HOMEWORK ASSIGNMENT 6

CSCI 571 – Fall 2022

Abstract

Server-side Scripting using Python, Flask, JSON, AJAX, and Yelp Fusion API

Prof. Marco Papa

papa@usc.edu

Homework 6: Server-side Scripting using Python, Flask, JSON, AJAX, and Yelp API

1 Objectives

- Get experience with the Python programming language and Flask framework.
- Get experience with the Google API, and Yelp Fusion API.
- Get experience creating web pages using HTML, CSS, JavaScript, DOM, JSON format and the XMLHttpRequest object.
- Get experience using JSON parsers in Python and JavaScript.
- Getting hands-on experience in GCP, AWS or Azure.

1.1 Cloud Exercise

The backend of this homework must be implemented in the cloud on Google Cloud App Engine, AWS or Azure, using Python.

- See assignment #5 for the installation of needed components on GCP, AWS or Azure.
- See the hints in Section 3; a lot of reference material is provided to you.
- For Python and Flask kick-start, please refer to the Lecture slides on the class website.
- You must refer to the grading guidelines, the video, the specs, and Piazza. Styling will be graded, and the point's breakup is mentioned in the grading guidelines.

2. Description

In this exercise, you are asked to create a webpage that allows you to search for business information using the Yelp API, and the results will be displayed in a card and tabular format.

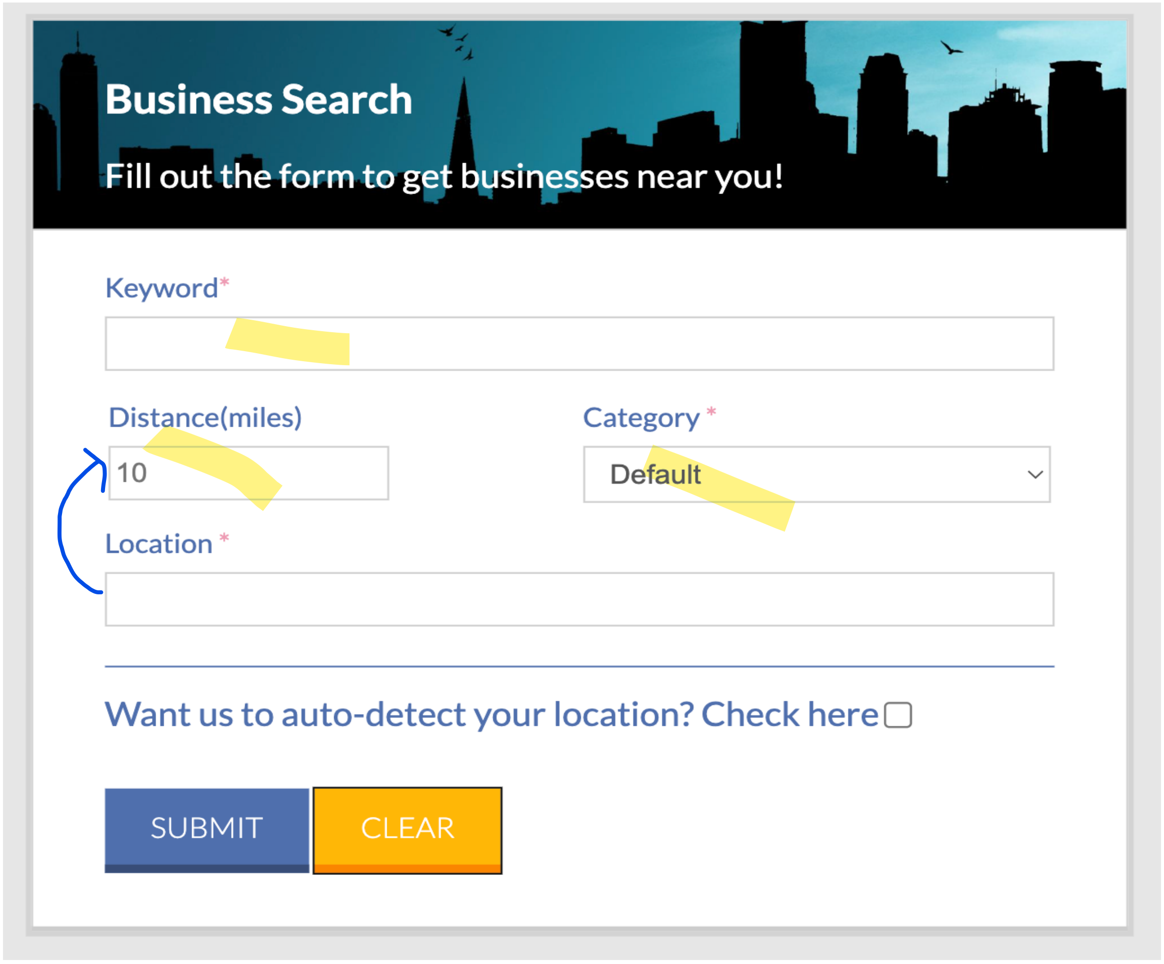
2.1. Description of the Search Form

The user first opens a web page (for example, **business.html,** or any valid web page name). The form has 5 components Keyword, Distance (miles), Category, Location, and a checkbox to auto-detect location. You should use the ipinfo.io API (See hint 3.3) to fetch the user's geolocation if the location checkbox is checked else the user must enter a location to search. Keyword, Distance and Location being the text box while Category being a dropdown.

The categories to include:

- Default
- Arts & Entertainment
- Health & Medical
- Hotels & Travel
- Food
- Professional Services

An example is shown in **Figure 1**.


Business Search
Fill out the form to get businesses near you!
Keyword*
Distance(miles)
10
Category *
Default
Location *
Want us to auto-detect your location? Check here
SUBMIT
CLEAR

If the **Check here** checkbox is checked then the location field should reset the **Location** textbox to blank and disable the field.

The search form has two buttons:

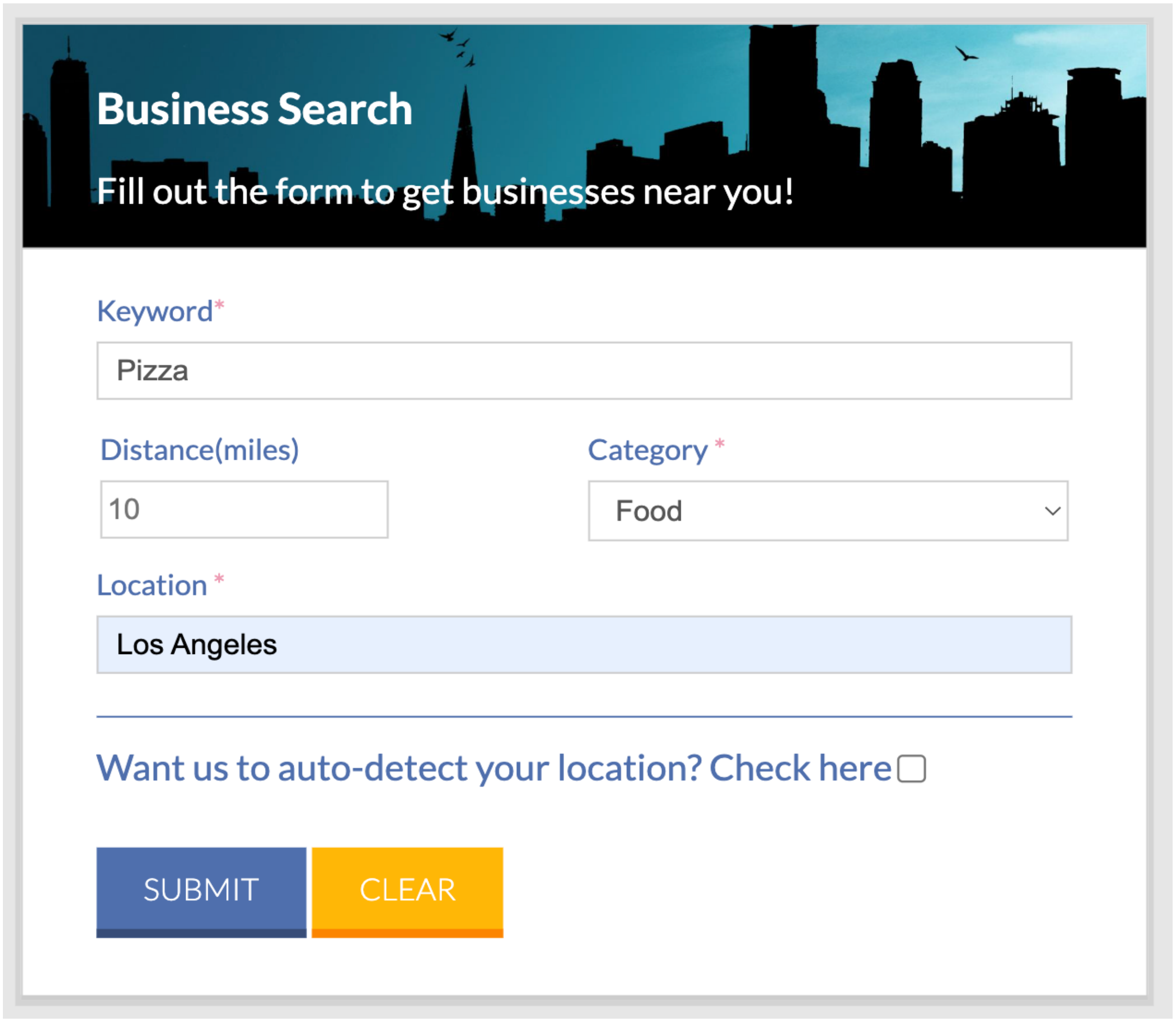
- **SUBMIT** button. Selecting this button performs a search using the given business keyword, the distance filter from the given location and the Category of businesses. An example of valid input is shown in **Figure 2**. Once the user has provided valid input, your client JavaScript should send a request to your web server Python script with the form inputs. You must use GET to transfer the form data to your web server (**do not use POST**, as you would be unable to provide a sample link to your cloud services). A Python script using Flask will extract the form inputs and use them to invoke the *YELP API* business information service. You need to use the *Flask* Python framework to make all the API calls. You may also use Django, though we can only provide limited support on Piazza for Django.

  If the user clicks on the SUBMIT button without providing a value in the “Keyword”, “Category” and “Location” fields or checking the location checkbox, you should show an error “tooltip” that indicates which field is missing. An example is shown in **Figure 2b**.

Using XMLHttpRequest or any other JavaScript calls for anything other than calling your own "cloud" backend will lead to a 4-point penalty. Do not call the Yelp API directly from JavaScript. You may use XMLHttpRequest to call the *Google Maps Geocoding API* service directly from JavaScript (see later).

Define the routing endpoints and make your API call from the Python backend. The recommended tutorial for *Flask* and more importantly, routing, can be found at the following link: https://flask.palletsprojects.com/en/1.1.x/

- **CLEAR** button. This button must clear the result area (below the search area) and set all the form fields to the default values in the search area. The **CLEAR** operation must be done using a JavaScript function.


Business Search
Fill out the form to get businesses near you!
Keyword*
Distance(miles)
Category *
Location *
Want us to auto-detect your location? Check here
SUBMIT
CLEAR

Keyword*	Pizza
Distance(miles)	10
Category *	Food
Location *	Los Angeles

Figure 2: An Example of a Valid Search

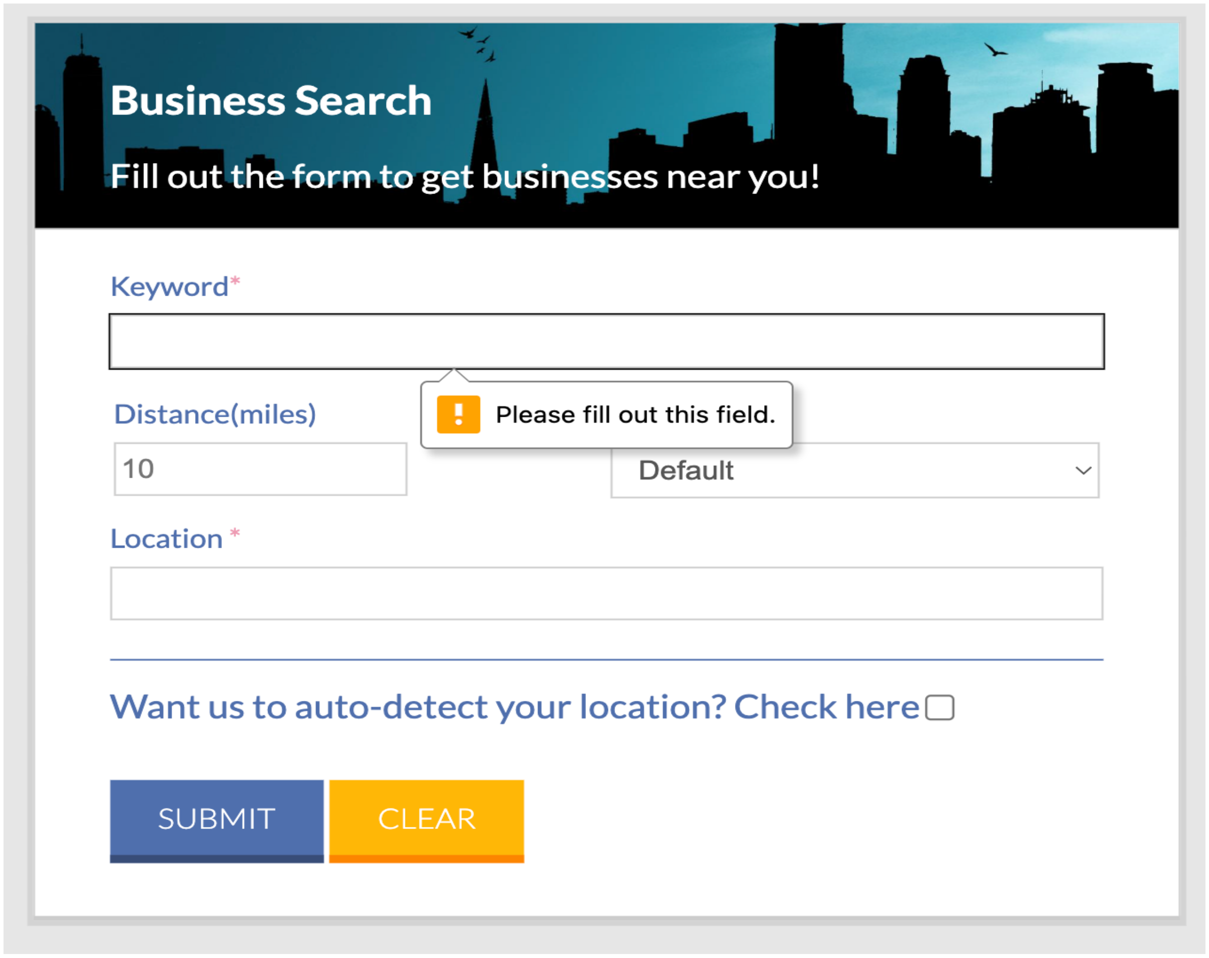

Business Search
Fill out the form to get businesses near you!
Keyword*
Distance(miles)
10
Please fill out this field.
Default
Location *
Want us to auto-detect your location? Check here
SUBMIT
CLEAR

Figure 2b: An Example of an Invalid Search

2.2 Displaying Business Results

In this section, we outline how to use the form inputs to construct the calls to the RESTful web services to the *Yelp API* service and display the results in the web page.

The *Yelp API* is documented here:

https://www.yelp.com/developers/documentation/v3/get_started

If the **Location** information is used to get business results, your client JavaScript should use the input address to get the geocoding via the *Google Maps Geocoding API*. The *Google Maps Geocoding API* is documented here:

https://developers.google.com/maps/documentation/geocoding/start

The Google Maps Geocoding API expects two parameters:

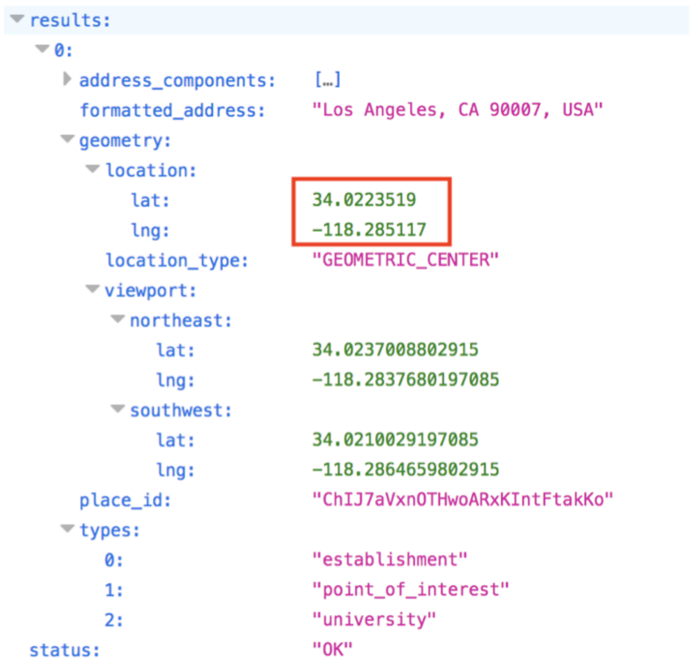
- ***address***: The location that you want to geocode, in the format used by the national postal service of the country concerned. Additional address elements such as business names and unit, suite or floor numbers should be avoided.
- ***key***: Your application's API key. This key identifies your application for purposes of quota management. (Explained in Section 3.2).

2.2.1 Geocoding

An example of an HTTP request to the Google Maps Geocoding API, when the location address is "University of Southern California, CA" is shown below:

https://maps.googleapis.com/maps/api/geocode/json?address=University+of+Southern+California+CA&key=YOUR_API_KEY

The response includes the latitude and longitude of the address.



```
results:
  0:
    address_components:      […]
    formatted_address:       "Los Angeles, CA 90007, USA"
    geometry:
      location:
        lat:                 34.0223519
        lng:                 -118.285117
      location_type:         "GEOMETRIC_CENTER"
      viewport:
        northeast:
          lat:               34.0237008802915
          lng:               -118.2837680197085
        southwest:
          lat:               34.0210029197085
          lng:               -118.2864659802915
    place_id:                "ChIJ7aVxnOTHwoARxKIntFtakKo"
    types:
      0:                     "establishment"
      1:                     "point_of_interest"
      2:                     "university"
status:                      "OK"
```

Figure 3: Example of object from Google Maps Geocoding

Figure 3 shows an example of the JSON object returned in the *Google Maps Geocoding API* web service response.

The latitude (lat) and longitude (lng) of the location are used when constructing a RESTful web service URL to retrieve matching search results.

2.2.2. Displaying table of businesses

We use the ***Yelp "business search" Service*** to get general information of the businesses that match our search criteria (the values on the search form). Please refer to

https://www.yelp.com/developers/documentation/v3/business_search

for more information.

API Sample:
https://api.yelp.com/v3/businesses/search?term=[KEYWORD]&latitude=[LAT]&longitude=[LONG]&categories=[CAT]&radius=[RAD]

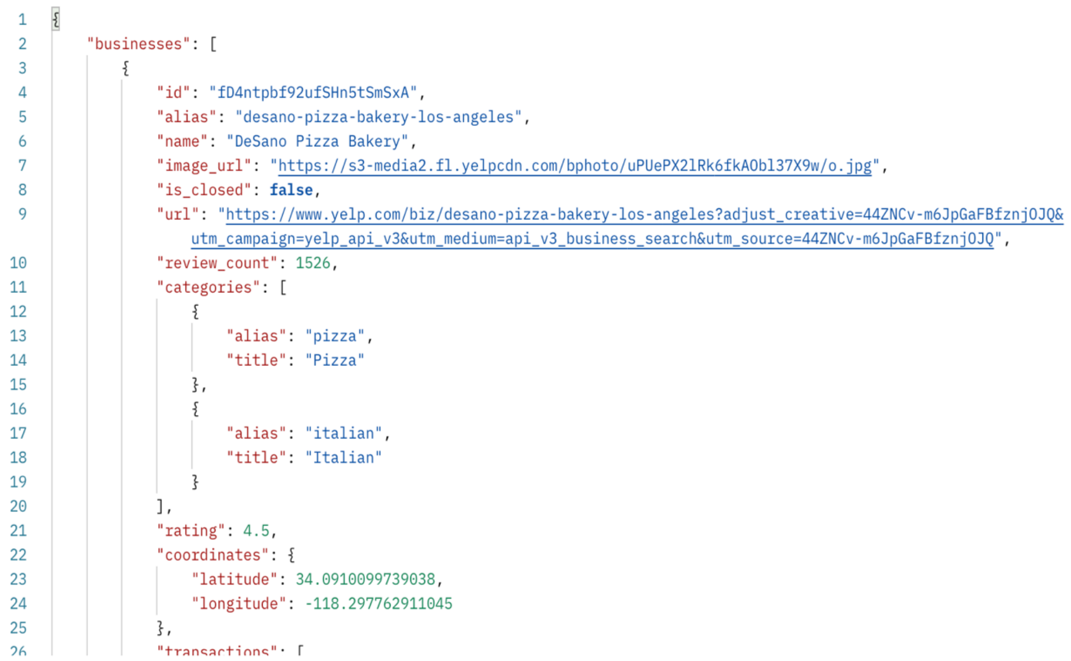
NOTE: Setup authorization is using request headers with the API key for the app. Refer to section 3.1 for more details.

The process to create your API key is explained in section 3. When constructing the Python request required to obtain the matching results, you need to provide five (5) values for the parameters in the URL:

1. The first is the **term**, i.e., the **keyword** the user entered in the form.
2. The second is the **latitude** of the location.
3. The third is the **longitude** of the location.
4. The fourth is the **categories**. We get this value from the category field of the search form. (In the case of **default** category, use **categories=all**)
5. The fifth is the **radius**. We can get this value from the **distance** field of the search form. Keep in mind that Yelp accepts radius in **meters**. You will have to convert miles to meters for calling the API using the radius parameter.

The **response** of this Python request is a **JSON-formatted object**. **Figure 5** shows a sample response from the above request. You need to parse this JSON object and extract some fields as required or you may pass the object unchanged to the client JavaScript. The mapping of the JSON response to the tabular data to be displayed on the screen is shown in **Table 1**.

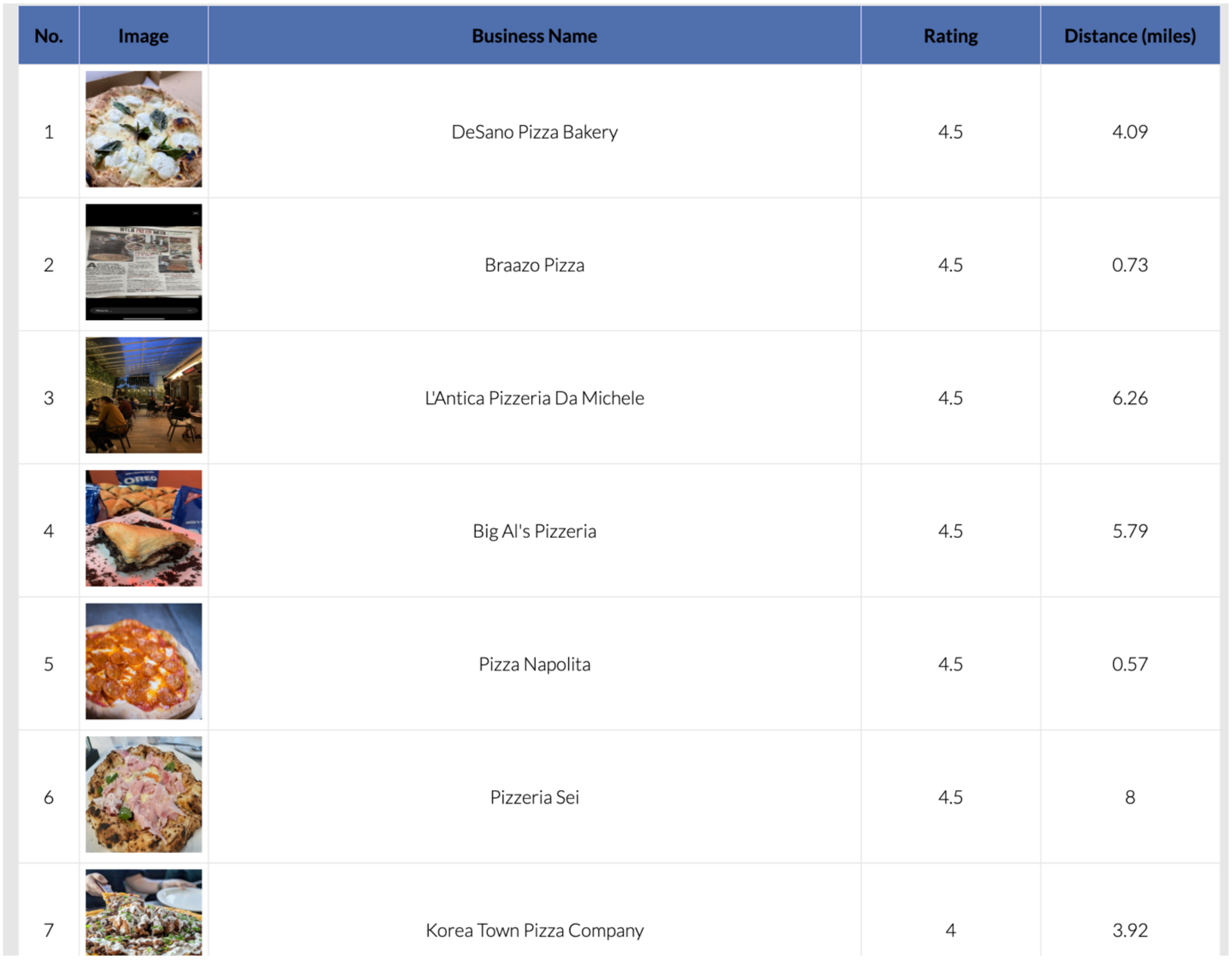
A sample response is shown in **Figure 5.** It shows only the first result.



```
{
    "businesses": [
        {
            "id": "fD4ntpbf92ufSHn5tSmSxA",
            "alias": "desano-pizza-bakery-los-angeles",
            "name": "DeSano Pizza Bakery",
            "image_url": "https://s3-media2.fl.yelpcdn.com/bphoto/uPUePX2lRk6fkAObl37X9w/o.jpg",
            "is_closed": false,
            "url": "https://www.yelp.com/biz/desano-pizza-bakery-los-angeles?adjust_creative=44ZNCv-m6JpGaFBfznjOJQ&
                utm_campaign=yelp_api_v3&utm_medium=api_v3_business_search&utm_source=44ZNCv-m6JpGaFBfznjOJQ",
            "review_count": 1526,
            "categories": [
                {
                    "alias": "pizza",
                    "title": "Pizza"
                },
                {
                    "alias": "italian",
                    "title": "Italian"
                }
            ],
            "rating": 4.5,
            "coordinates": {
                "latitude": 34.0910099739038,
                "longitude": -118.297762911045
            },
            "transactions": [
```

Figure 5: Example of the JSON response returned by the Yelp Fusion *API* service response.

The Python script may pass the returned JSON object to the client side "unmodified" (i.e., function as a pass-through") or parse the returned JSON and extract only the useful fields and pass these fields to the client side in a **JSON-formatted object**. You should use JavaScript to parse the JSON object, extract the needed fields, and display the results in a tabular format and a table view containing all the businesses in given distance, filtered by keyword and categories. A sample output is shown in **Figure 6**. The displayed table includes five columns: **No., Image, Business Name, Rating, and Distance in miles**. The columns may be sorted by clicking on the table header based on Name, Rating or Distance. By default, Yelp returns a maximum of 20 businesses if no limit parameter is provided. For this assignment we will be using the default limit of 20 and fetch the results of 20 businesses (or less, if the matches are less than 20).



No.	Image	Business Name	Rating	Distance (miles)
1		DeSano Pizza Bakery	4.5	4.09
2		Braazo Pizza	4.5	0.73
3		L'Antica Pizzeria Da Michele	4.5	6.26
4		Big Al's Pizzeria	4.5	5.79
5		Pizza Napolita	4.5	0.57
6		Pizzeria Sei	4.5	8
7		Korea Town Pizza Company	4	3.92

Figure 6: Output of Search Results

2.3 Displaying Detailed information of the business

In the search results, if the user clicks on the name of a business, a card should be displayed with detailed results of the business below the table of listed businesses. The page should request the ~~detailed information~~ using the *Yelp "Business Details" API* and direct to a section populated with this information, as shown in **Figure 8.**

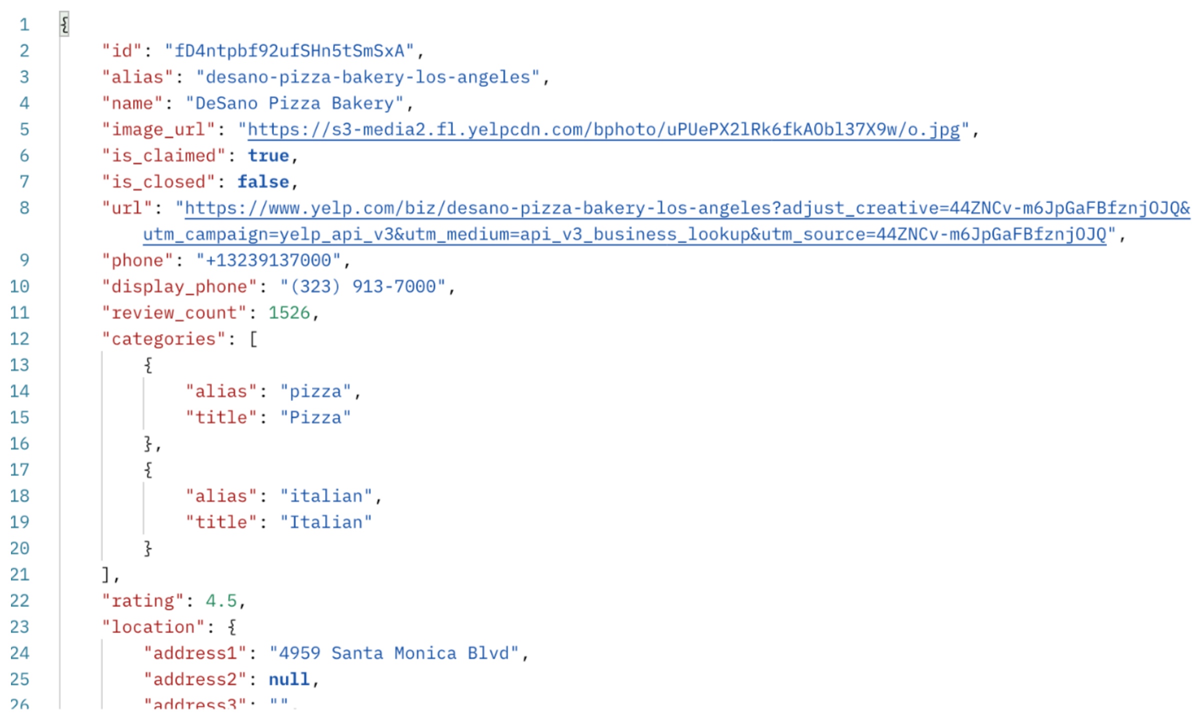
We use the ***Yelp "Business Details" service*** to get detailed information about the business. Please refer to

https://www.yelp.com/developers/documentation/v3/business

for more information.

API Sample: https://api.yelp.com/v3/businesses/{id}

A sample response is shown in **Figure 7**.



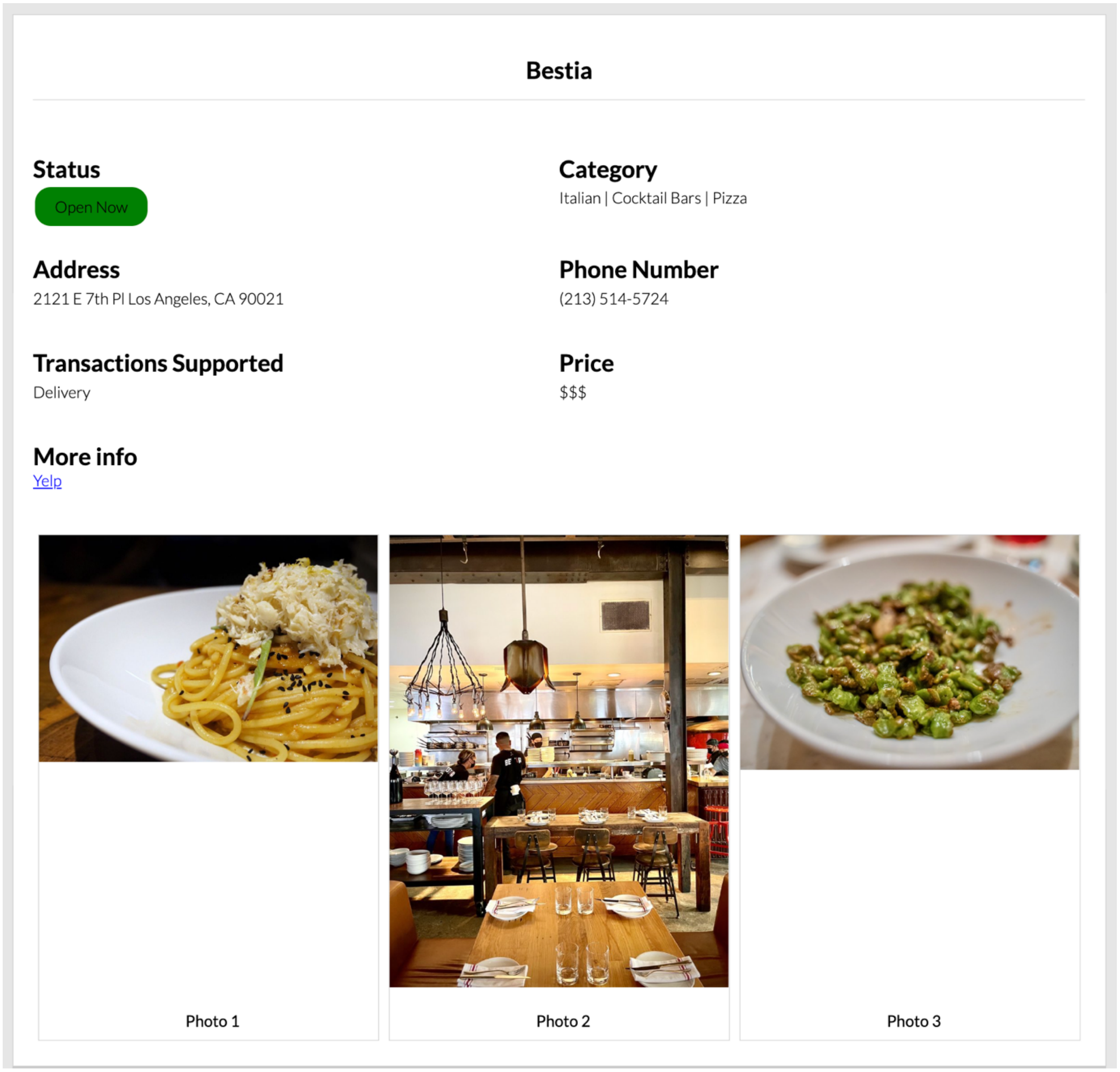
```
{
    "id": "fD4ntpbf92ufSHn5tSmSxA",
    "alias": "desano-pizza-bakery-los-angeles",
    "name": "DeSano Pizza Bakery",
    "image_url": "https://s3-media2.fl.yelpcdn.com/bphoto/uPUePX21Rk6fkA0bl37X9w/o.jpg",
    "is_claimed": true,
    "is_closed": false,
    "url": "https://www.yelp.com/biz/desano-pizza-bakery-los-angeles?adjust_creative=44ZNCv-m6JpGaFBfznjOJQ&
        utm_campaign=yelp_api_v3&utm_medium=api_v3_business_lookup&utm_source=44ZNCv-m6JpGaFBfznjOJQ",
    "phone": "+13239137000",
    "display_phone": "(323) 913-7000",
    "review_count": 1526,
    "categories": [
        {
            "alias": "pizza",
            "title": "Pizza"
        },
        {
            "alias": "italian",
            "title": "Italian"
        }
    ],
    "rating": 4.5,
    "location": {
        "address1": "4959 Santa Monica Blvd",
        "address2": null,
        "address3": "",
```

Figure 7: Example of the JSON response returned by the Yelp Fusion *API Business Details* service

The details to be included are:

- Status
- Address
- Transaction Supported
- Category
- Phone number
- Price
- More info

Note: Some businesses might not have all the details above. In that case the sub-heading for that detail should be SKIPPED. From the displayed output



Bestia

Status

Open Now

Category

Italian | Cocktail Bars | Pizza

Address

2121 E 7th Pl Los Angeles, CA 90021

Phone Number

(213) 514-5724

Transactions Supported

Delivery

Price

$$$

More info

Yelp

Photo 1

Photo 2

Photo 3

Figure 8: Business details

3. Hints

3.1 How to get the YELP Fusion API Key

To get a Yelp fusion API key, please follow these steps:

- Create a new account at:
  https://www.yelp.com/signup
- Go to Create App at:
  https://www.yelp.com/developers/v3/manage_app
- Create new app form, enter information about the app and submit.
- Go to the Manage App on the left panel and your API Key should be there.
- The Yelp fusion API documentation can be found at:
  https://www.yelp.com/developers/documentation/v3/get_started
- Add Authentication headers documentation can be found at:
  https://www.yelp.com/developers/documentation/v3/authentication
- Use Authentication Headers: put the API key inside of the request header as "**Authorization: Bearer <YOUR API KEY>**" and make requests against the API.

3.2 How to get the Google API Key

- To get a Google API key, please follow these steps:
- Go to the Google Developers Console:
  https://console.developers.google.com/flows/enableapi?apiid=geocoding_backend&keyType=SERVER_SIDE&reusekey=true.
- Create a project, if you do not have one already.
- At the Google APIs' guide page, click "Get a key" and select the created project.
  Note that you should NOT use a google account associated with a USC email. You must use a Gmail account.

3.3 How to get the IPInfo.io API Key

- Go to https://ipinfo.io/ and sign up for free
- A token would be provided after successful sign up

An example call is as follows: https://ipinfo.io/?token=YOUR_TOKEN_ID

3.4 Deploy Python file to the cloud (GCP/AWS/Azure)

You should use the domain name of the GAE/AWS/Azure service you created in Assignment #5 to make the request. For example, if your GAE/AWS/Azure server domain is called **example.appspot.com** or **example.elasticbeanstalk.com** or **example.azurewebsites.net**, the following links will be generated:

GAE - http://example.appspot.com/index.html

AWS - http://example.elasticbeanstalk.com/index.html

Azure - http://example.azurewebsites.net/index.html

The *example* subdomain in the above URLs will be replaced by your choice of subdomain from the cloud service during setup. You may also use a different page than index.html as your top level home page.

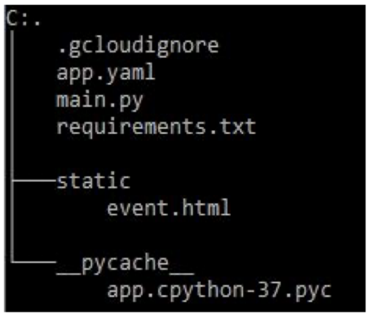
For example, if your GAE server domain is called **example.appspot.com**, the following links will be generated: http://example.appspot.com/index.html

The *example* subdomain in the above URLs will be replaced by your choice of subdomain from the cloud service. You may also use a different page than index.html. For example, the files to deploy on GCP would be:

1. client-side files (HTML+CSS+JS)

2. server-side file (main.py), .yaml, requirements.txt

The project structure may also look like the following one:



```
C:.
│   .gcloudignore
│   app.yaml
│   main.py
│   requirements.txt
│
├───static
│       event.html
│
└───__pycache__
        app.cpython-37.pyc
```

You are free to use any folder structure for your project.

3.5 Behavior for Search Forms

If the API service returns an empty result set with no matching businesses, the page should display "No records have been found" as shown in **Figure 9**:

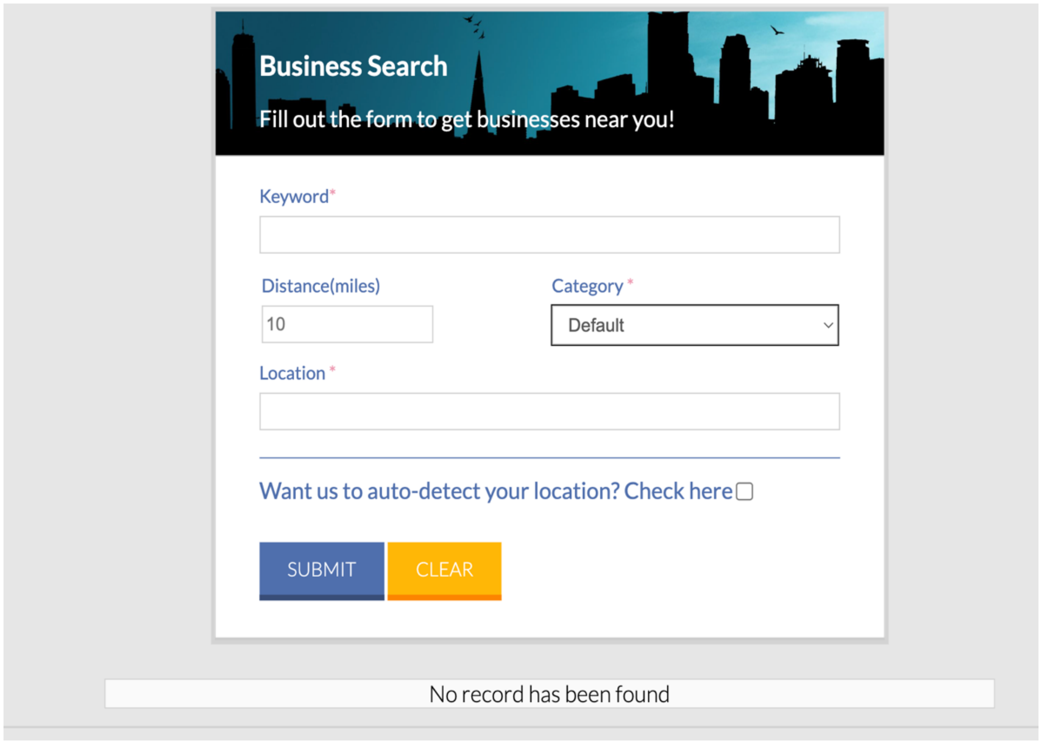


Figure 9: Display no matching businesses

3.6 Parsing JSON-formatted data in Python

Information on how to parse JSON-formatted data in Python is available here:

https://docs.python.org/3/library/json.html

If you use your cloud server as a "proxy" pass-through, you do not have to decode and encode the JSON.

4. Files to Submit

In your course homework page, you should update the **Homework 6** link to refer to your new initial web search page for this exercise (for example, **business.html**). Your files must be hosted on GAE, AWS or the Azure cloud service. Graders will verify that this link is indeed pointing to GAE.

Also, submit your source code files to DEN D2L. Submit a ZIP file of both front-end and back-end code, in separate folders, plus any additional files needed to build your app (e.g., yaml file). **The timestamp of the ZIP file will be used to verify if you used any "grace days."**

****IMPORTANT**:**

- All discussions and explanations in Piazza related to this homework are part of the homework description and grading guidelines. So please review all Piazza threads, before finishing the assignment. If there is a clarification on Piazza that conflicts with this description and/or the grading guidelines, **Piazza always rules**. In most cases, the clarification is related to an error in the description, or missing information from the description, but no additional functionality.
- You can use jQuery for Homework 6, but its use is not required.
- You can use Google Cloud Functions and other Faas, in lieu of building a monolithic application.
- You **should not call the Yelp APIs directly from JavaScript**, bypassing the Python proxy. Implementing any one of them in JavaScript instead of Python will result in a **4-point penalty.** Other APIs can be called from JavaScript.
- You may call the Google Maps Geocoding API directly from JavaScript.
- **APPEARANCE OF CARD VIEW and TABLE should be as similar as possible to the reference video.**