The UP Diliman E-Kiosk Project: Documentation

Cho, Brian

Sestoso, Mc Edrian

LIS 161 – PEREZ

I. Introduction

The E-Kiosk project is a website that offers an online food service on the Internet. It is meant to simulate the physical food service experience brought by the kiosks around the UP Diliman Campus.

II. Programming Language

The website is made in the Flask web framework with the help of the Jinja template engine and Werkzeug WSGI toolkit. It runs on both Python programming (major component) and HTML language (minor component).

III. Database Functions

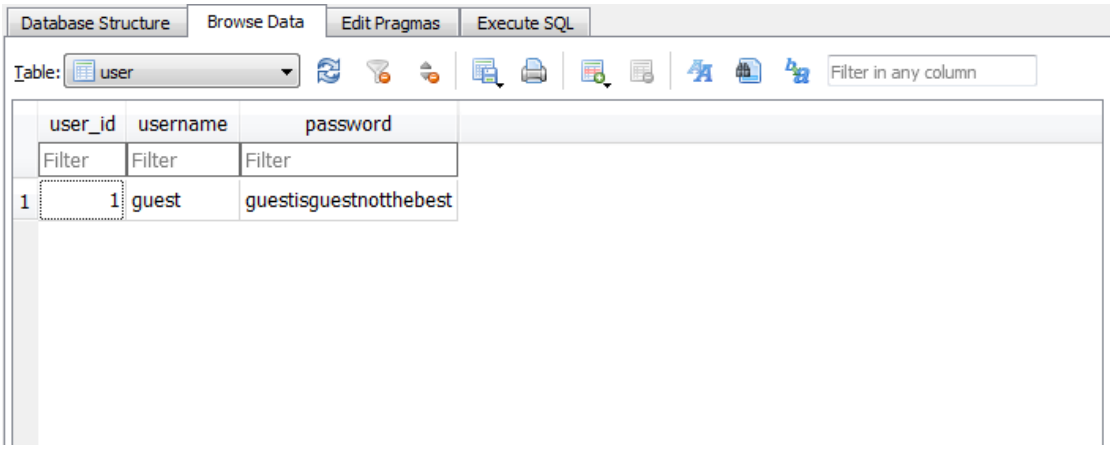
The website utilizes the SQLite database management system to handle user data storage, and access other important data used in the website. The website uses 'data.py', to connect to and access the database.
Inside ('data.py'):

```
import sqlite3

db_path = 'pb.db'

def connect_db(path):
    conn = sqlite3.connect(path)
    #converts tuples to dictionary
    conn.row_factory = sqlite3.Row
    return (conn, conn.cursor())
```

connect_db(path) connects to the database and converts the tuple into dictionary.

```
def select_dict_user():
    conn, cur = connect_db(db_path)
    query = 'SELECT * FROM user'
    results = cur.execute(query,).fetchall()
    return results
```

select_dict_user() grabs everything in the user table. The username column is unique.




```
def read_user_by_id(user_id):
    conn, cur = connect_db(db_path)
    query = 'SELECT * FROM user WHERE user_id=?'
    results = cur.execute(query, (user_id,)).fetchone()
    conn.close()
    return results
```

read_user_by_id(user_id) uses user_id from the user table to return specific values.

```
def insert_user_into_user(login_data):
    conn, cur = connect_db(db_path)
    query = 'INSERT INTO user (username,password) VALUES
(?,?)'
    values = (login_data['user'],
              login_data['password'],)
    cur.execute(query, values)
    conn.commit()
    conn.close()
```

insert_user_into_user(login_data) takes the user login and password input, which is defined inside the login_data dictionary, given by the user and puts them inside the user table.

```
def update_user_in_cart(user_data):
    conn, cur = connect_db(db_path)
    query = 'UPDATE cart SET user_id=?, user=? WHERE
cart_id=?'
    values = (
        user_data['user_id'],
        user_data['user'],
        user_data['cart_id'],
    )
    cur.execute(query,values)
    conn.commit()
    conn.close()
```

update_user_in_cart(user_data) takes the login data of the user (username and user_id), defined under the user_data dictionary, and changes the current user's name and id that is stored along with the cart.

```
def delete_user_from_user(username):
    conn, cur = connect_db(db_path)
    query = 'DELETE FROM user WHERE username=?'
    cur.execute(query, (username,))
    conn.commit()
    conn.close()
```

delete_user_from_user(username) deletes the user from the user table by using their username as key.

```
def select_dict_meals():
    conn, cur = connect_db(db_path)
    query = 'SELECT * FROM meals'
    results = cur.execute(query,).fetchall()
    return results
```

select_dict_meals() grabs everything in the meals table.

Table: meals

	meals_id	meals_type	url
	Filter	Filter	Filter
1	3	combomeals	img/combomeals.png
2	4	alacarte	img/alacarte.png
3	5	drinks	img/drinks.png

```
def read_meals_by_type(meals_type):
    conn, cur = connect_db(db_path)
    query = 'SELECT * FROM meals WHERE meals_type=?'
    results = cur.execute(query, (meals_type,)).fetchone()
    conn.close()
    return results
```

read_meals_by_type(meals_type) uses meals_type from the meals table to return specific values.

```
def select_dict_alacarte_type():
    conn, cur = connect_db(db_path)
    query = 'SELECT * FROM alacarte'
    results = cur.execute(query,).fetchall()
    return results
```

select_dict_alacarte_type() grabs everything in the alacarte table.

Table: alacarte

	alacarte_id	amount	price	alacarte_type	url
	Filter	Filter	Filter	Filter	Filter
1	1	0	20	Pancit Canton	img/pancitcanton.png
2	2	0	5	Kwek-kwek	img/index2.png
3	3	0	3	Kikiam	img/kikiam.png
4	4	0	2	Fish Ball	img/fishball.png
5	5	0	2	Squid Ball	img/squidball.png
6	6	0	2	Chicken Ball	img/chickenball.png

```
def read_alacarte_by_id(alacarte_id):
    conn, cur = connect_db(db_path)
    query = 'SELECT * FROM alacarte WHERE alacarte_id=?'
    results = cur.execute(query, (alacarte_id,)).fetchone()
    conn.close()
    return results
```

read_alacarte_by_id(alacarte_id) uses alacarte_id from the alacarte table to return specific values.

```
def select_dict_drinks_type():
    conn, cur = connect_db(db_path)
    query = 'SELECT * FROM drinks'
    results = cur.execute(query,).fetchall()
    return results
```

select_dict_drinks_type() grabs everything in the drinks table.

Table: drinks

	drinks_id	amount	price	drinks_type	url
	Filter	Filter	Filter	Filter	Filter
1	1	0	10	Water	img/water.png
2	2	0	20	Buko Juice	img/bukojuice.png

```
def read_drinks_by_id(drinks_id):
    conn, cur = connect_db(db_path)
    query = 'SELECT * FROM drinks WHERE drinks_id=?'
    results = cur.execute(query, (drinks_id,)).fetchone()
    conn.close()
    return results
```

read_drinks_by_id(drinks_id) uses drinks_id from the drinks table to return specific values.

```
def select_dict_flavors():
    conn, cur = connect_db(db_path)
```

```
    query = 'SELECT * FROM flavors'
    results = cur.execute(query, ()).fetchall()
    conn.close()
    return results
```

select_dict_flavors () grabs everything in the flavors table.

Table: flavors

	flavors_id	flavors_type	url
	Filter	Filter	Filter
1	1	classic	img/classic.png
2	2	calamansi	img/calamansi.png
3	3	chili-mansi	img/chilimansi.png
4	4	sweet & spicy	img/pcsweet&spicy.png
5	5	hot-chili	img/hotchili.png

```
def select_dict_sauces():
    conn, cur = connect_db(db_path)
    query = 'SELECT * FROM sauces'
    results = cur.execute(query, ()).fetchall()
    conn.close()
    return results
```

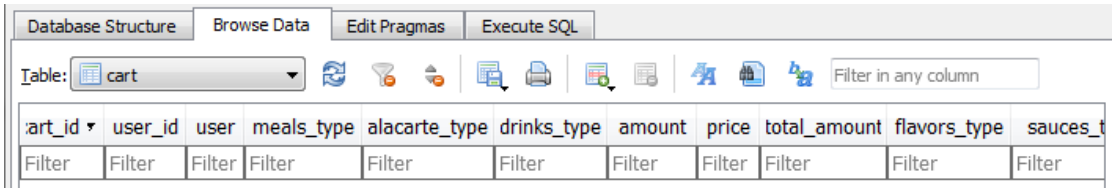
select_dict_sauces () grabs everything in the sauces table.

Table: sauces

	sauces_id	sauces_type	url
	Filter	Filter	Filter
1	1	plain	img/plain.png
2	2	sweet	img/sweet.png
3	3	sweet & spicy	img/sweet&spicy.png
4	4	vinegar dip	img/vinegardip.png
5	5	sweet & spicy and vinegar dip	img/sweet&spicyvinegardip.png

```
def select_dict_cart():
    conn, cur = connect_db(db_path)
    query = 'SELECT * FROM cart'
    results = cur.execute(query, ()).fetchall()
    conn.close()
    return results
```

select_dict_cart () grabs everything in the cart table.




```
def read_cart_by_id(cart_id):
    conn, cur = connect_db(db_path)
    query = 'SELECT * FROM cart WHERE cart_id=?'
    results = cur.execute(query, (cart_id,)).fetchone()
    conn.close()
    return results
```

read_cart_by_id(cart_id) uses cart_id from the cart table to return specific values.

```
def insert_cart_into_cart(cart_data):
    conn, cur = connect_db(db_path)
    query = 'INSERT INTO cart (user_id, user, meals_type,
alacarte_type, ' \
            'drinks_type, amount, price, flavors_type,
sauces_type, total_amount) ' \
            'VALUES (?,?,?,?,?,?,?,?,?,?)'
    values = (
              cart_data['user_id'],
              cart_data['user'],
              cart_data['meals_type'],
              cart_data['alacarte_type'],
              cart_data['drinks_type'],
              cart_data['amount'],
              cart_data['price'],
              cart_data['flavors_type'],
              cart_data['sauces_type'],
              cart_data['total_amount'],
    )
    cur.execute(query, values)
    conn.commit()
    conn.close()
```

insert_cart_into_cart(cart_data) takes the user data, which is defined inside the cart_data dictionary, given by the user and puts them inside the cart table.

```
def delete_cart_from_cart(cart_id):
    conn, cur = connect_db(db_path)
    query = 'DELETE FROM cart WHERE cart_id=?'
    cur.execute(query, (cart_id,))
    conn.commit()
    conn.close()
```

delete_cart_from_cart(cart_id) deletes a row from the cart table by using their cart_id as key.

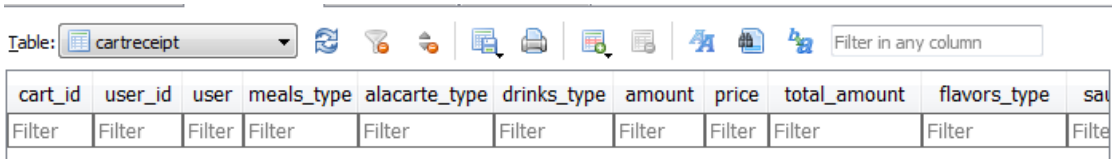
```
def update_cart(cart_data):
    conn, cur = connect_db(db_path)
```

```
    query ='UPDATE cart SET amount=?, flavors_type=?,
sauces_type=?, total_amount=? WHERE cart_id=?'
    values = (
            cart_data['amount'],
            cart_data['flavors_type'],
            cart_data['sauces_type'],
            cart_data['total_amount'],
            cart_data['cart_id'],
    )
    cur.execute(query,values)
    conn.commit()
    conn.close()
```

update_cart(cart_data) takes the data of the user, defined under the cart_data dictionary, and changes the current cart's data by using cart_id to specify the row needed to change.

```
def select_dict_cartreceipt():
    conn, cur = connect_db(db_path)
    query = 'SELECT * FROM cartreceipt'
    results = cur.execute(query, ()).fetchall()
    conn.close()
    return results
```

select_dict_cartreceipt() grabs everything in the cartreceipt table. The cartreceipt table is a duplicate of the cart table.




```
def insert_cart_into_cartreceipt(cart_data):
    conn, cur = connect_db(db_path)
    query = 'INSERT INTO cartreceipt (cart_id, user_id,
user, meals_type, alacarte_type, ' \
            'drinks_type, amount, price, flavors_type,
sauces_type, total_amount) ' \
            'VALUES (?,?,?,?,?,?,?,?,?,?,?)'
    values = (
              cart_data['cart_id'],
              cart_data['user_id'],
              cart_data['user'],
              cart_data['meals_type'],
              cart_data['alacarte_type'],
              cart_data['drinks_type'],
              cart_data['amount'],
              cart_data['price'],
              cart_data['flavors_type'],
              cart_data['sauces_type'],
              cart_data['total_amount'],
    )
```

```
    cur.execute(query, values)
    conn.commit()
    conn.close()
```

insert_cart_into_cartreceipt(cart_data) takes the user data, which is defined inside the cart_data dictionary, given by the user and puts them inside the cartreceipt table.

```
def insert_final_into_finalcart(finalcart_data):
    conn, cur = connect_db(db_path)
    query = 'INSERT INTO finalcart (finalcart_id,
order_type, order_location, cart_id, user_id, user, ' \
            'total_price, payment_amount, change,
payment_method) ' \
            'VALUES (?,?,?,?,?,?,?,?,?,?)'
    values = (
            finalcart_data['finalcart_id'],
            finalcart_data['order_type'],
            finalcart_data['order_location'],
            finalcart_data['cart_id'],
            finalcart_data['user_id'],
            finalcart_data['user'],
            finalcart_data['total_price'],
            finalcart_data['payment_amount'],
            finalcart_data['change'],
            finalcart_data['payment_method'],
    )
    cur.execute(query, values)
    conn.commit()
    conn.close()
```

insert_final_into_finalcart(finalcart_data) takes the user data, which is defined inside the finalcart_data dictionary, given by the user and puts them inside the finalcart table.

```
def read_finalcart_by_id(finalcart_id):
    conn, cur = connect_db(db_path)
    query = 'SELECT * FROM finalcart WHERE finalcart_id=?'
    results = cur.execute(query,
(finalcart_id,)).fetchone()
    conn.close()
    return results
```

read_finalcart_by_id(finalcart_id) uses finalcart_id from the finalcart table to return specific values.

```
def select_dict_finalcart():
    conn, cur = connect_db(db_path)
    query = 'SELECT * FROM finalcart'
    results = cur.execute(query, ()).fetchall()
    conn.close()
    return results
```

select_dict_finalcart() grabs everything in the finalcart table.

Table: finalcart

Filter in any column

finalcart_id	finalcart_no	order_type	order_location	user_id	user	cart_id	total_price	payment_amount	cha
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filte

IV. Website Functions

i. Navigation Functions

1. Navigation Bar

A navigation bar (‘nav.html’) is utilized in the website to make it easier for the user to access various parts of the website. The different sections in the nav bar lead to different parts of the website.

```
<ul class="nav navbar-nav" style="margin-left:25%;">
  <li>a href="{{ url_for('index') }}"
class="navcustom">Home</a></li>
  <li><a href="{{ url_for('basemenu') }}" class="link
navcustom">Menu</a></li>
  <li><a href="{{ url_for('kiosk') }}"
class="navcustom">Kiosks</a></li>
  <li><a href="{{ url_for('cart') }}"
class="navcustom">Cart</a></li>
  <li><a href="{{ url_for('about') }}"
class="navcustom">About</a></li>
</ul>
```

Home (‘index.html’) leads to the frontpage of the website. Menu (‘basemenu.html’) leads to the entire menu catalog featured in the website. Kiosks (‘order.html’) leads to the featured kiosks webpage. Cart (‘cart.html’) leads to the webpage that shows that items in the user’s cart. And About (‘about.html’) leads to the webpage giving brief information about the website.

Register (‘register.html’) leads to the account registration webpage. And Login (‘login.html’) leads to the account login webpage. When a user is logged in, Register

becomes Profile (‘profile.html’) and leads to the currently logged in user’s profile. Furthermore, the Login becomes a Logout when a user is also logged in the website.

```
{% if 'user' in session %}
  <li><a href="{{ url_for('profile',
username=session['user']) }}"><span
class="glyphicon"></span> Profile </a></li>
  <li><a href="{{ url_for('logout') }}"><span
class="glyphicon glyphicon-log-in"></span>
Logout</a></li>
{% else %}
  <li><a href="{{ url_for('register') }}"><span
class="glyphicon"></span> Register</a></li>
  <li><a href="{{ url_for('login') }}"><span
class="glyphicon glyphicon-log-in"></span> Login</a></li>
{% endif %}
```

2. Menu Side Bar

Inside Menu(‘basemenu.html’), a sidebar (‘menusidebar.html’), is utilized to make navigation in-between each meal type easier. On the Menu leads to Menu (‘menusidebar.html’), Combo meals lead to Combo meals, A la Carte leads to A la Carte and Drinks lead to Drinks.

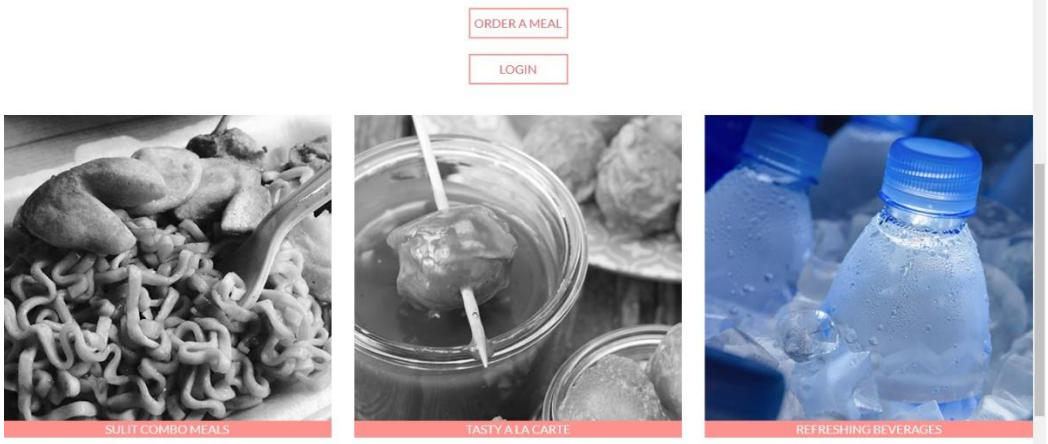
```
<ul class="nav">
    <li class=""><a href="/menu">On The Menu</a></li>
                <li class="colordivider"></li>
```

```
    <li class=""><a href="/menu/combomeals">Combo
Meals</a></li>
                <li class="colordivider"></li>
    <li class=""><a href="/menu/alacarte">À la
Carte</a></li>
                <li class="colordivider"></li>
    <li class=""><a href="/menu/drinks">Drinks</a></li>
                <li class="colordivider"></li>
</ul>
```

ii. Meal ordering functions

1. Home (index()), (‘index.html’)

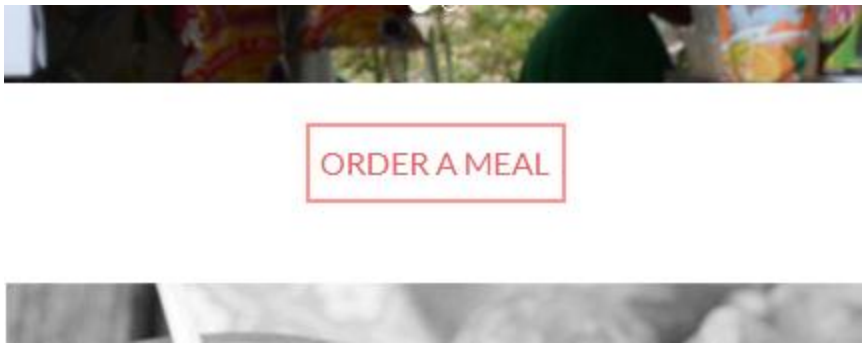
```
@app.route('/')
def index():
    return render_template('index.html')
```



Home is the homepage of the website and has several buttons and images that lead to other parts of the website. The first button ‘Order a meal’ (‘basemenu.html’), directs the user to the menu page.

```
<form action="{{ url_for('basemenu') }}">
<div class="container text-center">
  <br><h11><input class="custom-btn btn-11" type="submit"
value="ORDER A MEAL" name="action"></a></h11>
</div>
</form>
```

The second button ‘Login’ (‘login.html’), directs the user to the account login page. When a user is already logged in to the website, the login button disappears.




```
{% if 'user' in session %}
{% else %}
    <form action="{{ url_for('login') }}">
        <div class="container text-center col">
          <br><h11><input class="custom-btn btn-11"
type="submit" value="LOGIN" name="action"></a></h11>
        </div>
    </form>
{% endif %}
```

The images below direct the user to the different meal types offered by the website that they can order. From left to right, the first image directs the user to 'Combo Meals',

```
<form action="/menu/combomeals">
<button class="btn-img" style="width:100%">
<figure><img src="static/img/index1.png" /></figure>
</button>
</form>
<div class="text">SULIT COMBO MEALS</div>
```

the second image directs the user to 'A la Carte' ('alacartemeals.html'),

```
<form action="/menu/alacarte">
<button class="btn-img" style="width:100%">
<figure><img src="static/img/index2.png"></figure>
</button>
</form>
<div class="text">TASTY A LA CARTE</div>
```

and the third image directs the user to 'Drinks' ('drinks.html').

```
<form action="/menu/drinks">
<button class="btn-img" style="width:100%">
<figure><img src="static/img/index3.png"></figure>
</button>
</form>
<div class="text">REFRESHING BEVERAGES</div>
```

2. Menu (basemenu()), ('basemenu.html') and (menu(meals_type)), ('menu.html')

```
@app.route('/menu')
def basemenu():
    meals = select_dict_meals()
    return render_template('basemenu.html',meals=meals)
```

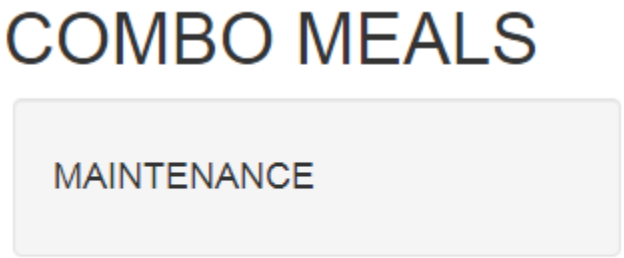
```
@app.route('/menu/<meals_type>/')
def menu(meals_type):
    meals = read_meals_by_type(meals_type,)
    alacartesdict = select_dict_alacarte_type()
    drinksdict = select_dict_drinks_type()
    return
render_template('menu.html',meals_type=meals_type,meals=m
eals, alacartesdict=alacartesdict,drinksdict=drinksdict)
```

Menu is the part of the website that features the items that the website offers to the users. basemenu() accesses the meal table in the database by defining select_dict_meals as meals. On ('basemenu.html'), the three images are links to the meals contained in each meal type. Combo meals meal type lead to the combo meals (not yet implemented and written as maintenance).
Inside ('basemenu.html'):

```
{% if mealtype.meals_type == 'combomeals' %}
<div class="col-md-8" >
<a href="/menu/{{ mealtype.meals_type }}/">
    <button class="btn-11" type="submit" name="menu"
style="width:100%;text-align:left;">
        <img src="{{ url_for('static',
filename=mealtype['url']) }}" style="width:100%;">
    </button>
</a></div>
```

Inside (‘menu.html’):

```
{% if meals['meals_type'] == 'combomeals' %}
    <h1 style="text-indent:10px">
        <input class="hidden" value="{{ meals_type }}"
readonly>COMBO MEALS</h1>
    <a href="/menu/{{ meals_type }}/"></a>
    <div class="col-sm-3">
        <div class="well">
    <p><h4><input class="btn-11" type="submit"
value="MAINTENANCE" name="action"></p></h4>
        </div>
    </div>
```



A la carte meal type leads to the a la carte meals.
Inside (‘basemenu.html’):

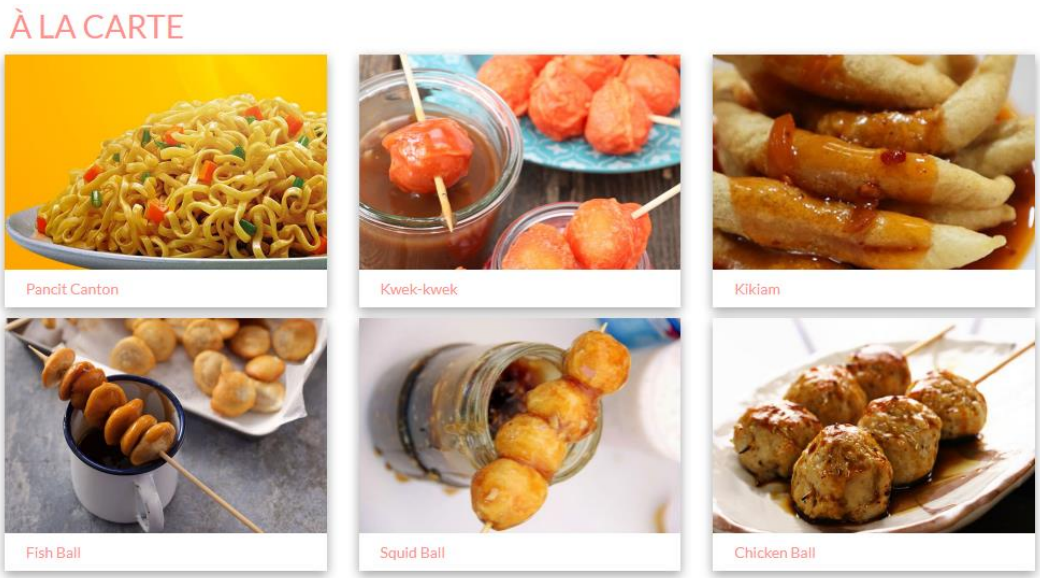
```
{% elif mealtype.meals_type == 'alacarte' %}
<div class="col-md-4">
<a href="/menu/{{ mealtype.meals_type }}/">
    <button class="btn-11" type="submit" name="menu"
style="width:100%;text-align:left;">
```

```
        <img src="{{ url_for('static',
filename=mealtype['url']) }}" style="width:100%;">
    </button>
</a></div>
```

Inside (‘menu.html’):

```
{% elif meals['meals_type'] == 'alacarte' %}
    <h2 style="…">
        <input class="hidden" value="{{ meals_type }}"
readonly>À LA CARTE</h2>
    {% for alacartemeals in alacartesdict %}
        <form action="{{
url_for('alacartes',meals_type='alacarte',alacarte_id=ala
cartemeals.alacarte_id,alacarte_type=alacartemeals.alacar
te_type) }}">
        <div class="col-sm-3">
            <div class="w3-container">
                <div class="w3-card-4"
style="width:100%">
                    <div class="w3-container pnktext">
                    <button class="btn-11" type="submit"
name="alacarte" style="width:100%;text-align:left;">
                        <img src="{{ url_for('static',
filename=alacartemeals.url) }}" style="width:100%;">
                        <p><h5 style="margin-
left:20px;">{{ alacartemeals.alacarte_type }}</h5>
                    </button>
                        </p>
                        </form>
                    </div>
                </div>
```

```
            </div>
        </div>
    {% endfor %}
```



And Drinks meal type leads to the drinks.
Inside (‘basemenu.html’):

```
    {% elif mealtype.meals_type == 'drinks' %}
    <div class="col-md-4">
    <a href="/menu/{{ mealtype.meals_type }}/">
        <button class="btn-11" type="submit" name="menu"
style="width:100%;text-align:left;">
            <img src="{{ url_for('static',
filename=mealtype['url']) }}" style="width:100%;">
        </button>
    </a></div>
    {% endif %}
{% endfor %}
```

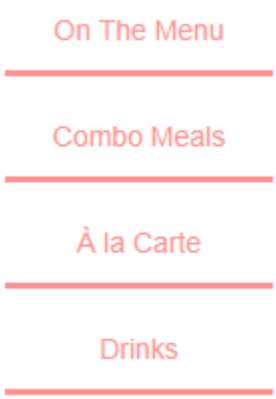
Inside (‘menu.html’):

```
{% elif meals['meals_type'] == 'drinks' %}
    <h2 style="…">
        <input class="hidden" value="{{ meals_type }}"
readonly>DRINKS</h1>
    {% for drinks in drinksdict %}
    <form action="{{
url_for('drinks',meals_type='drinks',drinks_id=drinks.dri
nks_id,drinks_type=drinks.drinks_type) }}">
        <div class="col-sm-3">
            <div class="w3-container">
                <div class="w3-card-4"
style="width:100%">
                    <div class="w3-container w3-left-
align pnktext">
                    <button class="btn-11" type="submit"
name="drinks" style="width:100%;text-align:left;">
                        <img src="{{ url_for('static',
filename=drinks.url) }}" style="width:100%">
                        <p><h5 style="margin-
left:20px;">{{ drinks.drinks_type }}</h5>
                    </button>
                    </p>
                    </form>
                    </div>
                </div>
            </div>
        </div>
    {% endfor %}
{% endif %}
```

Each meal in the meal type leads to their respective meal web pages.

```
@app.route('/menu/<meals_type>/')
def menu(meals_type):
    meals = read_meals_by_type(meals_type,)
    alacartesdict = select_dict_alacarte_type()
    drinksdict = select_dict_drinks_type()
    return
render_template('menu.html',meals_type=meals_type,meals=m
eals,

alacartesdict=alacartesdict,drinksdict=drinksdict)
```

Menu, its meal types and each individual meal/drink inside it has a sidebar (‘menusidebar.html’), that makes it easier to navigate in-between each meal type.

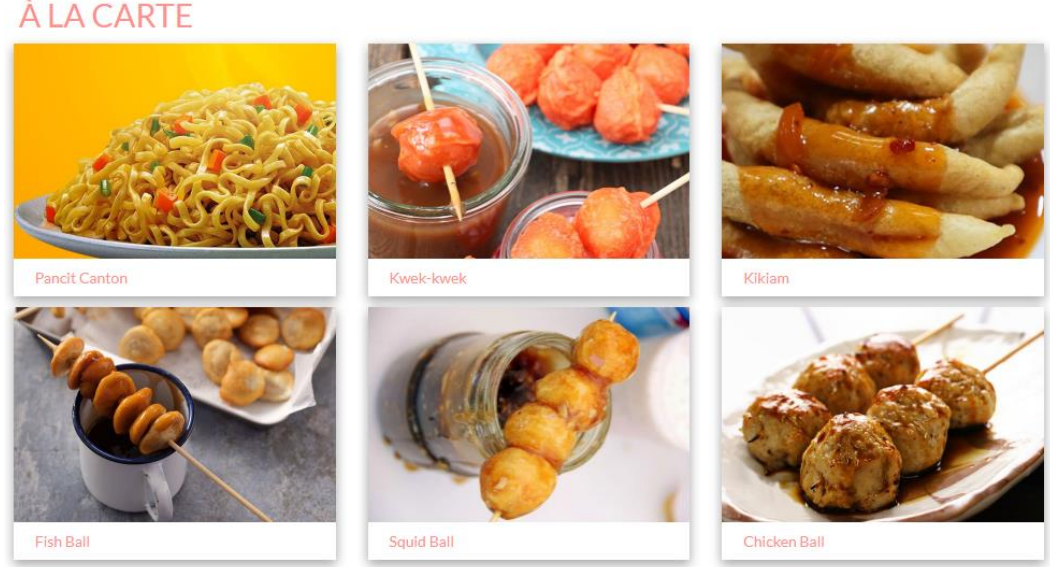


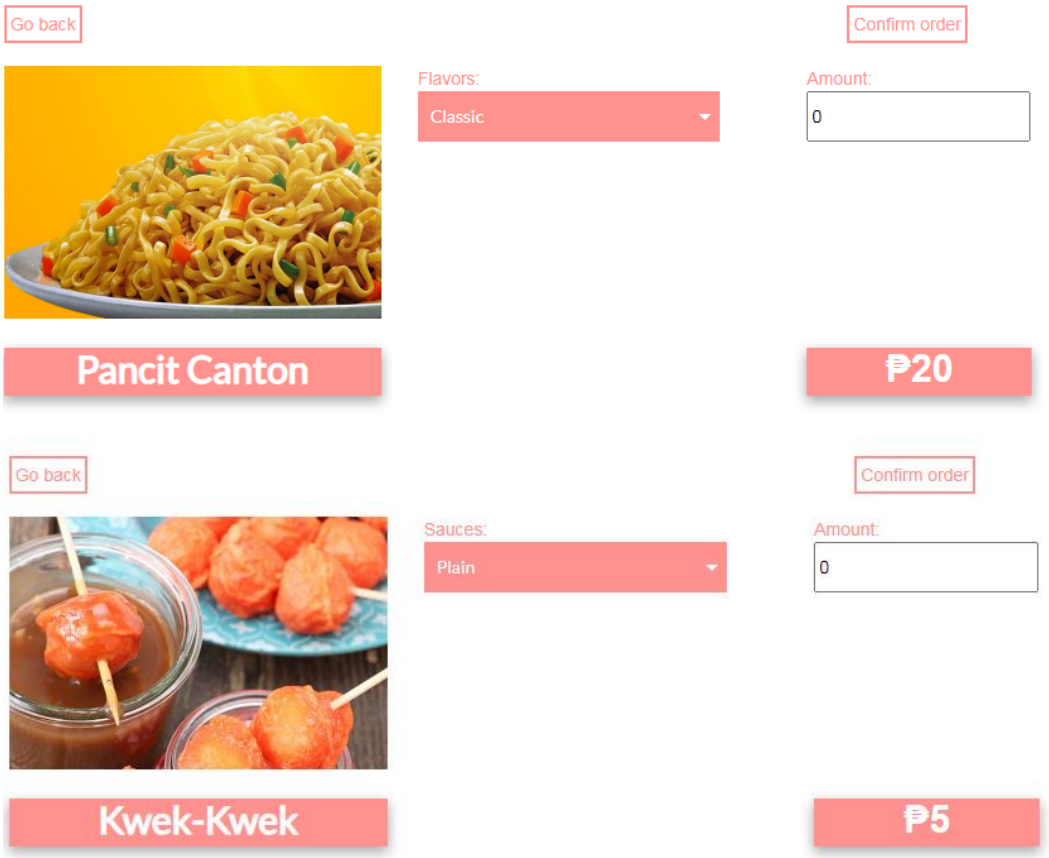
On the Menu leads to Menu (‘menusidebar.html’), Combo meals lead to the Combo meals, A la Carte leads to the A la Carte meals and Drinks lead to the Drinks.

3. A la Carte Meals (alacartes(alacarte_id, alacarte_type)), (‘alacartemeals.html’)

```
@app.route('/menu/alacarte/<int:alacarte_id>/<alacarte_ty
pe>/')
```

```
def alacartes(alacarte_id, alacarte_type):
    alacarteid = read_alacarte_by_id(alacarte_id)
    meals = select_dict_meals()
    flavordict = select_dict_flavors()
    saucedict = select_dict_sauces()
    loginuser = select_dict_user()

    for users in loginuser:
        if 'user' not in session:
            if 'guest' in users['username']:
                username = users['username']
                user_id = users['user_id']
        elif 'user' in session:
            if users['username'] == session['user']:
                users['username']
                username = users['username']
                user_id = users['user_id']

    return
render_template('alacartemeals.html',alacarte_id=alacarte
_id, alacarteid=alacarteid,

alacarte_type=alacarte_type,flavordict=flavordict,saucedi
ct=saucedict,

meals=meals,loginuser=loginuser,username=username,user_id
=user_id)
```





A la Carte meals contain the individual data of each meal under the A la Carte meal type. The user_id and user are changed based on the user inside of session. If the user is not in session, by default the user is guest. Included in the A la Carte meal data are: user_id (hidden), user (hidden),

```
<input class="hidden" id="username" name="username"
value="{{ username }}" readonly>
<input class="hidden" id="user_id" name="user_id"
value="{{ user_id }}" readonly>
```

meals_type (hidden),

```
{% for meal in meals %}
    {% if meal['meals_type'] == 'alacarte' %}
        <input class="hidden" name="meals_type" value="{{
meal['meals_type'] }}" readonly>
    {% endif %}
{% endfor %}
```

alacarte_type, drinks_type (hidden and NULL),

```
<input class="hidden" id="drinks_type" name="drinks_type"
value="NULL" readonly>
```

amount (user input), price (user input), and flavors_type and/or sauces_type (user input). ‘Go back’ takes the user to the previous page which is the meal type the meal is in.

```
<div class="grid-item">
  <form action="/menu/alacarte">
  <input class="custom-btn btn-11" type="submit"
value="Go back" name="action" style="margin-right:80%;">
  </form>
</div>
```

Depending on the A la Carte meal type, a flavors or sauces select element will appear with their corresponding choices. At the same time, either flavor or sauce set each other’s input values (hidden) into ‘NULL’ when they are present.

```
{% if alacarteid.alacarte_type == 'Pancit Canton' %}
     <span style="text-align:left;">Flavors:</span>
       <select name="alacarte_flavor" id="fl_type"
style="position:center;"  required>
       {% for flavors in flavordict %}
           <option value="{{ flavors.flavors_type }}"
name="alacarte_flavor">
               {{ flavors.flavors_type|title }}
           </option>

       {% endfor %}
       </select>
   <input class="hidden" id="alacarte_sauce"
name="alacarte_sauce" value=NULL readonly>
   {% else %}
       <span style="text-
align:left;color:#ff928e;">Sauces:</span>
       <select name="alacarte_sauce" id="scs_type"
style="position:center;" required>
       {% for sauces in saucedict %}
           <option value="{{ sauces.sauces_type }}"
name="alacarte_sauce">
               {{ sauces.sauces_type|title }}
           </option>
       {% endfor %}
       </select>
   <input class="hidden" id="alacarte_flavor"
name="alacarte_flavor" value=NULL readonly>
   {% endif %}
```

The user can input the amount of meal/s that they want to have with amount input. The input only accepts a greater than or equal to one numerical value.

```
<div class="number" style="text-
align:left;color:#ff928e;">
        Amount:
        <input type="number" min="1" value="{{
```

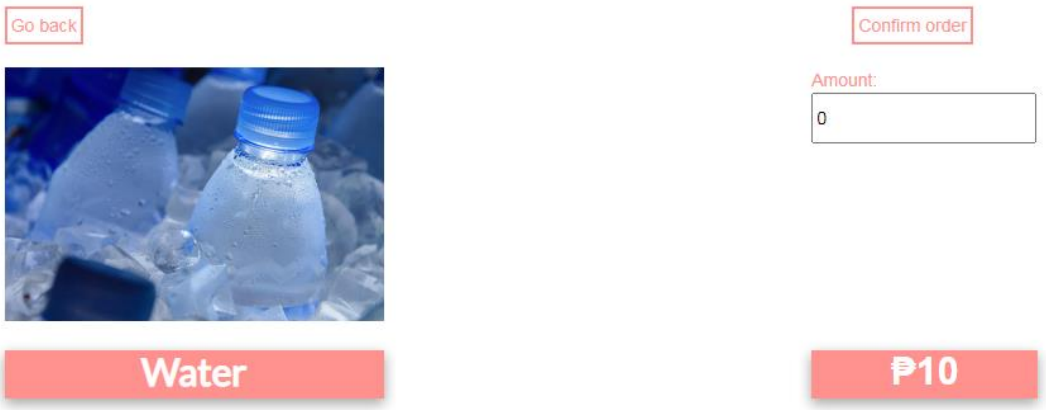
```
alacarteid.amount }}" name="amount"
                style="height:40px;color:black;" required>
</div>
```

'Confirm order' takes all the data in the web page and passes it onto mealprocessing(), which appends the data into the database cart table. Applying mealprocessing() redirects the user to Cart ('cart.html') .

```
<form action="{{ url_for('mealprocessing') }}"
method="post">
    <input class="custom-btn btn-11" type="submit"
value="Confirm order" name="action">
    .
    .
    .
</form>
```

4. Drinks (drinks (drinks_id, drinks_type)), ('drinks.html')

```
@app.route('/menu/drinks/<int:drinks_id>/<drinks_type>/')
def drinks(drinks_id,drinks_type):
    drinksid = read_drinks_by_id(drinks_id)
    meals = select_dict_meals()
    loginuser = select_dict_user()

    for users in loginuser:
        if 'user' not in session:
            if 'guest' in users['username']:
                username = users['username']
                user_id = users['user_id']
        elif 'user' in session:
            if users['username'] == session['user']:
                users['username']
                username = users['username']
                user_id = users['user_id']

    return
render_template('drinks.html',drinks_id=drinks_id,drinks_
type=drinks_type, drinksid=drinksid,

meals=meals,loginuser=loginuser,username=username,user_id
=user_id)
```



Drinks contain the individual data of each drink under the Drinks meal type. The user_id and user are changed based on the user inside of session. If the user is not in session, by default the user is guest. Included in the Drink meal data are: user_id (hidden), user (hidden), meals_type (hidden), alacarte_type (hidden and NULL), flavors_type and sauces_type (hidden and NULL).

```
<input class="hidden" id="username" name="username"
value="{{ username }}" readonly>
<input class="hidden" id="user_id" name="user_id"
value="{{ user_id }}" readonly>
<input class="hidden" id="alacarte_type"
name="alacarte_type" value="NULL" readonly>
<input class="hidden" id="alacarte_sauce"
name="alacarte_sauce" value="NULL" readonly>
<input class="hidden" id="alacarte_flavor"
name="alacarte_flavor" value="NULL" readonly>
  {% for meal in meals %}
    {% if meal['meals_type'] == 'drinks' %}
        <input class="hidden" name="meals_type" value="{{
meal['meals_type'] }}" readonly>
    {% endif %}
{% endfor %}
<input class="hidden" id="drinks_type" name="drinks_type"
value="NULL" readonly>
```

drinks_type, amount (user input), and price (user input).
'Go back' takes the user to the previous page which is the meal type the meal is in.

```
<form action="/menu/drinks">
<input class="custom-btn btn-11" type="submit" value="Go
back" name="action" style="margin-right:80%;">
</form>
```

Drinks have no flavors nor sauces select element and are set to NULL by default.

```
<input class="hidden" id="alacarte_sauce"
name="alacarte_sauce" value="NULL" readonly>
<input class="hidden" id="alacarte_flavor"
name="alacarte_flavor" value="NULL" readonly>
```

The user can input the amount of meal/s that they want to have with amount input. The input only accepts a greater than or equal to one numerical value.

```
<div class="number"style="text-
align:left;color:#ff928e;">
    Amount:
        <input type="number" min="1" value="{{
drinksid.amount }}" name="amount"
               style="height:40px;color:black;" required>
</div>
```

'Confirm order' takes all the data in the web page and passes it onto mealprocessing(), which appends the data into the database cart table.

```
<form action="{{ url_for('mealprocessing') }}"
method="post">
    <input class="custom-btn btn-11" type="submit"
value="Confirm order" name="action">
    .
    .
    .
  </form>
```

5. mealprocessing()

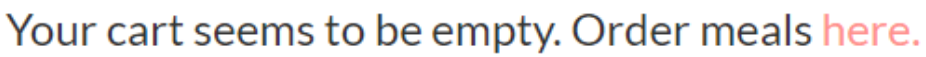
```
@app.route('/mealprocessing/', methods=['POST'])
def mealprocessing():

    totalamount = int(request.form['amount']) *
int(request.form['price'])

    cart_data = {
        'user_id': request.form['user_id'],
        'user' : request.form['username'],
        'meals_type' : request.form['meals_type'],
        'alacarte_type' : request.form['alacarte_type'],
        'drinks_type' : request.form['drinks_type'],
        'amount': int(request.form['amount']),
        'price': int(request.form['price']),
        'flavors_type' : request.form['alacarte_flavor'],
        'sauces_type' : request.form['alacarte_sauce'],
        'total_amount': int(totalamount),
    }

    insert_cart_into_cart(cart_data)

    return redirect(url_for('cart'))
```

mealprocessing() is a function that appends all the gathered data from a specific meal type into the database cart table. From the meal types earlier, some values are set to NULL. This is to avoid key errors and not null errors. total_amount is a data that takes

the variable totalamount which multiples the input amount with the set meal price. Applying mealprocessing() redirects the user to Cart ('cart.html') .

6. Cart (cart()), ('cart.html')

```
@app.route('/cart/',)
def cart():
    usercart = select_dict_cart()
    loginuser = select_dict_user()

    for users in loginuser:
        if 'user' not in session:
            if 'guest' in users['username']:
                username = users['username']
                user_id = users['user_id']
        elif 'user' in session:
            if users['username'] == session['user']:
                users['username']
                username = users['username']
                user_id = users['user_id']

    cartchecker = any(usercart)

    return
render_template('cart.html',loginuser=loginuser,usercart=
usercart, cartchecker=cartchecker,username=username,
user_id=user_id)
```



Cart shows the items added by a user into their cart database. When the user is not logged in, the default user and user_id is set to guest. cartchecker is a variable that returns true or false values. When it is false, the page shows a prompt and a link that leads to Menu ('basemenu.html').

```
{% if cartchecker == True %} …
{% else %}
<div class="w3-container w3-panel ">
```

```
        <div class="w3-container w3-center"
style="padding:220px;font-family:'Lato';">
    <h3> Your cart seems to be empty. Order meals
        <a href="{{ url_for('basemenu') }}"
style="padding:0px 0px;">here.</a></h3>
        </div>
</div>
{% endif %}
```

If cartchecker is true, it shows a table with all the data stored in the user's cart. A for loop for the variable usercart, which selects the cart table in the database, is ran to get its data. The data found in the Cart page are: cart_id (hidden), alacarte_amount (user input), drinks_type or alacarte_type (from the cart database, depending on which meal type is NULL), flavors_type or sauces_type (from the cart database, depending on which meal type is NULL, and also a user input), meal_type (depending on which meal type is NULL), price, and total amount.

```
{% for userdata in usercart %}
    <form action="{{ url_for('edit') }}" method="post">
            .
            .
            .
                <div class="table-data"><input
class="custom-btn btn-11" type="submit" value="EDIT"
name="action">
                </div>
    </form>

                <div class="table-data">
                    <form action ="{{ url_for('delete',
cart_id=userdata.cart_id) }}" method="post">
                        <input class="custom-btn btn-11"
type="submit" value="DELETE" name="action"
                        style="font-
family:'Lato';background-
color:#ff928e;color:white;margin-bottom:5%;">
                    </form>
                </div>
                <div class="table-data">
                </div>
            </div>
      </div>
{% endfor %}
```

Each row in the cart has subsequent 'Edit' and 'Delete' buttons. The user can change the amount of meal/s that they want. And depending on the meal type, can also change its subsequent flavor or sauce type. The user must press the edit button to commit these changes. The edit button leads to edit(). The delete button leads to delete(cart_id). Below the table are the 'Add more' and 'Check out' buttons. Add more goes back to Menu ('basemenu.html').

```
<form action="/menu">
    <input class="custom-btn btn-11" type="submit"
value="Add more" name="action"
    style="margin-left:37%;width:25%;">
</form>
```

Check out leads to checkout (‘checkout.html’).

```
<form action="{{ url_for('checkout', user_id=user_id,
users=username) }}" method="post">
    <input class="custom-btn btn-11" type="submit"
name="CHECK OUT" value="Check out"
   style="margin-right:37%;width:25%;">
</form>
```

7. edit()

```
@app.route('/cart/edit', methods=['post'])
def edit():
    usercart = select_dict_cart()

    alacarte_flavor = request.form['alacarte_flavor']
    alacarte_sauce = request.form['alacarte_sauce']
    alacarte_amount = request.form['alacarte_amount']
    cart_id = request.form['cart_id']
    price = request.form['price']

    totalamount = int(request.form['alacarte_amount']) *
int(request.form['price'])

    cart_data = {
        'amount': alacarte_amount,
        'flavors_type' : alacarte_flavor,
        'sauces_type' : alacarte_sauce,
        'cart_id': cart_id,
        'price' : price,
        'total_amount' : totalamount,
    }
    update_cart(cart_data)

    return redirect(url_for('cart', cart_id=cart_id))
```

Edit changes the values of some data in specific rows in the cart database. It uses the values inside the cart_data dictionary as replacement for the values in the cart table in the database. When edit() takes place, the user is then redirected back to the cart page.

8. delete(cart_id)

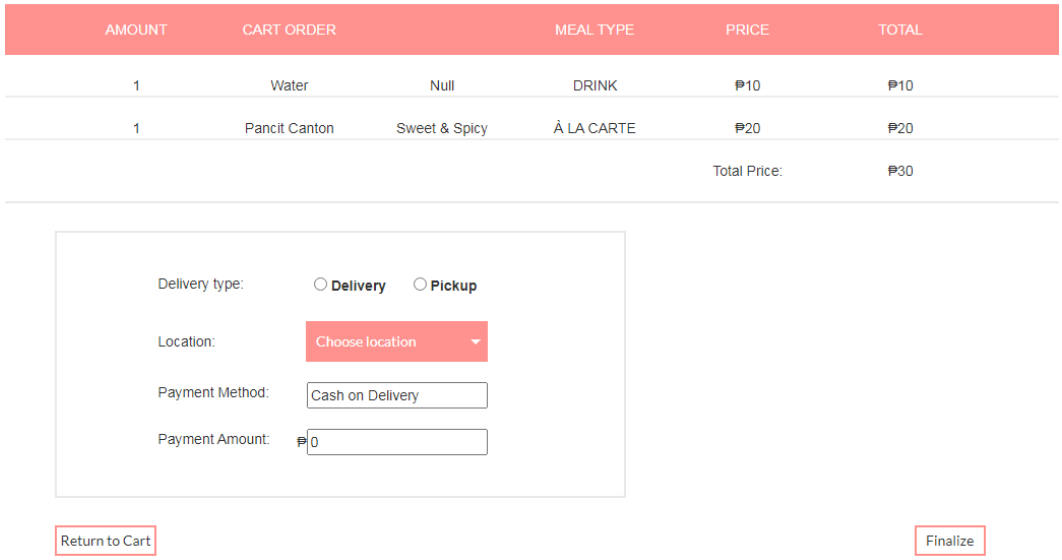
```
@app.route('/cart/delete/<int:cart_id>',
methods=['POST'])
def delete(cart_id):
    usercart = select_dict_cart()
    if request.form['action'] == 'DELETE':
        delete_cart_from_cart(cart_id)
```

```
        return redirect(url_for('cart', cart_id=cart_id,
usercart=usercart))
```

Delete removes the cart data that a user has added into the database. It uses cart_id to specify which cart will be deleted. Each row from the cart webpage table is produced by a for loop, which is the source of the cart_id.

9. Checkout (checkout(user_id, users)), ('checkout.html')

```
@app.route('/cart/<int:user_id>/<users>/checkout/',
methods=['post'])
def checkout(user_id,users):
    loginuser = select_dict_user()
    usercart = select_dict_cart()
    totalprice = 0
    for userdata in usercart:
        totalprice += int(userdata['total_amount'])

    if any(session) == False:
        for users in loginuser:
            users = users['username']
    else:
        if any(session) == True:
            for users in loginuser:
                users['username'] == session['user']
                users = users['username']

    return render_template('checkout.html',
users=users,user_id=user_id,usercart=usercart,totalprice=
totalprice)
```



Checkout is a page in the website that takes the user's final data. The data found in the Checkout page are similar to Cart: cart_id (hidden), alacarte_amount, drinks_type or alacarte_type (depending on which meal type is NULL), flavors_type or sauces_type (

depending on which meal type is NULL), meal_type (depending on which meal type is NULL), price, total amount, total price, order_location (user input), order_type (user input), payment_method and payment_amount (user input).

The cart data from Cart is passed on and displayed on this page after clicking the 'Check out' button. totalprice is also now being displayed. totalprice is a variable that adds all the values of total_amount. None of the data in Cart are now editable, instead the user is left to add their data for the delivery/pick up service. order_type is a radio input that makes the user choose between Delivery or Pickup.

```
<div class="col-sm" style="margin-left:16%;">
    Delivery type:
</div>
<div class="col-sm" style="margin-left:13%;">
    <input type="radio" id="delivery" name="order_type"
value="delivery" required>
<label for="delivery">Delivery</label>
</div>
<div class="col-sm" style="margin-left:5%;">
    <input type="radio" id="pickup" name="order_type"
value="pickup" required></d>
<label for="pickup">Pickup</label>
</div>
```

The order_location is a select input that makes the user choose which place their order is going to be picked up or delivered.

```
    <div class="col-sm-5" style="margin-left:14%;">
<select name="order_location" id="o_location"
style="width:94%;" required>
    <option hidden selected disabled>Choose location
</option>
    <option value="Vinzon's">Vinzon's</option>
    <option value="Arki">Arki</option>
    <option value="Math">Math</option>
    <option value="Palma">Palma</option>
    <option value="A2">A2</option>
</select>
    </div>
```

The payment_amount is a number input that requires the user to meet the total price required for their order to be purchased.

```
    <div class="col-sm" style="margin-left:5%;">
₱<input type="number" min="{{ totalprice }}"
name="payment_amount" value="0" required>
    </div>
```

The 'Return to Cart' button directs the user back to Cart.

```
<form action="{{ url_for('cart', loginuser=loginuser,
usercart=usercart, cartchecker=cartchecker,
totalprice=totalprice) }}">
```

```
    <input class="custom-btn btn-11" type="submit"
value="Return to Cart" name="action"
    style="position:left;width:100px;">
```

And the ‘Finalize’ button takes all the input data in checkout and sends it to finalize processing.

```
<form action="{{ url_for('finalizeprocessing') }}"
method="POST">
.
.
.
<input class="custom-btn btn-11" type="submit"
value="Finalize" name="action" for=""   >
</form>
```

10. finalizeprocessing()

```
@app.route('/finalizeprocessing', methods=['POST'])
def finalizeprocessing():
    usercart = select_dict_cart()
    finalcart = select_dict_finalcart()
    change = 0
    if int(request.form['payment_amount']) >=
int(request.form['totalprice']):
        change = int(request.form['payment_amount']) -
int(request.form['totalprice'])
    else:
        return redirect(url_for('checkout',
users=users,user_id=user_id,usercart=usercart,
                                totalprice=totalprice))

    if any(finalcart) == False:
        finalcart_id = 0
        finalcart_id += 1
    else:
        for data in finalcart:
            finalcart_id = data['finalcart_id']
            finalcart_id += 1

    for userdata in usercart:
        cart_data = {
            'cart_id': userdata['cart_id'],
            'user_id': userdata['user_id'],
            'user': userdata['user'],
            'meals_type': userdata['meals_type'],
            'alacarte_type': userdata['alacarte_type'],
            'drinks_type': userdata['drinks_type'],
            'amount': userdata['amount'],
            'price': userdata['price'],
            'flavors_type': userdata['flavors_type'],
```

```
            'sauces_type': userdata['sauces_type'],
            'total_amount': userdata['total_amount'],
        }
        insert_cart_into_cartreceipt(cart_data)

    try:
        for userdata in usercart:
            finalcart_data = {
                'finalcart_id' : int(finalcart_id),
                'order_type' :
request.form['order_type'],
                'order_location' :
request.form['order_location'],
                'cart_id' : userdata['cart_id'],
                'user_id' : userdata['user_id'],
                'user' : userdata['user'],
                'total_price' :
int(request.form['totalprice']),
                'payment_amount' :
int(request.form['payment_amount']),
                'change' : int(change),
                'payment_method' :
request.form['payment_method']
            }
            insert_final_into_finalcart(finalcart_data)

delete_cart_from_cart(finalcart_data['cart_id'])
    except:
        abort(400)
    return redirect(url_for('finalize',
user_id=request.form['user_id'],

users=request.form['username'],
finalcart_id=finalcart_id))
```

finalizeprocessing() is a function that appends all the gathered data from the Cart table and user data from Checkout into the database cartreceipt and finalcart table. From checkout, the data from cart are more or less just repeated, this is because the data needs to be appended to the cartreceipt table in the database. And when the order is finalized and processed through finalizeprocessing(), all of the data in the cart table are deleted. This is so that the user will have a new instance of orders that they can send to the cart table in the database, if they so want. Saving the cart data into cartreceipt after deleting the data itself from cart makes a backup system where the admin, and user through another website function, can see their specific order history.

All of the data from checkout are passed onto the cart_data and finalcart_data dictionaries. cart_data is added in the cartreceipt database table. finalcart_data is added in the finalcart database table. Notice that the finalcart_data doesn't specify the meals, only their cart_id. This makes for a cleaner design, having to look up which cart_id in

finalcart matches with the cart_id in cartreceipt. And justifies the existence of the cartreceipt table in the database as well.

change is a variable that calculates the payment_amount (user input) and its difference with totalprice. Applying mealprocessing() redirects the user to Cart (‘cart.html’) .

A return error is also added, in the case that the user forgets to change their order_location from checkout.

11. Finalize (finalize(user_id, users, finalcart_id)), (‘final.html’)

```
@app.route('/cart/<int:user_id>/<users>/<int:finalcart_id
>/finalize')
def finalize(user_id,users,finalcart_id):
    loginuser = select_dict_user()
    finalcartid = read_finalcart_by_id(finalcart_id)
    finalcart = select_dict_finalcart()
    cartreceipt = select_dict_cartreceipt()

    if 'user' not in session:
        for users in loginuser:
            users = users['username']
    else:
        if 'user' in session:
            for users in loginuser:
                users['username'] == session['user']
                users = users['username']

    return render_template('final.html', user_id=user_id,
users=users, finalcart_id=finalcart_id,
                           finalcartid=finalcartid,
finalcart=finalcart, cartreceipt=cartreceipt,
                           loginuser=loginuser)
```

THANK YOU FOR PURCHASING!

ORDER IS NOW BEING PROCESSED

Guest
Pickup at Arki
Cart No. 1

1	Water	₱10	₱ 10
1	Pancit Canton	₱ 20	₱ 20
		Total	**₱ 30**
		Payment (Cash on Delivery)	₱ 33
		Change	₱ 3

Enjoy your meal!

Finalize is the webpage that shows the receipt of the user after a successful transaction. If the user is not in session, by default the user is guest. The data shown, from the finalcart database table, are the user, order_type, order_location,

```
<td>{{ finalcartid.user|title }}
    {% if finalcartid.order_type == 'pickup' %}
    <br>Pickup at {{ finalcartid.order_location }}
    {% elif finalcartid.order_type == 'delivery' %}
    <br>Deliver near {{ finalcartid.order_location }}
    {% endif %}
        <br>Cart No. {{ finalcartid.finalcart_id }}
</td>
```

total_price, payment_type, payment_amount, change.

```
<tr class="total">
    <td class="alignright" width="80%">Total</td>
```

```
    <td class="alignright">₱ {{ finalcartid.total_price
}}</td>
</tr>

    <td class="alignright" width="80%">Payment ({{
finalcartid.payment_method }})</td>
    <td class="alignright">₱ {{
finalcartid.payment_amount }}</td>
</tr>
    <td class="alignright">Change</td>
    <td class="alignright">₱ {{ finalcartid.change
}}</td>
</tr>
```

The data shown, from the cartreceipt database table, are amount, drinks_type or alacarte_type, price and total_amount.

```
{% for data in finalcart %}
{% if data.finalcart_id == finalcart_id %}
{% for items in cartreceipt %}
{% if items.cart_id == data.cart_id %}
    {% if items['alacarte_type'] == 'NULL' %}
    <tr>
        <td class="alignleft" width="80%">
            <p><span style="margin-left:5%;"> {{
items.amount }}</span>
            <span style="margin-left:10%;"> {{
items['drinks_type'] }} </span>
            <span style="margin-left:10%;" >₱{{
items['price'] }}</span></p>
        </td>
        <td class="alignright" width="80%">
            <span style="margin-right:0%;">₱ {{
items['total_amount'] }}</span>
        </td>
    </tr>
    {% elif items['drinks_type'] == 'NULL' %}
    <tr>
        <td class="alignleft">
            <p><span style="margin-left:5%;"> {{
items['amount'] }}</span>
            <span style="margin-left:10%;"> {{
items['alacarte_type'] }} </span>
            <span style="margin-left:10%;">₱ {{
items['price'] }}</span></p>
        </td>
        <td class="alignright" width="80%">
            <span style="margin-right:0%;">₱ {{
items['total_amount'] }}</span>
        </td>
```

```
        </tr>
    {% endif %}
{% endif %}
{% endfor %}
{% endif %}
{% endfor %}
```

iii. User Login Functions

1. Login ('login.html')

```
@app.route('/login',)
def login():
    return render_template('login.html',)
```

The website has a login function that takes the user's inputs. The inputs are username and password respectively. The 'LOG-IN' button directs the user to loginprocessing(). The link 'Create one here' directs the user to register ('register.html')

2. loginprocessing()

```
@app.route('/loginprocessing', methods=['POST', 'GET'])
def loginprocessing():
    usercart = select_dict_cart()
    loginuser = select_dict_user()

    for users in loginuser:
        if request.form['name'] in users['username']:
            if request.form['name'] == users['username']
and request.form['password'] == users['password']:
                user = request.form['name']
```

```
                session['user'] = user
                if user in users:
                    users['user_id']
                    user_id = users['user_id']
                    if any(usercart) == True:
                        for ids in usercart:

                            user_data = {
                                'user_id': user_id,
                                'user': user,
                                'cart_id':
ids['cart_id'],
                            }


update_user_in_cart(user_data)
                            return redirect(url_for('user'))
                        else:
                            return redirect(url_for('user'))
            else:
                return redirect(url_for('login', ))
    else:
        return redirect(url_for('register', ))
```

loginprocessing() takes the user's input from Login ('login.html') and processes it. There are four different if statements that have different purposes for the function. From inside-out, the first if statement checks if there is any data inside of the cart table in the database. If there is, then upon login, the dictionary user_data changes the user_id and user of the current cart by using the cart_id to specify which cart will be changed. It then directs the user to user(). If cart is empty, then the user is directed to user(). The second if statement, assigns the variable user_id as users['user_id'] if the user input is in users, which is a variable defined in the loginuser for loop. The third if statement checks if the username and password given by the user are similar to a user and password in the user database. It also sets the input username as session['user']. If the username and password do not match, the user is sent back to Login ('login.html'). The outermost if statement checks if the input username is in the user database. If it is not, the user is then redirected to Register ('register.html').

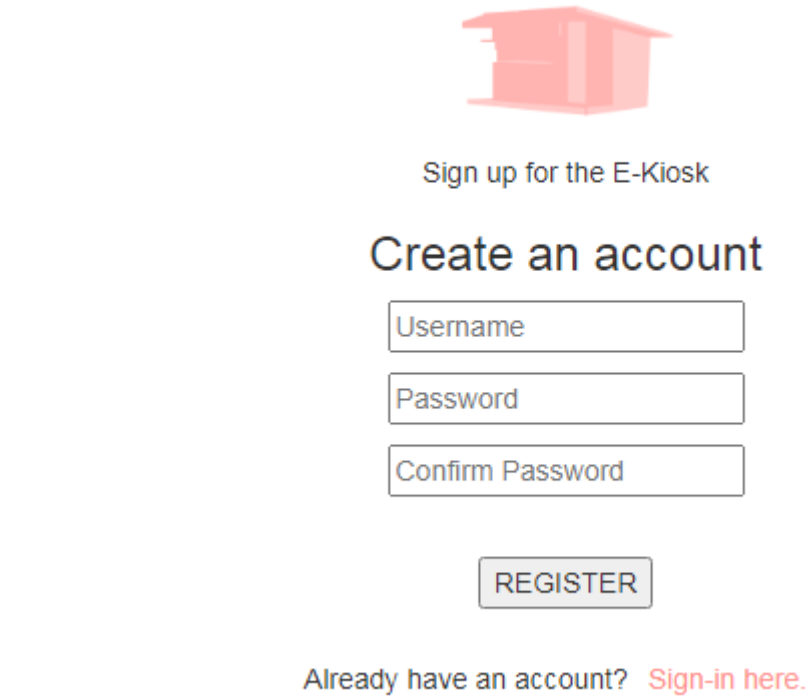
3. user()

```
@app.route('/user')
def user():
    if 'user' in session:
        user = session['user']
        return redirect(url_for('index'))
    else:
        return redirect(url_for('login'))
```

user() checks if the user is in session. If the user is in session, user() redirects the user back to frontpage('index.html'). If the user is not in session, user() redirects the user into Login ('login.html').

4. Register (‘register.html’)

```
@app.route('/register')
def register():
    return render_template('register.html',)
```



Register (‘register.html’) is the account register page. The webpage takes three inputs, name(username), password and confirmpassword.

```
    <form action="{{ url_for('registerprocessing') }}"
method="post">
    <p><input type="text" name="name"
placeholder="Username"></p>
    <p><input type="text" name="password"
placeholder="Password"></p>
    <p><input type="text" name="confirmpassword"
placeholder="Confirm Password"></p>
<br>
    <p><input type="submit" value="REGISTER"></p>

</form>
```

The ‘Sign-in here’ link below redirects the user to Login (‘login.html’).

```
Already have an account?<a href="{{ url_for('login')
}}">Sign-in here.</a>
```

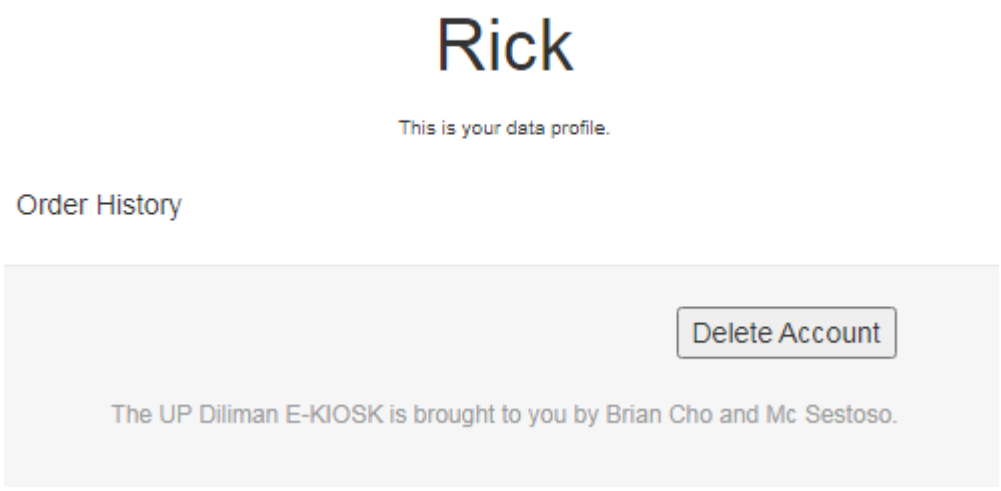
5. registerprocessing()

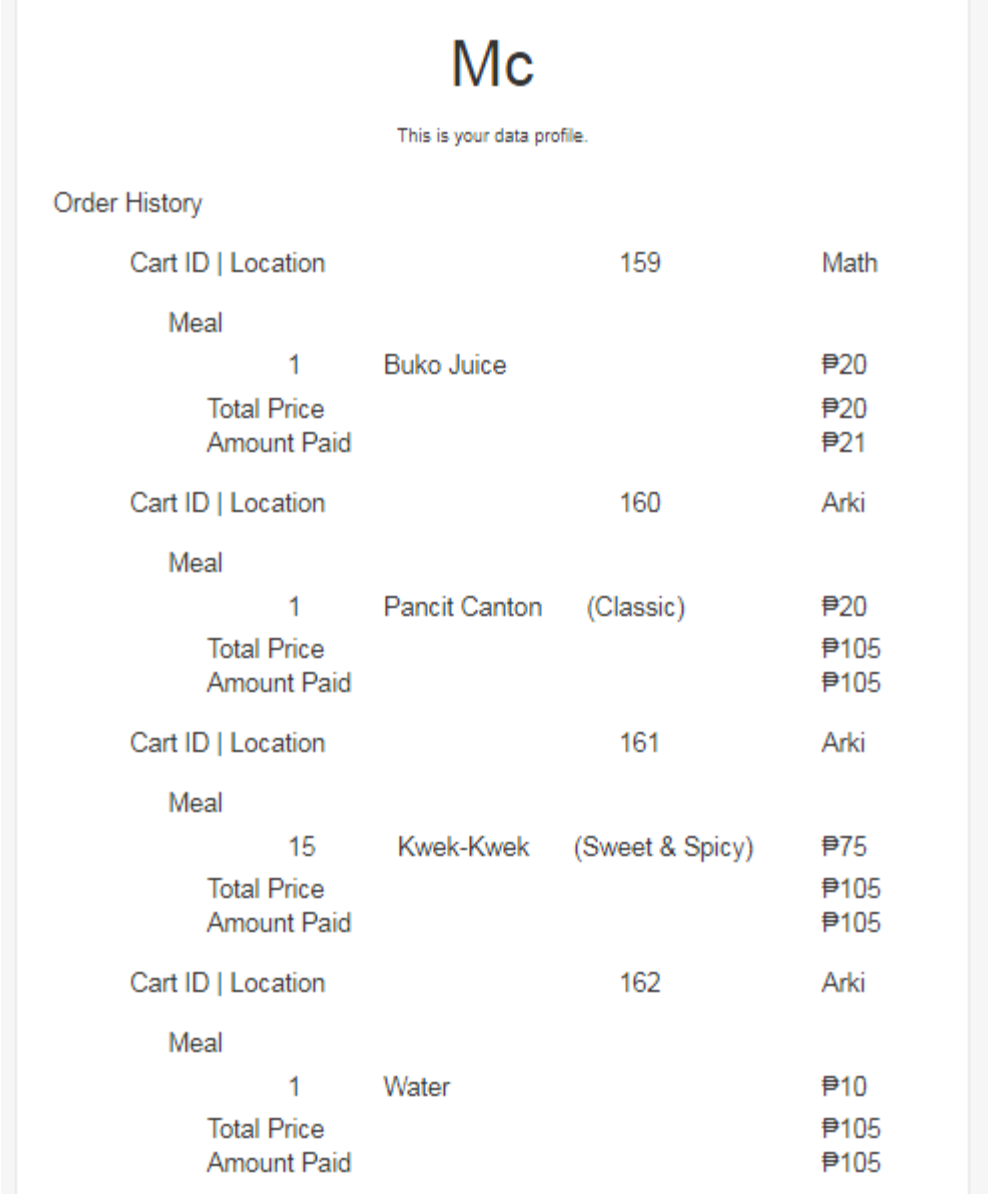
```
@app.route('/registerprocessing', methods=['POST'])
def registerprocessing():
    loginuser = select_dict_user()
    usercart = select_dict_cart()

    for user in loginuser:
        user['username']

    login_data = {
        'user' : request.form['name'],
        'password' : request.form['password'],
    }

    if request.form['password'] ==
request.form['confirmpassword']:
        if login_data['user'] in user['username']:
            return redirect(url_for('login',))
        elif login_data['user'] not in user['username']:
            user = request.form['name']
            session['user'] = user
            if any(usercart) == True:
                for ids in usercart:
                    ids['cart_id']
                    cart_id = ids['cart_id']
                    user_data = {
                        'user' : request.form['name'],
                        'user_id' : ids['user_id'],
                        'cart_id' : cart_id,
                    }
                    insert_user_into_user(login_data)
                    update_user_in_cart(user_data)
            elif any(usercart) == False:
                insert_user_into_user(login_data)
            return redirect(url_for('user'))
    else:
        return redirect(url_for('register'))
```

registerprocessing() processes the user input from Register ('register.html'). There are three different if statements that have different purposes for the function. From inside-out, the first if statement checks if there is any data in the database cart table. If there is, the user's input in the login_data dictionary is added into the database user table. And the user's input in the user_data dictionary is used to update and change the user and user_id of the user and user_id in the database cart table; while using the cart_id to specify which cart will be changed. If there is no data in the database cart table, the login_data is simply added to the user database. The second if statement, checks if the input user already exists in the user table. If it does, then the user is redirected to Login ('login.html'). If it doesn't, then it proceeds to the inner if statement. The outer most if statement requires the user to math the password and confirmpassword input. If the user fails, it redirects the user back to register ('register.html').

6. Profile (profile(username)), (‘profile.html’)

```
@app.route('/profile/<username>/')
def profile(username):
    loginuser = select_dict_user()
    finalcart = select_dict_finalcart()
    cartreceipt = select_dict_cartreceipt()
    cartchecker = any(finalcart)
    for users in loginuser:
        if users['username'] == session['user']:
            users['username']
            username = users['username']
            user_id = users['user_id']
```



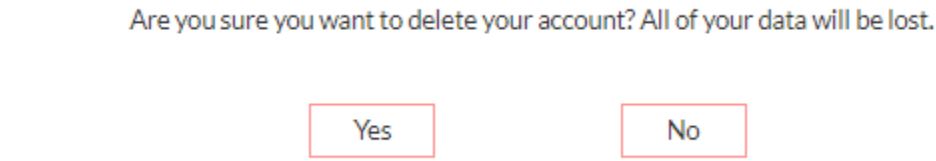


Profile ('profile.html') shows the order history of the current user in session. The data shown are a mix of items from cartreceipt and finalcart. The button 'Delete Account', sends the user to deleteconfirm().

```
<form action="{{ url_for('deleteconfirm') }}"
method="post">
    <span style="margin-left:65%">
    <input class="hidden" id="username" name="username"
value="{{ username }}" readonly>
    <input class="" name="action" type="submit"
value="Delete Account">
    </span>
</form>
```

7. deleteconfirm() ('choice.html')

```
@app.route('/deleteconfirm', methods=['POST'])
def deleteconfirm():
    #delete_user_from_user(username)
    if request.form['action'] == 'Delete Account':
        return render_template('choice.html')
```



deleteconfirm() returns 'choice.html', a web page that gives the user the option to confirm their account deletion or not. The button 'Yes' and 'No' direct the user to deleteaccount().

8. deleteaccount()

```
@app.route('/deleteaccount', methods=['POST'])
def deleteaccount():
    loginuser = select_dict_user()
    usercart = select_dict_cart()

    if request.form['action'] == 'Yes':
        for users in loginuser:
            if users['username'] == session['user']:
                users['username']
                username = users['username']
            for cart in usercart:
                cart_id = cart['cart_id']
                delete_cart_from_cart(cart_id)
        session.pop('user', None)
        delete_user_from_user(username)
        return redirect(url_for('index'))
    elif request.form['action'] == 'No':
        return redirect(url_for('profile',
username=session['user']))
```

deleteaccount() is a function that processes the user's input from 'choice.html'. If they selected Yes, the current user in session gets popped, the current user's data in the database user table is deleted and their cart in the cart table also deleted. They are then redirected back to the homepage ('index.html'). If they selected No, they will be redirected back to their profile ('profile.html') page.

9. Logout

```
@app.route('/logout')
def logout():
    loginuser = select_dict_user()
    usercart = select_dict_cart()
```

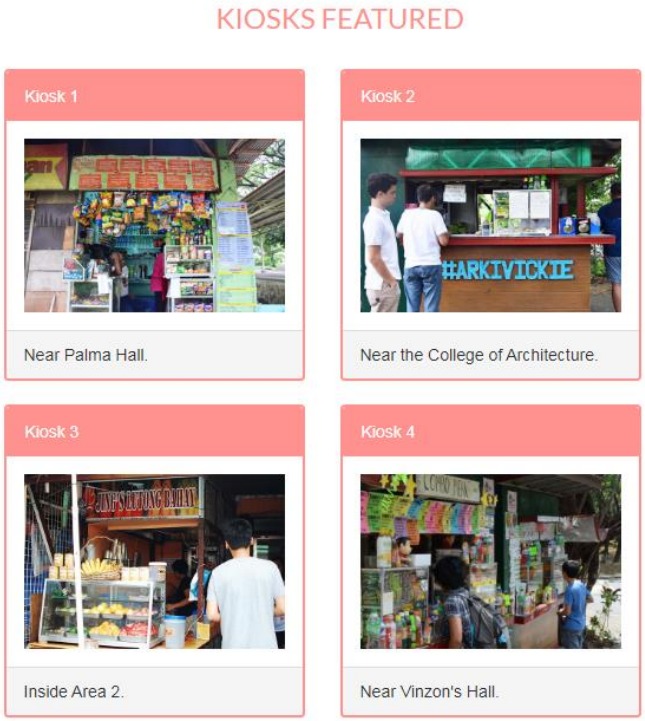
```
    for users in loginuser:
        if users['username'] == session['user']:
            users['username']
            username = users['username']
        for cart in usercart:
            cart_id = cart['cart_id']
            delete_cart_from_cart(cart_id)
    session.pop('user', None)
    return redirect(url_for('login'))
```

logout() is a function that pops the current user from the session and deletes their cart data in the cart table. The user is then redirected to the Login ('login.html') page.

iv. Miscellaneous Functions

1. Kiosks ('order.html')

```
@app.route('/kiosk')
def kiosk():
    return render_template('order.html')
```



Kiosks shows that various kiosk within UP Diliman. Ideally these are the kiosks that utilize the website and where the user can pick up their orders.

2. About ('about.html')

```
@app.route('/about')
def about():
    return render_template('about.html')
```

Welcome to the UP Diliman E-Kiosk. A project made by Brian Cho and Mc Edrian Sestoso. The E-Kiosk project is a website that offers an online food service on the Internet.

About shows information about the website and who its developers are.

v. Maybe – Improvements

1. Profile Receipt formatting. Ideally instead of per cart, the items are sectioned per finalcart_id. In that way, items in the same finalcart are in one space and don't take up much space.

vi. Known Issue(s)

1. After receiving the receipt and not deleting the cart data, an sqlite unique constraint failed (cart_id) error appears. Probably because the same data is being passed with no change. Implemented the deletion of cart_id in the finalcart database table to prevent the error. Probably could add a conditional that changes the cart_id after getting a receipt and ordering again. In the future maybe.