app.py

```
# imports Flask module and other specific functions
from flask import Flask, render_template, request, redirect, url_for
from data import *
app = Flask(__name__)
# routes to the index of the website and renders 'index.html' template
@app.route('/')
def index():
    return render_template('index.html')
# calls the 'read_restaurants_by_location()' function from the data.py to get the list of restaurants based on a certain location and renders 'store.html' template
@app.route('/stores/<location>')
def stores(location):
    restaurants_list = read_restaurants_by_location(location)
    return render_template("stores.html", location=location, restaurants=restaurants_list)
# calls the 'read_restaurant_by_id()' function from the data.py to get the details of a specific restaurant based on the provided ID and renders the 'restaurant.html' template
@app.route('/stores/<int:restaurant_id>')
def restaurant(restaurant_id):
    restaurant = read_restaurant_by_id(restaurant_id)
    return render_template("restaurant.html", restaurant=restaurant)
# renders the 'register.html' template
@app.route('/register')
def register():
    return render_template('register.html')
# processes the form data submitted via POST request, inserts the restaurant data into the storage, and redirects to the stores page for the specified location
@app.route('/processed', methods=['POST'])
def processing():
# handles the modifications (EDIT and DELETE) for a restaurant, either renders and update form or deletes a record, and redirects to the 'stores' page for the restaurant's location
@app.route('/modify', methods=['post'])
def modify():
# retrieves the updated restaurant data from the form, calls the update_restaurant() function to update the restaurant information, and then redirects to the "restaurant" page for the updated restaurant ID
@app.route('/update', methods=['post'])
def update():
# retrieves the search query from the URL parameters, calls the search_restaurants() function to get the search results, and then renders the search.html template, passing the search query and results to it
@app.route('/search', methods=['get'])
def search():
# ensures that the application is run only when the script is executed directly and starts the server with debugging enabled
if __name__ == "__main__":
    app.run(debug=True)
```

data.py

```
# imports the 'sqlite' module and defines the path to the SQLite database file
import sqlite3
```

```
db_path = "database.db"
# establishes a connection to the SQLite database file
def connect_to_db(path):
    conn = sqlite3.connect(path)
    conn.row_factory = sqlite3.Row
    return (conn, conn.cursor())
# reads all the restaurants from the database that match the specified 'location', fetches all the results returned by the query, and returns the results
def read_restaurants_by_location(location):
    conn, cur = connect_to_db(db_path)
    query = 'SELECT * FROM restaurants WHERE location = ?'
    value = location
    results = cur.execute(query,(value,)).fetchall()
    conn.close()
    return results
# reads a single restaurant from the database based on the specified id, fetches all the results returned by the query, and returns the results
def read_restaurant_by_id(id):
    conn, cur = connect_to_db(db_path)
    query = 'SELECT * FROM restaurants WHERE id = ?'
    value = id
    results = cur.execute(query,(value,)).fetchone()
    conn.close()
    return results
# inserts a new restaurant into the database using the provided restaurant_data, commits the changes to the database, and closes the connection
def insert_restaurant(restaurant_data):
    conn, cur = connect_to_db(db_path)
        query = 'INSERT INTO restaurants (location, name, price, review, rating, payment_method, url) VALUES (?,?,?,?,?,?,?)'
    values = (restaurant_data['location'], restaurant_data['name'],
            restaurant_data['price'], restaurant_data['review'],
            restaurant_data['rating'], restaurant_data['payment_method'],
            restaurant_data['url'])
    cur.execute(query,values)
    conn.commit()
    conn.close()
# updates an existing restaurant in the database based on the provided restaurant_data, commits the changes to the database, and closes the connection
def update_restaurant(restaurant_data):
    conn, cur = connect_to_db(db_path)
            query = "UPDATE restaurants SET location=?, name=?, price=?, review=?, rating=?, payment_method=?, url=? WHERE id=?"
    values = (restaurant_data['location'], restaurant_data['name'],
            restaurant_data['price'], restaurant_data['review'],
            restaurant_data['rating'], restaurant_data['payment_method'],
            restaurant_data['url'], restaurant_data['restaurant_id'])
    cur.execute(query, values)
    conn.commit()
```

```
    conn.close()
# deletes a restaurant from the database based on the provided restaurant_id, commits the changes to the database and closes the connection
def delete_restaurant(restaurant_id):
    con, cur = connect_to_db(db_path)
    query = "DELETE FROM restaurants WHERE id=?"
    values = (restaurant_id,)
    cur.execute(query, values)
    con.commit()
    con.close()
# searches for restaurants in the database that match the provided query, fetches all the results returned by the query, closes the database connection, and returns the results
def search_restaurants(query):
    conn, cur = connect_to_db(db_path)
    sql_query = "SELECT * FROM restaurants WHERE location LIKE ? OR name LIKE ?"
    value = "%{}%".format(query)
    results = cur.execute(sql_query, (value, value)).fetchall()
    conn.close()
    return results
```

base.html

```
# the <!DOCTYPE html> declaration specifies the document type and version of HTML being used.
# <meta> tags define character encoding, compatibility mode, and viewport properties.
#  <link> tags import external CSS files for styling the webpage
#   <style> includes custom CSS styles
# <title> sets the title of the webpage
# <body> encloses the main content of the webpage
<body>
#{% block content %} and {% endblock %} defines where specific content can be inserted later.
        {% block content %}
        {% endblock %}
    </div>
#script> tag imports a JavaScript file from a CDN.
```

#overall, this code establishes an HTML document's fundamental structure, incorporates external resources for styling, and acts as a placeholder for further information. For improved functionality, it also imports a JavaScript file and makes use of server-side templating.

index.html

```
#the {% extends 'base.html' %} indicates that this template base.html was extended in this file.
{% extends 'base.html' %}
#the code within the {% block content %} and {% endblock %} tags represents the specific content for this file.
#overall, this template defines a content block, extends a base template, and provides particular content for the webpage. It creates a container element containing a number of food items, each with an image, title, and description. The styling is used to improve the website's visual presentation.
```

nav.html

#in conclusion, this code generates a flexible navigation bar with a branding feature. It offers search functionality, home page navigation, and recommendation page navigation options. The styling is used to alter the navbar's visual style.

register.html

#in summary, this code represents a registration form where users can recommend their favorite food places at UP Diliman. The form collects information such as the location, restaurant/stall name, pricing, review, rating, modes of payment, and an image link. This information can be submitted for further processing.

search.html

#in summary, this code represents a search results page template that displays the search results for a specific query. It shows the search query in the heading, and if there are results, it lists the restaurants with their respective details such as name, image, pricing, review, rating, and mode of payment. If no results are found, it displays a message indicating that the keyword was not found.

stores.html

#in summary, this code serves as a template for showing a list of restaurants nearby. It starts out with a title and description to set the context before looping through the list of restaurants and displaying their names as clickable links. For uniform appearance and structure across the website, the template is intended to be used in conjunction with the base.html template.

update.html

#in summary, this code represents a template for editing the details of a restaurant. It includes a form that allows users to update various attributes of the restaurant, such as its location, name, pricing, review, rating, payment methods, and image link. The template is designed to be used with the base.html template for consistent styling and structure across the website.