HOMEWORK ASSIGNMENT 3

CSCI 571 – Fall 2023

Abstract

Ajax, JSON, Angular, Bootstrap, MongoDB, RWD, Node.js, and eBay API

Marco Papa

papa@usc.edu

Homework 3: Ajax, JSON, HTML5, Bootstrap, Angular, MongoDB, Node.js, Cloud and eBay API Assignment

1. Objectives

- Get familiar with the AJAX and JSON technologies.
- Use a combination of HTML5, Bootstrap and Angular on the client side.
- Use Node.js Express framework on the server side.
- Get familiar with Bootstrap to enhance the user experience using responsive design.
- Get experience with MongoDB as NoSQL database.
- Get hands-on experience on Amazon Web Services/Google Cloud Platform/Microsoft Azure.
- Learn to use popular APIs such as eBay API and Google Customized Search API.

2. General Directions

1. The backend of this homework must be implemented in JavaScript using the Node.js Express framework. Refer to Node.js website for installing Node.js and learning how to use it. Have a look at "Getting started" guides in the Express website to learn how to create backend applications using Express. Also, refer to the Node.js Express framework tutorial to learn more about it. The Axios library can be useful to make requests from your Node.js backend to eBay servers, or you can use any library with similar functionality as you like. **Implementing the backend in anything other than Node.js will result in a major point reduction.**

2. The frontend of this homework should be implemented using the Angular framework. Refer to the Angular setup docs for installing Angular and creating Angular projects. The Angular tutorial is a very good tutorial to see different Angular concepts in action. **Implementing the frontend in anything other than Angular or React will result in a major point reduction.**

3. You are expected to create a responsive website. For that reason, we require you to use **Bootstrap**, a CSS framework for responsive, mobile-first web development. Bootstrap will save you from the burden of dealing with CSS peculiarities and the website you create will be responsive automatically, if you develop within the framework provided by Bootstrap. Please refer to the Bootstrap docs for reference (especially look at the "Layout" section, and the components that you want to use). Refer to this post to learn how to add Bootstrap to Angular projects. **Not using Bootstrap will result in a major point reduction**.

4. The backend and frontend of this homework must be deployed to the cloud on GCP, Azure, or AWS. The backend should serve the frontend as well as other endpoints you may define.
5. In this homework you will use MongoDB as a database to store your product wish list. Refer to section 5.6 for more information on how to deploy MongoDB on GCP, AWS, or Azure.

6. You must refer to the homework description document (this document), grading guidelines, and the reference videos and instructions on Piazza while developing this homework.

7. The assignment will be graded using the latest version of the Google Chrome browser. Developing this assignment using the latest version of Google Chrome is recommended.

All APIs calls should be done through your Node.JS server, except calls to the ipinfo.io.

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. It includes three parts: search form, result table and item info card.

Your webpage should also support adding products to and removing products from the wish list and sharing products info using 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**.

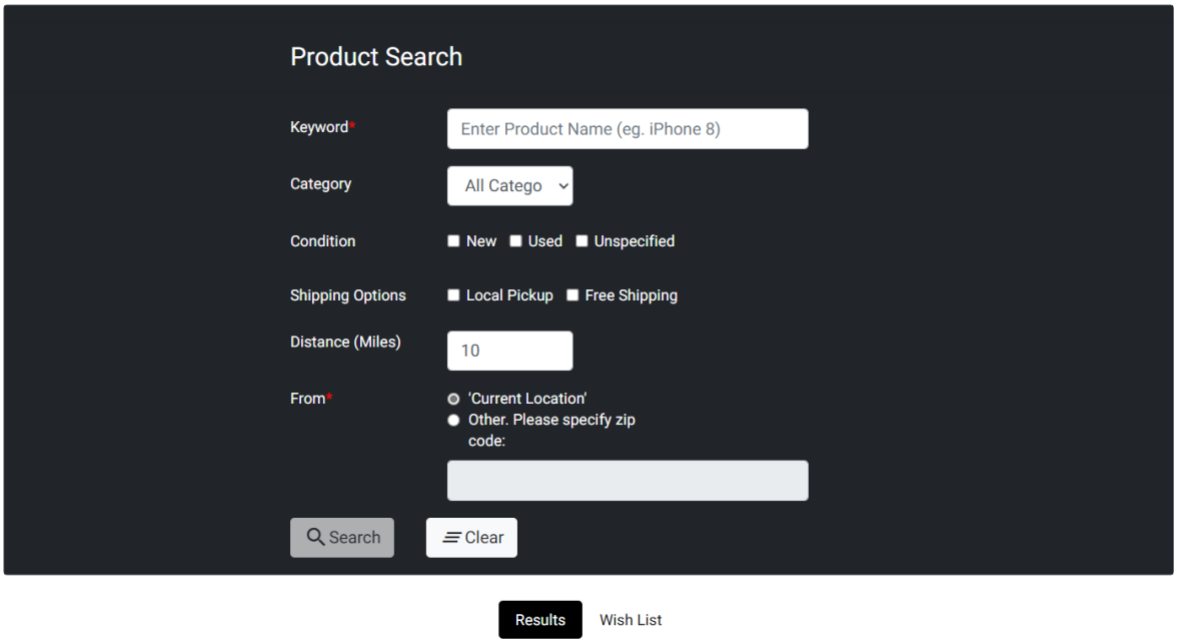


Figure 1 Initial Search Form

3.1 Search Form

3.1.1 Design

You must replicate the search form displayed in **Figure 1** using **Bootstrap**, so that it would be a responsive search form. The form fields are like Assignment #2.

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.

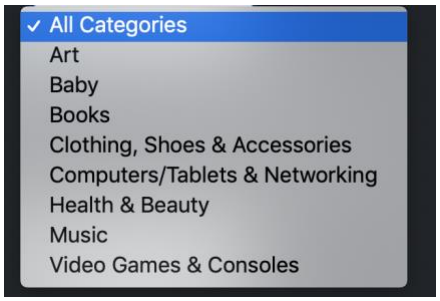


Figure 2 Category Options

4. **Shipping Options**: There are 2 options provided: **Local Pickup** and **Free Shipping**. The user has the option to select 1 or both.
5. **Distance (Miles)**: This is the radius of the area within which the search is performed. 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. When the second option is not chosen, the input box below the second radio button should be disabled. If the user chooses to provide a different location, this input field should be enabled.

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, and clear the results area including result table and item info card.

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&maxRows=5&username=[Username]&country=US

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.
3. Do not use the 'demo' account for your app or your tests. It is only meant for the sample links on the documentation pages. Create your own account instead.

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 can use **Angular Material** to implement the Autocomplete. (See section 5.3).

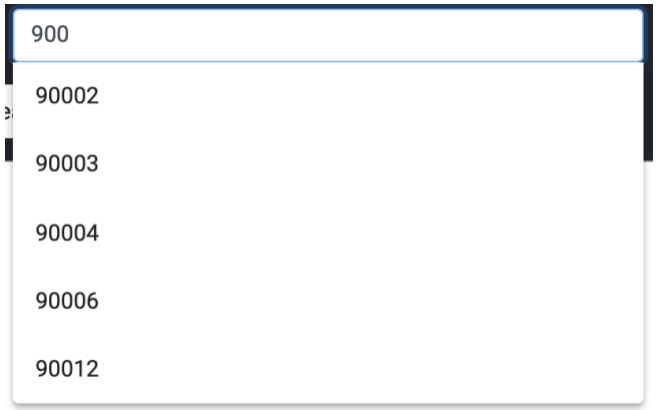


Figure 4 Autocomplete example

3.1.3 Validation

Your application should check if the "Keyword" text box contains something other than spaces. If not, then it's invalid, 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 the "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.

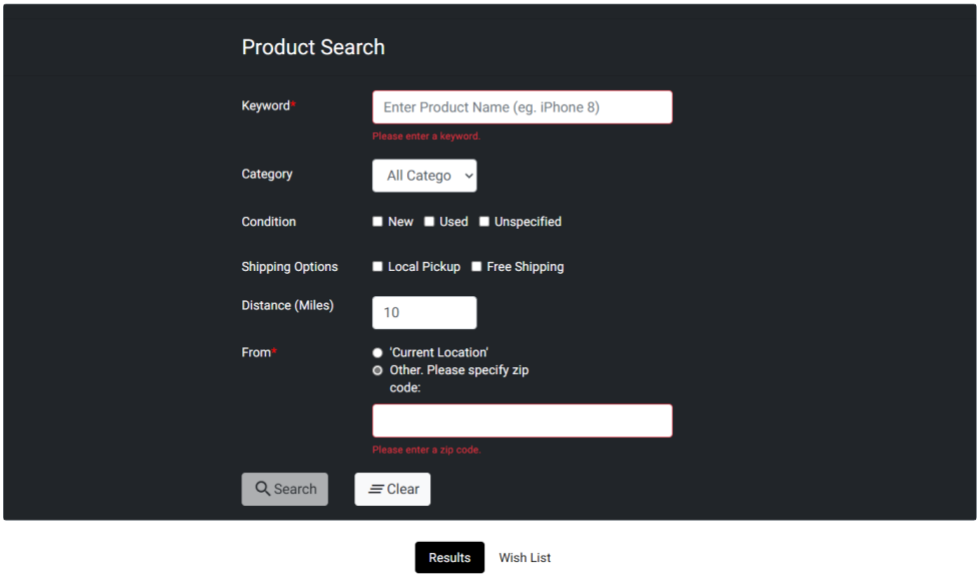


Figure 5 An Invalid Form

3.1.4 Obtaining User Location

As in Assignment #2, you should obtain the current user location using geolocation APIs. You can use the following api or any other apis with similar functionality.

https://ipinfo.io/json?token=your_token

3.1.5 Search Execution

Once the validation is successful and the user clicks on the "Search" button, your application should make a call to the Node.js backend hosted on GCP/AWS/Azure. The Node.js script on GCP/AWS/Azure will then make a request to the eBay API to get products' 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 usage of eBay APIs has been explained in the Assignment #2 documents.

The Node.js script should pass the JSON object returned by the eBay API 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, and Wish list.

Below the table there is a pagination feature which is used to navigate to the next set of products. There are 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.

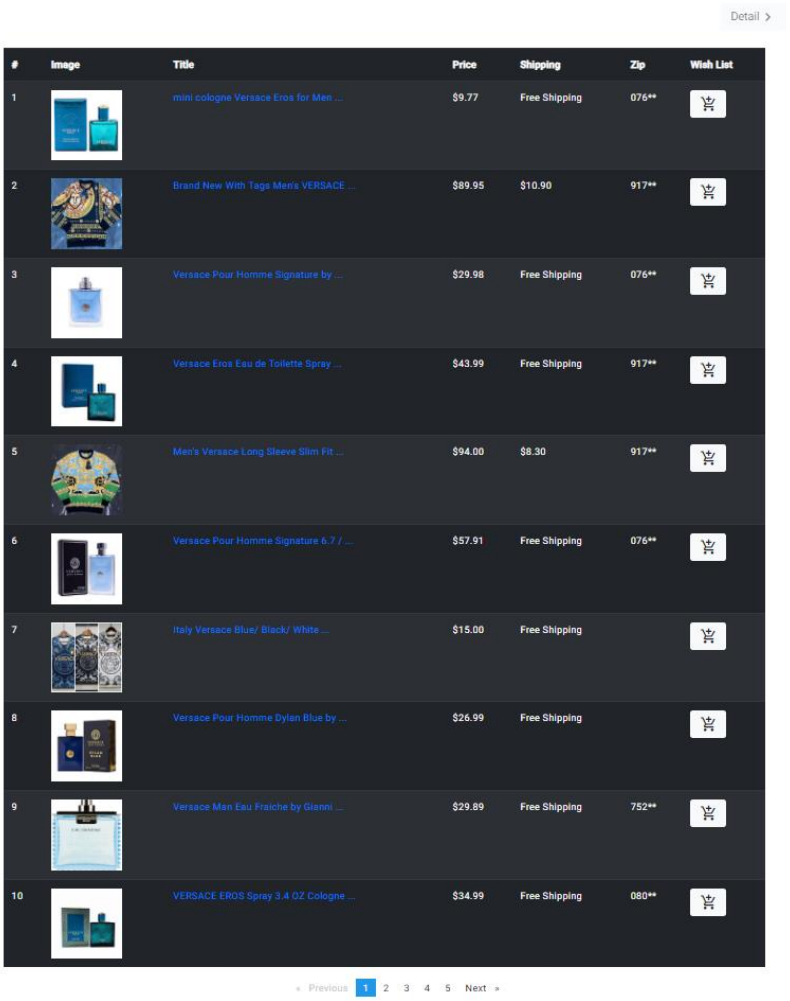


Figure 6 An Example of a Valid Search result

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 "*index*" is a number ranging from 1 to 50 for all results.
Image	The value of the "*Photos*" attribute is a part of the "*galleryURL*" object inside the "*item*" object.
Title	The value of the "*title*" attribute inside the "*item*" object.
Price	The value of the "__*value*__" attribute that is part of the "currentPrice" object inside the "*sellingStatus*" object which is part of the "*item*" object.
Shipping	The value of the "__*value*__" inside the "*shippingServiceCost*" attribute is part of the "*shippingInfo*" object inside the "*item*" object. Set the value to "*Free Shipping*" if the "*shippingServiceCost*" is 0.0 or set it to the value in dollars if it is not zero. Set it to "*N/A*" if no information is given.
Zip	The value of the "*postalCode*" attribute that is part of the "*item*" 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

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:
https://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 Assignment 2

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).**

If any value is missing in the first table, please display "N/A".

When you click the button under the Wish List column, it should add or remove this item's information into your MongoDB database that is on the cloud platform (GCP/AWS/Azure) See section 3.4 Wish List Tab and 5.6 MongoDB for more information.

If a particular image in the table is clicked, it should open in a new tab. The "#" column starts from 1 and goes until 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 ellipses "…" to avoid starting a new row. The "Title" column is clickable and triggers 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 these ideas to avoid starting a new row for a 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 ellipses '…' to the new string.
4. Show the tooltip of the whole product name (see **Figure 7**).

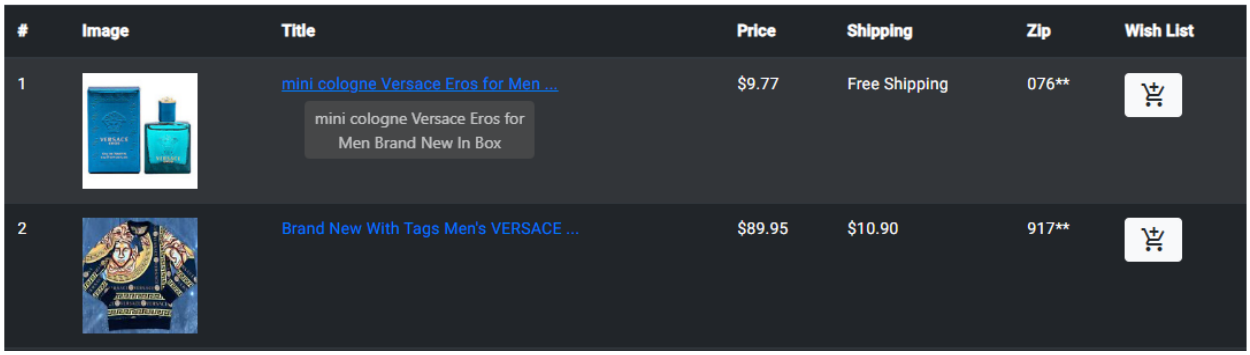


Figure 7 An Example of the tooltip for product name

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 (see **Figure 8**).

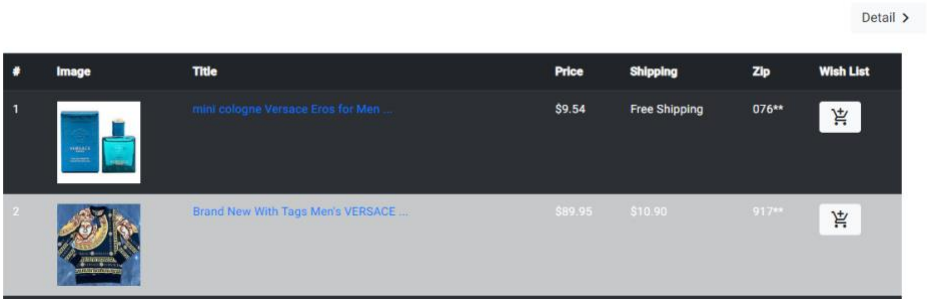


Figure 8 Highlight Selected Product

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. The Wish List button should have the same functionality as the Wish List button in the result table.

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:

https://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:

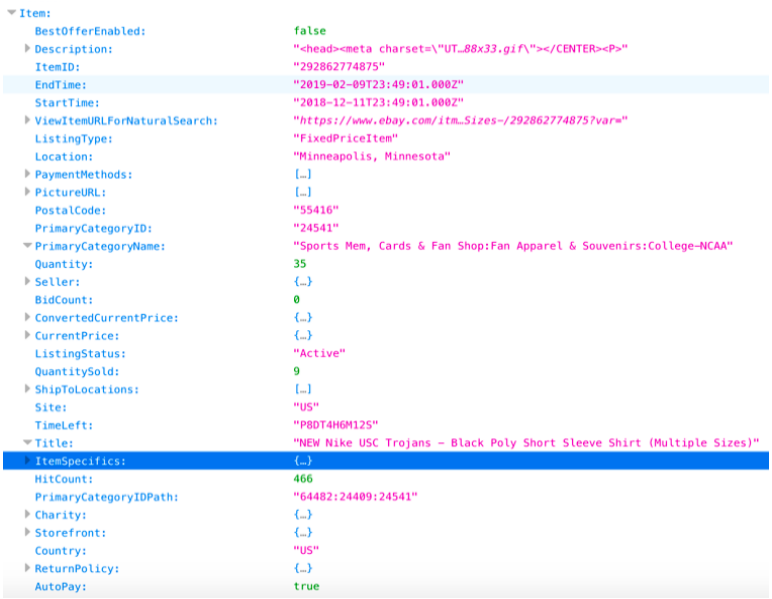


Figure 9: A sample JSON response from eBay Shopping API to get a Single Item.

3.3.1 Info Tab

A table containing the detailed info of the product is displayed in this tab. The table has the fields shown in **Table 2**, if they are available in the detail search results:

Fields in the table	Corresponding response object fields
Product Images	The value of the "*PictureURL*" attribute that is part of the "Item" object. It is a modal containing all pictures of the "*PictureURL*" array.
Price	The value of the "*Value*" attribute that is part of the "*currentPrice*" object inside the "*Item*" object.
Location	The value of the "*Location*" attribute that is part of the "*Item*" object.
Return Policy (US)	The value of the "*ReturnPolicy*" attribute that is part of the "*Item*" object. It is composed of two fields: "*ReturnsAccepted*" and "*ReturnsWithin*".
ItemSpecifics (Name)	The value of the all the "*NameValueList*" array corresponding to that name inside the "*ItemSpecifics*" object in the "Item" object.

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

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

When clicked on "Product Images", a modal should pop 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.

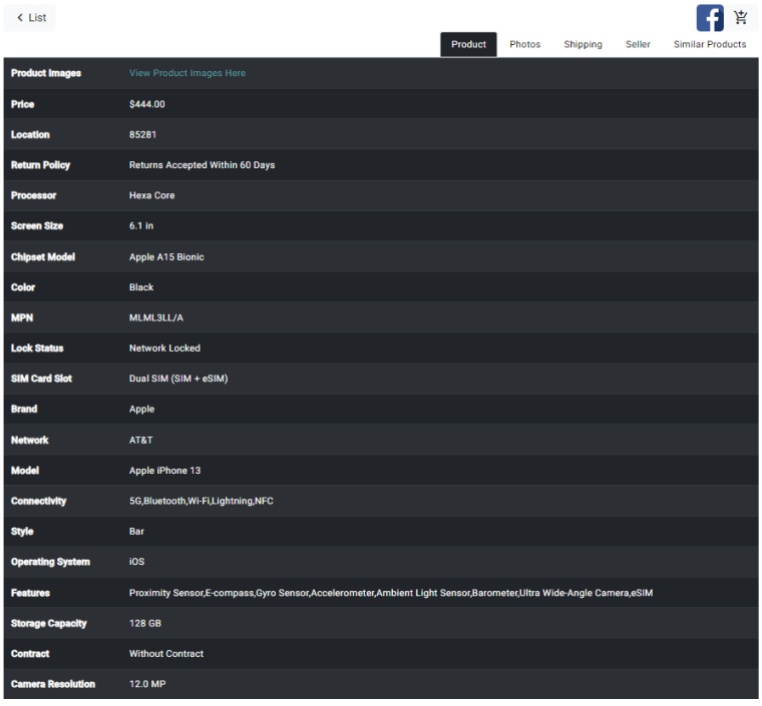


Figure 10 An Example of Product Detail Tab

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**. **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 are fine.

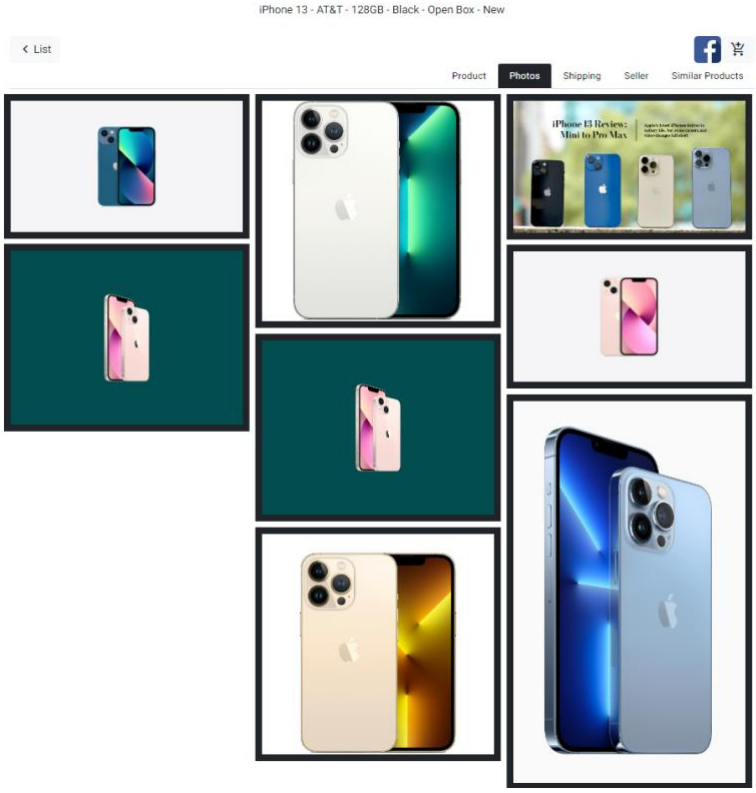


Figure 12 Multiple Photos Tab

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 in **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 to section 5.1 Image 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".

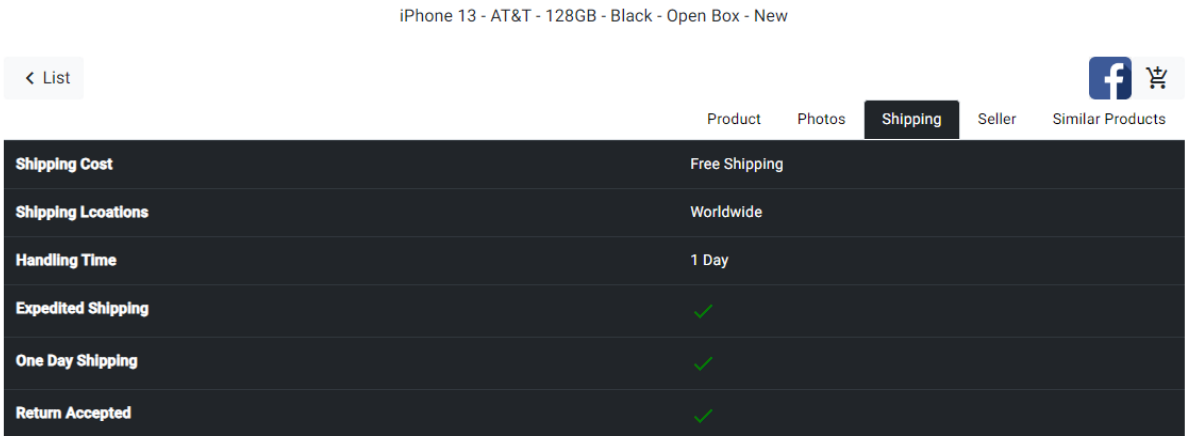


Figure 13 Shipping Tab

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 "*FeedbackScore*" in *"Seller"* object that is part of the "*item*" object.
Popularity	The value of the *"PositiveFeedbackPercent"* in "*Seller*" object that is part of "*item*" object.
Feedback Rating Star	The value of the *"FeedbackRatingStar" in* "*Seller*" object that is part of "*item*" object.

Top Rated	The value of the *"TopRatedSeller" in* "*Seller*" object that is part of "*item*" object.
Store Name	*The value of the "StoreName" in "Storefront" that is part of the "item" object.*
Buy Product At	*The value of the "StoreURL" in "Storefront" that is part of the "item" 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

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 to section 5.1 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.

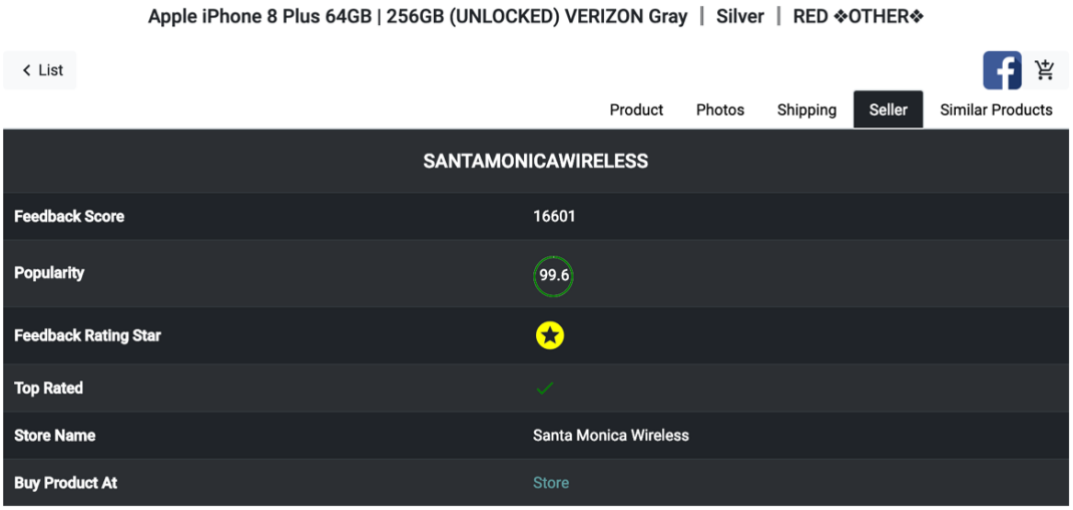


Figure 14 Seller Tab

3.3.5 Similar Products Tab

This tab displays the similar products of the product clicked from the first table. An example of an HTTP request to the *eBay Merchandise API* that searches for similar products to the product clicked initially is shown below:

https://svcs.ebay.com/MerchandisingService?OPERATION-NAME=getSimilarItems&SERVICE-NAME=MerchandisingService&SERVICE-VERSION=1.1.0&CONSUMER-ID=[APP-ID]&RESPONSE-DATA-FORMAT=JSON&REST-PAYLOAD&itemId=394884844909&maxResults=20

Note: Please set the maxResults to 20.

The sample API response will be:

Figure 15 A 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 *"title"* attribute. Add hyperlink for who has"*viewItemURL*".
Price	The value of the "*__value__*" of *"buyItNowPrice"* attribute.
Shipping Cost	The value of the "*__value__*" of *"shippingCost"* attribute.
Days Left	*The value of the "timeLeft" attribute.* Extract the value between '*P*' and '*D*'.

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 Shopping 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.

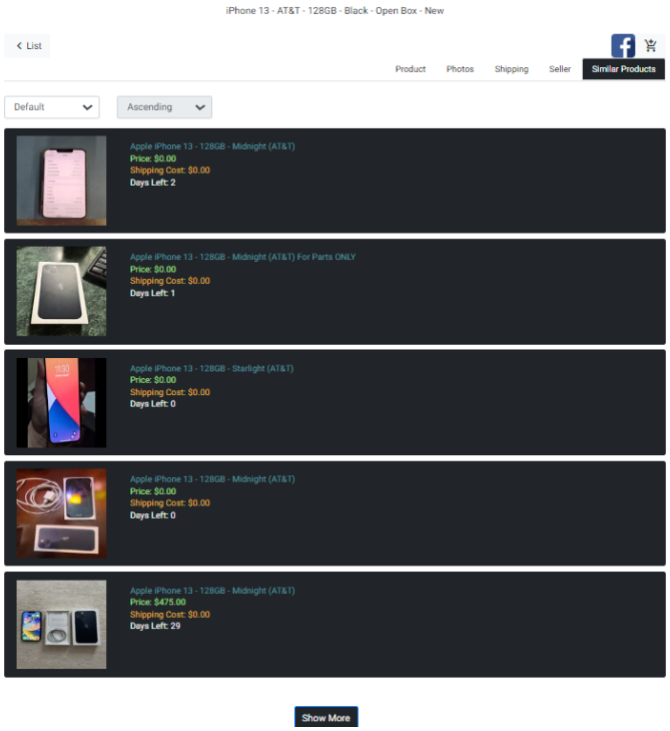


Figure 17(a): Show more button with 5 Products.

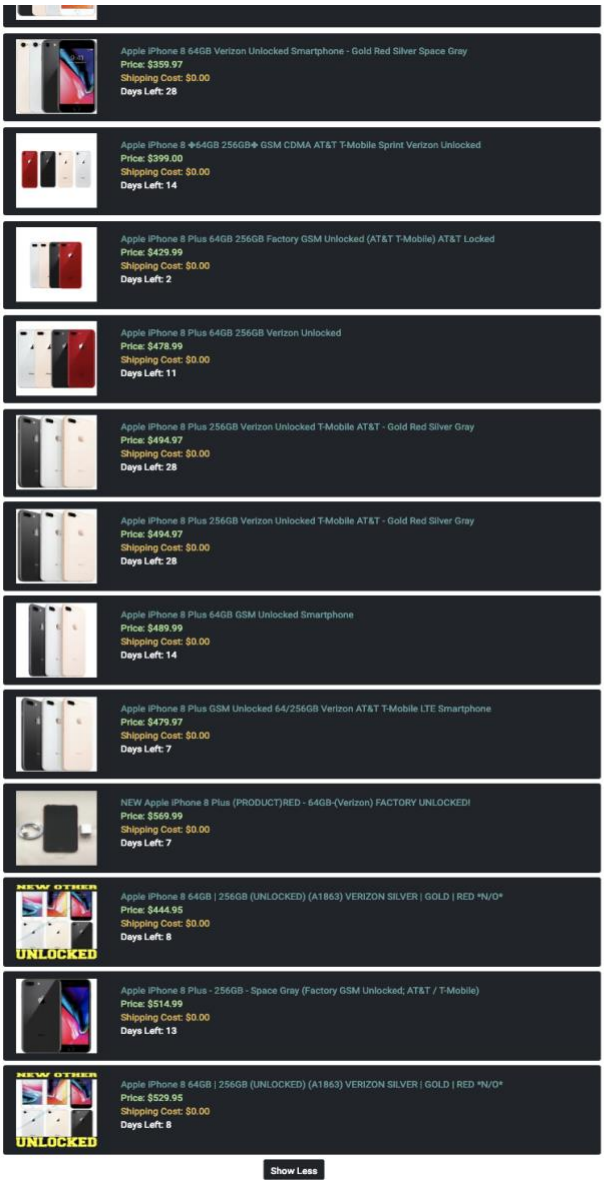


Figure 17(b): Show Less button with multiple Products.

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

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.

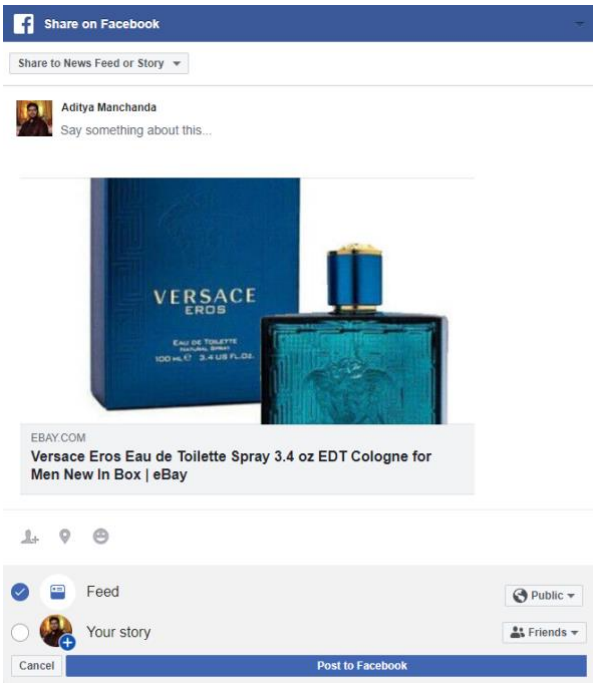


Figure 20 Facebook Share

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 name in 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 MongoDB on the cloud platform (GCP/AWS/Azure); 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.

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

There is a **Total Shopping** label which displays the total **Cost**, which is the sum of all products prices in the Wish List (see **Figure 22**).

Please note if a user closes and re-opens the browser, or changes browser, its Wish List should still be there. If there are no products in the Wish List, please show 'No Records'.

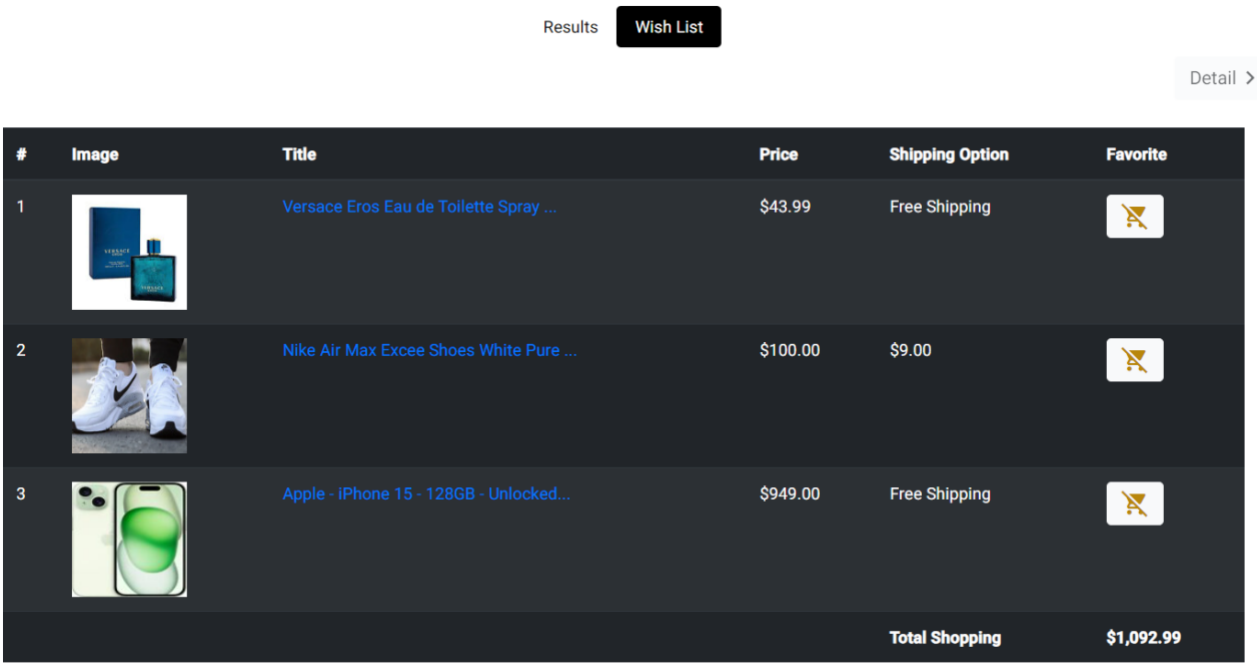


Figure 22 Wish List

3.5 No Result Messages

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

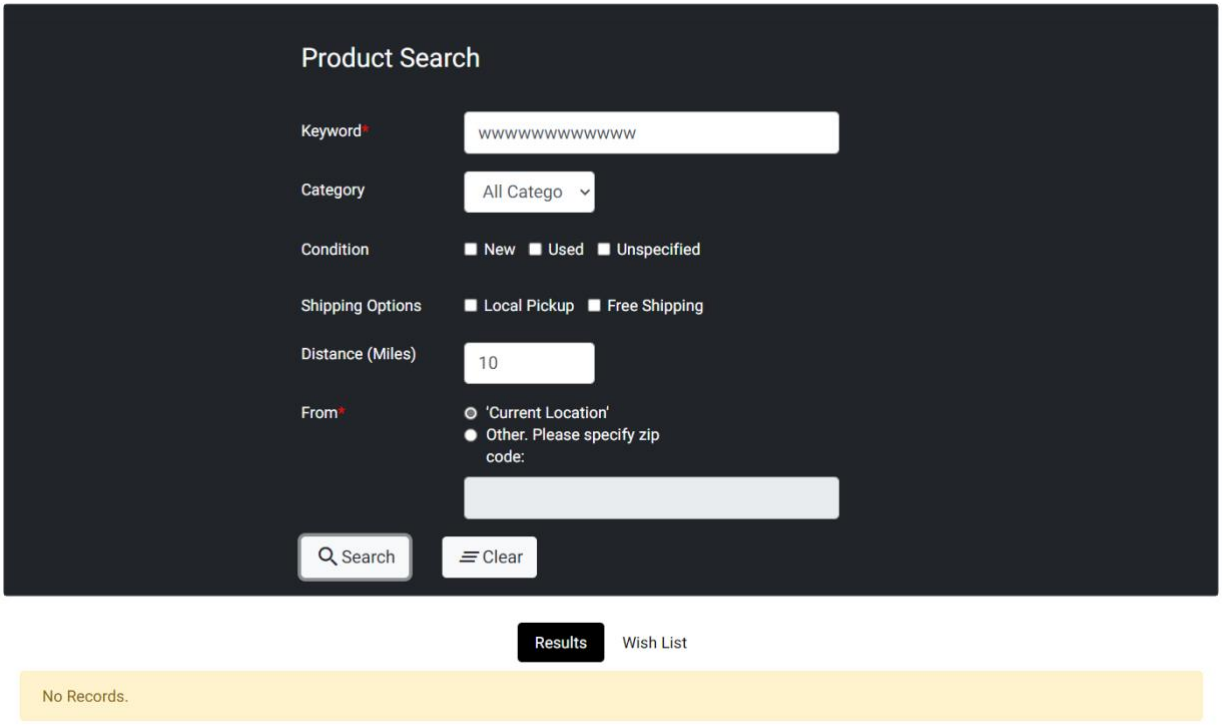


Figure 23 Error Message

3.6 No Records

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

Figure 24 No Records on Similar Products

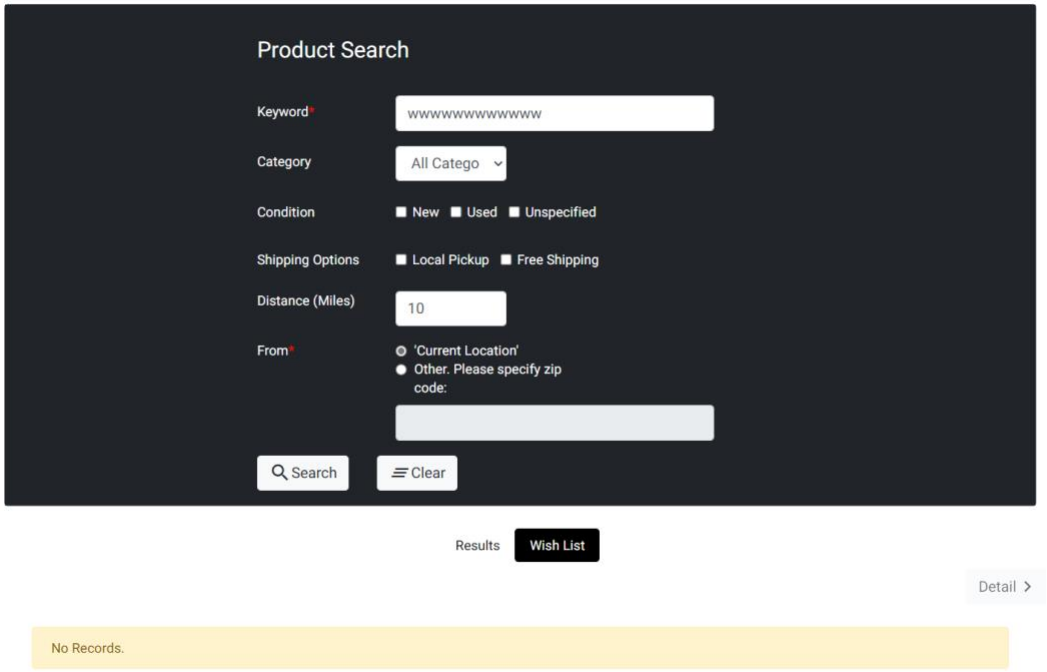


Figure 25 No Records on Wish List

3.7 Progress Bars

Whenever data is being fetched, dynamic progress bar must be displayed as shown in **Figure 26**.

You can use the progress bar component of **Bootstrap** to implement this feature.

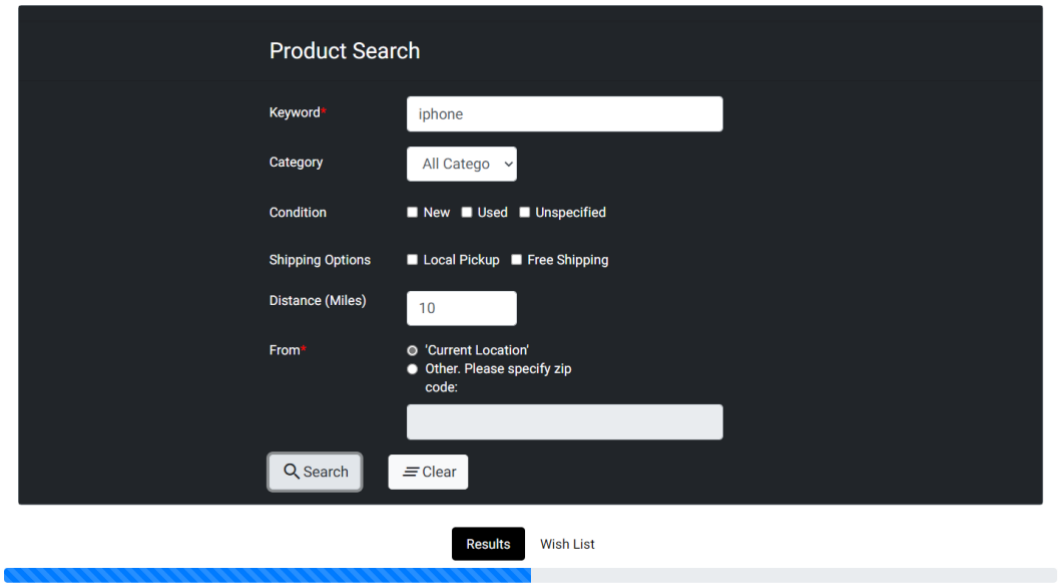
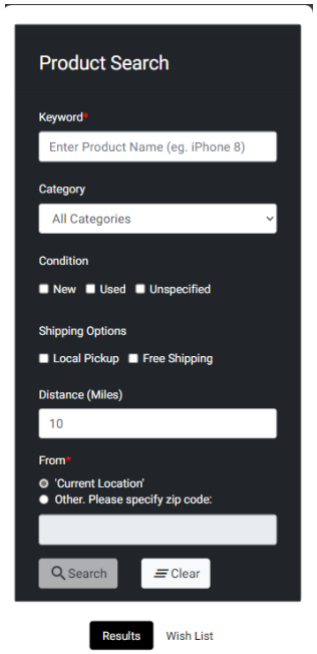
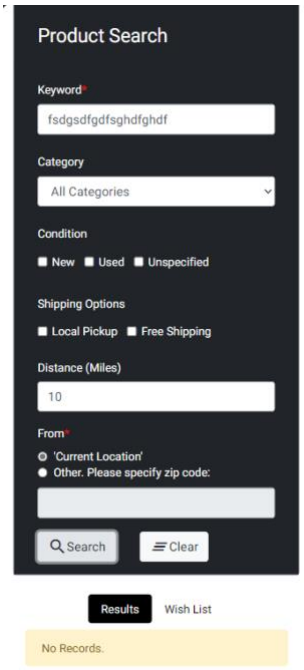


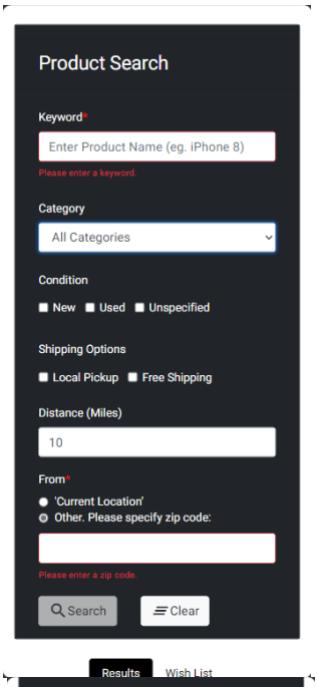
Figure 26 Progress Bar

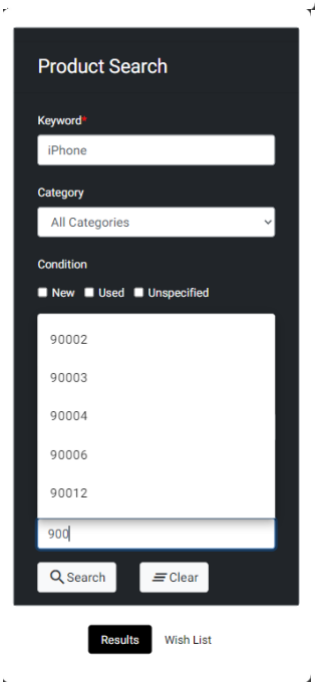
3.8 Responsive Design

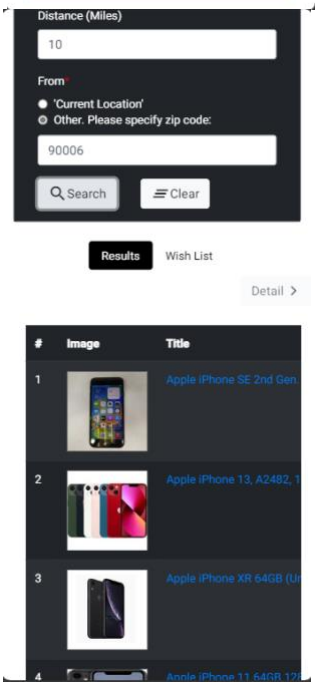
The following are snapshots of the webpage opened with **Chrome** on **iPhone 12 Pro** emulator. See the video for more details.

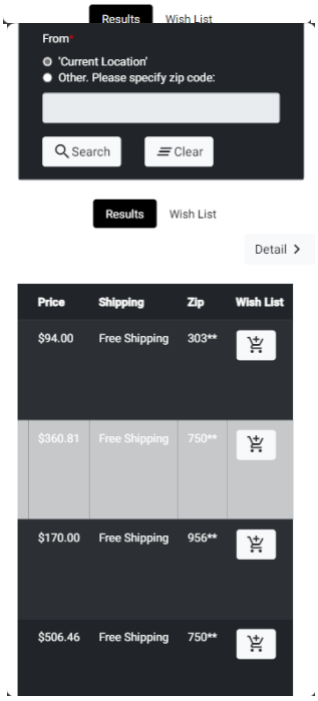

Product Search
Keyword*
Enter Product Name (eg. iPhone 8)
Category
All Categories
Condition
New Used Unspecified
Shipping Options
Local Pickup Free Shipping
Distance (Miles)
10
From*
'Current Location'
Other. Please specify zip code:
Search
Clear
Results
Wish List

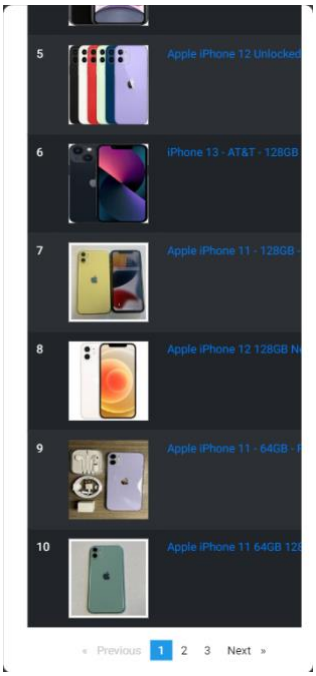

Product Search
Keyword*
fsdgsdfgdfsghdfghdf
Category
All Categories
Condition
New Used Unspecified
Shipping Options
Local Pickup Free Shipping
Distance (Miles)
10
From*
'Current Location'
Other. Please specify zip code:
Search
Clear
Results
Wish List
No Records.

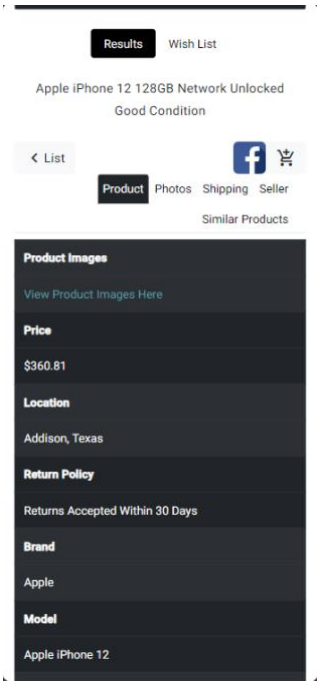

Product Search
Keyword*
Enter Product Name (eg. iPhone 8)
Please enter a keyword.
Category
All Categories
Condition
New Used Unspecified
Shipping Options
Local Pickup Free Shipping
Distance (Miles)
10
From*
'Current Location'
Other. Please specify zip code:
Please enter a zip code.
Search
Clear
Results
Wish List

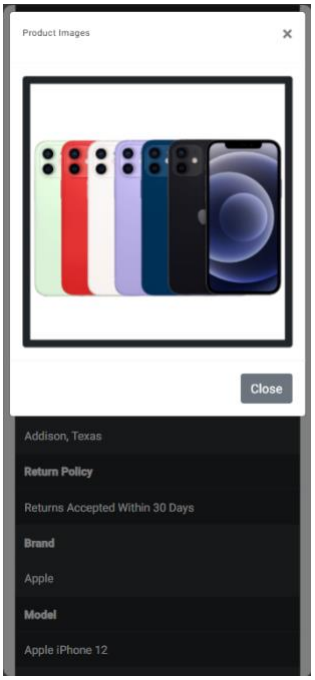

Product Search
Keyword*
iPhone
Category
All Categories
Condition
New Used Unspecified
90002
90003
90004
90006
90012
900
Search
Clear
Results
Wish List


Distance (Miles)
10
From*
'Current Location'
Other. Please specify zip code:
90006
Search
Clear
Results
Wish List
Detail
Image Title
1
Apple iPhone SE 2nd Gen
2
Apple iPhone 13, A2482,
3
Apple iPhone XR 64GB (U
4

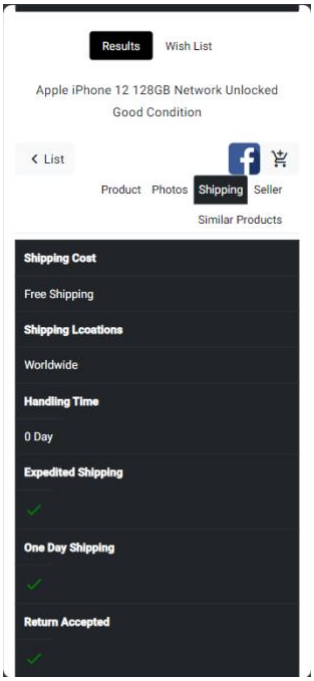

From*
'Current Location'
Other. Please specify zip code:
Search
Clear
Results
Wish List
Detail
Price Shipping Zip Wish List
$94.00 Free Shipping 303**
$360.81 Free Shipping 750**
$170.00 Free Shipping 956**
$506.46 Free Shipping 750**

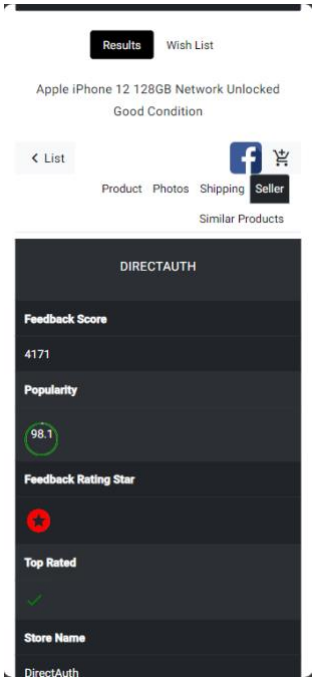

5
Apple iPhone 12 Unlocked
6
iPhone 13 - AT&T - 128GB
7
Apple iPhone 11 - 128GB
8
Apple iPhone 12 128GB N
9
Apple iPhone 11 - 64GB
10
Apple iPhone 11 64GB 12
« Previous
1
2
3
Next »

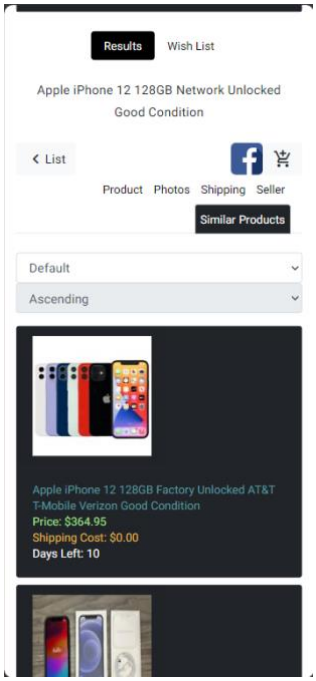

Results
Wish List
Apple iPhone 12 128GB Network Unlocked
Good Condition
List
Product
Photos
Shipping
Seller
Similar Products
Product Images
View Product Images Here
Price
$360.81
Location
Addison, Texas
Return Policy
Returns Accepted Within 30 Days
Brand
Apple
Model
Apple iPhone 12

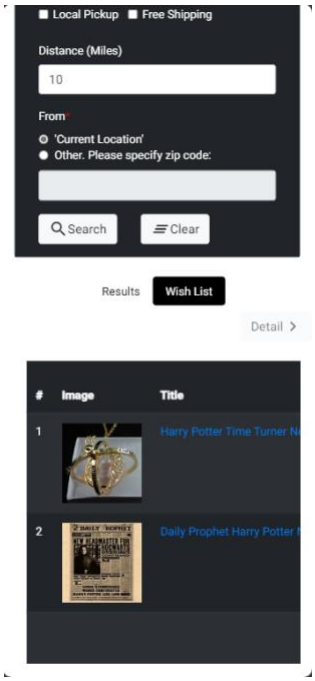

Product Images
Close
Addison, Texas
Return Policy
Returns Accepted Within 30 Days
Brand
Apple
Model
Apple iPhone 12


List
Product
Photos
Shipping
Seller
Similar Products


Results
Wish List
Apple iPhone 12 128GB Network Unlocked
Good Condition
List
Product
Photos
Shipping
Seller
Similar Products
Shipping Cost
Free Shipping
Shipping Locations
Worldwide
Handling Time
0 Day
Expedited Shipping
One Day Shipping
Return Accepted


Results
Wish List
Apple iPhone 12 128GB Network Unlocked
Good Condition
List
Product
Photos
Shipping
Seller
Similar Products
DIRECTAUTH
Feedback Score
4171
Popularity
98.1
Feedback Rating Star
Top Rated
Store Name
DirectAuth





Note: The Similar Products tab is replaced by the Related 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.

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

You should test your program in the mobile view (iPhone 12 Pro) in Chrome, since we will grade it using this simulator.

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 HW2. You can re-use it.

Note: Developers using "GetSingleItem" API calls must authenticate with an OAuth application access token in the HTTP header X-EBAY-API-IAF-TOKEN. Follow the steps below to create an OAuth token.

1. Download the *ebay_oauth_token.js* file shared with this assignment. You will need to use the **OAuthToken** class in your backend to generate OAuth application access token before calling *GetSingleItem*
2. Use the following code in your backend to generate the OAuth application access token and use it in your *GetSingleItem* call.

```
// Usage example
const client_id = 'your_client_id';
const client_secret = 'your_client_secret';

const oauthToken = new OAuthToken(client_id, client_secret);

oauthToken.getApplicationToken()
    .then((accessToken) => {
        console.log('Access Token:', accessToken);
    })
    .catch((error) => {
        console.error('Error:', error);
    });
```

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

You can search on some websites like
https://fonts.google.com/icons
https://google.github.io/material-design-icons/
https://material.io/tools/icons/

to find similar images in the homework description, like facebook icon, shopping cart icon, star, etc..

How to use star_border icon from Google, see this example:
https://www.w3schools.com/icons/tryit.asp?filename=tryicons_google-star_border

5.2 Get started with the Bootstrap Library

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

https://getbootstrap.com/docs/5.3/getting-started/introduction/

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

5.3 Angular Material

Angular Material: https://material.angular.io/
Autocomplete:https://material.angular.io/components/autocomplete/overview
Tooltip: https://material.angular.io/components/tooltip/overview

5.4 Google Cloud Platform/Amazon Web Services/ Microsoft Azure

You should use the domain name of the GCP/AWS/Azure service you created in Homework #2 to make the request. For example, if your GCP/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:

GCP - 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 Assignment #3, 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.

5.5 GET call

You **must use a GET method** to request the backend since you are required to provide the above additional link to your homework list to let graders check whether the Node.js script code is running in the “cloud” on GCP/AWS/Azure (see 5.4 above). Please refer to the grading guidelines for details.

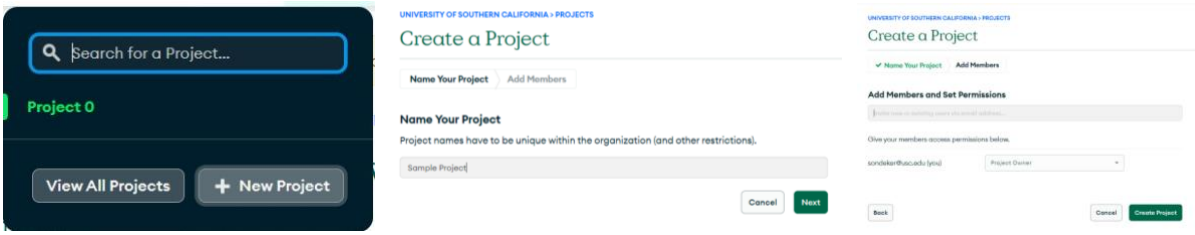
5.6 MongoDB

MongoDB is a source-available cross-platform document-oriented database program. It is classified as a NoSQL database program. MongoDB uses JSON-like documents with optional schemas. For more information, see: https://www.mongodb.com/docs/

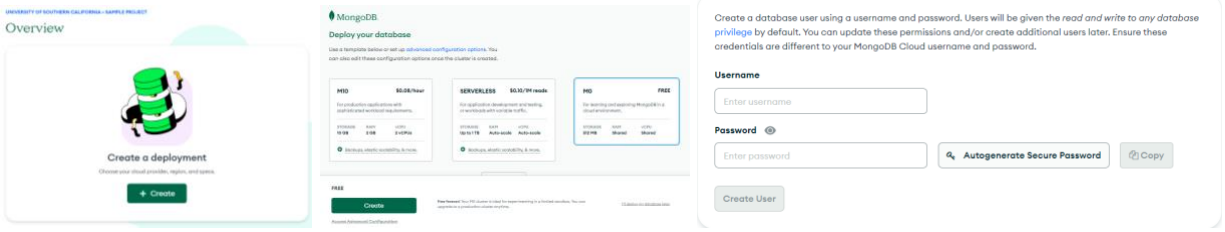
MongoDB on GCP: https://www.mongodb.com/mongodb-on-google-cloud
MongoDB on AWS: https://www.mongodb.com/mongodb-on-aws
MongoDB on Azure: https://www.mongodb.com/mongodb-on-azure

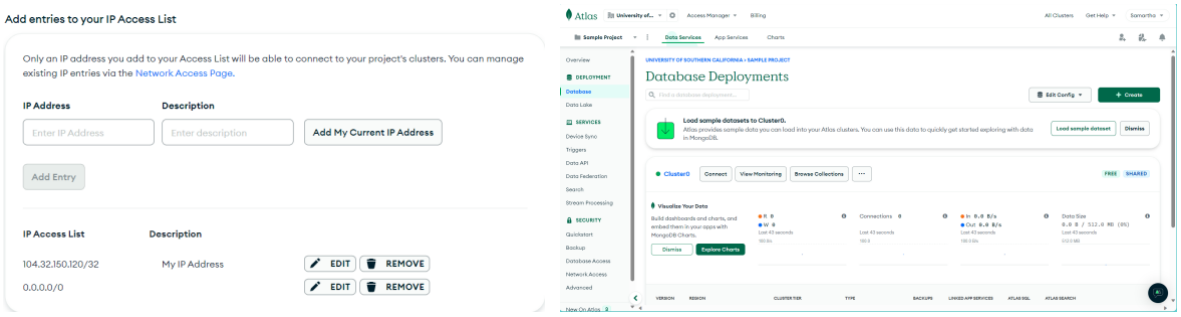
Once, you setup an account in MongoDB Atlas, you will have to create a project to store your databases. In a project you will create a database to store collections which hold the favorites information data in NoSQL format. Below are the steps to setup a project and create a database.

These steps create a project in which you can store databases for your application.

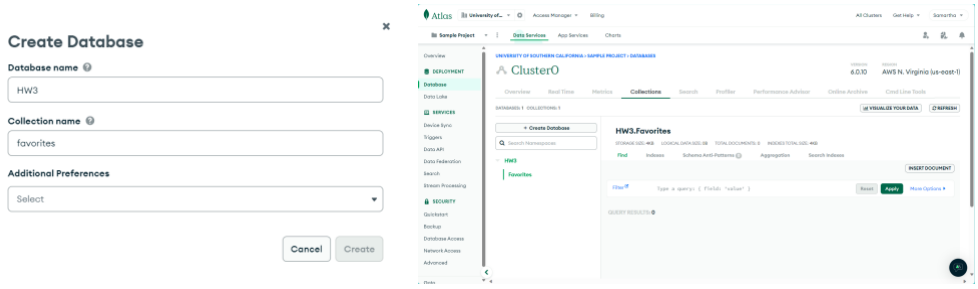


Follow these steps to create a deployment for the project by creating a cluster and providing user access and network access for the same. This would allow your application, running on a different server, connect to MongoDB in the cloud.

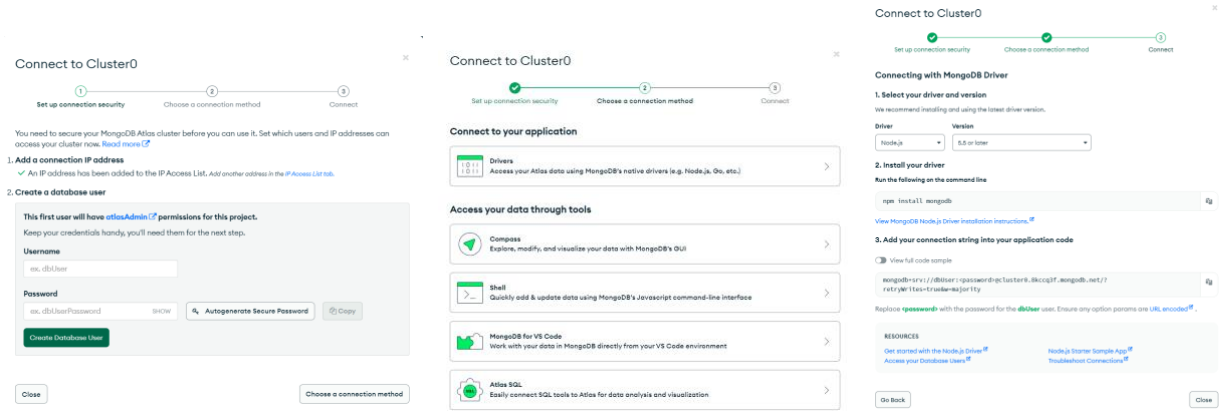




These steps will create a database and a collection in that database. MongoDB stores data records as documents (specifically in BSON format) which are gathered in collections. A database stores one or more collections of documents.



These steps provide the instructions on how to connect to your MongoDB database from your application.



Once you have setup the database and the collection, you can connect to the database by adding the MongoDB node driver to your application. For more information how to add the driver and run queries on the database, refer to [MongoDB Node Driver — Node.js]

6. Files to Submit

In your course homework page, you should update the Assignemnt #3 link to refer to your initial web page for this exercise. Additionally, you need to provide **an additional link** to the URL of the GCP/AWS/Azure service where the GET call is made with sample parameter values (i.e., a valid query, with keyword, location, etc. See section 5.4).

Also, submit all files (HTML, JS, CSS, TS), **both backend and frontend,** that you write to D2L, bundled in a **SINGLE ZIP file**. **Don't upload library, node-js modules, any angular-cli build files, any images or any code generated by the framework tools.**