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

1. Objectives

- Getting experience with the PHP programming language;
- Getting experience with the eBay Finding API;
- Getting experience using JSON parsers in PHP and JavaScript; and
- Getting 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, 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 products information using the *eBay APIs*, and the results will be displayed in a tabular format.
The page will also provide product details, seller details and related products.

2.1. Description of the Search Form

A user first opens a page, called **productSearch.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 users must enter a keyword and choose what *Category* of the product they want to search (categories include Art, Baby, Books, Clothing, Shoes & Accessories, Computers/Tablets & Networking, Health & Beauty, Music and Video Games & Consoles) from a drop-down list. The default value for the "Category" drop-down list is "All Categories", which covers all of the "types" provided by the *eBay Finding API*.

The users can also choose the Condition of the product they want based on new, used or unspecified. The default value for Condition is all for which none have to be checked. There is also a Shipping option for the users (Local Pickup and Free Shipping) which they choose as a filter. The default option is all for which neither of the options need to checked. Also, the users have option of enabling nearby search where they can enter the distance (in miles), which is the radius for the search where the center point is "Here" (zip code of current location returned from *ip-api.com HTTP API*) or the zip code entered in the edit box.

Only when the "Enable Nearby Search" is checked, the options to put the distance and select its center point should be enabled (it is initially greyed out and disabled when the page loads). When the "Here" radio button is selected, the zip code edit box must be disabled. When the zip code edit box is selected, it is a required field, and a 5-digit zip code must be entered. The default distance is 10 miles from the chosen location. Use HTML5 "placeholder" to show the string "zip

code" in the *zip code* edit box and "10" in the *distance* edit box as the initial values. An example is shown in **Figure 1**.

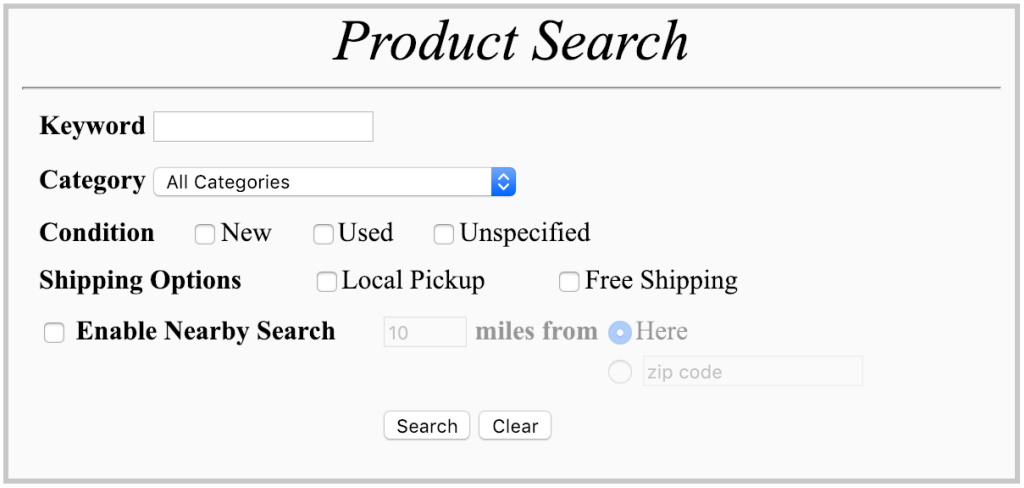


Figure 1(a): Initial Search Screen

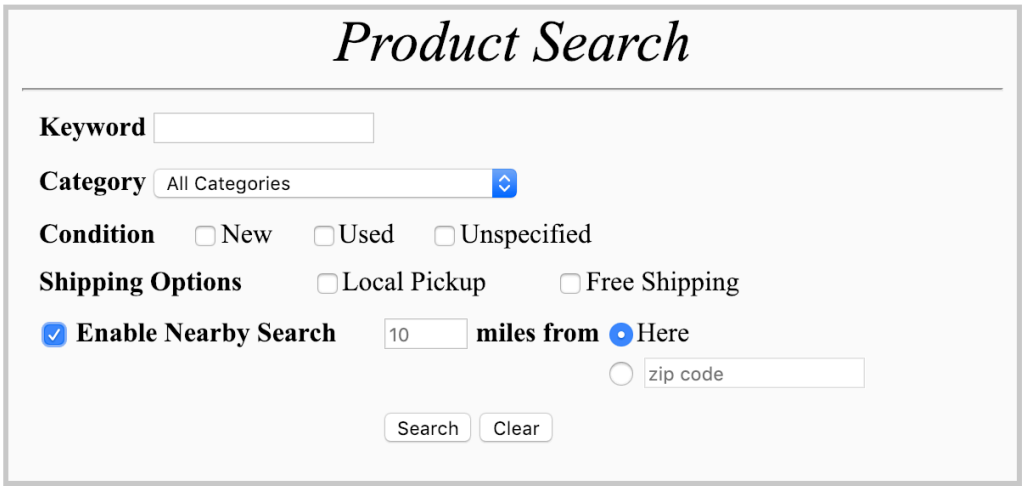


Figure 1(b): Search Screen (after Enabling Nearby Search)

The *Product 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 (written in JavaScript) should send a request to your server script **productSearch.php** with the form inputs. You can use either GET or POST to transfer the form data to the server script. The PHP server script will retrieve the form inputs, reformat them to the syntax of the API and send them to the *eBay Finding API*. If the user clicks on the search button without providing a value in the "Keyword" field or "zip code" edit box, you should show an error "tooltip" that indicates which field is missing. Examples are shown in **Figure 3(a)** and **3(b)**. If the input zip code is invalid, the page should display a corresponding error massage as shown in **Figure 3(c)**.

•

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

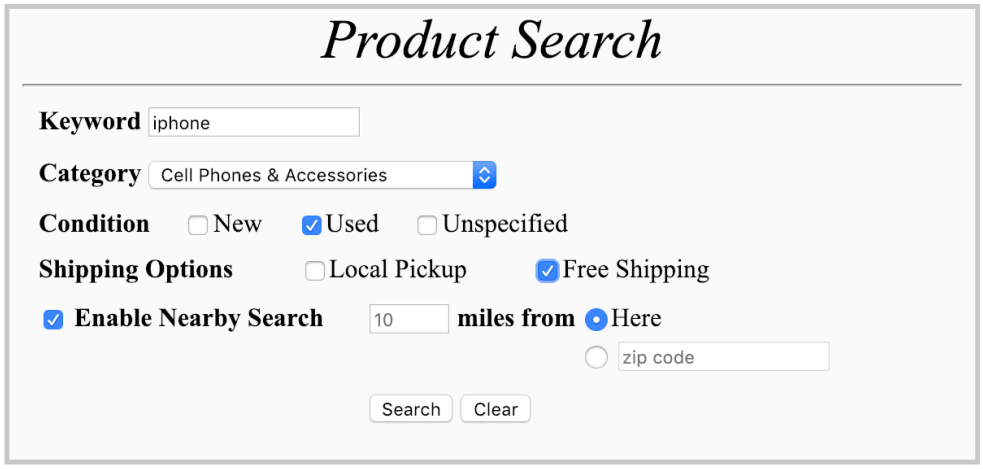


Figure 2: An Example of a valid Search

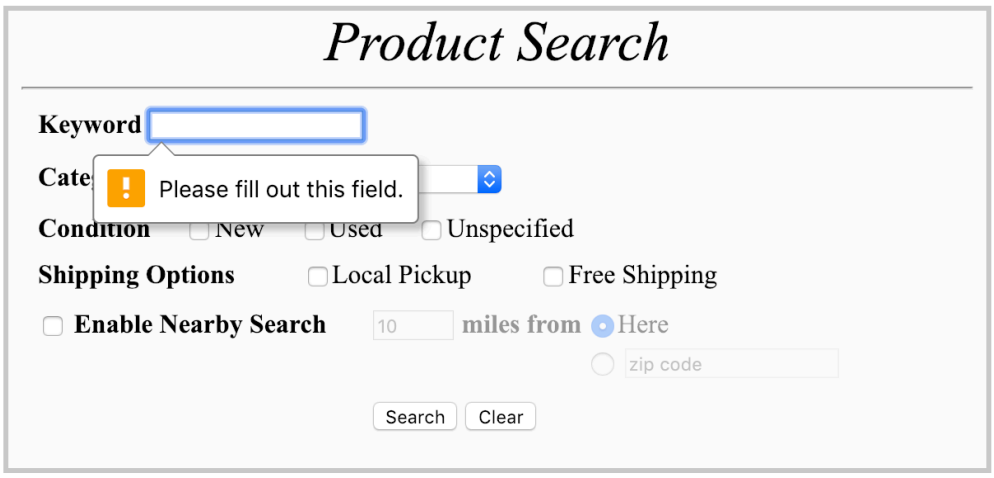


Figure 3(a): An Example of Invalid Search (empty input)

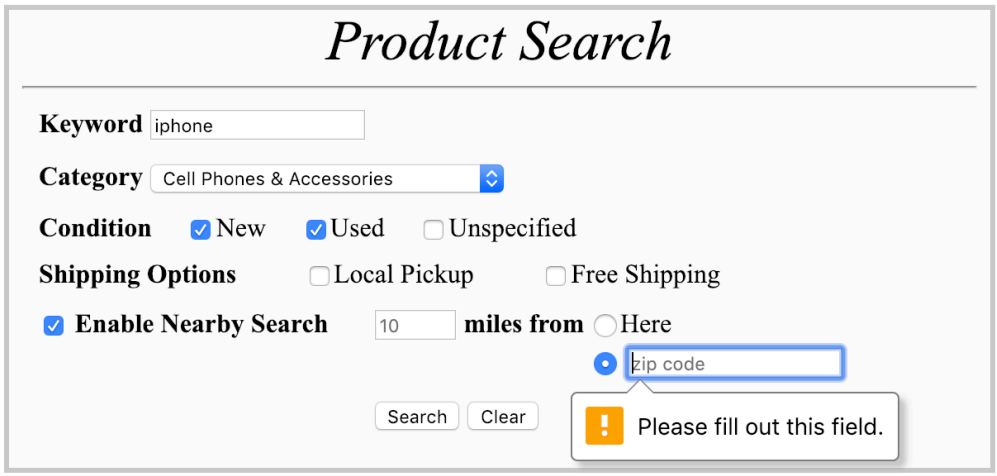


Figure 3(b): An Example of Invalid Search (empty zip code)

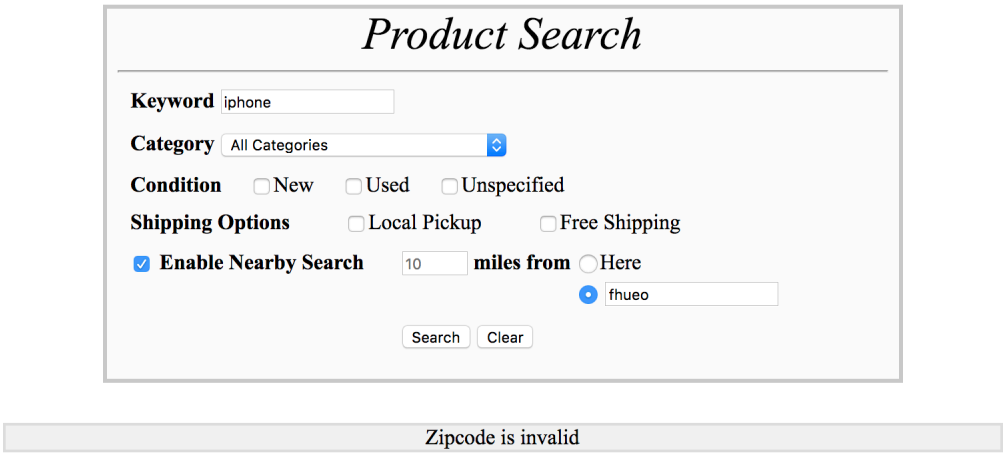


Figure 3(c): An Example of Invalid Search (invalid zip code)

2.2 Displaying Products Results Table

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

The *eBay Finding API* service is documented here:

https://developer.ebay.com/DevZone/finding/Concepts/FindingAPIGuide.html

The *eBay Finding API* service to "make a call" is documented here:

https://developer.ebay.com/DevZone/finding/Concepts/MakingACall.html

The *eBay Finding API* service expects the following parameters:

- ***OPERATION-NAME:*** Set this field to be 'findItemsAdvanced'.
- ***SERVICE-VERSION:*** Set this field to be '1.0.0'.
- ***SECURITY-APPNAME:*** Your application's API key. This key identifies your application for purposes of quota management.
- ***RESPONSE-DATA-FORMAT:*** Set this field to 'JSON'.
- ***REST-PAYLOAD:*** Add it to the API call.
- ***paginationInput.entriesPerPage:*** Set this to 20 to limit the number of results for a specific search.
- ***keywords***: A term to be matched against all content that eBay has indexed for this place, including name of the product to be searched.
- ***categoryId:*** Filters the results to products matching the specified type id. Only one category may be specified (see Table 1). Leaving the field to "-1" means searching in all categories.

Table 1: eBay Finding API - CategoryId Values for Various Categories

Category	***CategoryId***
Art	550
Baby	2984
Books	267
Clothing, Shoes & Accessories	11450
Computers/Tablets & Networking	58058
Health & Beauty	26395
Music	11233
Video Games & Consoles	1249

- There are 2 filters, namely *Condition* and *Shipping Options*. Every filter should have two parameters (name and value). When listing the filters, they should be indexed starting from ZERO. An Example of listing two parameters:

  itemFilter**[0]**.name=*filter1NAME*&itemFilter**[0]**.value=*filter1Value*&itemFilter**[1]**.name=*filter2NAME*&itemFilter**[1]**.value=*filter2Value*.

  If there are multiple values for a filter append it to the url using & operator:

  itemFilter[0].name=*filter1NAME*&itemFilter[0].value=*filter1Value*&itemFilter[0].value=*filter1Value.*
- ***buyerPostalCode:*** Zip code of where the product needs to be searched.
- ***MaxDistance***: By default, it is set to 10. The user can set it to any number which specifies the radius from his location.

An example of an HTTP request to the *eBay Finding API* that searches for the products related to USC within a 10 miles radius from the user's current location is shown below:

```
http://svcs.ebay.com/services/search/FindingService/v1?OPERATION-
NAME=findItemsAdvanced&SERVICE-VERSION=1.0.0&SECURITY-
```

```
APPNAME=[YOUR_APP_ID]&RESPONSE-DATA-FORMAT=JSON&REST-
PAYLOAD&paginationInput.entriesPerPage=20&keywords=usc&itemFilter.name=FreeSh
ippingOnly&itemFilter.value=true&itemFilter.name=LocalPickupOnly&itemFilter.v
alue=true&itemFilter.name=Condition&itemFilter.value=New&itemFilter.value=Use
d&itemFilter.value=Unspecified&buyerPostalCode=90007&MaxDistance=10
```

Figure 5 shows an example of the corresponding JSON response returned by the *eBay Finding API* service response.

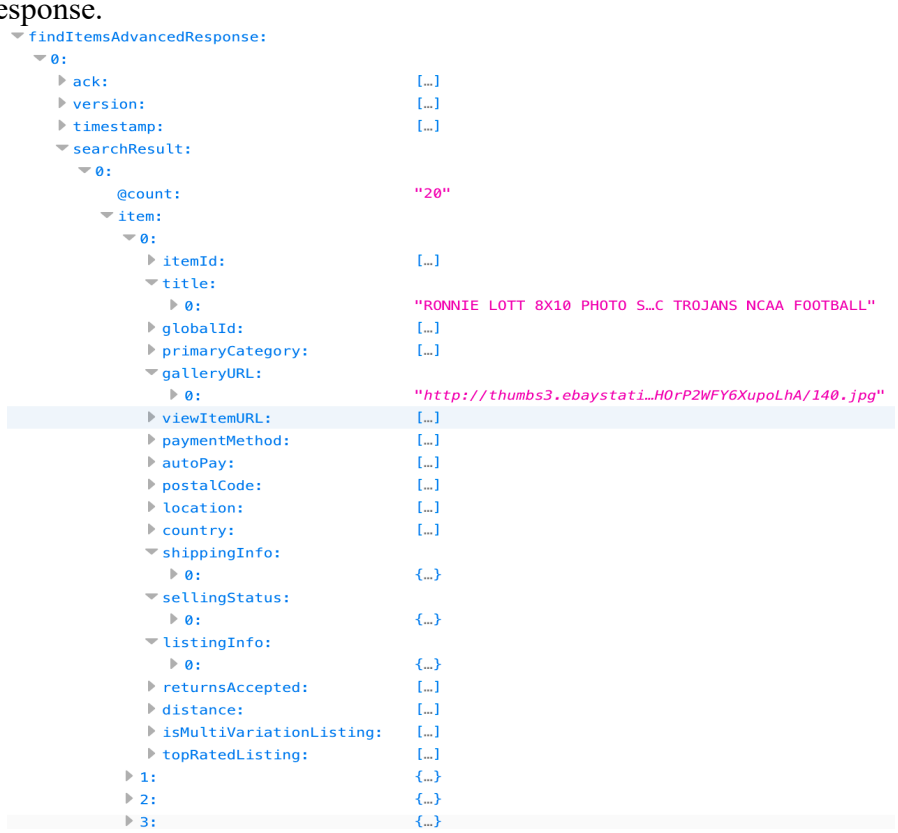
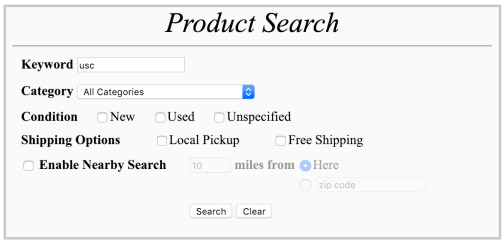


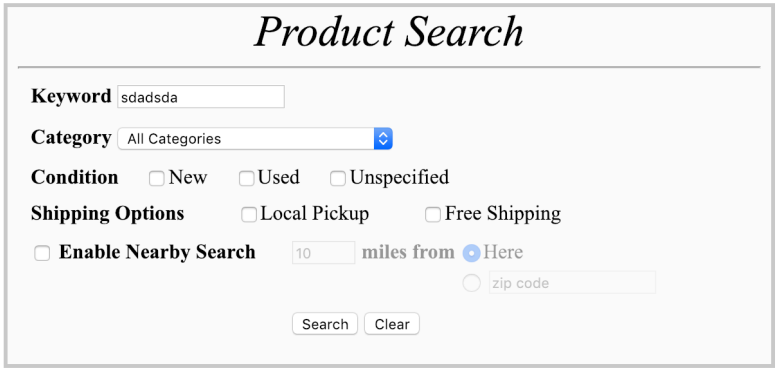
Figure 5: A sample JSON response returned by the *eBay Finding API*

The PHP script (i.e., **productSearch.php**) should pass the returned JSON object to the client side unmodified 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 six columns: Index, Photo, Name, Price, Condition and Shipping Option. If returned JSON object is missing a certain key value pair, display "N/A" in the corresponding field. If the API service returns an empty result set, the page should display "No records have been found" as shown in **Figure 7**.



Index	Photo	Name	Price	Condition	Shipping Option
1		NIKE USC Trojans retro Mark Prior baseball jersey Size Medium	$26.0	Pre Owned	$4.0
2		NEW Nike USC Trojans - Black Poly Short Sleeve Shirt (Multiple Sizes)	$9.86	Brand New	Free Shipping
3		Nike Dri Fit USC Trojans Mens M SS Polo Shirt NWOT	$21.99	Brand New	Free Shipping
4		USC Trojans 3.5" Iron or Sew On Embroidered Patch ~FREE Ship`!!	$6.95	Brand New	Free Shipping
5		USC Trojans license plate frame - chrome university of southern california	$11.99	Pre Owned	$4.99
6		Preowned NIKE USC TROJANS AUTHENTIC Basketball Shorts NCAA Men's Sz Medium	$35.0	Pre Owned	Free Shipping

Figure 6: An Example of a Valid Search result



No Records has been found

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 2**.

Table 2: Mapping the result from Graph API into HTML table

HTML Table Column	**API service response**
Index	The value of the "*index*" is a number ranging from 1 to 20 for all results.
Photo	The value of the "*Photos*" attribute is a part of the "*galleryURL*" object inside the "*item*" object.

Name	The value of the "N*ame*" attribute is part of the "*title*" object inside the "*item*" object.
Price	The value of the "*Price*" attribute is part of the "*currentPrice*" object inside the "*sellingStatus*" object which is part of "*item*" object.
Condition	The value of the "*Condition*" attribute is part of the "*condition*" object inside "*item*" object. Set the value to "*N/A*" if it doesn't exist.
Shipping Option	The value of the "*Shipping Option*" attribute is part of the "*shippingInfo*" object inside "*item*" object. Set the value to "*Free Shipping*" if the "*shippingServiceCost*" is zero or set it to the value in dollars if it is not zero. Set it to "*N/A*" if no information is given.

2.3 Displaying Product Details (Product details and Seller Message and Similar Items)

In the search result table, if the user clicks on the name of a product, the page should make a request for the detailed information using the *eBay shopping API* documented at:

https://developer.ebay.com/devzone/shopping/docs/Concepts/ShoppingAPI_FormatOverview.html
https://developer.ebay.com/devzone/shopping/docs/CallRef/GetSingleItem.html

To retrieve the details of a single item, the request needs the following parameters (output should be `JSON`):

- ***callName:*** Set it to "*GetSingleItem*" to get information for a specific product.
- ***responseencoding:*** Set it to "*JSON*" to get a JSON response.
- ***appid***: Your application's API key. This key identifies your application for purposes of quota management.
- ***siteid***: Set it to '*0'* for siteId purposes.
- ***version:*** Set it to '*967*' for API version purposes.
- ***ItemId:*** It is the "*itemId"* of the product the user clicked.
- ***IncludeSelector:*** Set it to "*Description,Details,ItemSpecifics*" to get required fields for that product.

An example of an HTTP request to the *eBay Shopping API* is shown below:

```
http://open.api.ebay.com/shopping?callname=GetSingleItem&responseencoding=JSON&appid=[APPID]&siteid=0&version=967&ItemID=[ITEMID]&IncludeSelector=Description,Details,ItemSpecifics
```

Figure 8 shows a sample corresponding response.

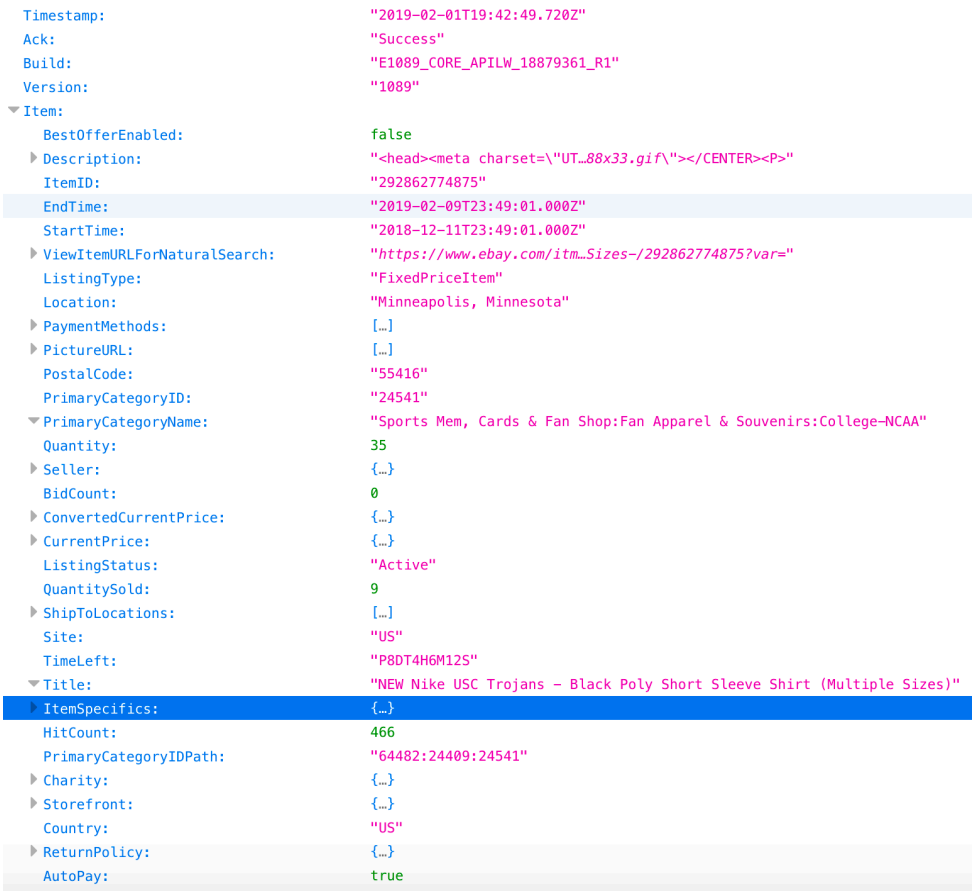


Figure 8: A sample JSON response from eBay Shopping API to get a Single Item.

The PHP script (i.e., **productSearch.php**) should pass the returned JSON object to the client side unmodified 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 similar format as **Figure 9**. 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 such as *"Subtitle"* and other item specific fields.

Item Details

Photo	
Title	Apple iPhone 7 128GB GSM Unlocked Smartphone
Subtitle	90 DAY WARRANTY - FREE SHIPPING AND ACCESSORIES
Price	259.99 USD
Location	Astoria, New York, 11103
Seller	cellfeee
Return Policy(US)	Returns Accepted within 60 Days
Brand	Apple
Model	iPhone 7
MPN	MN9H2LL/A
Storage Capacity	128GB
Network	Unlocked
Manufacturer Color	See color
Style	Touch Screen
Contract	Without Contract
Features	3G Data Capable
Camera Resolution	12.0MP
Operating System	iOS - Apple
Cellular Band	GSM
Screen Size	4.7"
Lock Status	Factory Unlocked
Processor	Quad Core
RAM	2GB
California Prop 65 Warning	www.P65warnings.ca.gov

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

Item Details

Photo	
Title	LG V30 H931 64GB 4G LTE (Unlocked) GSM Smartphone 1-Year Warranty A
Price	249.99 USD
Location	Chatsworth, California, 91311
Seller	qualitycellz
Return Policy(US)	Returns Accepted within 30 Days
Network	Unlocked
Model	V30
Brand	LG
Storage Capacity	64 GB
MPN	H931
Contract	Without Contract
Features	3G Data Capable
Lock Status	Unlocked
Model Number	H931

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 3**.

Table 3: Mapping the result from *eBay Shopping API* into HTML Table

HTML Key	**API service response**
Title	The value of the "*Title*" attribute that is part of the "*Item*" object.
SubTitle	The value of the "*Subtitle*" attribute that is part of the "*Item*" object.
Price	The value of the *"price"* attribute that is part of the "*currentPrice*" object inside the "*Item*" object.
Location	The value of the "*Location*" attribute that is part of the "*Item*" object along with its "*postalcode*" which is also a part of the "*Item*" object.
Seller	The value of the "*UserId*" attribute that is part of the "S*eller*" object inside the "*Item*" object.
Return Policy (US)	The value of the "*ReturnPolicy*" attribute that is part of the "*Item*" object.
ItemSpecifics (Name)	The value of the all the "*NameValueList*" array corresponding to that name inside the "*ItemSpecifics*" object in the "*Item*" object.

Regarding the Seller Message on toggling Click to show seller message:

To retrieve the Seller Message, you should use the IFRAME tag to implement it. The height of that message is dynamic depending on the height of the inner html page embedded in it.

The information to be embedded in the IFRAME tag is the "*Description*" attribute which is present in the "*Item*" object.

An example of a Seller Message for an iPhone as a keyword is given below in **Figure 11**:

Figure 11: An Example of a Valid Seller Message

The details information includes two sub-sections: Seller Message and Similar Items which are by default hidden (i.e., collapsed) (as shown in **Figure 12**). 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 "seller message" sub-section should be expanded, and the "similar items" sub-section should be hidden (if it is open) and vice versa (see the video for the behavior).

Figure 12: Both the seller message and similar items are hidden

The “seller message” sub-section should display the seller message, as shown in **Figure 13**.

Figure 13: When seller message is clicked, similar items are hidden.

The “similar products photos” sub-section should display all photos (as shown in **Figure 14**) in a tabular format.

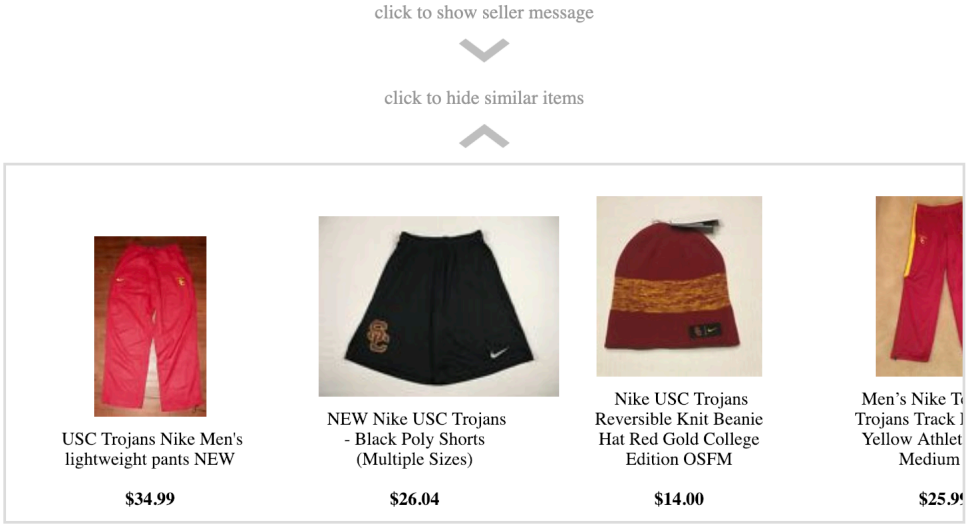


Figure 14: When similar items are clicked, seller message is hidden.

On clicking the button to show similar items, the page should request the detailed information using the *eBay Merchandising API* documented at:

https://developer.ebay.com/devzone/merchandising/docs/Concepts/MerchandisingAPI_FormatOverview.html

To retrieve details of similar items, the request needs the following parameters (output should be `JSON`):

- ***OPERATION-NAME:*** Set it to “*getSimilarItems*” to get information for a related product.
- ***SERVICE-NAME:*** Set it to “*MerchandisingService*” to specify Merchandising API calls.
- ***SERVICE-VERSION***: Set it to 1.1.0 for API version support.
- ***CONSUMER-ID***: Your application's API key. This key identifies your application for purposes of quota management.
- ***RESPONSE-DATA-FORMAT:*** Set it to ‘*JSON*’.

- ***REST-PAYLOAD:*** Add it to the API call.
- ***itemId:*** Set it to "*itemid*" of the product the user clicked.
- ***maxResults:*** Set it to 8 to limit the related products to 8 in a row.

An example of an HTTP request to the *eBay Merchandise API* that searches for similar products to the product clicked initially is shown below:

http://svcs.ebay.com/MerchandisingService?OPERATION-NAME=getSimilarItems&SERVICE-NAME=MerchandisingService&SERVICE-VERSION=1.1.0&CONSUMER-ID=[Your_APP_ID]&RESPONSE-DATA-FORMAT=JSON&REST-PAYLOAD&itemId=292862774875&maxResults=8

Figure 15 shows a sample corresponding response.

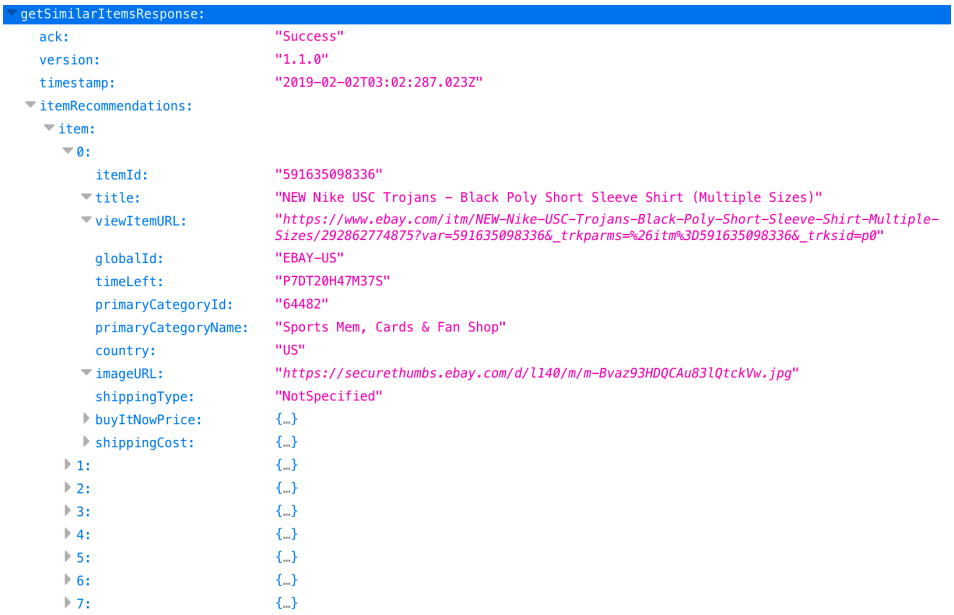


Figure 5: A sample JSON response returned by the *eBay Merchandise API for getting similar items.*

If the API service returns an empty result set, the page should display "No Seller Message Found" instead of the Seller Message section and should display "No Similar Items Found" instead of Similar Items section. Sample outputs are shown in **Figure 16** and **Figure 17**.

Figure 16: When no Seller Message is found.

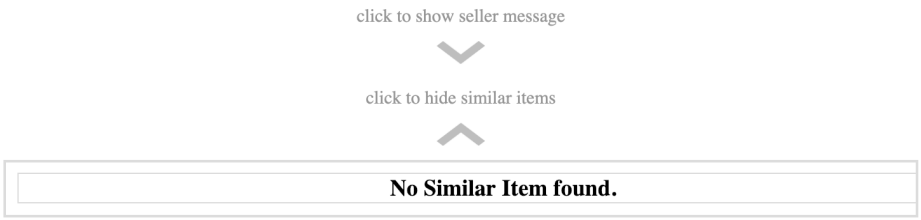


Figure 17: When no Similar Items are found.

Note that:

- **You must use PHP server scripts 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 for this exercise.**

2.4 Saving Previous Inputs

In addition to displaying the results, the web page should maintain the provided values. For example, if a user searches for "Keyword: USC, Category: Books, Condition: Used, Shipping Options: Free Shipping, Enable Nearby Search with 15 miles from Here", the user should see what was provided in the search form when displaying the results. In addition, when clicking on a "Product", the page should display the information about the product, seller message and similar products and keep the values provided in the search form. It follows that you need to keep the whole search box/input fields and buttons unmodified 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 eBay *API* service.
- Pass the (possibly edited) JSON to the client side and parse JSON using JavaScript.
- Display the product information and seller message along with the similar items in the proper format.

3. Hints

3.1 How to get eBay API Key

- To get an eBay API key, please follow these steps given in the PDF file below:

https://developer.ebay.com/DevZone/building-blocks/eBB_Join.pdf

3.2 Get geolocation using IP-API.com

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

http://ip-api.com/json

The response is a JSON object shown in **Figure 22**.

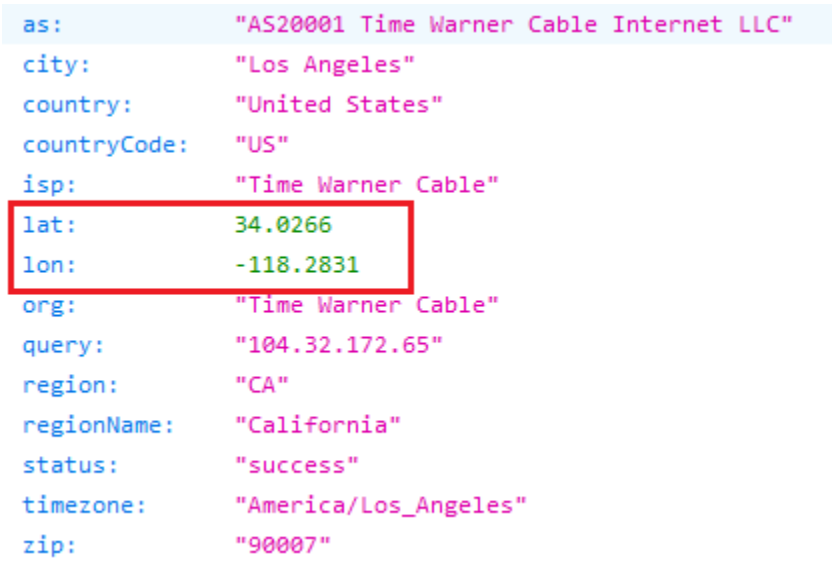


Figure 22: Response from *ip-api.com API*

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.

3.3 Parsing JSON-formatted data in PHP

In PHP 5 and 7, 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.

3.4 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.

3.5 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.co or example.elasticbeanstalk.com or example.azurewebsites.net, the following links will be generated:

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

example in the above URLs will be replaced by your choice of subdomain.

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, **productSearch.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, e.g. productSearch.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.

****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 eBay 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/dfIZgmp6KtU