Homework 6: Search Server-side Scripting using PHP, JSON and Ticketmaster API

1. Objectives

- Get experience with the PHP programming language;
- Get experience with the Google API and Ticketmaster API;
- Get experience using JSON parsers in PHP and JavaScript.
- Get hands-on experience in GCP App Engine, AWS or Azure

1.1. Cloud exercise

The back-end of this homework must be implemented in the cloud on GCP App Engine, AWS or Azure using PHP.

See homework 5 for installation of either one of these platforms. You only have to select one platform to implement your back-end.

2. Description

In this exercise, you are asked to create a webpage that allows you to search for events information using the Ticketmaster API, and the results will be displayed in a tabular format. The page will also provide event details and venue details.

2.1. Description of the Search Form

A user first opens a page, called **event.php** (or any valid web page name). You should use the *ip-api.com HTTP API* (See hint 3.3) to fetch the user's geolocation, after which the search button should be enabled (it is initially greyed out and disabled when the page loads). The user must enter a keyword and choose what *Category* of event he/she wants to search (categories include Music, Sports, Arts & Theatre, Film, Miscellaneous) from a drop-down list. The default value for the "Category" drop-down list is "default", which covers all of the "types" provided by the *Ticketmaster API*. Also, the user can choose the distance (in miles), which is the radius for the search where the center is "Here" (the current location returned from *ip-api.com HTTP API*) or the location listed in the edit box. When the "Here" radio button is selected, the location edit box must be disabled. When the location edit box is selected, it is a required field, and a location string must be entered. The default distance is 10 miles from the chosen location. Use HTML5 "placeholder" to show the string "location" in the location edit box and "10" in the Distance edit box as the initial values. An example is shown in Figure 1.

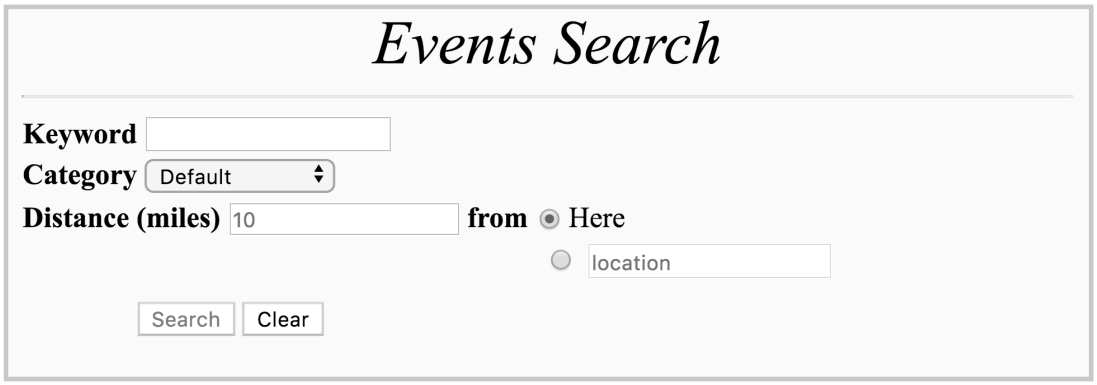


Figure 1(a): Initial Search Screen (Search button disabled)

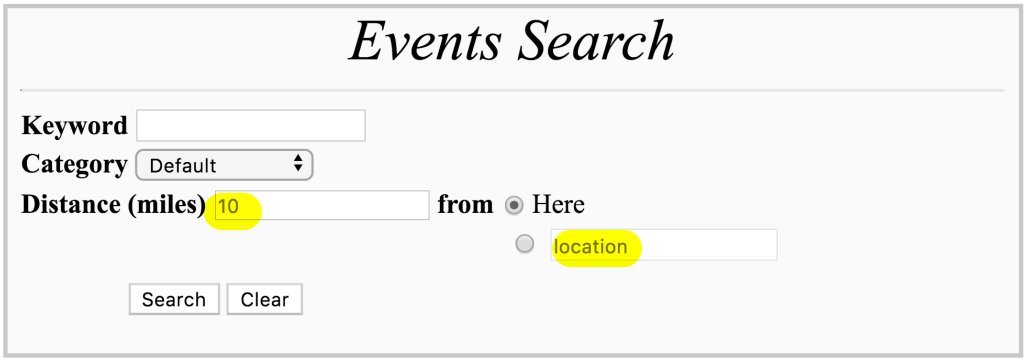


Figure 1(b): Search Screen (after fetched location)

The search form has two buttons:

- **Search** button: The button must be disabled while the page is fetching the user's geolocation and must be enabled once the geolocation is obtained. An example of valid input is shown in Figure 2. Once the user has provided valid input, your client script should send a request to your web server script **event.php** with the form inputs. You can use either GET or POST to transfer the form data to the web server script. The PHP script will retrieve the form inputs, reformat it to the syntax of the API and send it to the *Ticketmaster API* event search service. If the user clicks on the search button without providing a value in the "Keyword" field or "location" edit box, you should show an error "tooltip" that indicates which field is missing. Examples are shown in Figure 3a and 3b.

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

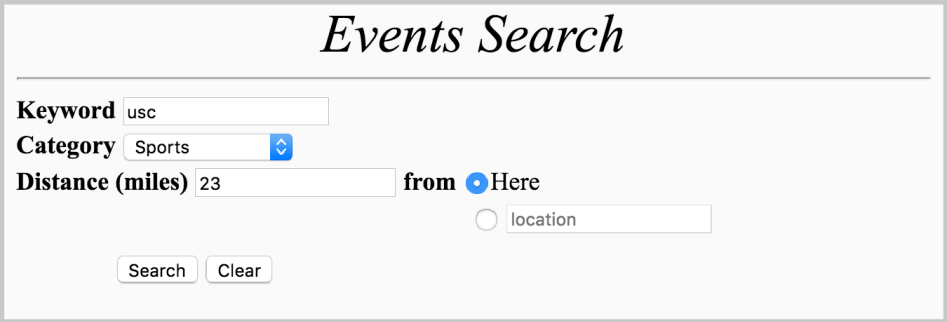


Figure 2: An Example of valid Search

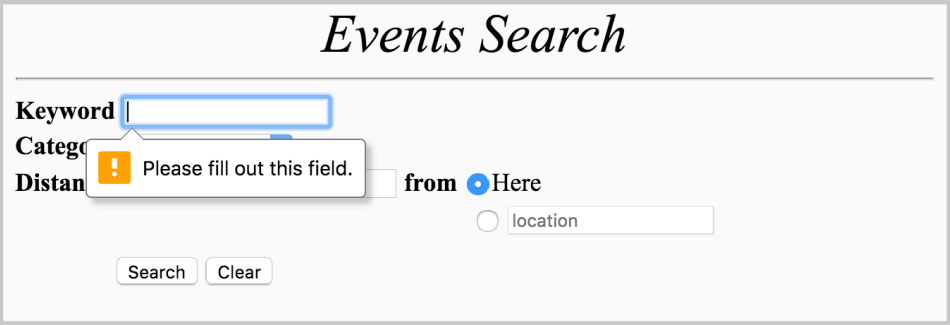


Figure 3(a): An Example of Invalid Search (empty input)

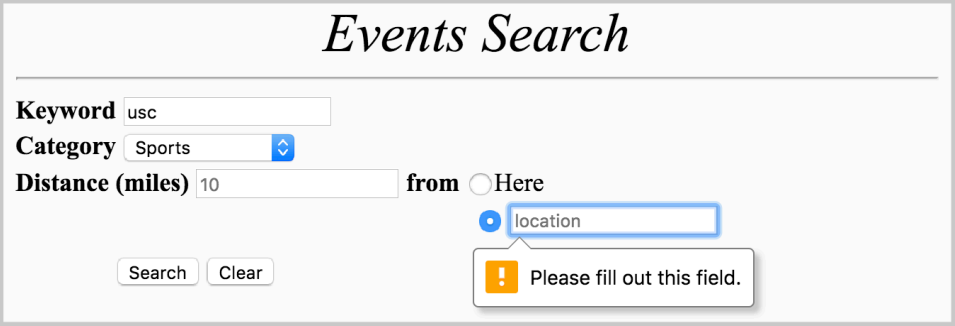


Figure 3(b): An Example of Invalid Search (empty location)

2.2 Displaying Events Results Table

In this section, we outline how to use the form inputs to construct HTTP requests to the Ticketmaster API service and display the result in the web page.

The *Ticketmaster API* is documented here:
https://developer.ticketmaster.com

If the location edit box is selected, the PHP script (i.e., **event.php**) uses the input address to get the geocoding via *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 street address 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 3.1)

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. **Figure 4** shows an example of the JSON object returned in the Google Maps Geocoding API web service response.

Figure 4: A sample result of Google Maps Geocoding query

The latitude and longitude of the address are needed when constructing a restful web service URL to retrieve all entities matching the user query. *Ticketmaster API* "Event Search" service uses *geohash* to represent the address location, instead of *latitude* and *longitude*. We provide an external PHP file to do the conversion. The source code can be found here:

http://csci571.com/hw/hw6/geoHash.txt

download the file to your local directory and change the extension from ".txt" to ".php". Call the function *encode()* by including the external file.

The *Ticketmaster API* Event Search service is documented here:

https://developer.ticketmaster.com/products-and-docs/apis/discovery-api/v2/#search-events-v2

The *Ticketmaster API* Event Search service expects the following parameters:

- ***apikey:*** Your application's API key. This key identifies your application for purposes of quota management.
- ***geoPoint:*** The *geohash* around which to retrieve event information. The geohash is calculated by latitude and longitude values.
- ***radius:*** The distance within which to return event results.
- ***segmentId:*** Filters the results to events matching the specified type id. Only one category may be specified. Leave the field empty means searching in all categories.

Category	***SegmentId***
Music	KZFzniwnSyZfZ7v7nJ
Sports	KZFzniwnSyZfZ7v7nE
Arts & Theatre	KZFzniwnSyZfZ7v7na
Film	KZFzniwnSyZfZ7v7nn
Miscellaneous	KZFzniwnSyZfZ7v7n1

- ***unit:*** Unit of the radius. There are two options, "miles" and "km". Use "miles".
- ***keyword***: A term to be matched against all content that Google has indexed for this place, including but not limited to name, type, and address, as well as customer reviews and other third-party content.

An example of an HTTP request to the *Ticketmaster API* Event Search that searches for the nearby sport events near the University of Southern California within a 10 miles radius is shown below:

```
https://app.ticketmaster.com/discovery/v2/events.json?apikey= YOUR_API_KEY
&keyword=University+of+Southern+California&segmentId=KZFzniwnSyZfZ7v7nE
&radius=10&unit=miles&geoPoint=9q5cs
```

Figure 5 shows an example of the JSON response returned by the *Ticketmaster API* Event Search service response.

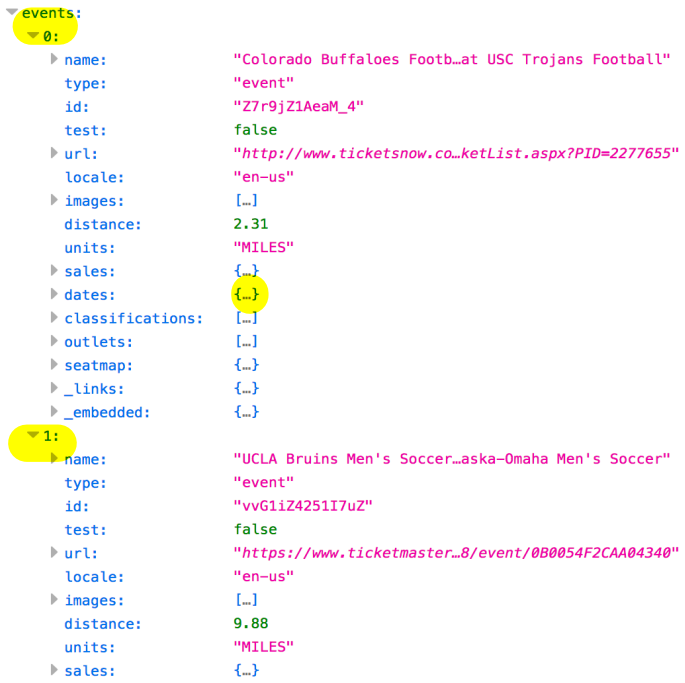
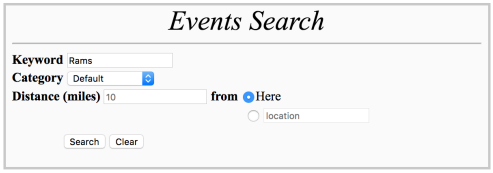


Figure 5: A sample JSON response returned by the *Ticketmaster API* Event Search

The PHP script (i.e., **event.php**) should pass the returned JSON object to the client side or parse the returned JSON and extract useful fields and pass these fields to the client side in **JSON format**. You should use JavaScript to parse the JSON object and display the results in a tabular format. A sample output is shown in Figure 6. The displayed table includes five columns: Date, Icon, Event Name, Genre, and Venue Name. If the API service returns an empty result set, the page should display "No records have been found" as shown in **Figure 7**.



Date	Icon	Event	Genre	Venue
2018-12-30 13:25:00		Los Angeles Rams vs. San Francisco 49ers	Sports	Los Angeles Memorial Coliseum
2018-10-28 13:25:00		Los Angeles Rams vs. Green Bay Packers	Sports	Los Angeles Memorial Coliseum
2018-09-23 13:05:00		Los Angeles Rams vs. Los Angeles Chargers	Sports	Los Angeles Memorial Coliseum
2018-09-27 17:20:00		Los Angeles Rams vs. Minnesota Vikings	Sports	Los Angeles Memorial Coliseum
2018-09-16 13:05:00		Los Angeles Rams vs. Arizona Cardinals	Sports	Los Angeles Memorial Coliseum
2018-11-11 13:25:00		Los Angeles Rams vs. Seattle Seahawks	Sports	Los Angeles Memorial Coliseum
2018-12-16 17:20:00		Los Angeles Rams vs. Philadelphia Eagles	Sports	Los Angeles Memorial Coliseum
2018-12-31		Los Angeles Rams VIP Packages	Sports	Los Angeles Memorial Coliseum
2018-11-19		Los Angeles Rams VIP Packages (Mexico City Game)	Sports	Los Angeles Memorial Coliseum

Figure 6: An Example of a Valid Search result

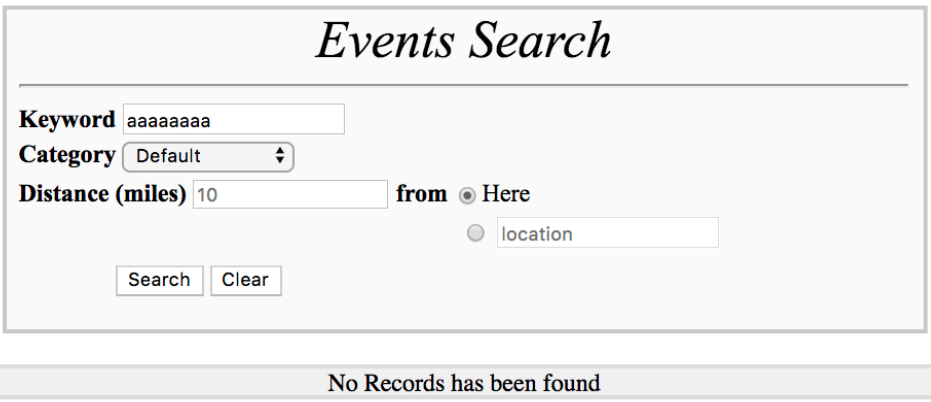


Figure 7: An Example of an Empty Search result

When the search result contains at least one record, you need to map the data extracted from the API result to render the HTML result table as described in Table 1.

HTML Table Column	API service response
Date	The value of the "*localDate*" and "*localTime*" attributes that is part of "*events*" object
Icon	The value of the "*images*" attribute that is part of the "*events*" object.
Event	The value of the "*name*" attribute that is part of the "*events*" object.
Genre	The value of the "segment" attributes that is part of the "*events*" object.
Venue	The value of the "*name*" attribute that is part of the "*venue*" object inside "*events*" object.

Table 1: Mapping the result from Graph API into HTML table

2.3 Displaying Event Details (Event details and Venue details)

In the search result table, if the user clicks on the name of an event, the page should request the detailed information using the *Event Details API* and *Venue Search API*, documented at:

https://developer.ticketmaster.com/products-and-docs/apis/discovery-api/v2/#event-details-v2
https://developer.ticketmaster.com/products-and-docs/apis/discovery-api/v2/#search-venues-v2

To retrieve event details, the request needs two parameters (output should be `JSON`):

- ***id:*** ID of the event
- ***apikey:*** Your application's API key. This key identifies your application for purposes of quota management.

An example of an HTTP request to the *Event Details API* is shown below:

```
https://app.ticketmaster.com/discovery/v2/events/{id}?apikey=YOUR_API_KEY&
```

Figure 8 shows a sample response.

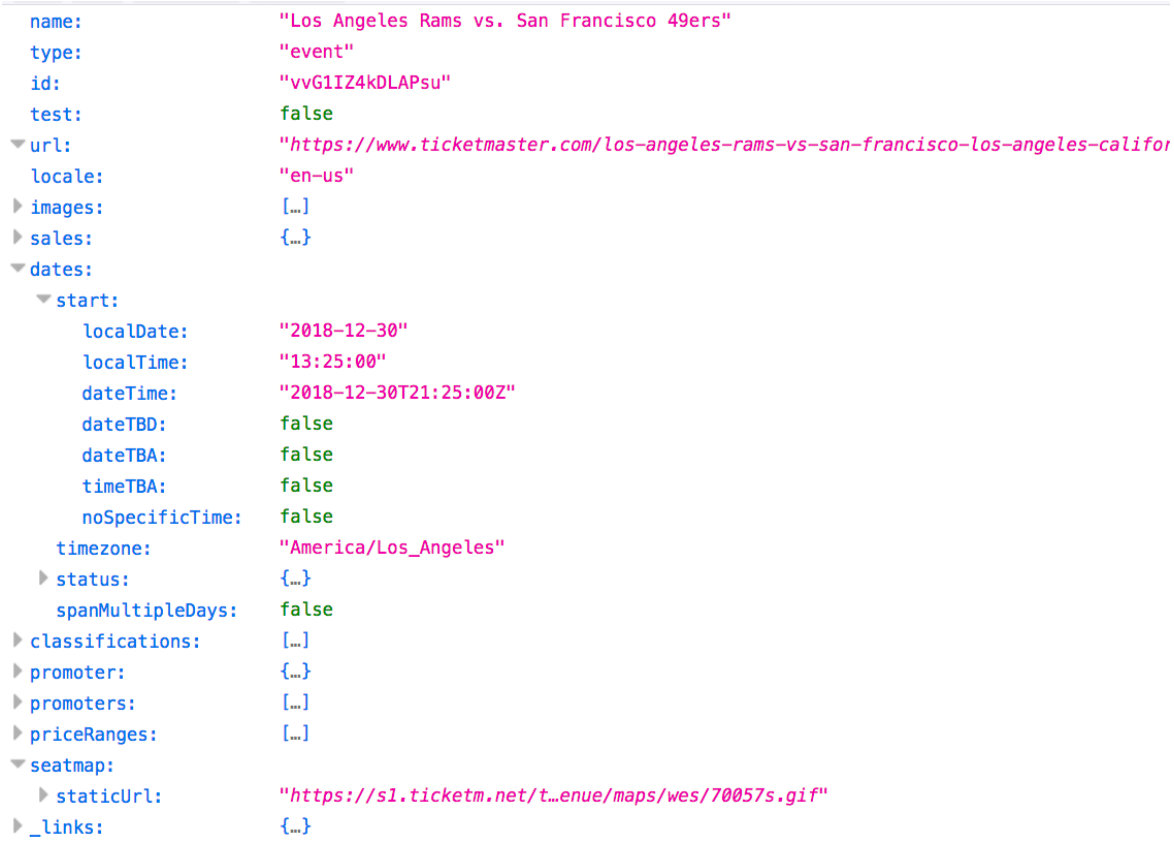


Figure 8: An example of a team photo search response (Keyword: Rams)

The PHP script (i.e., **event.php**) should pass the returned JSON object to the client side or parse the returned JSON and extract useful fields and pass these fields to the client side in **JSON format**. You should use JavaScript to parse the JSON object and display the results in similar format as Figure 9. When click on the artist name (or team name), a page with artist's upcoming events will open in a new tab. When clicking on the "Ticketmaster" link, under "Buy Ticket At", a page to buy tickets online will open in a new page. If the returned JSON stream doesn't contain certain fields, those fields will not appear on the detail page. A sample output is shown in **Figure 9**. Figure 9(a) shows a result with all fields, Figure 9(b) shows a result with missing fields like *"artist", "genre", "price Range",* and *"seat map"*.

Los Angeles Rams vs. San Francisco 49ers

Date

2018-12-30 13:25:00

Artist / Team

Los Angeles Rams | San Francisco 49ers

Venue

Los Angeles Memorial Coliseum

Genres

NFL | Football | Sports | Team | Group

Price Ranges

62 - 275 USD

Ticket Status

Onsale

Buy Ticket At:

Ticketmaster

Figure 9(a): An Example of a Valid Search result

Jen's Test Site

Date

2019-09-30 08:45:00

Venue

NOS Events Center

Ticket Status

Onsale

Buy Ticket At:

Ticketmaster

Figure 9(b): An Example of a Valid Search result with missing fields

When the search result contains at least one field, you need to map the data extracted from the API result to render the HTML result table as described in Table 2.

HTML Key	**API service response**
Date	The value of the "*localDate*" and "*localTime"* attributes that is part of the "*dates*" object
Artist/Team	The value of the *"name"* attribute that is part of the "*attractions*" object, segmented by " | "
Venue	The value of the *"name"* attribute that is part of the "*venue*" object.
Genre	The value of the *"subGenre", "genre", "segment", "subType",* and *"type"* attributes that is part of the "*classifications*" object, segmented by *" | "*
Price Ranges	The value of the "*min*" and *"max"* attributes that

	are part of the "*priceRanges*" object, combined by " -"
Ticket Status	The value of the "*status*" attribute that is part of the "*dates*" object.
Buy Ticket At	The value of the *"url"* attribute.
Seat Map	The value of the "*staticUrl*" attribute that is part of the *"seatmap"* object.

Table 2: Mapping the result from *Event Details API* into HTML Table

To retrieve the venue details, the request to *Venue Search API* needs two parameters (output should be `JSON`):

- ***keyword:*** Name of the venue
- ***apikey:*** Your application's API key. This key identifies your application for purposes of quota management.

An example of an HTTP request to the *Venue Details API* is shown below:

```
https://app.ticketmaster.com/discovery/v2/venues?apikey=YOUR_API_KEY
&keyword=Los%20Angeles%20Memorial%20Coliseum
```

Figure 10 shows a sample response.

Figure 10: An example of a venue detail result (Keyword: Los Angeles Memorial Coliseum)

The PHP script (i.e., **event.php**) should pass the returned JSON object to the client side or parse the returned JSON and extract useful fields and pass these fields to the client side in **JSON format**. You should use JavaScript to parse the JSON object and display the results in similar format as Figure 11 and Figure 12.

There are two parts in Venue details. The first part is a table with venue's location information. When click on in value of "Upcoming Events", a page which upcoming events in this venue will open in new page. If returned JSON file doesn't contain certain fields, the value of those fields will be set as "N/A". A sample output is shown in **Figure 11**.

The second part is venue's photos in tabular format. A sample output is shown in **Figure 12**.

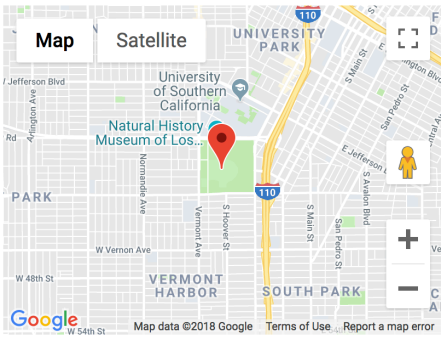
Name	Los Angeles Memorial Coliseum
Map	Walk there Bike there Drive there 
Address	3911 S. Figueroa St
city	Los Angeles, CA
Postal Code	90037
Upcoming Events	Los Angeles Memorial Coliseum Tickets

Figure 11: An Example of a Valid Venue Detail

Figure 12: An Example of a Valid Venue photo(s)

When the search result contains at least one field, you need to map the data extracted from the API result to render the HTML result table as described in Table 3 and Table 4.

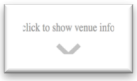
HTML Table Row	**API service response**
Name	The value of the "*name*" attributes
Map	The value of the *"latitude"* and *"longitude"* attribute that is part of the "*location*" object
Address	The value of the *"line1"* attribute that is part of the "*address*" object.
City	The value of the *"name"* attribute of "city" object and "stateCode" attribute of "state" object, connected by a comma.
Postal Code	The value of the "*postalCode*
Upcoming Events	The value of the "*url*" attribute

Table 3: Mapping the result from *Venue Search API* into HTML Table

HTML Table Column	API service response
photos	The value of the "*images*" attribute

Table 4: Mapping the result from *Venue Search API* into HTML Table

The details information includes two sub-sections: Info and Photos which are by default hidden (i.e., collapsed) (as shown in **Figure 13**).

The details information should over-write the result table and needs to be displayed under the search form. When the user clicks on the 

button, the "venue info" sub-section should be expanded, and the "venue photo" sub-section should be hidden (if it is open) and vice versa (see the video for the behavior).

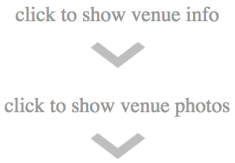


Figure 13: Both the venue info and photos are hidden

The "venue info" sub-section should display the venue info, as shown in **Figure 14**.

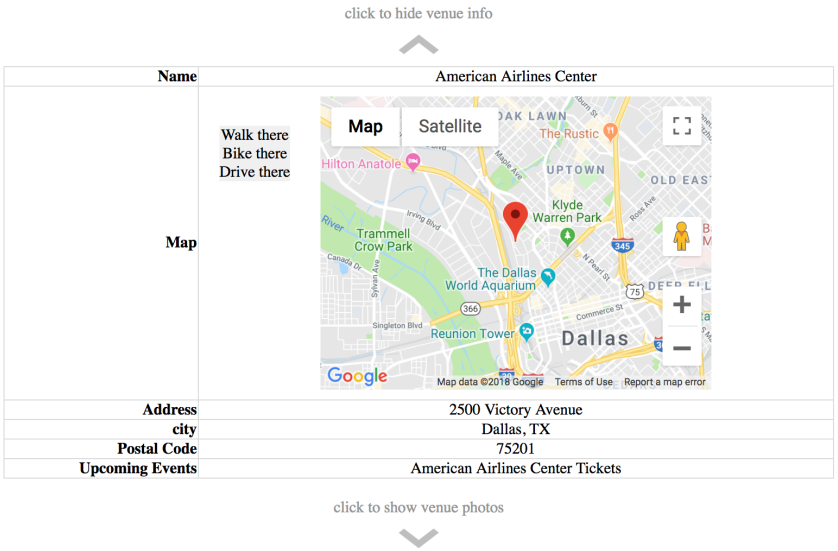


Figure 14: When venue info is clicked, venue photos are hidden.

The "venue photos" sub-section should display all photos (as shown in **Figure 15**) in tabular format.

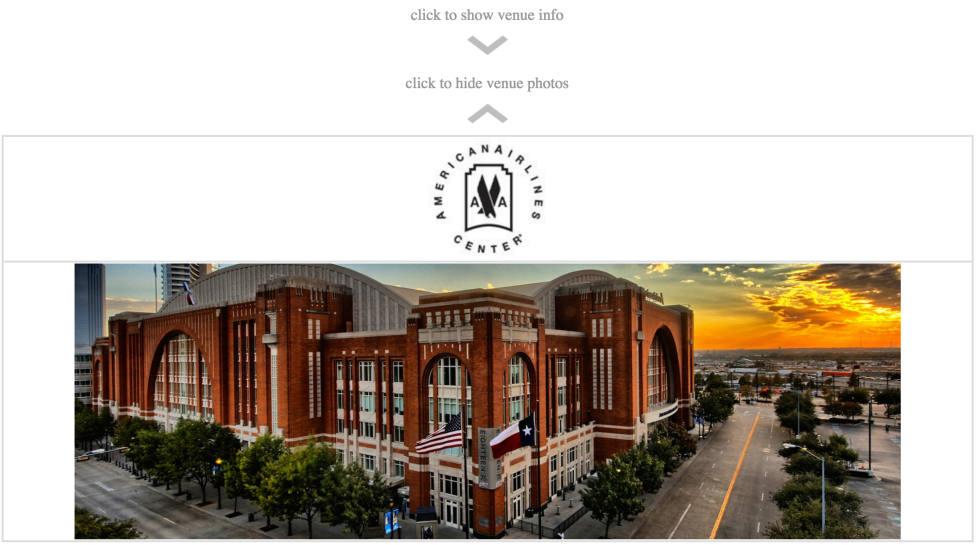


Figure 15: When venue photos are clicked, venue info is hidden.

If the API service returns an empty result set, the page should display "No Venue Info Found" instead of venue info section and "No Venue Photos Found" instead of venue photo section. A sample output is shown in Figure 16 and Figure 17.

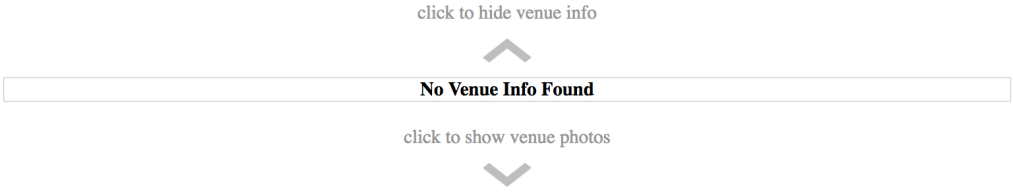


Figure 16: When no info is found.

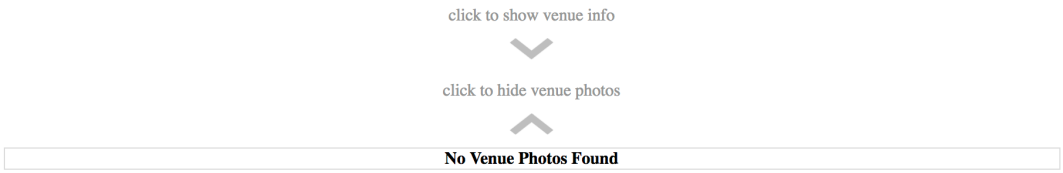


Figure 17: When no photos are found.

Note that:

- **Please DO NOT copy the external php function (geohash.php) into your own php file, otherwise MOSS will find those code similar with others. Include it as an external file. Also do not upload the geohash.php file to GitHub Classroom.**
- **You must use PHP to request all JSON objects except when calling the *ip-api.com API* which should be called on the client side using JavaScript.**

- **Expanding or hiding sub-areas should be implemented using JavaScript and you are not allowed to use JQuery.**

2.4 Displaying Map and Directions

In the search result table, when the corresponding address of a certain record is clicked, a Google Map with a marker of the place should pop up. If the Google Map is already displayed, clicking it will make the map hidden again. The map should not over-write the result table and needs to be displayed right under the address that you click on. Please see the video for the details.

You should use the Google Maps JavaScript Library to construct the map, documented at:

https://developers.google.com/maps/documentation/javascript/adding-a-google-map

A sample is shown in **Figure 18** when selecting the venue "Los Angeles Memorial Coliseum".

Date	Icon	Event	Genre	Venue
2018-12-30 13:25:00		Los Angeles Rams vs. San Francisco 49ers	Sports	Los Angeles Memorial Coliseum
2018-09-27 17:20:00		Los Angeles Rams vs. Minnesota Vikings	Sports	
2018-09-23 13:05:00		Los Angeles Rams vs. Los Angeles Chargers	Sports	
2018-09-16 13:05:00		Los Angeles Rams vs. Arizona Cardinals	Sports	
2018-11-11 13:25:00		Los Angeles Rams vs. Seattle Seahawks	Sports	
2018-10-28 13:25:00		Los Angeles Rams vs. Green Bay Packers	Sports	
2018-12-16 17:20:00		Los Angeles Rams vs. Philadelphia Eagles	Sports	
2018-12-31		Los Angeles Rams VIP Packages	Sports	Los Angeles Memorial Coliseum
2018-11-19		Los Angeles Rams VIP Packages (Mexico City Game)	Sports	Los Angeles Memorial Coliseum

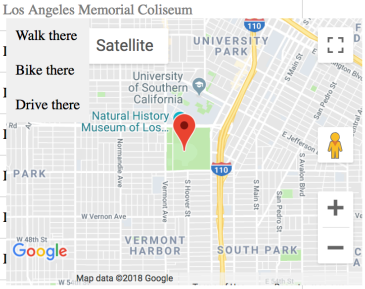


Figure 18: Maps shown when clicking the address of a record.

At the top left corner of the map, there should be a travel mode list (including Walk there, Bike there, and Drive there). If a user clicks on an option, the Google Map with a Marker should be replaced by a Google Map with directions from the location that you choose as the "center point" on the search form to the selected record on a Google Map. A sample is shown in Figure 19 when choosing "Walk there" based on Figure 18. Also watch the video to see the behavior.

You need to use the Direction service to construct the direction route map, documented here:

https://developers.google.com/maps/documentation/javascript/directions

Date	Icon	Event	Genre	Venue
2018-12-30 13:25:00		Los Angeles Rams vs. San Francisco 49ers	Sports	Los Angeles Memorial Coliseum
2018-09-27 17:20:00		Los Angeles Rams vs. Minnesota Vikings	Sports	
2018-09-23 13:05:00		Los Angeles Rams vs. Los Angeles Chargers	Sports	
2018-09-16 13:05:00		Los Angeles Rams vs. Arizona Cardinals	Sports	
2018-11-11 13:25:00		Los Angeles Rams vs. Seattle Seahawks	Sports	
2018-10-28 13:25:00		Los Angeles Rams vs. Green Bay Packers	Sports	
2018-12-16 17:20:00		Los Angeles Rams vs. Philadelphia Eagles	Sports	
2018-12-31		Los Angeles Rams VIP Packages	Sports	Los Angeles Memorial Coliseum
2018-11-19		Los Angeles Rams VIP Packages (Mexico City Game)	Sports	Los Angeles Memorial Coliseum

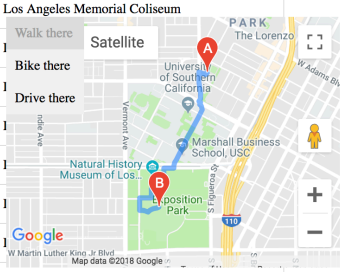
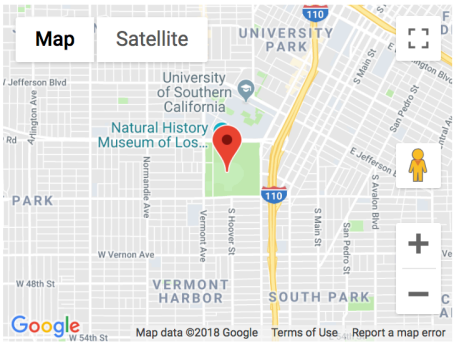


Figure 19: Directions after clicking "Walk there"

In the venue detail table, there is also a Google map to show where the venue is. It has three options like the one in the search table. Every time the venue info section is re-opened, the map should return to the initial state, no direction is shown, only a Google Map with Marker of the location. A sample is shown in Figure 21 when choosing "Bike there" based on Figure 20.

click to hide venue info

Name	Los Angeles Memorial Coliseum
Map	Walk there Bike there Drive there
Address	3911 S. Figueroa St
city	Los Angeles, CA
Postal Code	90037
Upcoming Events	Los Angeles Memorial Coliseum Tickets



click to show venue photos

Figure 20: Maps shown in venue info table.

Figure 21: Directions after clicking "Bike there"

2.5 Saving Previous Inputs

In addition to displaying the results, the PHP page should maintain the provided values. For example, if a user searches for "Keyword: USC, Category: sports, Distance: 15 from Here", the user should see what was provided in the search form when displaying the results. In addition, when clicking on a "Event", the page should display the reviews/photos and keep the values provided in the search form. It follows that you need to keep the whole search box/input fields and buttons even while displaying results/errors.

In summary, the search mechanism to be implemented behaves as follows:

- Based on the query in the search form, construct a web service URL to retrieve the output from the Ticketmaster *API* service.
- Pass the (possibly edited) JSON to the client side and parse JSON using JavaScript.
- Display the events information and venue information in proper format.
- Display the map and directions.

3. Hints

3.1 How to get Ticketmaster API Key

- To get a Ticketmaster API key, please follow these steps:
- Create a new account at:

https://developer-acct.ticketmaster.com/user/register

- Click your name on the right top corner and select "My Apps".

- you can see a Consumer Key

3.2 How to get 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.
- At every Google APIs' guide page, click "Get a key" and select a created project.

Note that you should NOT use a google account associated with a USC e-mail. Preferably use a gmail account.

3.3 Google Maps JavaScript API on demand API Documentation

- Adding a Google Map with a Marker to Your Website:

https://developers.google.com/maps/documentation/javascript/adding-a-google-map

- Directions Service:

https://developers.google.com/maps/documentation/javascript/directions

3.4 Get geolocation using IP-API.com

You need to use *ip-api.com* for searching the geolocation based on IP addresses. An example call looks like:

`http://ip-api.com/json`

The response is a JSON object shown in Figure 22.

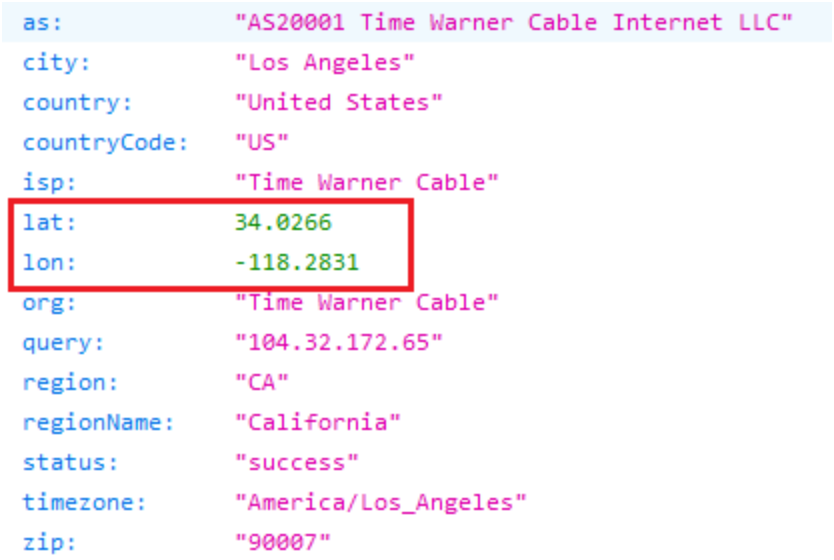


Figure 22: Response from *ip-api.com API*

This article introduces some similar APIs, so you have more choice for your homework 6:

https://ahmadawais.com/best-api-geolocating-an-ip-address/

Use of Freegeoip API is not recommended.

3.4 Parsing JSON-formatted data in PHP

In PHP 5, you can parse JSON-formatted data using the "*json_decode*" function. For more information, please go to http://php.net/manual/en/function.json-decode.php.

You can encode data into JSON-formatted objects using the "*json_encode*" function. For more information, please go to http://php.net/manual/en/function.json-encode.php.

3.5 Read and save contents in PHP

To read the contents of a JSON-formatted object, you can use the "*file_get_contents*" function. To save contents on the server side, you can use "*file_put_contents*" function.

3.6 Deploy PHP file to the cloud (GAE/AWS/Azure)

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

GAE - http://example.appspot.com/event.php
AWS - http://example.elasticbeanstalk.com/event.php
Azure - http://example.azurewebsites.net/event.php

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, **event.php**). This PHP file must be hosted on GAE, AWS or Azure cloud service. Graders will verify that this link is indeed pointing to one of the cloud services.

Also, submit your source code file (**it must be a single .php file, like event.php)** to the GitHub Classroom repository so that it can be graded and compared to all other students' source code via the MOSS code comparison tool. **Do not upload the geoHash.php that we provided to you.**

****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 conflict between Piazza and this description and/or the grading guidelines, **Piazza always rules**.
- You should not use JQuery for Homework 6.
- You **should not call the Ticketmaster APIs directly from JavaScript**, bypassing the Apache/HTTP proxy. Implementing any one of them in JavaScript instead of PHP will result in a **4-point penalty.**
- **The link to the video is**: https://youtu.be/8uCm-1l6pLc