

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

Product Search

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

1. Objectives

- Get familiar with the AJAX and JSON technologies.
- Use a combination of HTML5, Bootstrap, Angular and jQuery 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 Amazon Web Services/Google Cloud Platform App Engine/Microsoft Azure.
- Learn to use popular APIs such as eBay API, Google Customized Search API and Facebook API.

1.1 Prerequisite

Please register Facebook API as soon as possible. In previous semesters, Facebook may take a long time to check your account if you did not use Facebook frequently.

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 Amazon Web Services (AWS)

AWS is Amazon's implementation of cloud computing. Included in AWS is Amazon Elastic Compute Cloud (EC2), which delivers scalable, pay-as-you-go compute capacity in the cloud, and AWS Elastic Beanstalk, an even easier way to quickly deploy and manage applications in the AWS cloud. You simply upload your application, and Elastic Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring. Elastic Beanstalk is built using familiar software stacks such as the Apache HTTP Server, PHP, and Python, Passenger for Ruby, IIS for .NET, and Apache Tomcat for Java.

The Amazon Web Services homepage is available at: <http://aws.amazon.com/>

2.4 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/>

2.5 Microsoft Azure

Microsoft Azure is a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through a global network of Microsoft-managed data centers. It provides software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS) and supports many different programming languages, tools and frameworks, including both Microsoft-specific and third-party software and systems.

To learn more about Azure support for Node.js visit this page:

<https://docs.microsoft.com/en-us/javascript/azure/?view=azure-node-latest>

2.6 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, either AngularJS or Angular 2+ (Angular 2,4,5,6 or 7) can be used, but Angular 7 is recommended as AngularJS entered LTS. 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 AngularJS visit this page:

<https://angularjs.org/>

To learn more about Angular 2+, visit this page:

<https://angular.io/>

2.7 jQuery

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

To learn more about jQuery visit this page:

<https://jquery.com/>

2.8 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/>

Note: In this document when you see “jQuery/Angular” it means that you can either use a jQuery or Angular function; and when you see GAE/AWS/Azure it means that you can either use Google App Engine, Amazon Web Services or Microsoft Azure Services.

There are typically three ways to implement the client side:

1. Use jQuery + AngularJS
2. Use entirely AngularJS
3. Use entirely Angular2+

You can use either way.

All APIs calls should be done through your Node.JS server, except calls to the ip-api.

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. Once the user clicks on a product name to search for product details, your webpage should display 5 tabs which contain a Product info tab, Product photos tab, Shipping info tab, Seller info tab and a Similar products tab related to the product, respectively. Your webpage should also support adding products to and removing products from the wish list and sharing products info to Facebook. All the implementation details and requirements will be explained in the following sections.

When a user initially opens your webpage, your page should look like in Figure 1.

The image shows a 'Product Search' form on a dark background. The form has several sections: 'Keyword' with a text input field containing the placeholder 'Enter Product Name (eg. iPhone 8)'; 'Category' with a dropdown menu showing 'All Categories'; 'Condition' with three checkboxes labeled 'New', 'Used', and 'Unspecified'; 'Shipping Options' with two checkboxes labeled 'Local Pickup' and 'Free Shipping'; 'Distance (Miles)' with a text input field containing '10'; and 'From' with two radio button options: 'Current Location' (selected) and 'Other. Please specify zip code:' followed by a text input field. At the bottom of the form are two buttons: 'Search' and 'Clear'. Below the form, there are two tabs: 'Results' and 'Wish List'.

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 similar to Homework #6.

There are six input fields in the search form:

1. **Keyword:** This field is required. Validation is performed on this field. Please refer to section 3.1.3 for details of validation. Initially, please show the placeholder shown in the picture.
2. **Category:** The default value of “Category” is “All Categories”, which covers most of the “types” provided by the *eBay API*. The other options are shown in Figure 2.
3. **Condition:** There are 3 options for the user to select from: New, Used and Unspecified. Multiple options could be selected to get results for all of them.
4. **Shipping Options:** There are 2 options provided: Local Pickup and Free Shipping. The user again has the option to select 1 or both of them.
5. **Distance (Miles):** This is the radius of the area within which the search is performed. The center of the area is specified in the “Current Location” field. The default value is 10 miles.

6. **From:** The center of the area within which the search is performed. The user can choose between their current location or other location based on the zip code. This field is required (the user must either choose the first radio button or choose the second one and provide a location) and must be validated. Please refer to section 3.1.3 for details of validation. The input box below the second radio button is disabled by default. If the user chooses to provide a different location, this input field should be enabled.

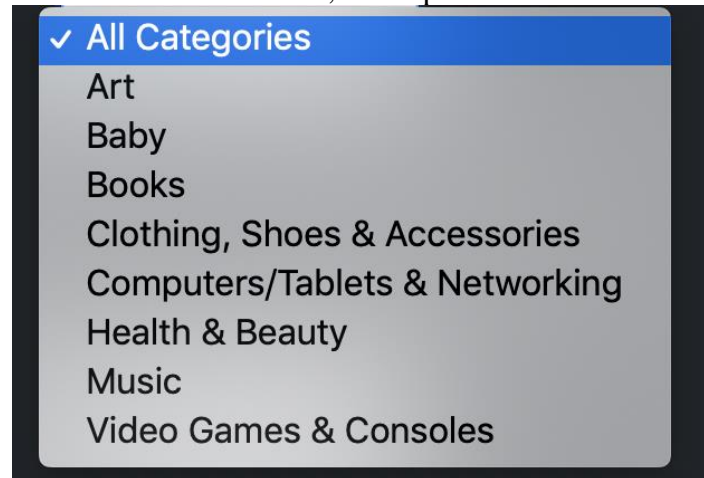


Figure 2 Category Options

The search form has two buttons:

1. **Search:** The “Search” button should be disabled whenever either of the required fields is empty or validation fails, or the user location is not obtained yet. Please refer to section 3.1.3 for details of validation. Please refer to section 3.1.4 for details of obtaining user location.
2. **Clear:** This button must reset the form fields, clear all validation errors if present, switch the view to the results tab and clear the results area.

3.1.2 AutoComplete

Autocomplete for zip code is implemented by using the suggestion service provided by *Geonames*. Please go to this page to learn more about this service:

<https://www.geonames.org/export/web-services.html>

An example of an HTTP request to the *Geonames* API Get Suggestion that searches for the zip code “900” is shown below:

[http://api.geonames.org/postalCodeSearchJSON?postalcode_startsWith=900&username=\[Username\]&country=US&maxRows=5](http://api.geonames.org/postalCodeSearchJSON?postalcode_startsWith=900&username=[Username]&country=US&maxRows=5)

To get Username in the above URL:

1. Register at <http://www.geonames.org/login>
2. In the API call, the “Username” is the username you entered during registration.

Note:

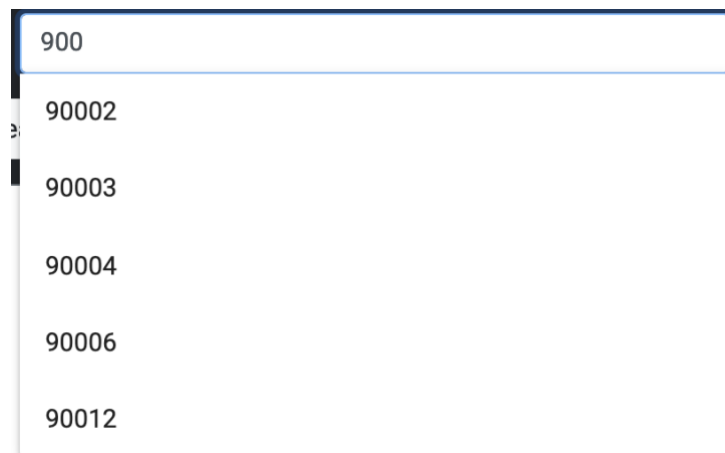
1. Set the **maxRows** to 5 to limit the number of suggestions.
2. If the **Geonames** API doesn’t work, **hide the autocomplete** feature. This situation should be handled and not throw any error.

The response is a JSON object shown in Figure 3.

```
{
  "postalCodes": [
    {
      "adminCode2": "037",
      "adminCode1": "CA",
      "adminName2": "Los Angeles",
      "lng": -118.246213,
      "countryCode": "US",
      "postalCode": "90002",
      "adminName1": "California",
      "ISO3166-2": "CA",
      "placeName": "Los Angeles",
      "lat": 33.94969
    },
    {
      "adminCode2": "037",
      "adminCode1": "CA",
      "adminName2": "Los Angeles",
      "lng": -118.272739,
      "countryCode": "US",
      "postalCode": "90003",
      "adminName1": "California",
      "ISO3166-2": "CA",
      "placeName": "Los Angeles",
      "lat": 33.965335
    },
    { },
    { },
    { }
  ]
}
```

Figure 3 Autocomplete JSON Response

You must use **Angular Material** to implement the Autocomplete. (See section 5.4).



A screenshot of an Angular Material autocomplete dropdown menu. The input field at the top contains the text "900". Below the input field, a list of suggestions is displayed, each on a new line. The suggestions are "90002", "90003", "90004", "90006", and "90012". The dropdown menu has a light gray background and a thin border.

Figure 4 Autocomplete example

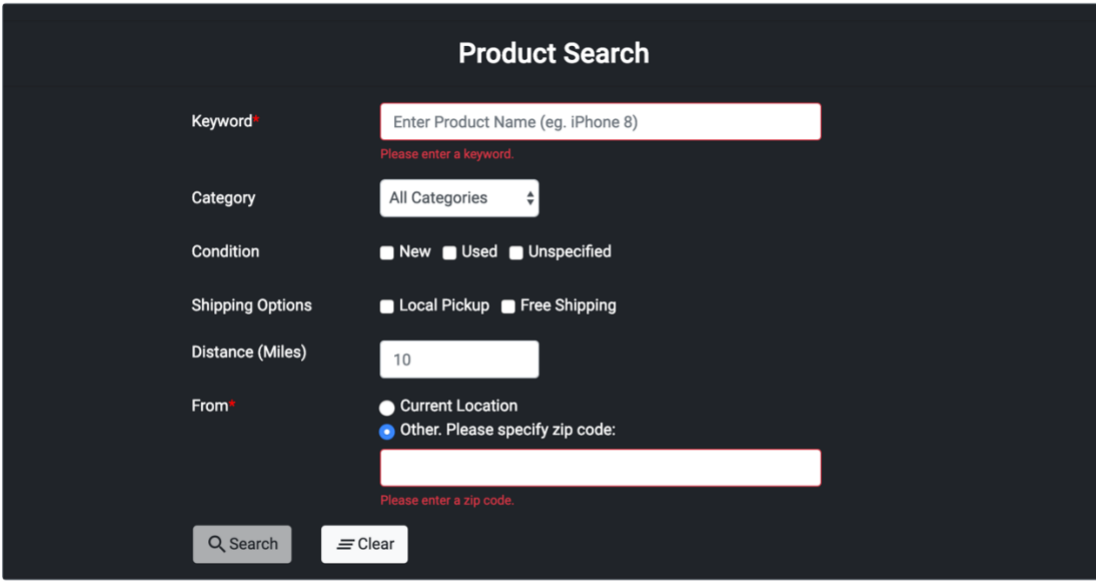
3.1.3 Validation

Your application should check if the “Keyword” contains something other than spaces. If not, then it’s invalid and an error message should be shown, and the border of the input field should turn red as in Figure 5.

If the second radio button of “From” field is chosen, the same validation should be performed for the input field below the second radio button.

The zip code is restricted to only 5 digits. The search button should be disabled until and unless a 5-digit zip code is provided. If any other characters are included, the search button should not be enabled.

Please watch the reference video carefully to understand the validation behavior.



The screenshot shows a 'Product Search' form with the following elements:

- Keyword:** A text input field with a red border and a red error message 'Please enter a keyword.' below it.
- Category:** A dropdown menu showing 'All Categories'.
- Condition:** Three radio buttons: 'New', 'Used', and 'Unspecified'.
- Shipping Options:** Two radio buttons: 'Local Pickup' and 'Free Shipping'.
- Distance (Miles):** A text input field containing the value '10'.
- From:** Two radio buttons: 'Current Location' and 'Other. Please specify zip code:'. The 'Other' option is selected, and its associated text input field has a red border and a red error message 'Please enter a zip code.' below it.
- Buttons:** A 'Search' button with a magnifying glass icon and a 'Clear' button with a close icon.
- Footer:** Two buttons: 'Results' and 'Wish List'.

Figure 5 An Invalid Form

3.1.4 Obtaining User Location

As in Homework #6, you should obtain the current user location using one of the geolocation APIs. The usage of this API has been explained in the Homework #6 documents.

<http://ip-api.com/json>

3.1.5 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 GAE/AWS/Azure. The Node.js script on GAE/AWS/Azure will then make a request to eBay API to get the product’s information. This will be explained in the next section.

3.2 Results Tab

3.2.1 Results Table

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

The *eBay API Search* 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 usage of these two APIs has been explained in the Homework #6 documents.

An example of an HTTP request to the eBay Finding API that searches for the products related to iPhone 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=\[APP-ID\]&RESPONSE-DATA-FORMAT=JSON&REST-PAYLOAD&paginationInput.entriesPerPage=50&keywords=iphone&buyerPostalCode=90007&itemFilter\(0\).name=MaxDistance&itemFilter\(0\).value=10&itemFilter\(1\).name=FreeShippingOnly&itemFilter\(1\).value=true&itemFilter\(2\).name=LocalPickupOnly&itemFilter\(2\).value=true&itemFilter\(3\).name=HideDuplicateItems&itemFilter\(3\).value=true&itemFilter\(4\).name=Condition&itemFilter\(4\).value\(0\)=New&itemFilter\(4\).value\(1\)=Used&itemFilter\(4\).value\(2\)=Unspecified&outputSelector\(0\)=SellerInfo&outputSelector\(1\)=StoreInfo](http://svcs.ebay.com/services/search/FindingService/v1?OPERATION-NAME=findItemsAdvanced&SERVICE-VERSION=1.0.0&SECURITY-APPNAME=[APP-ID]&RESPONSE-DATA-FORMAT=JSON&REST-PAYLOAD&paginationInput.entriesPerPage=50&keywords=iphone&buyerPostalCode=90007&itemFilter(0).name=MaxDistance&itemFilter(0).value=10&itemFilter(1).name=FreeShippingOnly&itemFilter(1).value=true&itemFilter(2).name=LocalPickupOnly&itemFilter(2).value=true&itemFilter(3).name=HideDuplicateItems&itemFilter(3).value=true&itemFilter(4).name=Condition&itemFilter(4).value(0)=New&itemFilter(4).value(1)=Used&itemFilter(4).value(2)=Unspecified&outputSelector(0)=SellerInfo&outputSelector(1)=StoreInfo)













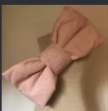







Note: Changes in the URL from Homework 6

1. **paginationInput.entriesPerPage=50 (limit to 50 products).**
2. **outputSelector(0)=SellerInfo (To get Seller details for each product).**
3. **outputSelector(1)=StoreInfo (To get Store Info for each product).**

The Node.js script should pass the JSON object returned by the *Product Search* 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 eight columns: # (Index number), Image, Title, Price, Shipping, Zip, Seller and Wish list.

Below the table there is a pagination feature which is used to navigate to the next set of products. There is a maximum of 10 products on a page. The user can navigate to a different page directly by clicking on the page number, or by using previous and next buttons.

Detail >

#	Image	Title	Price	Shipping	Zip	Seller	Wish List
1		Puma Premium Scuderia Ferrari Team ...	\$26.55	Free Shipping	90021	VKAPPAREL	
2		Puma Fenty bag	\$25.00	\$7.85	90007	MICPIT_74	
3		PUMA TEAMSPORT FORMATION BACKPACK ...	\$29.99	Free Shipping	90058	APPARELZOO	
4		Unisex Puma Ferrari Shoulder Bag, ...	\$29.99	Free Shipping	90003	FRIENDFOREVERGUITAR	
5		NEW NWT PUMA PREMIUM SCUDERIA ...	\$26.55	Free Shipping	90021	VKAPPAREL	
6		Puma Men's Scuderia Ferrari F1 Racer...	\$29.99	Free Shipping	90019	KRASSDEALS	
7		Puma Fenty by Rihanna Bow CrossPack ...	\$129.99	\$9.99	90023	MASSI747	
8		NEW PUMA SCUDERIA FERRARI F1 ...	\$47.45	Free Shipping	90021	VKAPPAREL	
9		Unisex Puma Ferrari Fanwear Portable...	\$29.99	Free Shipping	90003	FRIENDFOREVERGUITAR	
10		NEW PUMA Equation 21\" Duffel Bag ...	\$49.99	Free Shipping	91205	SEIZETHEDEALS	

« Previous 1 2 3 4 5 Next »

Figure 6 An Example of a Valid Search result

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

HTML Table Column	API service response
Index	The value of the “ <i>index</i> ” is a number ranging from 1 to 50 for all results.
Image	The value of the “ <i>Photos</i> ” attribute is a part of the “ <i>galleryURL</i> ” object inside the “ <i>item</i> ” object.
Title	The value of the “ <i>title</i> ” attribute inside the “ <i>item</i> ” object.
Price	The value of the “ <i>__value__</i> ” attribute that is part of the “ <i>currentPrice</i> ” object inside the “ <i>sellingStatus</i> ” object which is part of “ <i>item</i> ” object.
Shipping	The value of the “ <i>__value__</i> ” inside the “ <i>shippingServiceCost</i> ” attribute is part of the “ <i>shippingInfo</i> ” object inside “ <i>item</i> ” object. Set the value to “ <i>Free Shipping</i> ” if the “ <i>shippingServiceCost</i> ” is zero or set it to the value in dollars if it is not zero. Set it to “ <i>N/A</i> ” if no information is given.
Zip	The value of the “ <i>postalCode</i> ” attribute that is part of the “ <i>item</i> ” object.
Seller	The value of the “ <i>sellerUserName</i> ” attribute that is part of the “ <i>sellerInfo</i> ” object.
Wish List	This contains a button that can add the product to, or remove the product from, the Wish List.

Table 1: Mapping the result from eBay Search API into HTML table

Note: If any value is missing in the first table please display “N/A”.

If a particular image in the table is clicked it should open in a new tab. The “#” column starts from 1 and goes till 50. Since every product may not have 50 results, you need to show as many results returned by eBay, limiting to 50 results in total and only 10 results per page. This needs to be done using pagination.

The “Title” column might not be long enough to show the entire name of the product and should show “...” to avoid starting a new row. The “Title” column is clickable to trigger the detail search for the corresponding product. If a product is on the Wish List, the cart icon is full (Figure 7 item 1). Otherwise, the cart icon is empty (Figure 7, item 2).

You can follow this idea to avoid starting a new row for product name:

1. Judge whether the product name’s length is larger than 35 characters. (Or other reasonable number)
2. If yes, please cut the string to the first 35 characters, and if the cut position is not a white space, please find the last index of white space before the cut position and use that as the substring’s end index.
3. Add ‘...’ to the new string.
4. Show the tooltip of the whole product name (Figure 7).

Detail >





#	Image	Title	Price	Shipping	Zip	Seller	Wish List
1		Puma Premium Scuderia Ferrari Team ... <div style="background-color: #333; color: white; padding: 2px; font-size: 0.8em;"> Puma Premium Scuderia Ferrari Team Lifestyle Fanny Waist Bag Black PMMO1007 </div>	\$26.55	Free Shipping	90021	VKAPPAREL	
2		Puma Fenty bag	\$25.00	\$7.85	90007	MICPIT_74	

Figure 7 An Example of the tooltip for product name

For tooltip component, if you use entirely AngularJS/Angular2+, please use Angular Material to implement this. If you use jQuery, you could use the bootstrap tooltip. The style between bootstrap tooltip and Angular Material tooltip will be a slightly different, but it is fine. (See section 5.3 and 5.4)

3.2.3 Details Button and Highlighting

The “Details >” button, right above the results table, is initially disabled. It will be enabled once a product details search is triggered. If this button is enabled and clicked, the page will be taken to the Product detail tabs. After a product details search is performed, the corresponding product row in the results table should be highlighted to indicate the product whose details are displayed in the Details tabs (Figure 8).

Detail >





#	Image	Title	Price	Shipping	Zip	Seller	Wish List
1		Puma Premium Scuderia Ferrari Team ...	\$26.55	Free Shipping	90021	VKAPPAREL	
2		Puma Fenty bag	\$25.00	\$7.85	90007	MICPIT_74	

Figure 8 Highlight Selected Product

3.3 Details

Once a product name in the “Title” column is clicked, your webpage should search for the details of that product. Above the tabs in the details view, you should display the whole name of the Product, a button that allows you to go back to the previous list, a Facebook button and a Wish List button.

An example of an HTTP request to the eBay Shopping API that searches for details for a single product based on the ItemID is shown below:

[http://open.api.ebay.com/shopping?callname=GetSingleItem&responseencoding=JSON&appid=\[APP-ID\]&siteid=0&version=967&ItemID=132961484706&IncludeSelector=Description,Details,ItemSpecifics](http://open.api.ebay.com/shopping?callname=GetSingleItem&responseencoding=JSON&appid=[APP-ID]&siteid=0&version=967&ItemID=132961484706&IncludeSelector=Description,Details,ItemSpecifics)

The response for a single item is shown below:

```

Timestamp:      "2019-02-01T19:42:49.720Z"
Ack:            "Success"
Build:          "E1089_CORE_APILW_18879361_R1"
Version:        "1089"
▼ Item:
  BestOfferEnabled: false
  ▶ Description:  "<head><meta charset=\"UT_88x33.gif\"></CENTER><P>"
  ItemID:        "292862774875"
  EndTime:       "2019-02-09T23:49:01.000Z"
  StartTime:     "2018-12-11T23:49:01.000Z"
  ▶ ViewItemURLForNaturalSearch: "https://www.ebay.com/itm_Sizes-/292862774875?var="
  ListingType:   "FixedPriceItem"
  Location:      "Minneapolis, Minnesota"
  ▶ PaymentMethods: [...]
  ▶ PictureURL:  [...]
  PostalCode:    "55416"
  PrimaryCategoryID: "24541"
  ▼ PrimaryCategoryName: "Sports Mem, Cards & Fan Shop:Fan Apparel & Souvenirs:College-NCAA"
  Quantity:      35
  ▶ Seller:      {...}
  BidCount:      0
  ▶ ConvertedCurrentPrice: {...}
  ▶ CurrentPrice: {...}
  ListingStatus: "Active"
  QuantitySold:  9
  ▶ ShipToLocations: [...]
  Site:          "US"
  TimeLeft:      "P8DT4H6M12S"
  ▼ Title:       "NEW Nike USC Trojans - Black Poly Short Sleeve Shirt (Multiple Sizes)"
  ▶ ItemSpecifics: {...}
  HitCount:      466
  PrimaryCategoryIDPath: "64482:24409:24541"
  ▶ Charity:      {...}
  ▶ Storefront:   {...}
  Country:       "US"
  ▶ ReturnPolicy: {...}
  AutoPay:       true

```

Figure 9: A sample JSON response from eBay Shopping API to get a Single Item.

3.3.1 Info Tab

A table containing the detailed info of the product is displayed in this tab. The table has the following fields if they are available in the detail search results:

Fields in the table	Corresponding response object fields
Product Images	The value of the “ <i>PictureURL</i> ” attribute that is part of the “ <i>Item</i> ” object. It is a modal containing all pictures of the “ <i>PictureURL</i> ” array.
SubTitle	The value of the “ <i>Subtitle</i> ” attribute that is part of the “ <i>Item</i> ” object.
Price	The value of the “ <i>Value</i> ” attribute that is part of the “ <i>currentPrice</i> ” object inside the “ <i>Item</i> ” object.

Location	The value of the “ <i>Location</i> ” attribute that is part of the “ <i>Item</i> ” object.
Return Policy (US)	The value of the “ <i>ReturnPolicy</i> ” attribute that is part of the “ <i>Item</i> ” object. It is composed of two fields: “ <i>ReturnsAccepted</i> ” and “ <i>ReturnsWithin</i> ”.
ItemSpecifics (Name)	The value of the all the “ <i>NameValueList</i> ” array corresponding to that name inside the “ <i>ItemSpecifics</i> ” object in the “ <i>Item</i> ” object.

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

Note: If any value is missing, don’t display the row in product info tab.

When clicked on “Product Images”, a modal should be popped up with all product images. If a particular image in the modal is clicked it should open in a new tab. See video for reference.

iPhone 8 Plus 64/256GB+ GRAY SILVER GOLD AT&T T-Mobile Sprint Verizon Unlocked

< List

Product Photos Shipping Seller Similar Products

Product Images	View Product Images Here
Subtitle	1-YR WARRANTY ✓ 3-5 DAY SHIP ✓ A/B/C Grade ✓ Charger+
Price	\$479.00
Location	Alhambra, California
Return Policy	Returns Accepted Within 30 Days
Camera Resolution	12.0MP
Model	iPhone 8 Plus
Operating System	iOS
Contract	Without Contract
Style	Smartphone
Cellular Band	LTE
Network Technology	GSM
Features	3G Data Capable,4G Data Capable,Accelerometer,Bluetooth Enabled,Fingerprint Sensor,Global Ready
Model Number	A1897 (GSM)
Lock Status	Factory Unlocked
MPN	Does Not Apply
Network	Factory Unlocked
Brand	Apple
Screen Size	5.5"

Figure 10 An Example of Product Detail Tab

3.3.2 Photos Tab

A table containing the images related to the product is displayed in this tab. The Google Custom Search Engine is documented at:

<https://developers.google.com/custom-search/json-api/v1/overview>

To retrieve photos about the Product, the request needs 6 parameters (output should be JSON):

- **q**: The search expression
- **cx**: The custom search engine ID to use for this request.
- **imgSize**: Returns images of a specified size.
- **num**: Number of search results to return. (Valid values are integers between 1 and 10, inclusive.)
- **searchType**: Specifies the search type: image. If unspecified, results are limited to webpages.
- **key**: Your application's API key. This key identifies your application for purposes of quota management.

An example of an HTTP request to the Google custom search API is shown below:

```
https://www.googleapis.com/customsearch/v1?q=[Product_Title]&cx=[YOUR_SEARCH_ENGINE_ID]&imgSize=huge&imgType=news&num=8&searchType=image&key=[YOUR_API_KEY]
```

Figure 11 shows a sample response.

```
{
  "kind": "customsearch#search",
  "url": {},
  "queries": {},
  "context": {},
  "searchInformation": {},
  "items": [
    {
      "kind": "customsearch#result",
      "title": "Apple iPhone 8 Plus a1897 64GB GSM Unlocked - Good | eBay",
      "htmlTitle": "<b>Apple iPhone 8</b> Plus a1897 <b>64GB GSM Unlocked</b> - Good | eBay",
      "link": "https://d3d71ba2asa5oz.cloudfront.net/12015576/images/apple%20iphone%208%20plus%20rose%20gold%20front%20122717.jpg",
      "displayLink": "www.ebay.com",
      "snippet": "Apple iPhone 8 Plus a1897 64GB GSM Unlocked - Good | eBay",
      "htmlSnippet": "<b>Apple iPhone 8</b> Plus a1897 <b>64GB GSM Unlocked</b> - Good | eBay",
      "mime": "image/jpeg",
      "image": {
        "contextLink": "https://www.ebay.com/itm/Apple-iPhone-8-Plus-a1897-64GB-GSM-Unlocked-Good-/233044091828",
        "height": 1727,
        "width": 1727,
        "byteSize": 944257,
        "thumbnailLink": "https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSF0DLhZW0maVjf5a90n3VBtnkE1pUkp4soTXeS1onRhMvWFNehEJIkrAFnmQ",
        "thumbnailHeight": 150,
        "thumbnailWidth": 150
      }
    }
  ]
}
```

Figure 11 Google Customized Search API response

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

HTML Table Column	API service response
Photo	You should display 8 photos, which is present in “link” attribute inside the ‘item’ object.

Table 3: Mapping the result from Google Custom Search API into HTML Table

You should display the photos in three columns and arrange them in the same manner as in Figure 12 (from left to right, top to bottom). When a photo is clicked, a new tab is opened to display that photo in its original size.

Note: Broken Images in the Photos tab is fine.

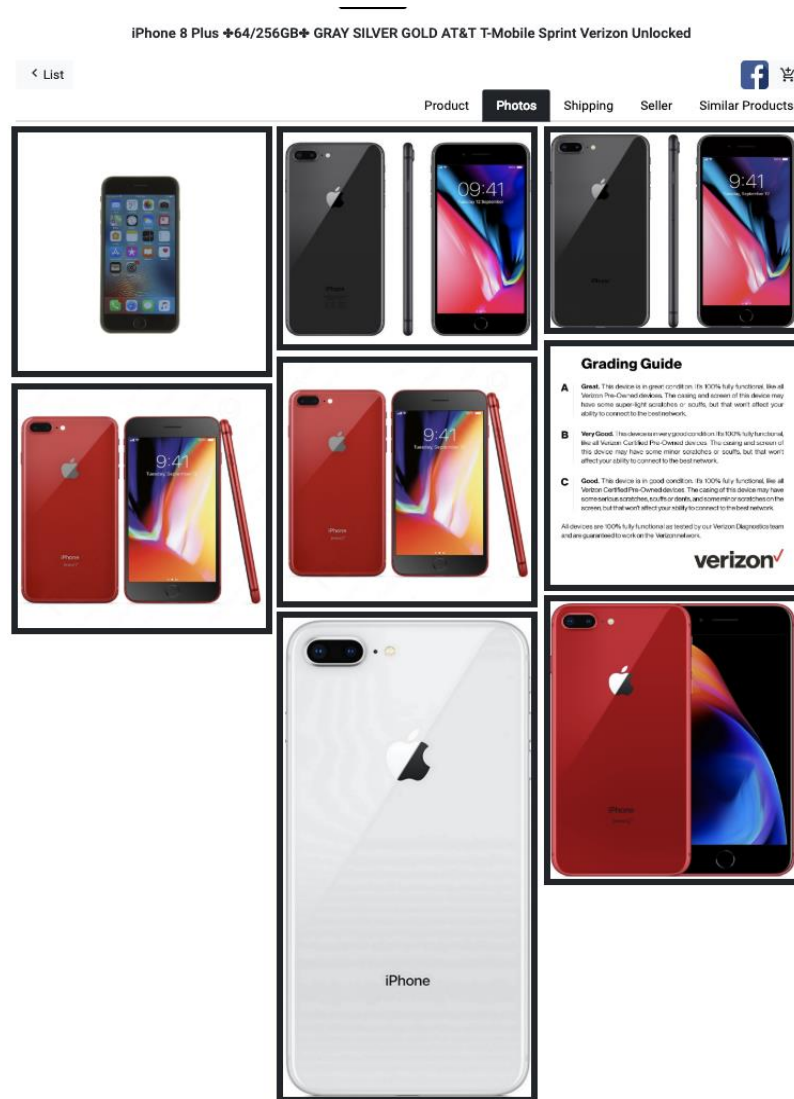


Figure 12 Multiple Photos Tab

3.3.3 Shipping Tab

To get the Shipping info, with the help of first API call you get the shipping information for each product.

A table containing the detailed info of the Shipping details is displayed in this tab. The table has the following fields if they are available in the detail search results:

Fields in the info table	Corresponding response object fields
Shipping Cost	The value of the “__value__” in “shippingServiceCost” object that is part of the “shippingInfo” object that is part of the “item” object.
Shipping Locations	The value of the “shipToLocation” attribute of “shippingInfo” object that is part of “item” object.
Handling time	The value of the “handlingTime” attribute of “shippingInfo” object that is part of “item” object.
Expedited Shipping	The value of the “expeditedShipping” attribute of “shippingInfo” object that is part of “item” object.
One Day Shipping	The value of the “oneDayShippingAvailable” attribute of “shippingInfo” that is part of the “item” object.
Return Accepted	The value of the “returnsAccepted” attribute that is part of the “item” object.



Table 4: Mapping the result from eBay API into HTML table

A sample of Shipping tab is shown as Figure 13.

Note: If any value is missing, don’t display that row in shipping info tab.

For Expedited Shipping, One Day Shipping and Return Accepted if the value is true then a green tick mark is to be displayed else a red cross is to be displayed. Please refer Material Icons for the tick marks and red cross. For **Handling Time** if the value is 0 or 1 then append “Day”, if greater than 1 then append “Days”.

iPhone 8 Plus 64/256GB GRAY SILVER GOLD AT&T T-Mobile Sprint Verizon Unlocked

< List  

Product Photos **Shipping** Seller Similar Products

Shipping Cost	Free Shipping
Shipping Locations	Worldwide
Handling Time	1 Day
Expedited Shipping	✓
One Day Shipping	✓
Return Accepted	✓

Figure 13 Shipping Tab

3.3.4 Seller Tab

To get the Seller info, the eBay Shopping API will give you the details for the seller and the store.

A table containing the detailed info of the store details of the seller is displayed in this tab. The table has the following fields, if they are available in the detail search results:

Fields in the info table	Corresponding response object fields
Feedback Score	The value of the “ <i>FeedbackScore</i> ” in “ <i>Seller</i> ” object that is part of the “ <i>item</i> ” object.
Popularity	The value of the “ <i>PositiveFeedbackPercent</i> ” in “ <i>Seller</i> ” object that is part of “ <i>item</i> ” object.
Feedback Rating Star	The value of the “ <i>FeedbackRatingStar</i> ” in “ <i>Seller</i> ” object that is part of “ <i>item</i> ” object.
Top Rated	The value of the “ <i>TopRatedSeller</i> ” in “ <i>Seller</i> ” object that is part of “ <i>item</i> ” object.
Store Name	The value of the “ <i>StoreName</i> ” in “ <i>Storefront</i> ” that is part of the “ <i>item</i> ” object.
Buy Product At	The value of the “ <i>StoreURL</i> ” in “ <i>Storefront</i> ” that is part of the “ <i>item</i> ” object.

Table 5: Mapping the result from eBay API into HTML table

Note: If any value is missing don’t display the row in Seller info tab.

The **Popularity** should be provided with a round progress bar. For Angular2+ users, please use <https://github.com/crisbeto/angular-svg-round-progressbar>

For AngularJS users, please use <https://github.com/crisbeto/angular-svg-round-progressbar/tree/angular-1.x>

The **Feedback Rating Star** will be based on the color, the corresponding material icon star with that color should be displayed. For the color of stars: <https://developer.ebay.com/devzone/finding/callref/types/SellerInfo.html>

For a value of Feedback score greater than or equal to 10000, use **stars** else use **star_border** with the colors mentioned from material icons. (Refer 5.5 for the icons) For **Top Rated** if true display a green tick else display a red cross. On Clicking **Store**, the link for the store should open in a new tab.

Apple iPhone 8 Plus 64GB | 256GB (UNLOCKED) VERIZON Gray | Silver | RED ♦OTHER♦

< List

Product Photos Shipping **Seller** Similar Products

SANTAMONICAWIRELESS	
Feedback Score	16601
Popularity	99.6
Feedback Rating Star	★
Top Rated	✓
Store Name	Santa Monica Wireless
Buy Product At	Store

Figure 14 Seller Tab

3.3.5 Similar Products Tab

This tab displays the similar products of the product clicked from the first table. Details for the parameter of the API call is given in the HW6 description. 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=20](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=20)

Note: Please set the maxResults to 20.

The sample API response will be:

```
getSimilarItemsResponse:
  ack: "Success"
  version: "1.1.0"
  timestamp: "2019-02-02T03:02:287.023Z"
  itemRecommendations:
    item:
      0:
        itemId: "591635098336"
        title: "NEW Nike USC Trojans - Black Poly Short Sleeve Shirt (Multiple Sizes)"
        viewItemURL: "https://www.ebay.com/itm/NEW-Nike-USC-Trojans-Black-Poly-Short-Sleeve-Shirt-Multiple-Sizes/292862774875?var=591635098336_trkparms%3D%3A591635098336_trksid%3D%3A292862774875"
        globalId: "EBAY-US"
        timeLeft: "P7DT20H47M37S"
        primaryCategoryId: "64482"
        primaryCategoryName: "Sports Mem, Cards & Fan Shop"
        country: "US"
        imageURL: "https://securethumbs.ebay.com/d/1140/m/m-Bvaz93HDQCAu831QtckVw.jpg"
        shippingType: "NotSpecified"
        buyItNowPrice: {}
        shippingCost: {}
      1: {}
      2: {}
      3: {}
      4: {}
      5: {}
      6: {}
      7: {}
```

Figure 15 A sample JSON response returned by the *eBay Merchandise API* for getting similar items.

All fields in the table are present in the “*item*” object in the “*itemRecommendations*” object.

Fields in the info table	Corresponding response object fields
Product Name	The value of the “ <i>title</i> ” attribute. Add hyperlink whose value is “ <i>viewItemURL</i> ”.
Price	The value of the “ <i>__value__</i> ” of “ <i>buyItNowPrice</i> ” attribute.
Shipping Cost	The value of the “ <i>__value__</i> ” of “ <i>shippingCost</i> ” attribute.
Days Left	The value of the “ <i>timeLeft</i> ” attribute. Extract the value between ‘ <i>P</i> ’ and ‘ <i>D</i> ’.

Table 6: Mapping the result from eBay Similar Items API into HTML card

By default, Similar Products are displayed in the default order (the order in which the items are returned by the API). There are two dropdowns in this tab. The first one allows the user to sort the products in several different categories: Default, Product Name, Days Left, Price, and Shipping Cost. The second one allows the user to sort in ascending or descending order. When the sort category is “Default”, the sort order dropdown should be disabled (either Ascending or Descending).



Figure 16: Dropdown to Sort the Products.

By default, only 5 similar products are displayed, like shown in Figure 17(a). After clicking the “Show More” button, all upcoming products in the returned JSON should be displayed, and the button changes to “Show Less”, like in Figure 17(b). After clicking “Show Less” only the top 5 similar products in given sorting order should remain. If the number of products is less than 5, then don’t display “Show More” and “Shore Less” button.

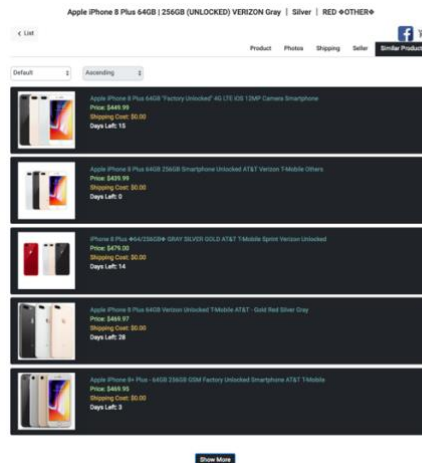


Figure 17(a): Show more button with 5 Products.

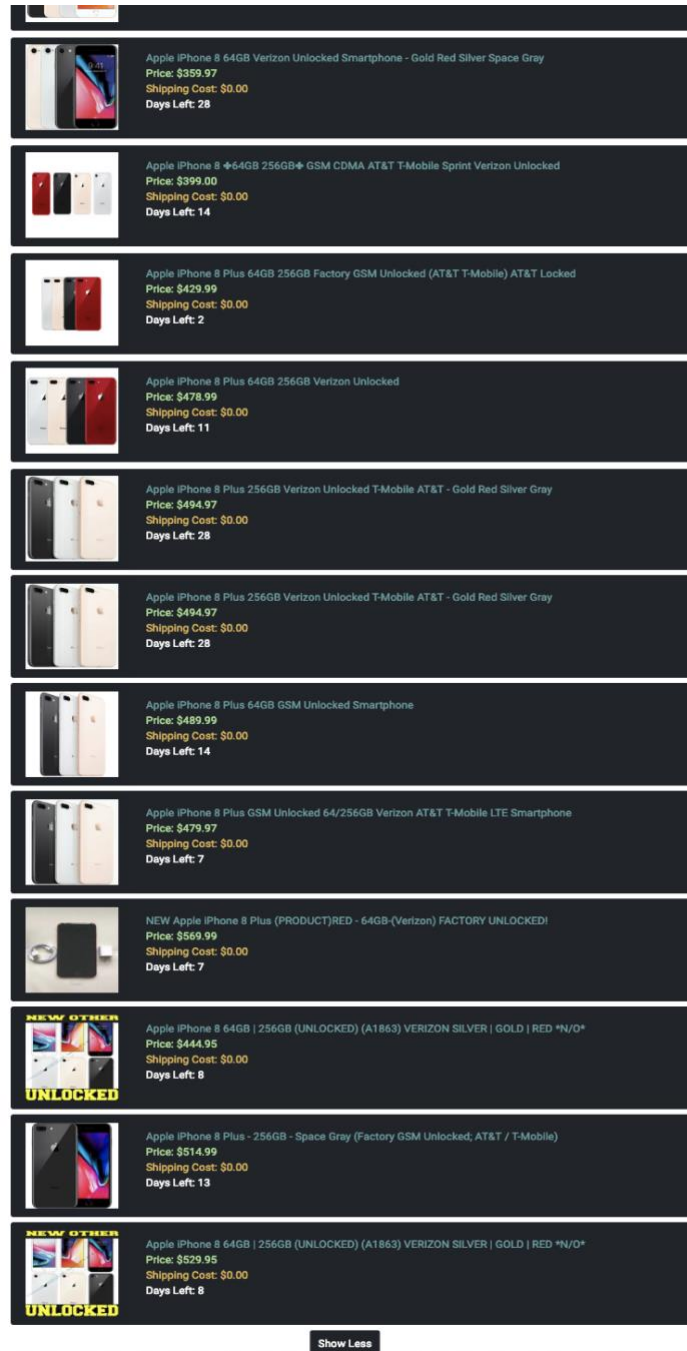


Figure 17(b): Show Less button with multiple Products.

If the API service returns an empty result set, the page should display “No Records Found”.

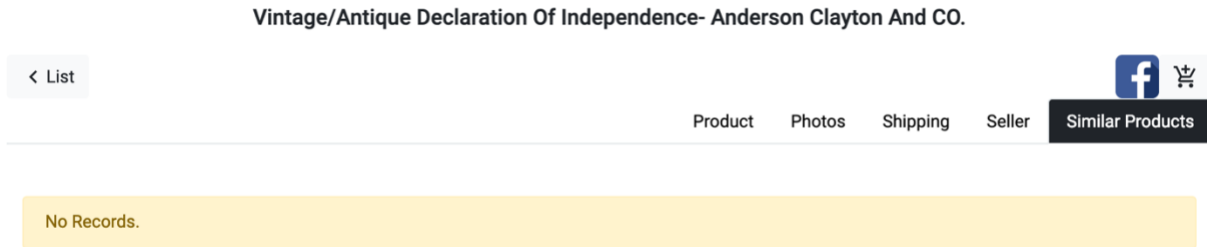


Figure 18: When no Similar Items are found.

3.3.6 List Button, Wish List Button and Facebook Button

Once the **List** button is clicked, your webpage should go back to the previous list view, whether it's the **result list or the Wish List.**



Figure 19 List Button

The **Wish List** button works the same way as the ones in the result list. The **Facebook** button allows the user to share product link and post it to Facebook. Once the button is clicked, a new dialog should be opened and display the default Facebook content in this format: “*Buy PRODUCT_NAME at PRICE from LINK below.*”. Replace PRODUCT_NAME, PRICE and LINK with the real product name, price and viewItemURLForNaturalSearch in the “item” object in eBay Shopping API call.

Link for Facebook share: <https://developers.facebook.com/docs/sharing/reference/share-dialog>

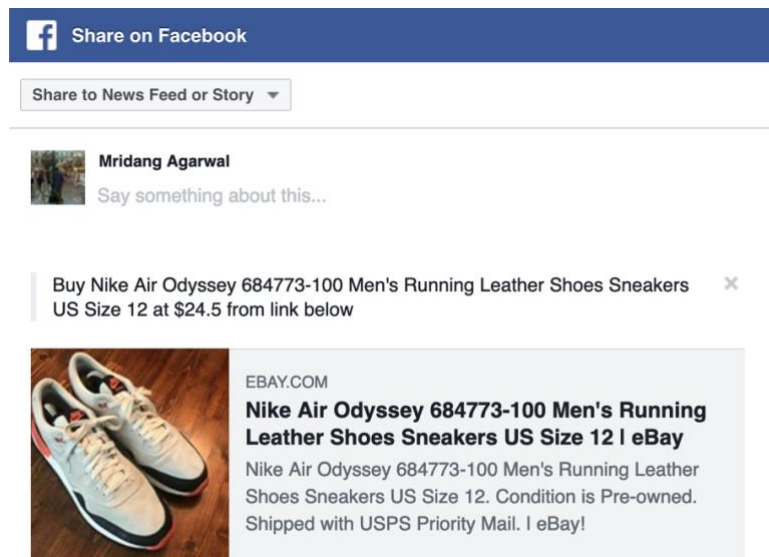


Figure 20 Facebook Share

The Wish List button should maintain the consistency if that product was already added in the Wish List previously or it should not be added.

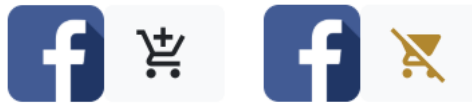


Figure 21 Wish List and Facebook Buttons

3.4 Wish List Tab

The Wish List tab is very similar to the Results tab: the products on the list are displayed in a table; there is a button that allows the user to go to the details view and is disabled initially; the user can search for product details by clicking on the “product” column.

The major differences between these two tabs are: the product information displayed in the Wish List tab is saved in and loaded from the **local storage** of the browser; the buttons in the “Wish List” column of the Wish List tab is only used to remove a product from the list and has an icon for it to be removed from the Wish List. (Refer 5.5 for the icons)

Note: There is no Zip column in the Wish List tab.

There is a **Total Shopping** which stores the total **Price** which is the sum of all products in Wish List.

Please note if a user closes and re-opens the browser, its Wish List should still be there. If there are no products in Wish List, please show ‘No Records’. (see Figure 22)

Results

Wish List

Detail >





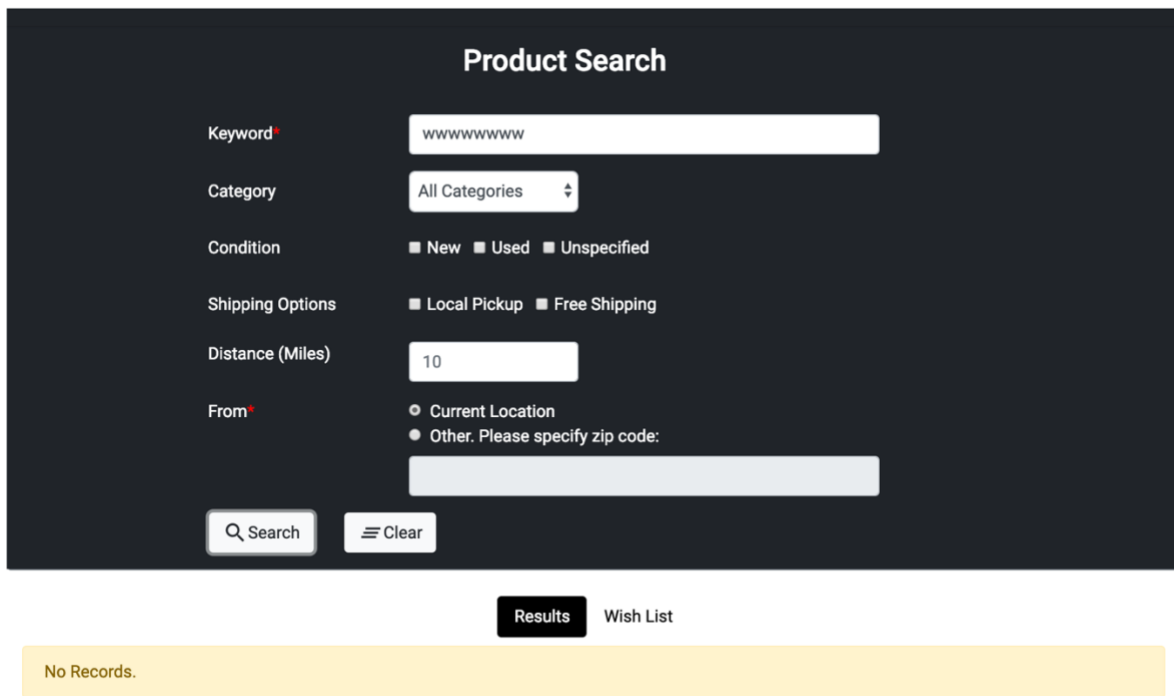
#	Title	Price	Shipping Option	Seller	Favorite
1		Apple iPhone 6+ Plus 16GB 64GB 128GB...	\$164.99	Free Shipping	DEALSCALY 
2		Apple iPhone 5s 16GB 32GB GSM ...	\$80.80	Free Shipping	123321IMKT 
Total Shopping					\$245.79

Figure 22 Wish List

3.5 Error Messages

If for any reason an error occurs whether its products search or details search, an appropriate error message should be displayed.

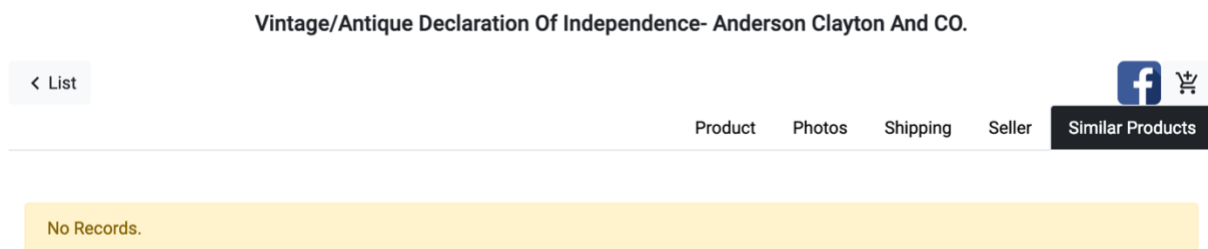


The image shows a 'Product Search' form on a dark background. The form includes fields for 'Keyword' (containing 'wwwwwwww'), 'Category' (a dropdown menu set to 'All Categories'), 'Condition' (checkboxes for 'New', 'Used', and 'Unspecified'), 'Shipping Options' (checkboxes for 'Local Pickup' and 'Free Shipping'), 'Distance (Miles)' (a text input with '10'), and 'From' (radio buttons for 'Current Location' and 'Other. Please specify zip code:'). Below these fields are 'Search' and 'Clear' buttons. At the bottom of the form are 'Results' and 'Wish List' buttons. A yellow banner at the bottom of the form area displays the message 'No Records.'

Figure 23 Error Message

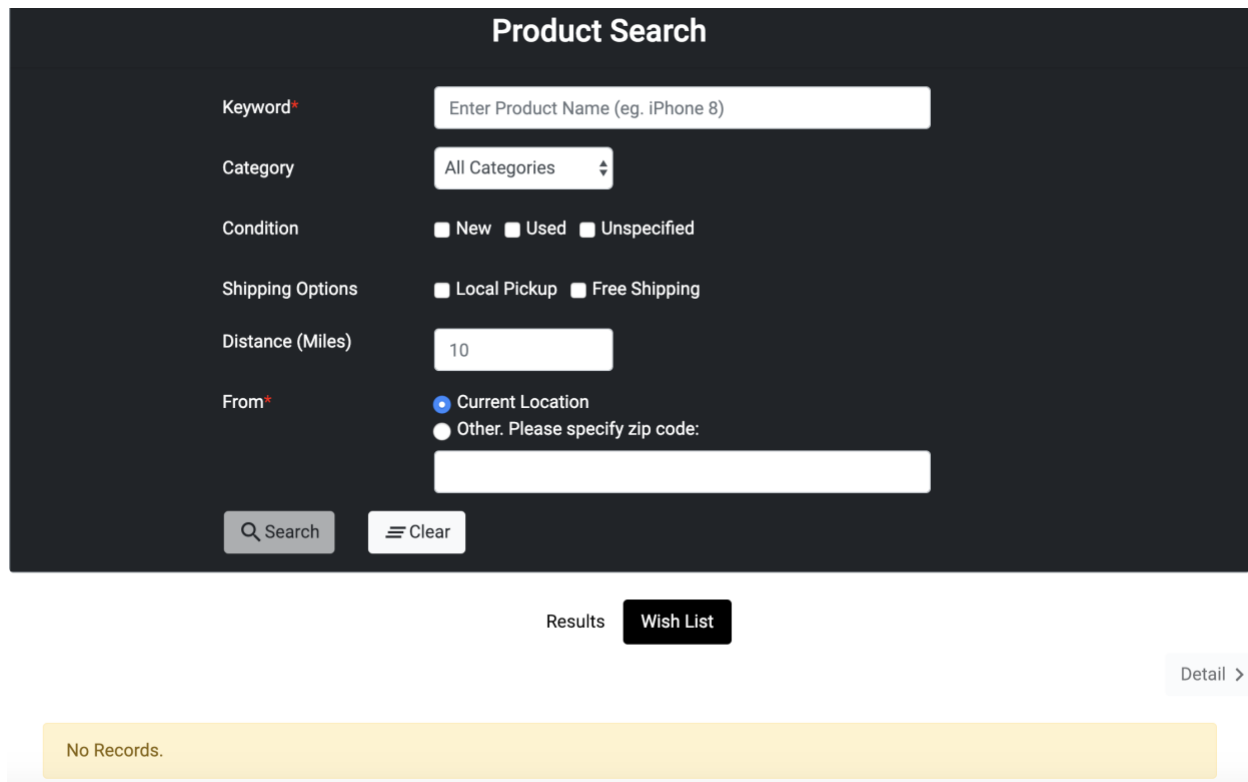
3.6 No Records

Whenever the search receives no records, an appropriate message should be displayed.



The image shows a product details page for 'Vintage/Antique Declaration Of Independence- Anderson Clayton And CO.'. The page has a header with a '< List' button and a navigation bar with links for 'Product', 'Photos', 'Shipping', 'Seller', and 'Similar Products'. The 'Similar Products' link is highlighted. A yellow banner at the bottom of the page displays the message 'No Records.'

Figure 24 No Records on Similar Products



Product Search

Keyword*

Category

Condition ☐ New ☐ Used ☐ Unspecified

Shipping Options ☐ Local Pickup ☐ Free Shipping

Distance (Miles)

From* ☒ Current Location ☐ Other. Please specify zip code:

Results

[Detail >](#)

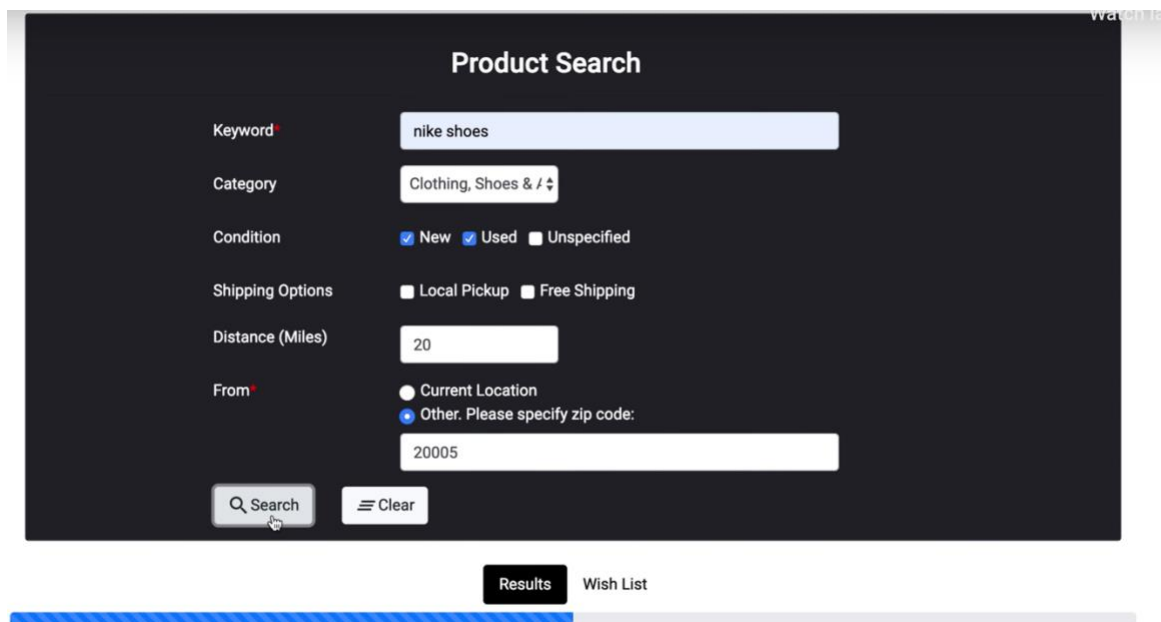
No Records.

Figure 25 No Records on Wish List

3.7 Progress Bars

Whenever data is being fetched, a dynamic progress bar must be displayed as shown in Figure 23.

You can use the progress bar component of **Bootstrap** to implement this feature. You can find hints about the bootstrap components in the Hints section.



Product Search

Keyword*

Category

Condition ☒ New ☒ Used ☐ Unspecified

Shipping Options ☐ Local Pickup ☐ Free Shipping

Distance (Miles)

From* ☐ Current Location ☒ Other. Please specify zip code:

Progress Bar: A blue progress bar is shown at the bottom of the form, indicating that data is being fetched.

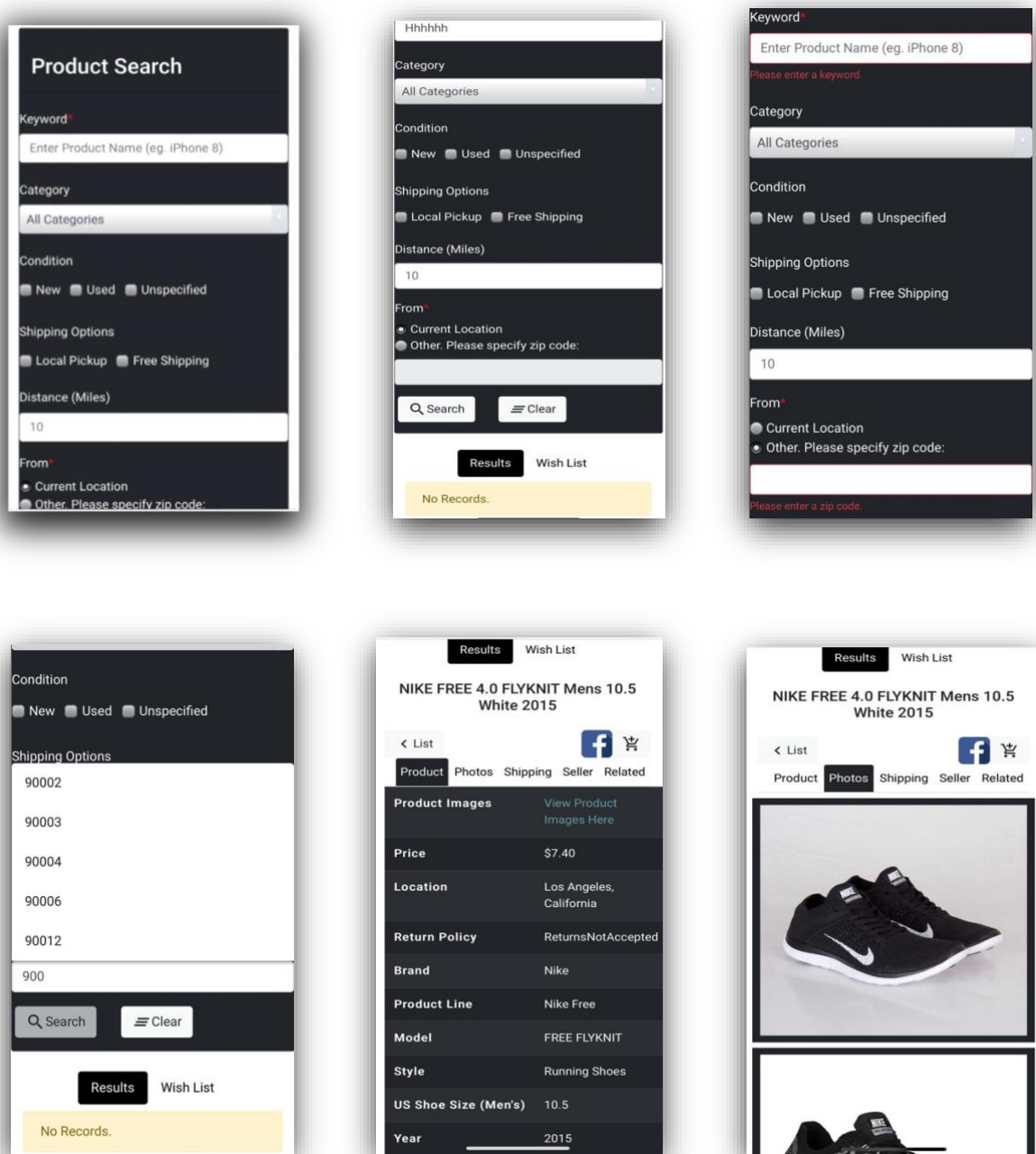
Figure 26 Progress Bar

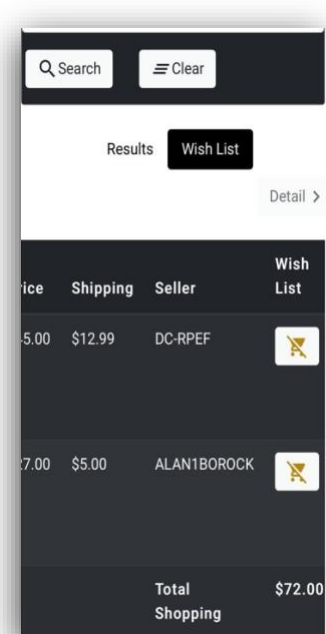
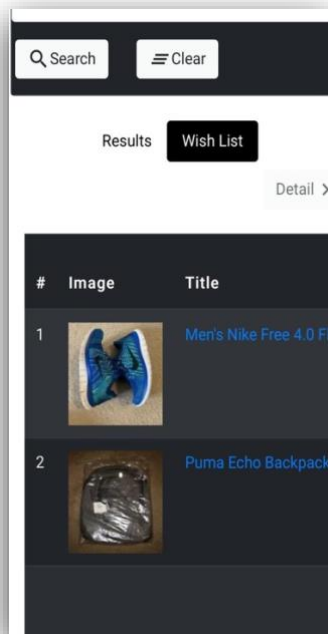
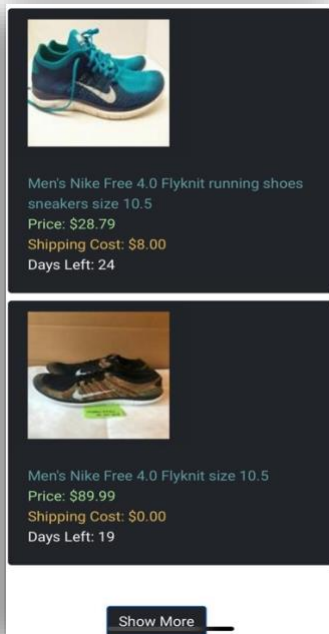
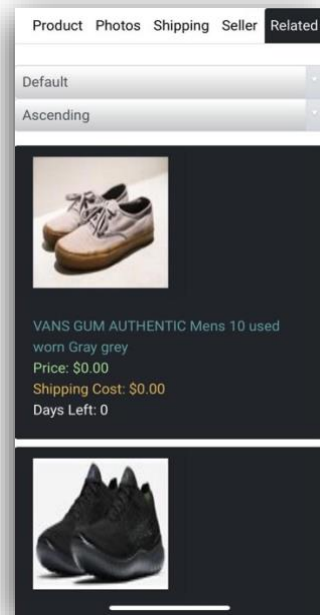
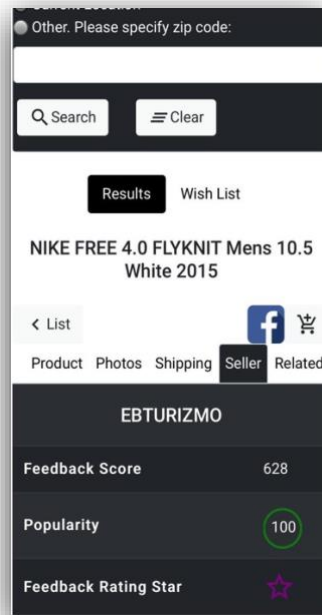
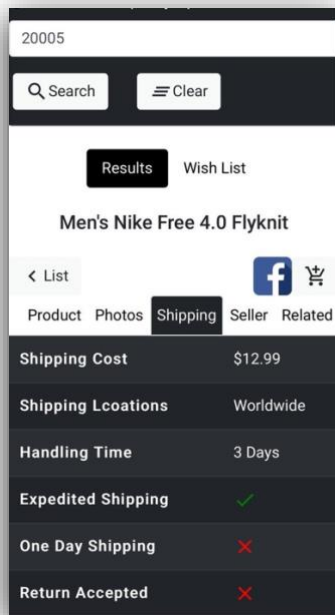
3.8 Animation

Whenever the view switches between Results/Wish List and details, there should be a sliding effect. Please check out the video to see the effect. **The sliding animation must be implemented with AngularJS/Angular2+.**

3.9 Responsive Design

The following are snapshots of the webpage opened with Safari on iPhone 7 Plus. See the video for more details.





Note: The Similar Products in the tabs is replaced by Related in the Responsive design. Make sure all tabs remain in 1 line for responsive design.

Some of the requirements in the mobile view are listed here:

- The search form should display each component in a vertical way (“stacked”) on smaller screens.

- All tables can be scrolled horizontally (“panned”).
- The photos should be aligned vertically, and use 100% width, on smaller screens.
- Animations must work on mobile devices.

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 webpage works perfectly on a desktop, it may not work as perfect 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. API Documentation

4.1 eBay API

To use eBay API, you need first to register for an eBay Account. This is the same App id used for the HW6. You can re-use it.

4.2 Google Customized Search API

This link will provide the details to get the API key:

<https://developers.google.com/custom-search/json-api/v1/overview>

Note: You can use any additional Angular libraries and Node.js modules you like.

5. Implementation Hints

5.1 Images

The images needed for this homework are available here:

<http://csci571.com/hw/hw8/Images/facebook.png>

5.2 Get started with the Bootstrap Library

To get started with the Bootstrap toolkit, please refer to the link:

<https://getbootstrap.com/docs/4.0/getting-started/introduction/>.

You need to import the necessary CSS file and JS file provided by Bootstrap.

5.3 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.

At a minimum, you will need to use Bootstrap Forms, Tabs, Progress Bars and Alerts to implement the required functionality.

Bootstrap Forms <https://getbootstrap.com/docs/4.0/components/forms/>

Bootstrap Tabs <https://getbootstrap.com/docs/4.0/components/navs/#tabs>

Bootstrap Progress Bars <https://getbootstrap.com/docs/4.0/components/progress/>

Bootstrap Alerts <https://getbootstrap.com/docs/4.0/components/alerts/>
Bootstrap Tooltip <https://getbootstrap.com/docs/4.1/components/tooltips/>
Bootstrap Cards <https://getbootstrap.com/docs/4.0/components/card/>

5.4 Angular Material

AngularJS Material: <https://material.angularjs.org/latest/>
Autocomplete: <https://material.angularjs.org/latest/demo/autocomplete>
<https://material.angularjs.org/latest/api/directive/mdAutocomplete>
Tooltip: <https://material.angularjs.org/latest/demo/tooltip>

Angular Material (Angular 2+) : <https://material.angular.io/>
Autocomplete: <https://material.angular.io/components/autocomplete/overview>
Tooltip: <https://material.angular.io/components/tooltip/overview>

5.5 Material Icon

Icons for the search button, clear button, left arrow, right arrow, add_shopping_cart, remove_shopping_cart, stars and star_border can be viewed here:

<https://google.github.io/material-design-icons/>
<https://material.io/tools/icons/>

5.6 Google App Engine/Amazon Web Services/ Microsoft Azure

You should use the domain name of the GAE/AWS/Azure service you created in Homework #7 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 JavaScript program will perform a GET request with keyword="xxx", and an example query of the following type will be generated:

GAE - <http://example.appspot.com/searchProducts?keyword=xxx>
AWS - <http://example.elasticbeanstalk.com/searchProducts?keyword=xxx>
Azure – <http://example.azurewebsites.net/searchProducts?keyword=xxx>

Your URLs don't need to be the same as the ones above. You can use whatever paths and parameters you want. Please note that in addition to the link to your Homework #8, you should also **provide a link like this URL in the table of your Node.js backend link**. When your grader clicks on this additional link, a valid link should return a JSON object with appropriate data.

5.7 Deploy Node.js application on GAE/AWS/Azure

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

5.8 AJAX call

You should send the request to the Node.js script(s) by calling an Ajax function (Angular or jQuery). You **must use a GET method** to request the resource since you are required to provide this link to your homework list to let graders check whether the Node.js script code is running in the "cloud" on Google GAE/AWS/Azure (see 5.6 above). Please refer to the grading guidelines for details.

5.9 HTML5 Local Storage

Local storage is more secure, and large amounts of data can be stored locally, without affecting website performance. Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server. There are two methods, getItem() and setItem(), that you can use. The local storage can only store strings. Therefore, you need to convert the data to string format before storing it in the local storage. For more information, see:

<https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage>

http://www.w3schools.com/html/html5_webstorage.asp

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/AWS/Azure service where the AJAX call is made with sample parameter values (i.e. a valid query, with keyword, location, etc. See 5.6).

Also, electronically submit all files (HTML, JS, CSS, TS) the you have personally written to the GitHub Classroom repository so that they can be compared to all other students' code. **Don't upload / push library, node-js modules, any config files, any angular-cli build files, or any images that we provided or that are included in any library or any code generated by the tools.**

Please note, you need to submit your files (HTML, JS, CSS, TS), both backend and frontend, directly to the root folder of your homework 8 GitHub repository. That is, **do not submit any subfolders on GitHub.** If you submit some subfolders, you will receive 3-points penalty.

****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.