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

1. Objectives

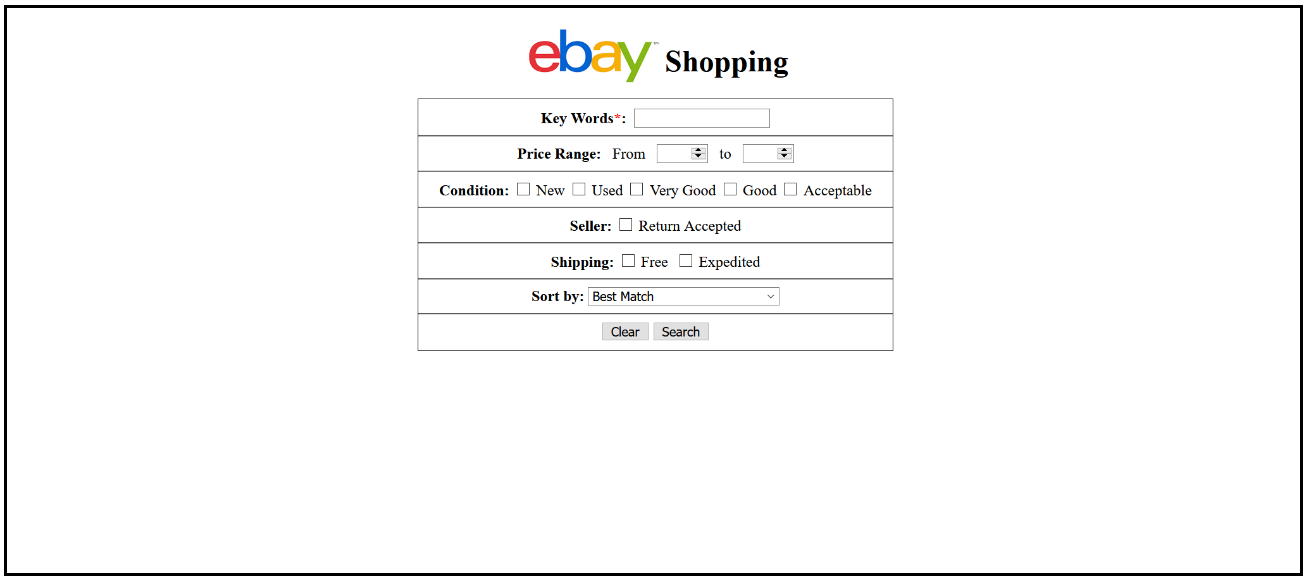
- Get experience with Python programming language and Flask framework.
- Get experience creating web pages using HTML, CSS, JavaScript, HTML DOM and XMLHTTPRequest object
- Get experience with eBay Finding API.
- Get experience using JSON format.

1.1 Cloud Exercise

• The backend of this homework must be implemented in the cloud on GCP using Python.
• See homework 5 for installation of Google Cloud platform.
• See the hints (section 3) at the bottom; a lot of reference material is given to you.
• For Python and Flask kickstart please refer to the Lecture slides on the class website.
• You must refer to the grading guidelines, the video, the specs and Piazza. Styling is graded this time and the points breakup is mentioned in guidelines.

2. Description

In this exercise, you are asked to create a webpage that allows users to search items for sale on eBay.com using their API, and the results will be displayed in a tabular format. Instructions on how to use the API are given in section 3.1. The user first opens a page as shown below in Figure 1, where he/she can enter a query in the keywords textbox.


Shopping

Key Words*:	
Price Range:	From ☐ to ☐
Condition:	☐ New ☐ Used ☐ Very Good ☐ Good ☐ Acceptable
Seller:	☐ Return Accepted
Shipping:	☐ Free ☐ Expedited
Sort by:	Best Match

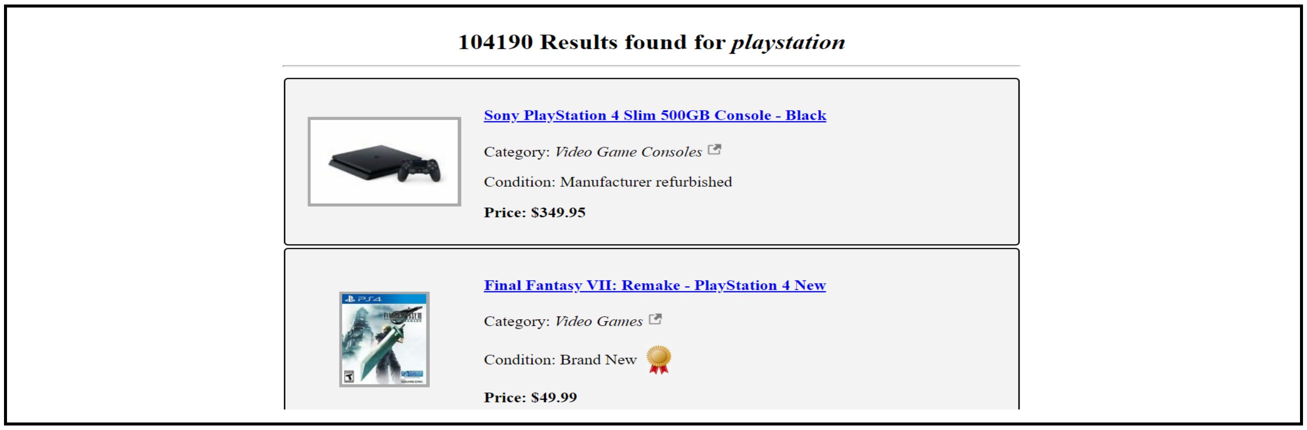
Figure 1: Initial search page

Once the user has provided valid data (a keyword is required), your script will make a request to your web server providing it with the form data that was entered. 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 retrieve the data and send it to the *eBay API Web*

Service “***findItemsAdvanced***”. The API call to eBay is done simply using a URL REST request. For example, if you are just searching for “harry potter” and you want to retrieve the results sorted by ascending price (i.e., from low to high price including shipping cost) in JSON format, the URL will be:

> https://svcs.eBay.com/services/search/FindingService/v1?**OPERATION-NAME=findItemsAdvanced**&SERVICE-VERSION=1.0.0&**SECURITY-APPNAME=**YourEBayAppKey&**RESPONSE-DATA-FORMAT=**JSON&**keywords=**harry%20potter&**sortOrder**= PricePlusShippingLowest

A sample result snippet of 2 results is shown below in **Figure 2.**



104190 Results found for *playstation*

Sony PlayStation 4 Slim 500GB Console - Black

Category: *Video Game Consoles*

Condition: Manufacturer refurbished

Price: $349.95

Final Fantasy VII: Remake - PlayStation 4 New

Category: *Video Games*

Condition: Brand New

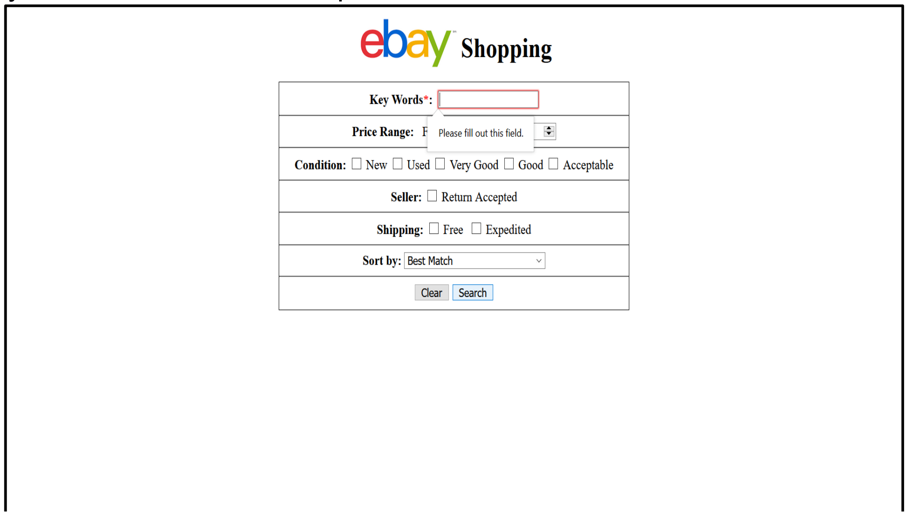
Price: $49.99

Figure 2. Results page

2.1. DESCRIPTION OF THE SEARCH FORM

The following is a brief description of the fields in the search form (see **Figure 1**) which you need to implement in your homework.

1. **Key Words**: This is a text box, which enables the user to search for matching items by entering keywords. This must be a non-empty string. If the **search** button is clicked while this text box is empty, an alert should inform the user that this is a mandatory field (see **Figure 3**), and no query to the server should be performed.



Shopping

Key Words*:

Price Range:

Please fill out this field.

Condition: ☐ New ☐ Used ☐ Very Good ☐ Good ☐ Acceptable

Seller: ☐ Return Accepted

Shipping: ☐ Free ☐ Expedited

Sort by: Best Match

Figure 3. Empty query

2. **Price Range:** These are two boxes, displaying numeric values, which enable the user to assign a price range for the matching items. These can be positive integers or decimal numbers [0.0, ∞). You must ensure that **Minimum Price <= Maximum Price and MinimumPrice, Maximum Price >=0.0.** For specifying the price range**,** it is possible to assign only the lower price range or the higher price range, or both of them, or none of them. If the minimum price > maximum price, an error message should be displayed as shown in **Figure 4**. An error message for negative price value should be displayed as shown in **Figure 5**.

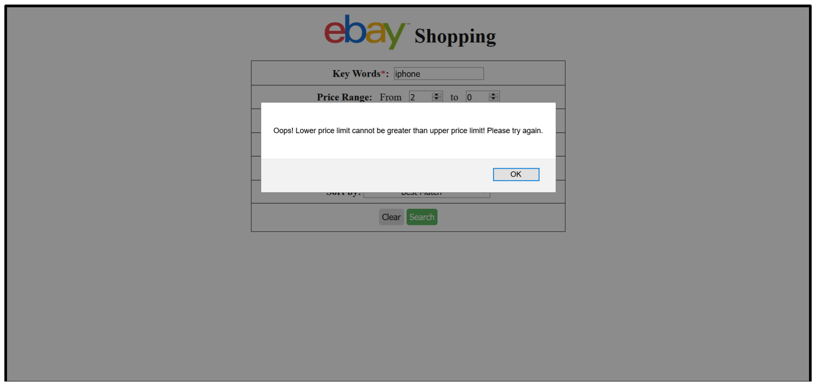

Oops! Lower price limit cannot be greater than upper price limit! Please try again.

Figure 4. Incorrect Price Range

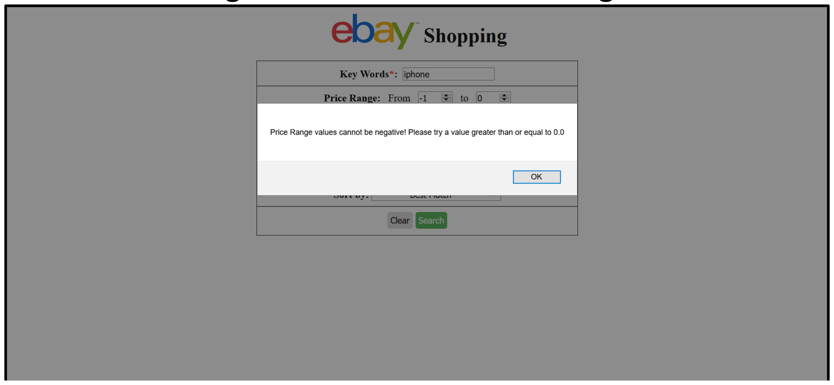

Price Range values cannot be negative! Please try a value greater than or equal to 0.0

Figure 5. Negative Price Value

3. **Condition**: These are five checkboxes "*New*, *Used*, *Very Good*, *Good*, and *Acceptable*" which enable the user to control the item condition retrieved in the results. The user can select one of the options or a combination of them to search for items in specific conditions.
4. **Seller**: Specifies if the user wants only items sold by a seller who accepts returns.
5. **Shipping**: Specify if only "free shipping" items should be returned; if only items available with expedited shipping should be returned or combination of both. Default is not specified which means "any" filter.
6. **Sort By**: This specifies the ordering of the results table, which can be one of:
   a. Best Match - ***THIS IS THE DEFAULT***
   b. Price: highest first
   c. Price + Shipping: highest first
   d. Price + Shipping: lowest first

The search form has two buttons:

1. **SEARCH** button**:** this button validates whether the user provided all the mandatory field (i.e., key words) and verifies the correctness of the provided data in the fields (the price range is valid). The validation should be implemented in a JavaScript function. Then, an HTTP

request is made to your web server in the cloud providing it with the form data that was entered. You need to use **GET** to transfer the form data as query/path parameters to the web server.

2. **CLEAR** button: This button **must** clear the result area, all text fields, uncheck all checkboxes and reset the "sort by" field to its default value mentioned above. The clear operation is done using a JavaScript function.

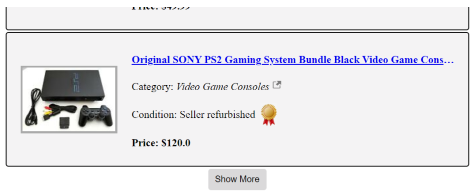
NOTE: In the video, you should notice the change in styling for the two buttons when hovering. Also, notice that the selected sort criteria in the Sort By filter is center aligned.

2.2. DISPLAYING RESULTS

The list of results should be displayed as shown in **Figure 2** along with ta headline indicating the number of results, i.e. the total number of entries on eBay retrieved for a given keyword, and the corresponding filters. The headline is separated from the list of results by a horizontal line. Items are to be stacked vertically (1 item per row). Inside the item card, to the left, is the item's display image. On the top right, the item title is displayed in bold. Below it, the Category (in which the item is tagged) is displayed in the second line, "condition" of the item is displayed in the third line and lastly the "Price" in USD is displayed. Additionally, if applicable, you must also indicate whether the item is top rated to the right of the Category field using the image *topRatedImage.jpg*.

NOTE: If any attribute is missing or empty or blank, do not display the item to the user.

If more than 3 articles are returned in the results, you should display the first 3 items along with the **Show More** button underneath. On clicking the button, a maximum of 7 more items (for a total of 10 items) should be displayed and the **Show More** button changes to **Show Less** as shown in **Figure 6**. After clicking the **Show More** button, all the items will be displayed and a **Show Less** button will be displayed. Clicking the **Show Less** button will again display just 3 results. Additionally, when the **Show More** button is clicked, the page should automatically scroll to the bottom and it should scroll to the top of the page when the **Show Less** button is clicked. The **Show More** and **Show Less** buttons should have the same styling as the clear and search buttons of the form. See the behavior in the video.

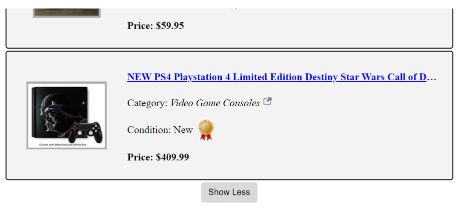


Original SONY PS2 Gaming System Bundle Black Video Game Cons…

Category: *Video Game Consoles*

Condition: Seller refurbished

Price: $120.0



Price: $59.95

NEW PS4 Playstation 4 Limited Edition Destiny Star Wars Call of D…

Category: *Video Game Consoles*

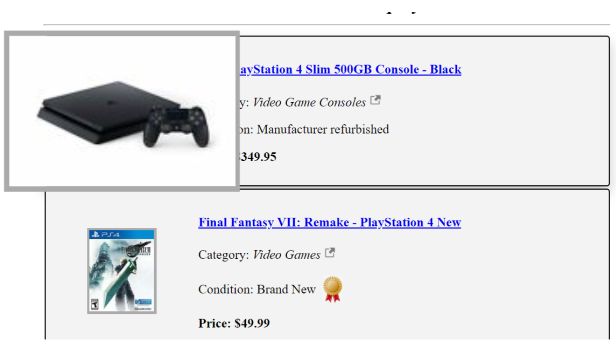
Condition: New

Price: $409.99

Figure 6: Show More and Show Less Buttons

In case the item does not contain images, the eBay server returns a default image with the URI as [https://thumbs1.ebaystatic.com/ pict/04040_0.jpg](https://thumbs1.ebaystatic.com/pict/04040_0.jpg). Replace this image with the **Item Default Image** given to you (See section 5, Images).

Additionally, the item image magnifies when the user hovers on the image and has a transition while magnifying, as shown in figure 7. See the video for the effect.


ayStation 4 Slim 500GB Console - Black

y: *Video Game Consoles*

on: Manufacturer refurbished

349.95

Final Fantasy VII: Remake - PlayStation 4 New

Category: *Video Games*

Condition: Brand New

Price: $49.99

Figure 7: Zoomed in image on mouse hover

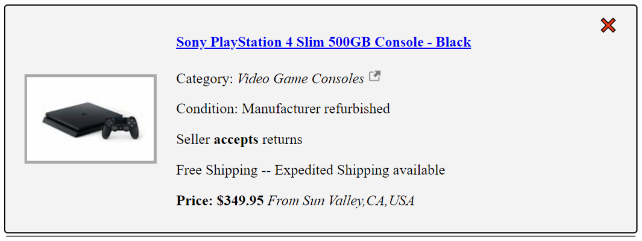
The item is initially in the summary state as shown in **Figure 7**. The fields used from the JSON Response for creating the item summary card and the number of results headlines are shown in **Table 1** below.

Item Information in the Card	**Tags in eBay JSON response**
Total Results Found	*paginationOutput->totalEntries*
Item's image URL	*searchResult->item->galleryURL*
Item Title	*searchResult->item->title*
Item Category tag	*searchResult->item->primaryCategory->categoryName*
eBay product link for redirection	*searchResult->item->viewItemURL*
Condition of the item	*searchResult->item->condition->conditionDisplayName*
Item Top Rated Image (If the current listing is top rated for the given item)	if the value of *searchResult-->item->topRatedListing* is true, display the image topRatedImage.jpg. You can find the image at **URL Link**
Item Price	The price of the item is displayed in the format: *Price: $x (+ $y for shipping)* The item price value is read from *searchResult->item->sellingStatus-> convertedCurrentPrice* The shipping cost is read from *searchResult->item->shippingInfo->shippingServiceCost* The shipping cost phrase (+ *$YY for shipping*) is mentioned if and only if the value of ***shippingServiceCost*** *field* is greater than ZERO

Table 1. Summary Card information

Note: All the numeric values in the JSON response are strings and need to be converted into JavaScript Number type for comparisons.

Each summary card is clickable and results in additional information being displayed as shown in **Figure 8** below.



Sony PlayStation 4 Slim 500GB Console - Black

Category: *Video Game Consoles*

Condition: Manufacturer refurbished

Seller **accepts** returns

Free Shipping -- Expedited Shipping available

Price: $349.95 *From Sun Valley,CA,USA*

Figure 8: Detailed Item card

The mapping between the information populated in the expanded card result and the JSON object is shown in **Table 2** below. For styling of the card, refer to **Figure 8** and the video.

Item Information in the Result Card	**Tags in eBay JSON response**
Item image URL	*searchResult->item->galleryURL*
Item Title	*searchResult->item->title*
Item Category	*searchResult->item->primaryCategory->categoryName*
eBay product link for redirection	*searchResult->item->viewItemURL*
Condition of the item	*searchResult->item->condition->conditionDisplayName*
Item Top Rated Image (If the current listing is top rated for the given item)	if the value of *searchResult-->item->topRatedListing* is true, display the image itemTopRated.jp. You can find the image at **URL LINK**
Seller Accepts return?	*searchResult->item->returnsAccepted* is either true or false. If field is false, show "Seller **does not accept returns"** and "Seller **accepts** returns" otherwise.
Free Shipping available?	If the value of the tag *searchResult-->item->shippingInfo->shippingServiceCost* is equal to 0.0, print "Free Shipping". Otherwise, print "No Free Shipping". (Refer video for styling)
Expedited Shipping available?	*searchResult->item->shippingInfo->expeditedShipping* is either true or false. If the field is true, append "-- Expedited Shipping available" to the Free Shipping status. (Refer Video for styling)
Price	The price is displayed in the format: *Price: $20 (+ $3 for shipping)* The price value is read from *searchResult->item->sellingStatus-> convertedCurrentPrice* The shipping cost is read from *searchResult->item->shippingInfo->shippingServiceCost* The shipping cost phrase (+ *$$$ for shipping*) is mentioned if and only if the value of *shippingServiceCost* is greater than ZERO
Ships from Location	searchResult->item->location

Table 2. Detailed Card Information

In the expanded card, you must indicate, if applicable, if the seller accepts returns.

Shipping information about the item also needs to be displayed:

1. Whether it is a “Free Shipping” item;
2. Whether expedited shipping is available for that item;

In the last line of the details, the price of the item (+ $x for shipping, if applicable) is displayed in **bold**, followed by the sender location for the item in *italics.*

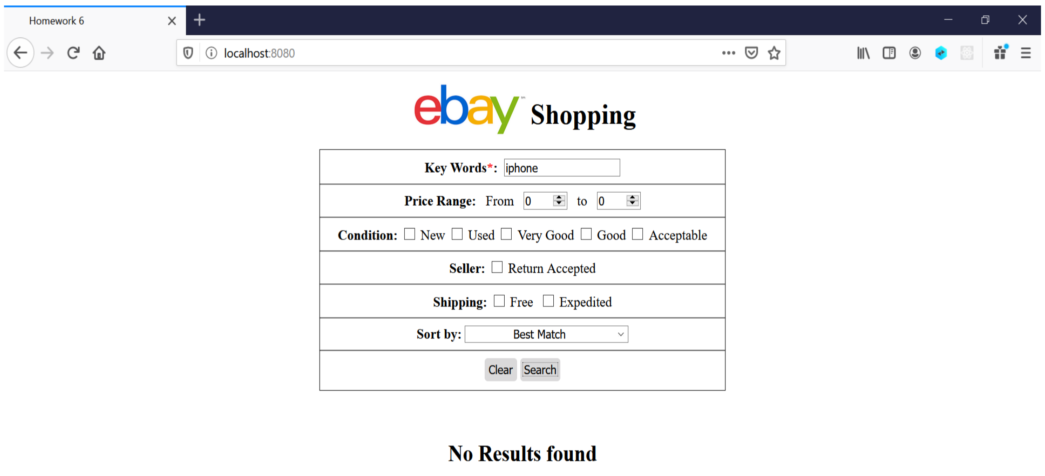
At the top of the results, you should mention the total of the entries matching the search criteria and the key words used in the search (the headline containing the the number of results). An example of the result title is shown in **Figure 2**.

2.3. SAVING PREVIOUS INPUTS

In addition to displaying the results, your page should maintain the provided filter values while displaying the current results. For example, if one searches for “Free Shipping” items for “harry potter”, one should see what was provided in the search form and the corresponding results. Same goes for **all** fields and input types. It follows that you need to keep the whole search box/input fields and buttons, even while displaying results/errors. Also, the keyword text field should provide previous keywords searched as suggestions (default behavior for text fields but can be toggled to enable this feature).

2.4. NO RESULTS

In case where the eBay API returns no items for a specific keyword such as “*alcnkajcsnkacnka*”, you should display “No results found” or any similar message as shown in **Figure 9**.



No Results found

Figure 9: An example when No Results is found

3. How to Use the eBay “findItemsAdvanced” API

3.1. HOW TO CREATE THE eBay App ID

To use any of the eBay APIs, you should first register for membership in the eBay Developer Program at https://developer.ebay.com/signin?tab=register. **The activation of the account takes 1 business day so register early to avoid delays.**

After registration, login into the developer portal and create an application with any title. Once you have a valid application title, complete the registration form by selecting ***individual*** as the account type. Then, create a “Production Keyset”. Do not create more than one Key Set (set of 3 keys, DEVID, AppID, CertID). When you have created your Key Set, your account page may look something like **Figure 10** (ignore the “Sandbox Keys”)

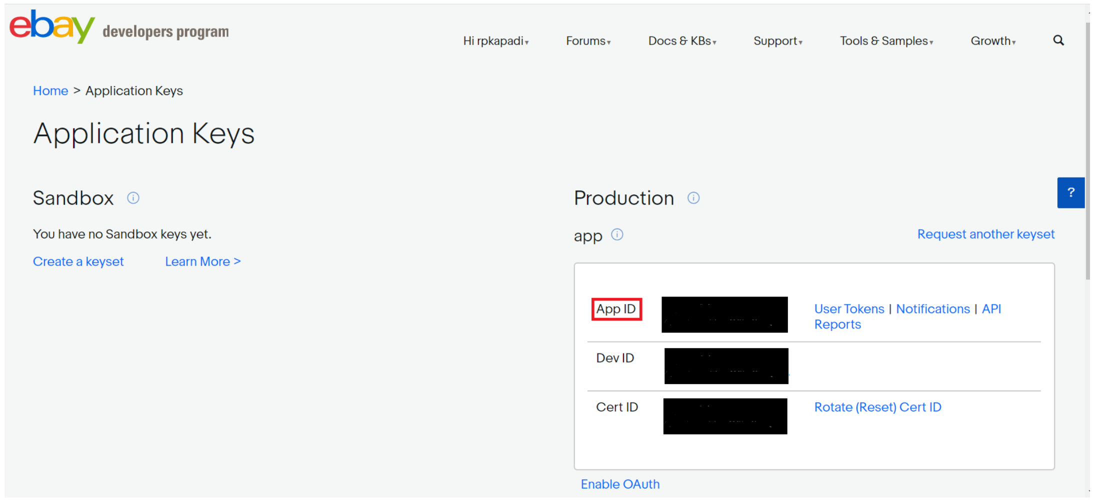


Figure 10: How to Create an eBay App ID

Under the “Application Keys” section, “Production” sub-section, you’ll find your AppID. In **Figure 10**, the application ID is marked in red. When constructing the URL to call the eBay APIs, your Application ID should be provided as a value for the parameter SECURITY-APPNAME. Each AppID only allows 5000 API calls a day. Please ensure you do not exceed the daily usage limit. After reaching the limit, any subsequent calls will fail for 24 hours.

3.2 CONSTRUCTING URL FOR eBay API CALLS.

In this homework, we will use the eBay ***“findItemsAdvanced”*** API. A comprehensive reference about this API is available at:

http://developer.eBay.com/DevZone/finding/CallRef/findItemsAdvanced.html.

Table 3 shows the mapping between the search fields and the URL parameters to call the eBay API. Some of the search fields are handled through ItemFilter. The item filter is specified using two parameters: *itemFilterName* and *itemFilterValue*. The Condition parameter can take multiple values. The reference for the URL parameters can be found at:

https://developer.ebay.com/DevZone/finding/CallRef/extra/fnditmsadvncd.rqst.tmfltr.nm.html

Search Field	**URL parameter in the API Call**
Key words	The name of the query parameter is *Keywords*
Sort by (A drop-down list displaying: 1. Best Match 2. Price: highest first 3. Price + Shipping: highest first 4. Price + Shipping: lowest first)	*sortOrder.* The possible values are: 1. BestMatch 2. CurrentPriceHighest 3. PricePlusShippingHighest 4. PricePlusShippingLowest
Price Range From	The name of the **item filter** is *MinPrice*. The value of the filter is the value of “Price Range From” field. Values have to be decimal and greater than or equal to 0.0. Additional parameters to pass with this filter are **paramName=Currency** and **paramValue=USD**
Price Range To	The name of the **item filter** is *MaxPrice*. The value of the filter is the value of “Price Range To” field. Values have to be decimal and greater than or equal to 0.0. Additional parameters to pass with this filter are **paramName=Currency** and **paramValue=USD**
Seller - Returns Accepted	The name of the **item filter** is *ReturnsAcceptedOnly*. The possible values are true or false.
Shipping - Free Shipping	The name of the item filter is FreeShippingOnly. The value can be true/false.
Shipping - Expedited shipping available	The name of the item filter is ExpeditedShippingType. One of the allowed values is: Expedited. If checked, pass Expedited value else do not apply this filter in the API call.
Condition (a set of check boxes: New, Used, Very Good, Good, and Acceptable)	The name of the **item filter** is Condition. This filter defaults to OR logic if multiple values are provided. The possible values are: 1. 1000 (means New) 2. 3000 (means Used) 3. 4000 (means Very Good) 4. 5000 (means Good) 5. 6000 (means Acceptable) If no value is checked, do not apply this filter in the API call.

Table 3: Mapping between the search fields and the URL parameters

In addition to the parameters mentioned in the above table, there are five parameters which should be included in every call. The name of these parameters and their values are listed below:

- OPERATION-NAME=findItemsAdvanced
- SERVICE-VERSION=1.0.0
- SECURITY-APPNAME=**YourEBayAppKey**
- RESPONSE-DATA-FORMAT=JSON
- REST-PAYLOAD

Every item filter should have two parameters (name and value). When listing the filters, they should be indexed starting from ZERO. An Example of listing three parameters:

itemFilter**(0)**.name=*filter1NAME***&**itemFilter**(0)**.value=*filter1Value*&itemFilter**(1)**.name=*filter2NAME*&
&itemFilter**(1)**.value=*filter2Value*&itemFilter**(2)**.name=*filter3NAME*&itemFilter**(2)**.value=*filter3Value*

If the filter is assigned multiple values, the values should be mentioned in a list of parameters and the list of the values should be indexed from ZERO. For example, in order to filter items based on their *Condition* to be *NEW* or *USED* or *Very Good* can be written as:

itemFilter**(X)**.name=Condition**&**itemFilter**(X)**.value**(0)**=1000**&**itemFilter**(X)**.value**(1)**=3000**&**itemFilter**(X)**.value**(2)**=4000

For more information about filters, you can read this page:

http://developer.ebay.com/DevZone/finding/CallRef/types/ItemFilterType.html

An example URL constructed from the parameters will look similar to:

https://svcs.ebay.com/services/search/FindingService/v1?OPERATION-NAME=findItemsAdvanced&SERVICE-VERSION=1.0.0&SECURITY-APPNAME=**YourAppID**&RESPONSE-DATA-FORMAT=JSON&REST-PAYLOAD&keywords=iphone&paginationInput.entriesPerPage=10&sortOrder=BestMatch&itemFilter(0).name=MaxPrice&itemFilter(0).value=25&itemFilter(0).paramName=Currency&itemFilter(0).paramValue=USD

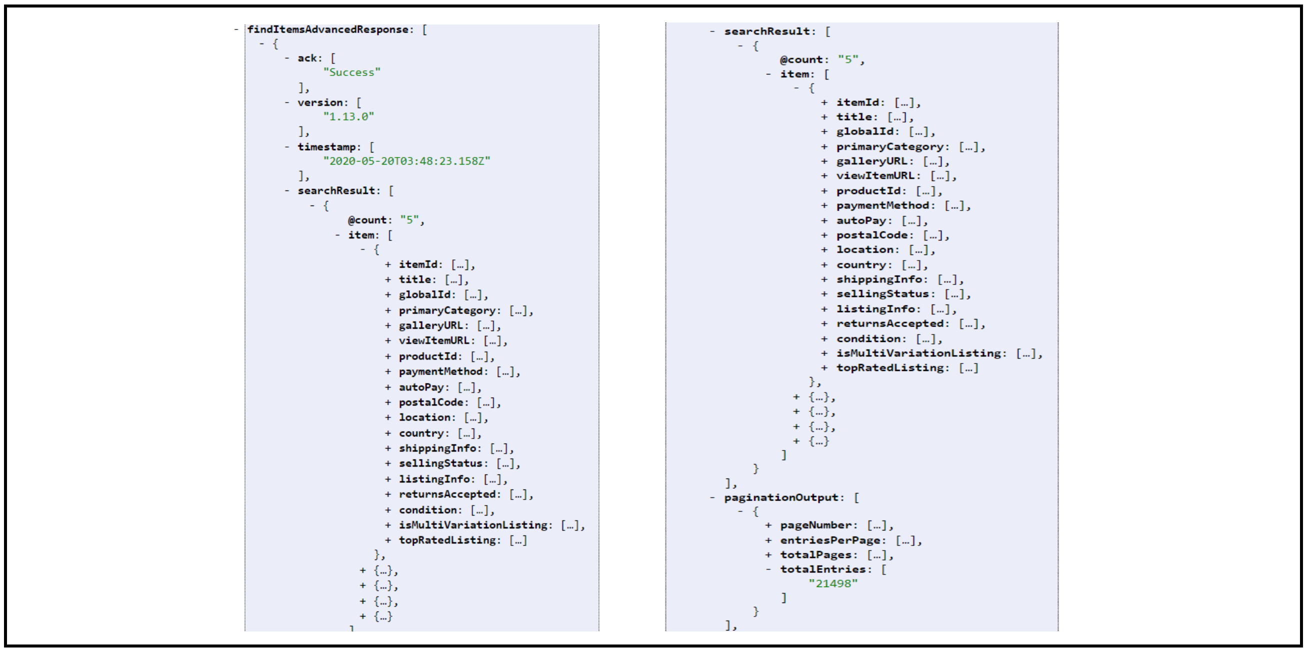
In the example URL above, the **paginationInput.entriesPerPage** parameter can be used to reduce the number of items returned by the eBay API. This parameter is **optional** and if omitted, the eBay API will return a maximum of 100 items by default. You need to only display a maximum of 10 items on the web page but might need to request more items to ensure having 10 items even when some listings have to be removed due to missing fields or errors.

3.3 PARSING JSON RESULTS

To view a sample JSON response from the *findItemsAdvanced* API, just copy and paste the URL below and hit Enter in your **Firefox** browser.

https://svcs.ebay.com/services/search/FindingService/v1?OPERATION-NAME=findItemsAdvanced&SERVICE-VERSION=1.0.0&SECURITY-APPNAME=**YourAPIKey**&RESPONSE-DATA-FORMAT=JSON&REST-PAYLOAD&keywords=iphone&paginationInput.entriesPerPage=25&sortOrder=BestMatch&itemFilter(0).name=MaxPrice&itemFilter(0).value=25&itemFilter(0).paramName=Currency&itemFilter(0).paramValue=USD&itemFilter(1).name=MinPrice&itemFilter(1).value=10&itemFilter(1).paramName=Currency&itemFilter(1).paramValue=USD&itemFilter(2).name=ReturnsAcceptedOnly&itemFilter(2).value=false&itemFilter(3).name=Condition&itemFilter(3).value(0)=2000&itemFilter(3).value(1)=3000

Please note that the SECURITY-APPNAME parameter is developer-specific as described in section 3.2. **Figure 11** below is an example of the high level JSON response of the *findItemsAdvanced* API.



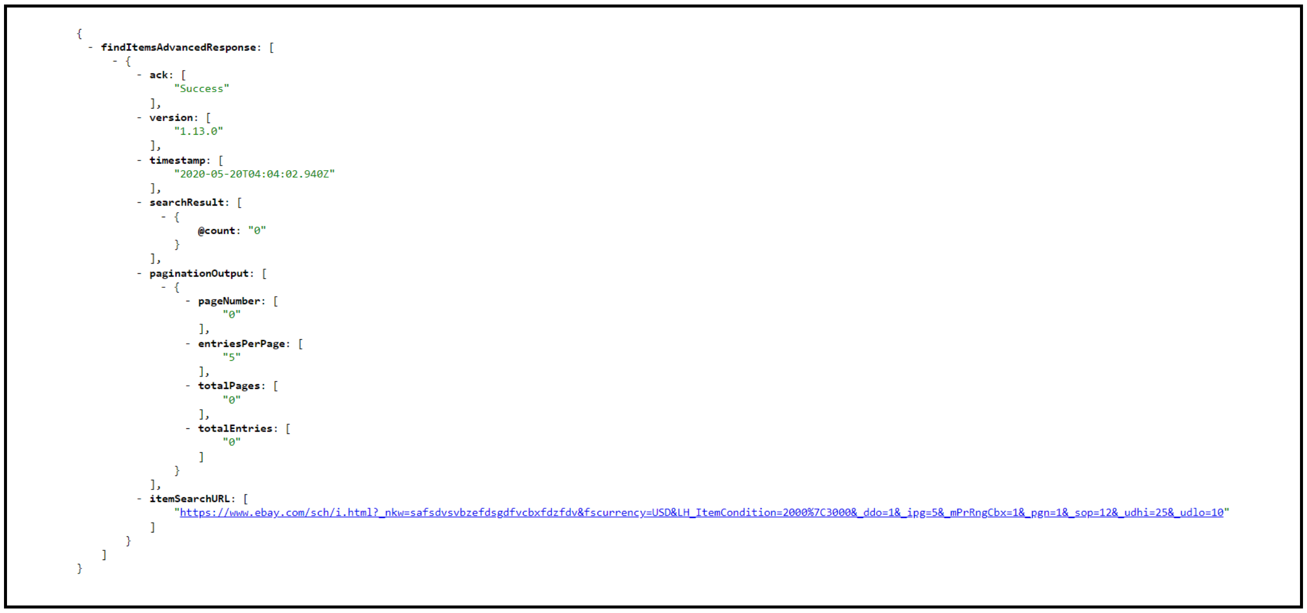
```
- findItemsAdvancedResponse: [
  - {
      - ack: [
            "Success"
        ],
      - version: [
            "1.13.0"
        ],
      - timestamp: [
            "2020-05-20T03:48:23.158Z"
        ],
      - searchResult: [
          - {
                @count: "5",
              - item: [
                  - {
                      + itemId: […],
                      + title: […],
                      + globalId: […],
                      + primaryCategory: […],
                      + galleryURL: […],
                      + viewItemURL: […],
                      + productId: […],
                      + paymentMethod: […],
                      + autoPay: […],
                      + postalCode: […],
                      + location: […],
                      + country: […],
                      + shippingInfo: […],
                      + sellingStatus: […],
                      + listingInfo: […],
                      + returnsAccepted: […],
                      + condition: […],
                      + isMultiVariationListing: […],
                      + topRatedListing: […]
                    },
                  + {…},
                  + {…},
                  + {…},
                  + {…}
                ]
```

```
- searchResult: [
  - {
        @count: "5",
      - item: [
          - {
              + itemId: […],
              + title: […],
              + globalId: […],
              + primaryCategory: […],
              + galleryURL: […],
              + viewItemURL: […],
              + productId: […],
              + paymentMethod: […],
              + autoPay: […],
              + postalCode: […],
              + location: […],
              + country: […],
              + shippingInfo: […],
              + sellingStatus: […],
              + listingInfo: […],
              + returnsAccepted: […],
              + condition: […],
              + isMultiVariationListing: […],
              + topRatedListing: […]
            },
          + {…},
          + {…},
          + {…},
          + {…}
        ]
    }
],
- paginationOutput: [
  - {
      + pageNumber: […],
      + entriesPerPage: […],
      + totalPages: […],
      - totalEntries: [
            "21498"
        ]
    }
],
```

Figure 11: High-level successful response JSON

In **Figure 11**, the item tags have been minimized. Each value for a field is enclosed inside an array in the JSON response. The mandatory requirement for the homework is to reduce the number of items returned to 10 and only those items that have all required fields be sent to the web page. You can send the additional fields which are not used by the webpage as well in your response. However, the preferred approach will be to parse the Ebay API JSON for validation and retrieval of data and then create a new JSON in python that will only contain the required information.

The case where no results are found is shown in Figure 12 below. You can then query the value of *paginationOutput->totalEntries. If* the value of *totalEntries* is zero, display “no results were found” in the results area.



```
{
  - findItemsAdvancedResponse: [
      - {
          - ack: [
                "Success"
            ],
          - version: [
                "1.13.0"
            ],
          - timestamp: [
                "2020-05-20T04:04:02.940Z"
            ],
          - searchResult: [
              - {
                    @count: "0"
                }
            ],
          - paginationOutput: [
              - {
                  - pageNumber: [
                        "0"
                    ],
                  - entriesPerPage: [
                        "5"
                    ],
                  - totalPages: [
                        "0"
                    ],
                  - totalEntries: [
                        "0"
                    ]
                }
            ],
          - itemSearchURL: [
                "https://www.ebay.com/sch/i.html?_nkw=safsdvsvbzefdsgdfvcbxfdzfdv&fscurrency=USD&LH_ItemCondition=2000%7C3000&_ddo=1&_ipg=5&_mPrRngCbx=1&_pgn=1&_sop=12&_udhi=25&_udlo=10"
            ]
        }
    ]
}
```

Figure 12: No Results JSON

The item tags from which you should extract information for display are given in Table 1. Your **Python program should parse the resulting JSON and extract the information for all items**.

All information specific to an item will be within its item element as shown above. The root “item” tag for will be findItemsAdvancedResponse->searchResult->item[index].

4. Hints

- For Flask/Python Backend, refer:
  - For setting up a server - https://flask.palletsprojects.com/en/1.1.x/
  - For making requests from Python to the eBay APIs, use the Requests module - https://requests.readthedocs.io/en/master/
- Styling hints
  - Button styling - https://www.w3schools.com/css/css3_buttons.asp
  - Shadow on a card - https://www.w3schools.com/howto/tryit.asp?filename=tryhow_css_cards2
  - For close button - https://wesbos.com/times-html-entity-close-button
- Flask/Python
  1. Flask should be used to implement RESTful services to eBay
  2. Flask templates should not be used
  3. Flask send_static_file() or send_from_directory() should be used to send "static" HTML, CSS and JavaScript
- Google Cloud Platform (GCP)
  1. Follow the steps mentioned in Homework 5 to deploy your web application on GCP.

5. Images

eBay Logo: https://www.csci571.com/hw/hw6/images/eBayLogo.png
Top Rate Image: https://www.csci571.com/hw/hw6/images/topRatedImage.png
Item Default Image*:* https://www.csci571.com/hw/hw6/images/ebay_default.jpg
Redirect: https://www.csci571.com/hw/hw6/images/redirect.png

5. Files to Submit

- In your course “Table of HomeWorks” page, you should update the Homework 6 link to refer to your new initial web search page for this exercise (for example, index.html).
- Your files must be hosted on GCP.
- Also, add a link to the hardcoded endpoint which returns the result for the filters mentioned in the grading guidelines, below the “Homework 6” hyperlink.
- Graders will verify that this link is indeed pointing to one of the cloud services.

Also, save your source code files so that they can be shared with the graders in case of plagiarism concerns and compared to other students’ source code files. **You do not need to submit any source code unless asked to do so.**

****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** or **Bootstrap** for Homework 6.
• You should not call the **eBay APIs directly from JavaScript, bypassing the Python proxy**. Implementing the eBay API call in JavaScript instead of Python will result in a **<u>4-point penalty</u>**.