Development Roadmap: Cloud

Specifications:

* Server: Amazon Linux 32bit, Amazon Web Service (AWS, EC2)
* Backend stack: Apache2, mod-wsgi, Django 1.7, MySQL (sqlite3 for development)

Roadmap Overview:

1. Functional analysis

2. Database design

3. API design and implementation

4. Security

5. Testing

6. Template packaging

# 1. Functional breakdown

## 1.1 Specific requirements

**Overall vision**

The cloud must serve three groups of users: store operators, franchise managers, and customers. While implementation can be flexible, several goals must be strictly kept to avoid falling short of our vision:

1. Highly robust system for heavy load peak times: performance is of utmost priority. As such, development style should be simple and static where possible.
2. Sensitive information such as payment information must be handled accord to PCI compliance.
3. User permissions must be thoroughly tested to prevent unauthorized API access.
4. User interface design can be kept at a minimum; only store operators and managers require interfacing with the cloud directly.

**Store interface**

1. Read and edit (and maybe delete) order status.
2. Read, edit, add, delete inventory
3. Read, edit, add QR codes for store

**Franchise management interface**

1. Read, edit, add, delete stores.
2. Read statistics

**Customer interface**

1. Read, edit, add, delete customer information
2. Read menu and QR codes
3. Transaction on their own account (add order done by cloud after transaction, not by customer directly)
4. Read order status

# 2. Database design

MySQL chosen for deployment due to its functional simplicity and scalability at high volume writes (as compared to PostgreSQL and sqlite). File-based sqlite3 will be used for development stage.

**Store DB functionality:**

1. Inventory (store id, products, options, price, stock, QR id)
2. Orders (order no., customer id/name, product, options, order time, order group)

**Customer DB functionality:**

1. Customer information (name, email, password, credit card(optional), in-store status)
2. Payment information (customer id/name, credit card no, expiry date)

**Franchise DB functionality:**

1. Stores information (store id, location, active, )
2. Marketing and management statistics
3. Store all QR codes in use (QR id, store id, URL)

# 3. API design and implementation

## 3.1 General specifications

* Restful API, subset callable based on user permissions
* REST framework used: django-rest-framework, django-filters
* Hashing libraries for sensitive information: Django PBKDF2, Bcrypt
* SSL used for all API interaction.

## 3.2 API design

**URL structure**

|  |  |  |
| --- | --- | --- |
| URL | Description | Permissions (\* = logged in user) |
| /s/ | Store root | Retrieve/list: STORE |
| /s/inv | Store inventory | Retrieve/list: ALL  Create: STORE\*  Update: STORE\*  Destroy: STORE\* |
| /s/order | Store orders | Retrieve/list: ALL  Create: N/A (handled by cloud)  Update: STORE\*  Destroy: N/A |
| /f/qr | QR code functions | Retrieve/list: ALL  Create: STORE, FRANCHISE  Update: STORE, FRANCHISE  Destroy: STORE, FRANCHISE |
| /f/ | Franchise root | Retrieve/list: FRANCHISE |
| /f/store | Add/Modify stores | Retrieve/list: FRANCHISE  Create: FRANCHISE  Update: FRANCHISE  Delete: FRANCHISE |
| /c/info | Customer info | Retrieve/list: CUSTOMER\*, STORE, FRANCHISE  Create: CUSTOMER\*  Update: CUSTOMER\*  Destroy: CUSTOMER\* |
| /c/buy | Transaction | Create: CUSTOMER\* |

# 4. Security

# 5. Testing

# 6. Template packaging