Homework 8: Ajax, JSON, Responsive Design and Node.js

Product Search

(AJAX/JSON/HTML5/Bootstrap/Angular /Node.js/Cloud Exercise)

1. Objectives

- Get familiar with the AJAX and JSON technologies.
- Use a combination of HTML5, Bootstrap and Angular on client side.
- Use Node.js on server side.
- Get familiar with Bootstrap to enhance the user experience using responsive design.
- Get hands-on experience of Google Cloud Platform App Engine
- Learn to use popular APIs such as eBay API,

2. Background

2.1 AJAX and JSON

AJAX (Asynchronous JavaScript + XML) incorporates several technologies:

- Standards-based presentation using XHTML and CSS;
- Result display and interaction using the Document Object Model (DOM);
- Data interchange and manipulation using XML and JSON;
- Asynchronous data retrieval using XMLHttpRequest;
- JavaScript binding everything together.

See the class slides at http://csci571.com/slides/ajax.pdf

JSON, short for JavaScript Object Notation, is a lightweight data interchange format. Its main application is in AJAX web application programming, where it serves as an alternative to the use of the XML format for data exchange between client and server. See the class slides at:

http://csci571.com/slides/JSON1.pdf

2.2 Bootstrap

Bootstrap is a free collection of tools for creating responsive websites and web applications. It contains HTML and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. To learn more details about Bootstrap please refer to the lecture material on Responsive Web Design (RWD). Please use **Bootstrap 4** in this homework. See the class slides at:
http://csci571.com/slides/Responsive.pdf

2.3 Google App Engine (GAE)

Google App Engine applications are easy to create, easy to maintain, and easy to scale as your traffic and data storage needs change. With App Engine, there are no servers to maintain. You simply upload your application and it's ready to go. App Engine applications automatically scale based on incoming traffic. Load balancing, micro services, authorization, SQL and noSQL

databases, memcache, traffic splitting, logging, search, versioning, roll out and roll backs, and security scanning are all supported natively and are highly customizable.

To learn more about GAE support for Node.js visit this page:
https://cloud.google.com/appengine/docs/standard/nodejs/

2.4 Angular

Angular is a toolset for building the framework most suited to your application development. It is fully extensible and works well with other libraries. Every feature can be modified or replaced to suit your unique development workflow and feature needs. Angular combines declarative templates, dependency injection, end to end tooling, and integrated best practices to solve development challenges. Angular empowers developers to build applications that live on the web, mobile, or the desktop.

For this homework, Angular 2+ (Angular 2,4,5,6,7,8 or 9) can be used, but Angular 9 is recommended. However, please note Angular 2+ will be a little difficult to learn if the developer is not familiar with Typescript and component-based programming.

To learn more about Angular 2+, visit this page:
https://angular.io/

2.5 Node.js

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js package ecosystem, **npm**, is the largest ecosystem of open source libraries in the world.

To learn more about Node.js, visit:
https://Node.js.org/en/

Also, **Express.js** is strongly recommended. Express.js is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. It is in fact the standard server framework for Node.js.

To learn more about Express.js, visit:
http://expressjs.com/

All APIs calls should be done through your Node.JS server

3. High Level Description

In this exercise you will create a webpage that allows users to search for products using the eBay API and display the results on the same page below the form. The application evolves from previous homework. When the application loads, a form is displayed to the user. After successful search, the matching products need to be displayed. All the implementation details and requirements will be explained in the following sections.

When a user initially opens your web page, the initial search form should look like in **Figure 1**.

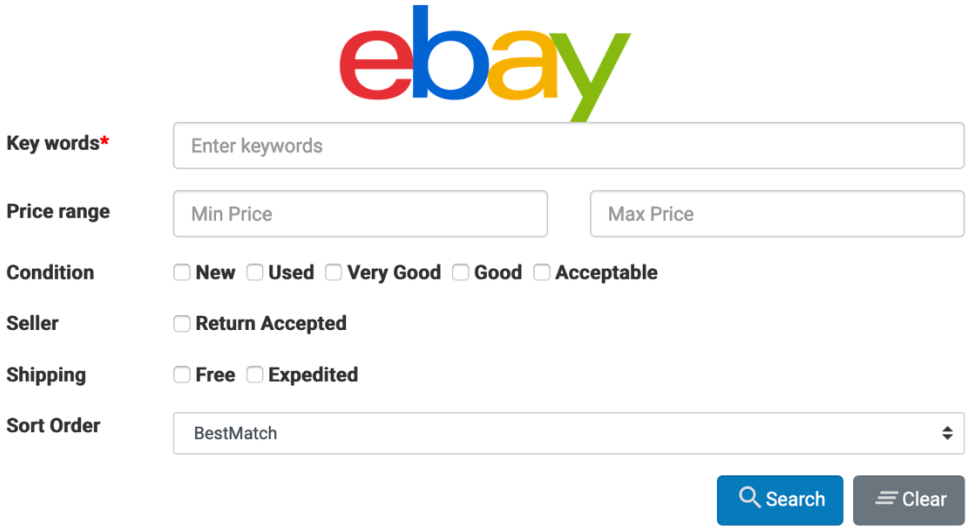


Figure 1: Initial Search Form

3.1 Search Form

3.1.1 Design

You must replicate the search form displayed in **Figure 1** using a **Bootstrap form**. The form fields are the same as Homework #6. There are six input fields in the search form explained below.

1. **Keyword**: This is a text box, which enables the user to search for matching items by entering keywords. This must be a non-empty string.

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, ∞). Please ensure that Minimum Price <= Maximum Price and values are

positive. 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.

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. The default when neither is specified should be "any" filter.

6. **Sort By**: This field specifies the ordering of the results table, which can be one of –

    a. Best Match (**Default Option**)
    b. Price: highest first
    c. Price + Shipping: highest first
    d. Price + Shipping: lowest first

The search form has two buttons:

1. **Search**: The "Search" button will read values from the form, validate the values and send the request to the backend server. On successful request the values in the form persist and matching results are displayed.

2. **Clear**: This button must reset the form fields, clear all validation errors if present and clear the results.

3.1.2 Search Execution

Once the validation is successful and the user clicks on the "Search" button, your application should make an AJAX call to the Node.js script hosted on Google App Engine (GAE). The Node.js script on GAE will then make a request to the eBay API service to get the product's information.

3.2 Displaying Results

3.2.1 Results Table

After the user clicks search, a list of results should be displayed. The following needs to be displayed on successful search. Please refer to **Figure 2** below.

- Searched keyword – The keyword entered by the user in the form.

- Pagination – The component helps users to paginate through the results.
- Products – List of matching products are displayed

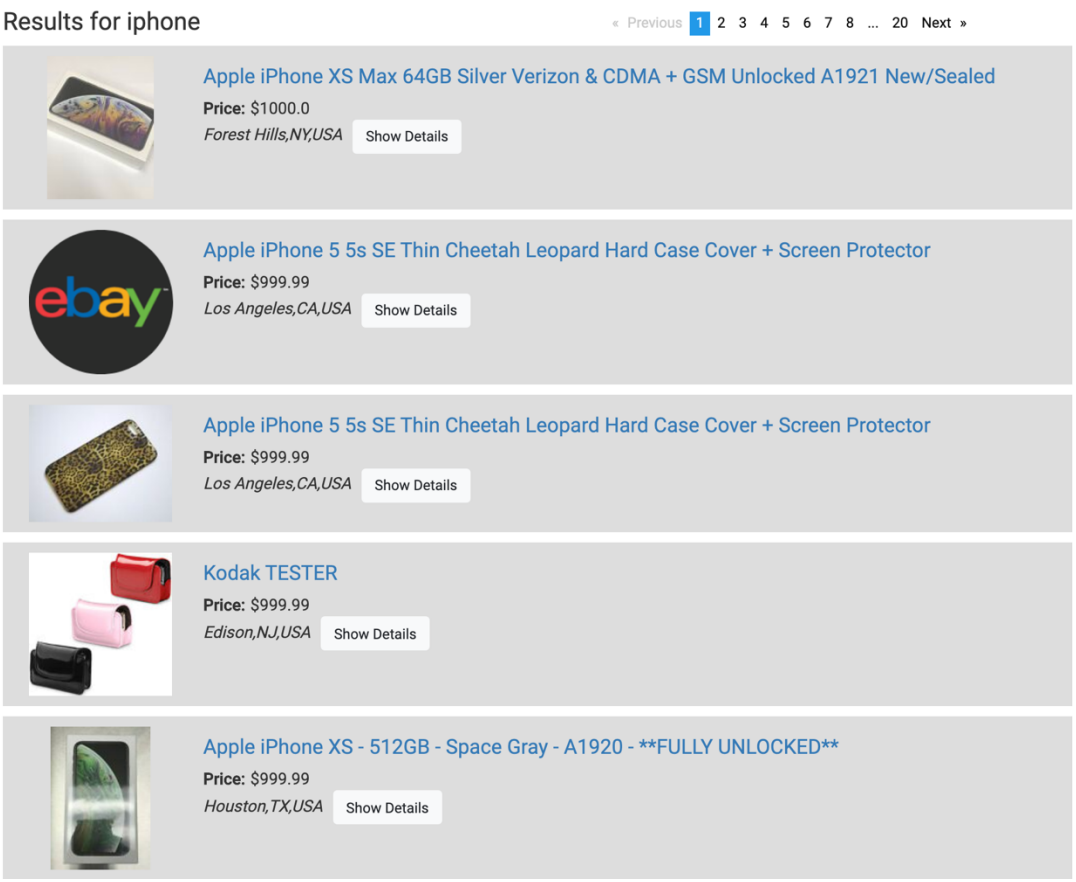


Figure 2: Search Result rows

Every product displayed will have an image to its left. On the top right, the item title is displayed in bold. On clicking the title, the user should be able to view product on the eBay website. Below the title, the product price should be displayed. Below the price, the product location and a Show/Hide details button should be displayed.

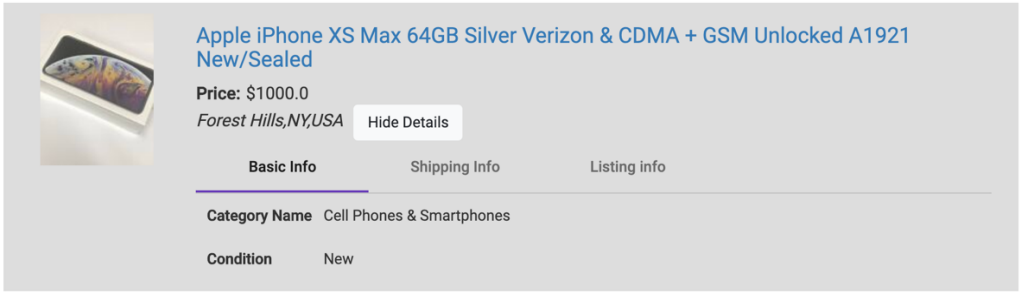
Note:

1. If any attribute is missing or empty or blank, do not display the item to the user.
2. If image is missing, display the image shared below.

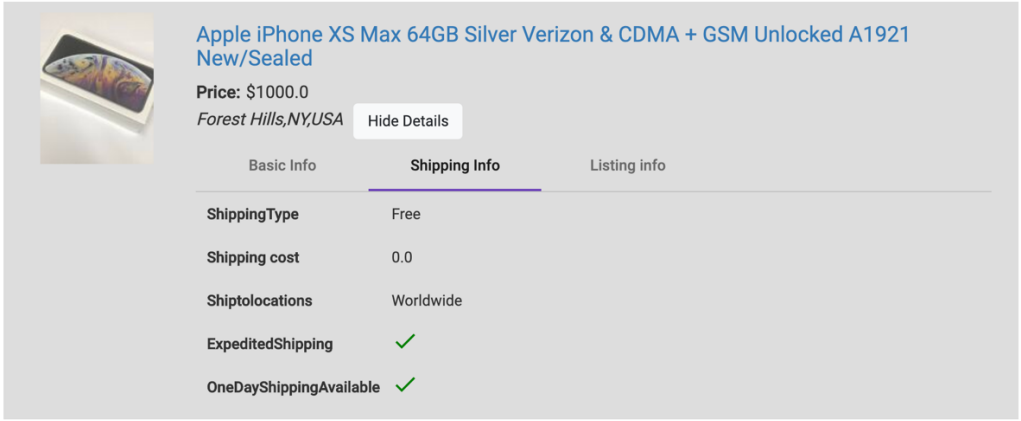
3.2.2 Product details

If the user clicks the **Show Details** button, more details about the product should be displayed. The Angular material library is used to implement tabs. Please check the expanded view of each section below. The mapping of fields used in each of the sections is explained in Section 4, eBay API.

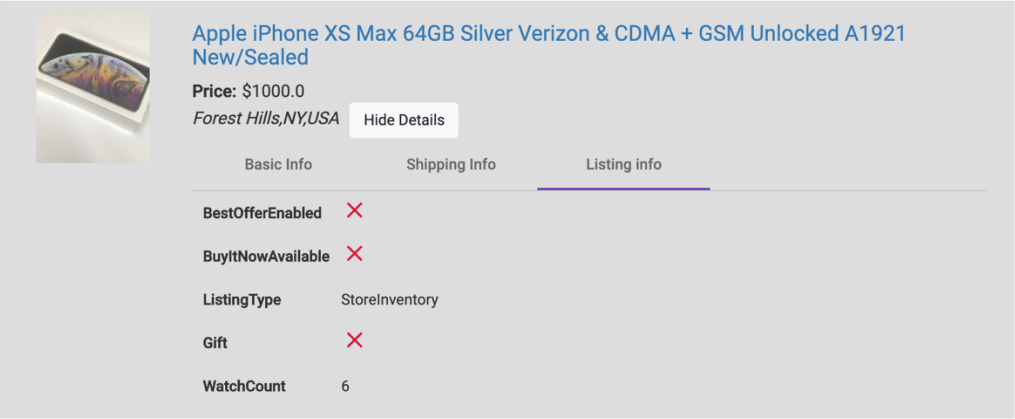
a) **Basic Info** -



b) **Shipping Info** –



c) **Listing Info** –



3.2.3 Validation Cases

Case 1: Key word is a mandatory field. If keyword is not entered by users, display the warning as shown in the image below.

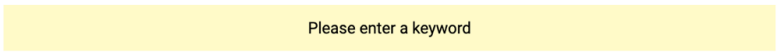


Figure 3: Validate entered keyword

Case 2: As explained in the search form description earlier, validate Min and Max price entered by the user.

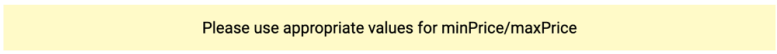


Figure 4: Validate Min and Max price

Case 3: If the result set is empty, display appropriate error message.

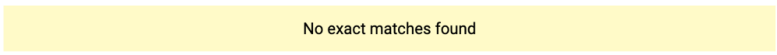


Figure 5: Validate Search Response

Case 4: Both Case 1 and Case 2 can happen simultaneously. In such scenario display both the warnings one below another.

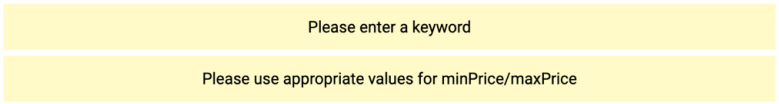
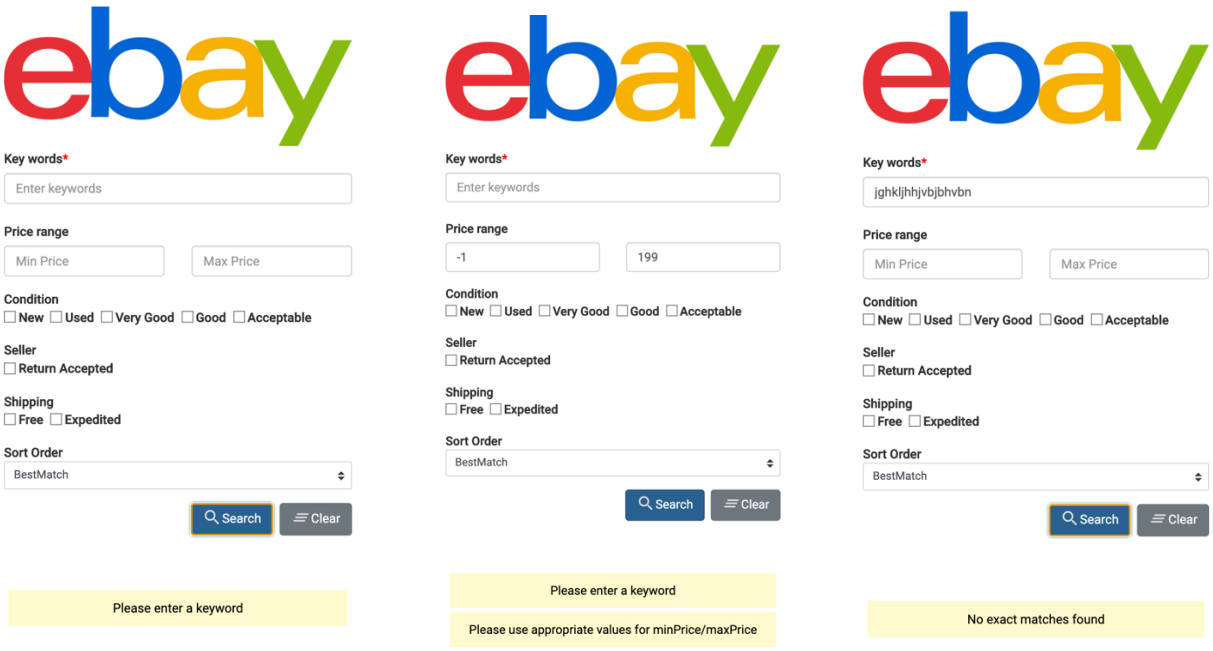
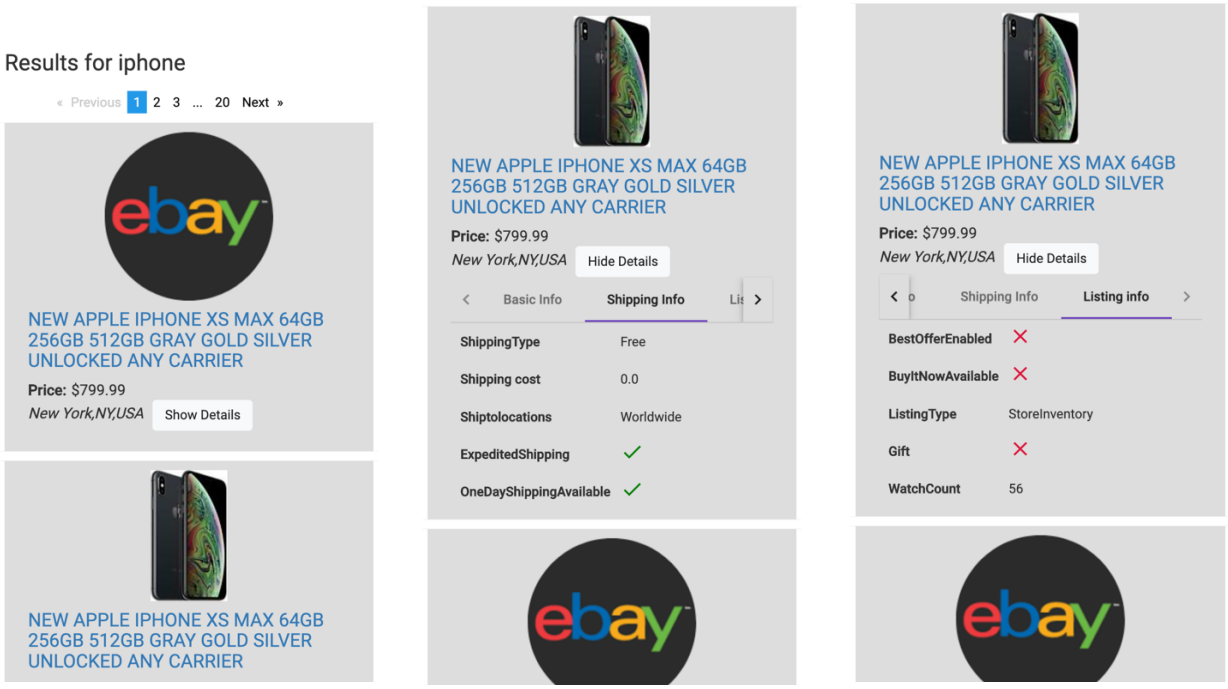


Figure 6: Validate Search Response

3.2.4 Responsive Design

The following are few snapshots of the webpage opened with chrome on an iPhone.





You must watch the video carefully to see how the page looks like on mobile devices. All functions must work on mobile devices.

Mobile browsers are different from desktop browsers. Even if your web page works perfectly on a desktop, it may not work as perfectly as you think on mobile devices. It's important that you also test your webpage on a real mobile device. Testing it in the mobile view of a desktop browser will not guarantee that it works on mobile devices.

4. eBay API

4.1 Constructing URL for API call similar to Homework 6

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 1 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 API Call**
Key words	*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 1: 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=100&sortOrder=BestMatch&itemFilter(0).name=MaxPrice&itemFilter(0).value=25&itemFilter(0).paramName=Currency&itemFilter(0).paramValue=USD

4.2 Sample API Response

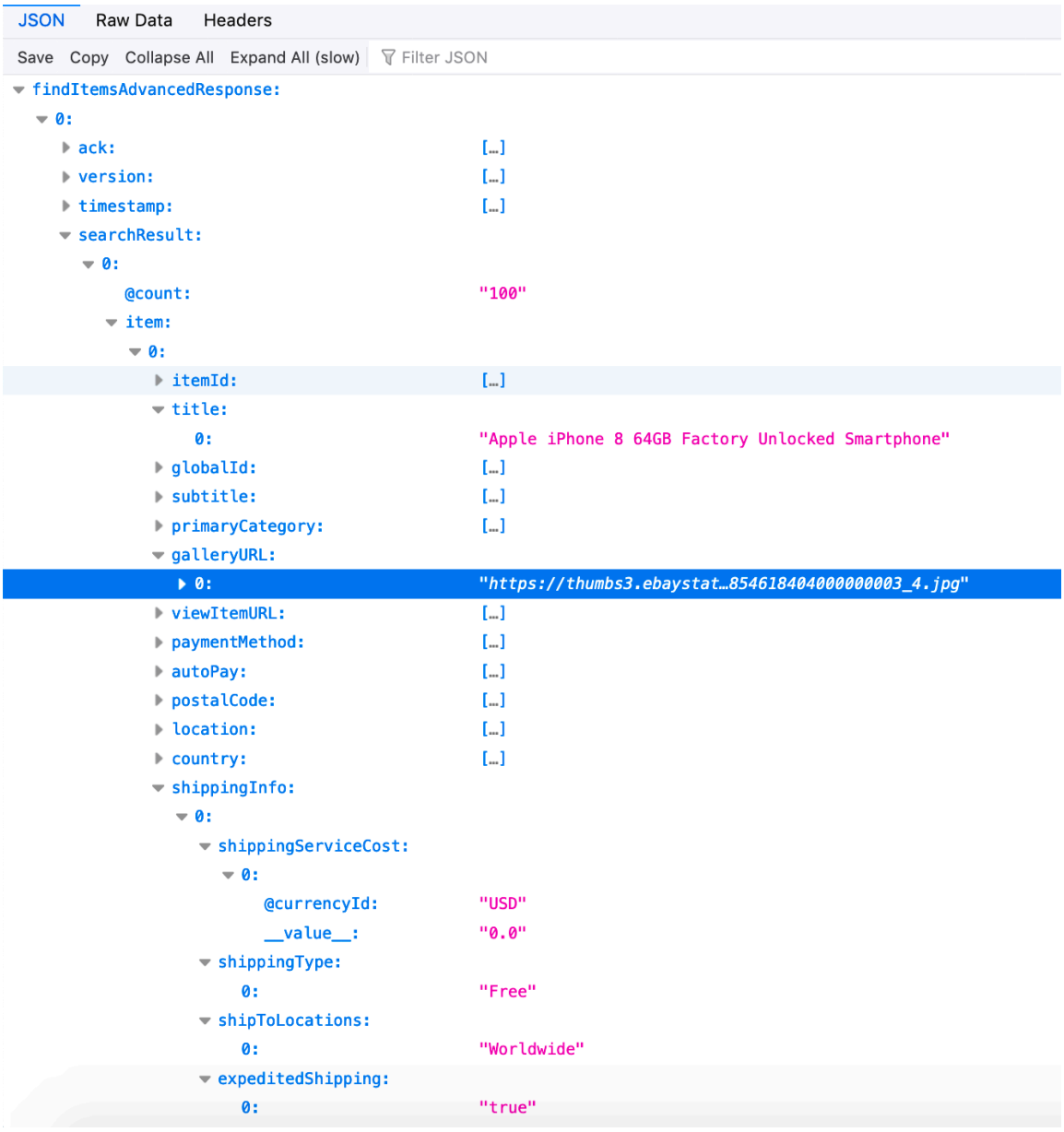


Figure 7: Ebay API response sample 1

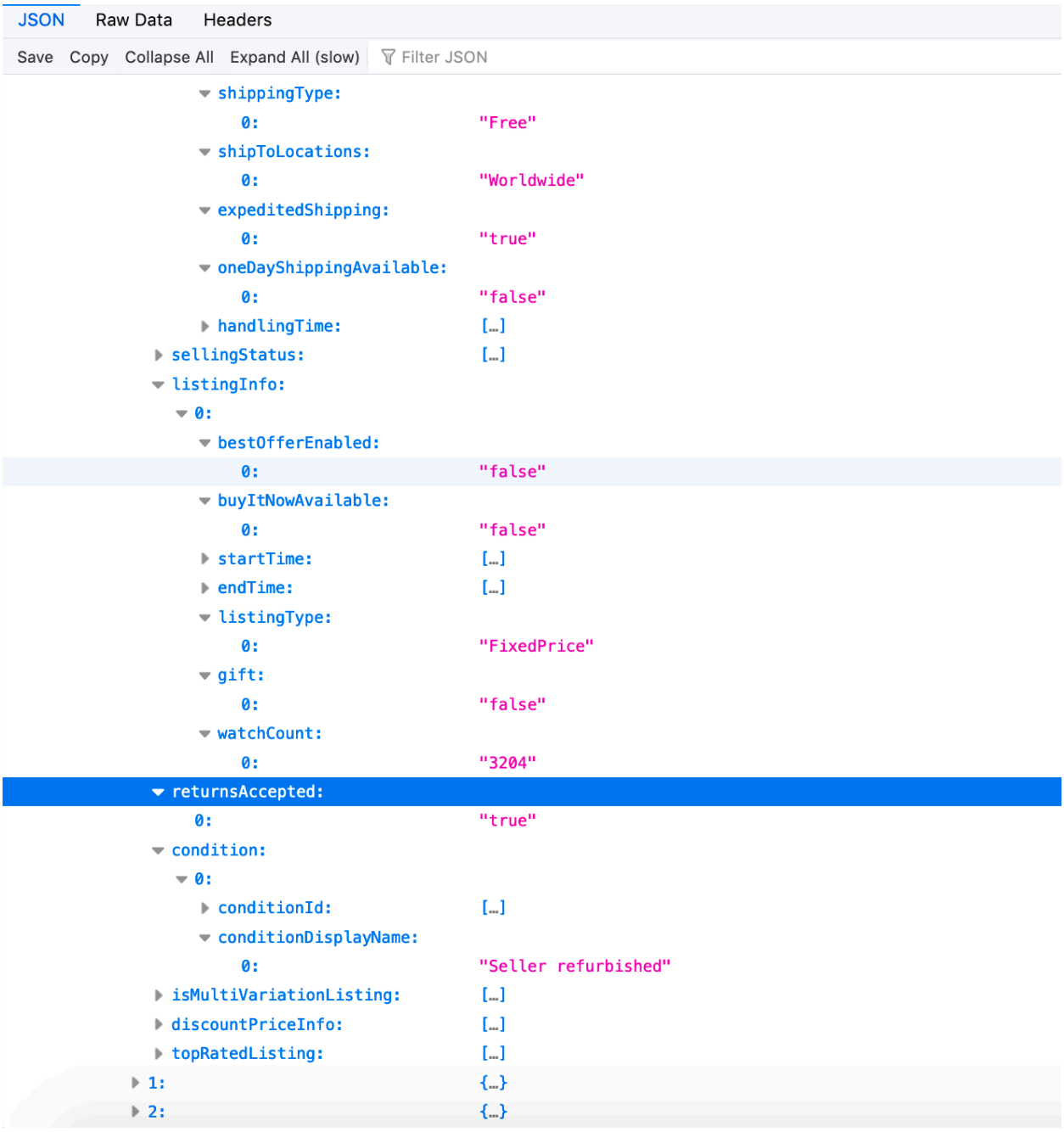


Figure 8: Ebay API response sample 2

4.3 Mapping of API Response to User Interface

UI Element	**API Response Field**
Title	*item* -> title
Image	*item -> galleryURL*
Price	*item -> sellingStatus -> currentPrice -> __value__*
Location	*item -> location*
Category Name	*item -> primaryCategory[0] - > categoryName[0]*
Condition	*item -> condition[0]- > conditionDisplayName[0]*
Shipping Type	*item -> shippingInfo[0] -> shippingType[0]*
Shipping Cost	*item -> shippingInfo[0] -> shippingServiceCost [0] -> __value__*
Ship to Locations	*item -> shippingInfo[0] -> shipToLocations [0]*
Expedited Shipping	*item -> shippingInfo[0] -> expeditedShipping [0]*
OneDayShippingAvailable	*item -> shippingInfo[0] -> oneDayShippingAvailable [0]*
BestOfferEnabled	*Item -> listingInfo[0] -> bestOfferEnabled[0]*
BuyItNowAvailable	*Item -> listingInfo[0] -> buyItNowAvailable [0]*
ListingType	*Item -> listingInfo[0] -> listingType [0]*
Gift	*Item -> listingInfo[0] -> gift [0]*
WatchCount	*Item -> listingInfo[0] -> watchCount [0]*

Table 2: UI – API Mappings

5. Implementation Hints

5.1 Bootstrap Library

To get started with the Bootstrap toolkit, please see:
https://getbootstrap.com/docs/4.0/getting-started/introduction/.

Bootstrap can be imported to Angular in couple of ways. See:

https://www.techiediaries.com/angular-bootstrap/

5.2 Bootstrap UI Components

Bootstrap provides a complete mechanism to make Web pages responsive to different mobile devices. In this exercise, you will get hands-on experience with responsive design using the Bootstrap Grid System.

https://getbootstrap.com/docs/4.0/layout/grid/

At a minimum, you will need to use Bootstrap Forms, Alerts, Cards and Buttons to implement the required functionality.

Bootstrap Forms	https://getbootstrap.com/docs/4.0/components/forms/
Bootstrap Alerts	https://getbootstrap.com/docs/4.0/components/alerts/
Bootstrap Cards	https://getbootstrap.com/docs/4.0/components/card/
Bootstrap Buttons	https://getbootstrap.com/docs/4.0/components/buttons/

5.3 Angular

- Angular Set up –
  https://angular.io/

- Angular Material Installation -
  https://material.angular.io/guide/getting-started

- Pagination -
  https://www.npmjs.com/package/ngx-pagination#simple-example

- Angular Material Tabs -
  https://material.angular.io/components/tabs/overview

5.4 Images

The image needed for this homework is available here:
https://csci571.com/hw/hw8/Images/ebayDefault.png

5.5 Icons

- https://material.io/resources/icons/?style=baseline
- https://material.io/resources/icons/?search=done&icon=done&style=baseline
- https://material.io/resources/icons/?search=clear&icon=clear_all&style=baseline
- https://material.io/resources/icons/?search=clear&icon=clear&style=baseline

5.6 Deploy Node.js application on GAE

Since Homework #8 is implemented with Node.js and GAE, you should **select Nginx as your proxy server (if available)**, which should be the default option.

6. Files to Submit

In your course homework page, you should update the Homework #8 link to refer to your new initial web page for this exercise. Additionally, you need to provide **an additional link** to the URL of the GAE service where the AJAX call is made with sample parameter values.

****IMPORTANT**:**

All videos are part of the homework description. All discussions and explanations in Piazza related to this homework are part of the homework description and will be accounted into grading. So please review all Piazza threads before finishing the assignment.