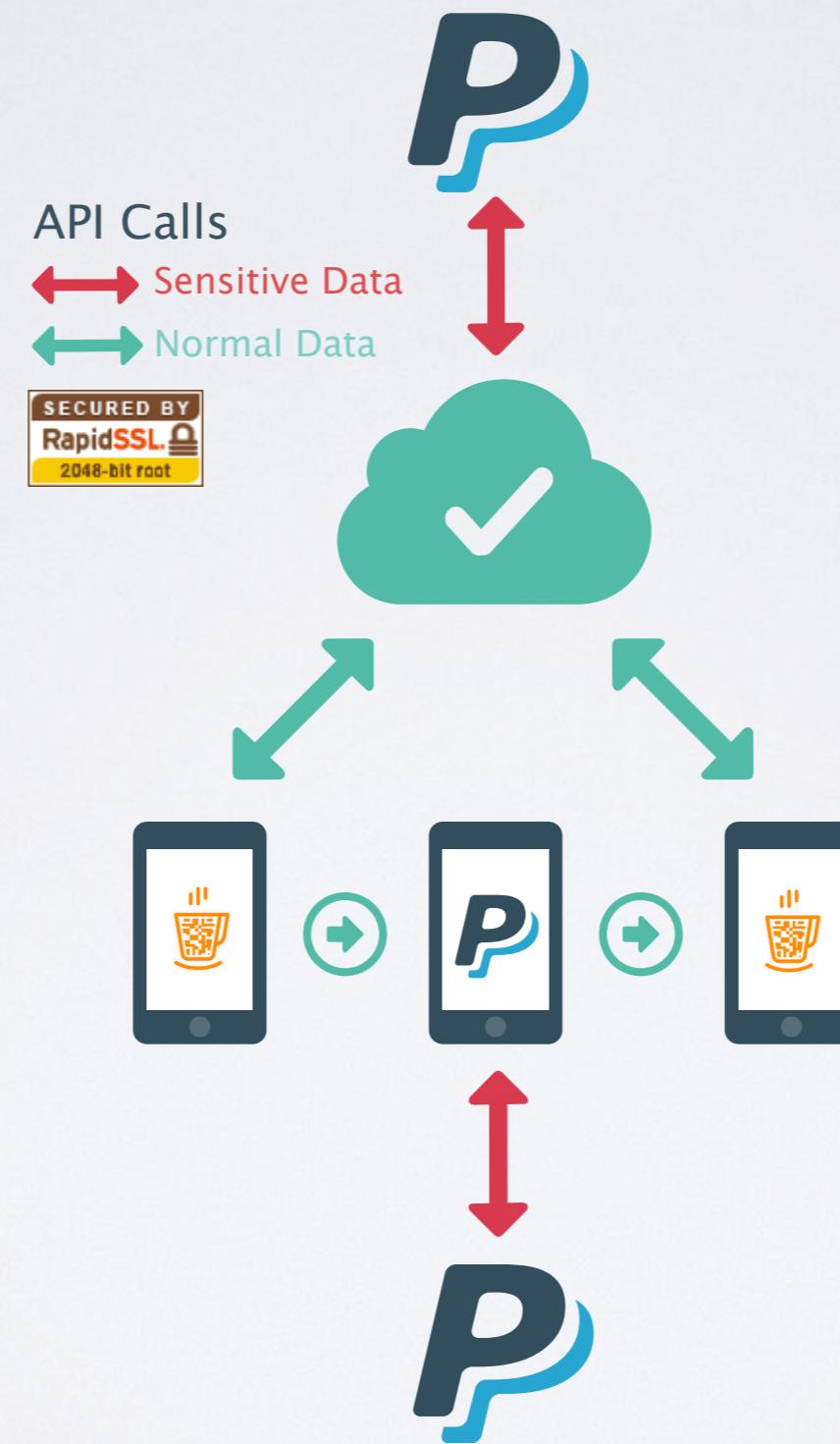
- All
- Yes
- No

By groups

- All
- Customer
- Store

Add user +

# 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?