

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

## Entertainment Event 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 App Engine/Microsoft Azure.
- Learn to use popular APIs such as **Ticketmaster APIs, Spotify APIs, Google Maps APIs, Google Customized Search APIs, Songkick APIs, and Twitter APIs.**

#### 1.1 Prerequisite

Please apply **Songkick APIs as soon as possible**, as it will take **2 to 7 days to obtain your API key**. See more on section 4.2.

### 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/flexible/Node.js/>

## 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, Angular 2, Angular 4, Angular 5, or Angular 6 can be used, but **Angular 6 is recommended since AngularJS enter LTS support**. 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+

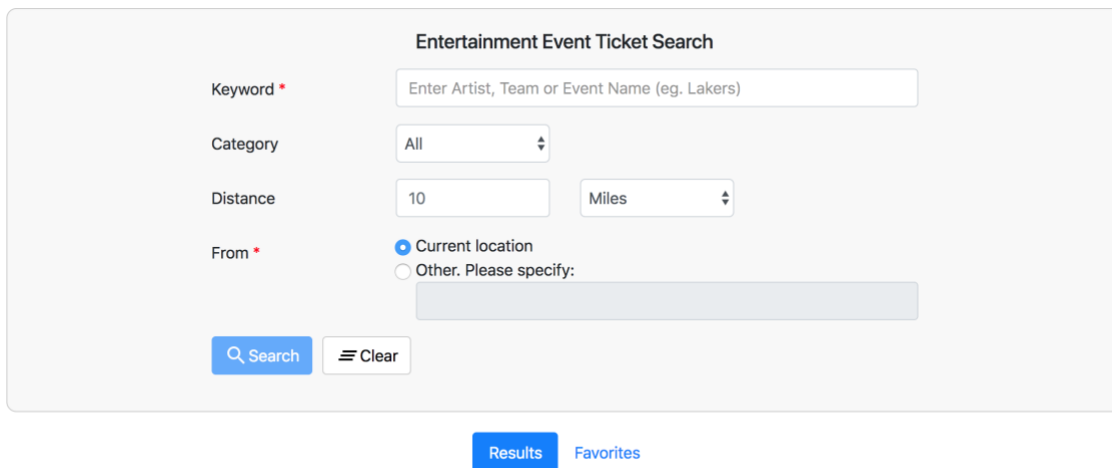
You can use either way.

**All APIs calls should be done through your Node.JS server, except calls to the ip-api and Google Maps display.**

### 3. High Level Description

In this exercise you will create a webpage that allows users to **search** for **events** using the **Ticketmaster API** and display the results on the same page below the form. Once the user clicks on a button to search for event details, your webpage should display **several tabs** which contain an **event info table**, **artist info table**, **venue info table**, and **upcoming events** related to this event respectively. Your webpage should also support adding events to and removing events from **favorites list** and **posting** events info to **Twitter**. 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 Figure 1.



**Figure 1** Initial Search Form

### 3.1 Search Form

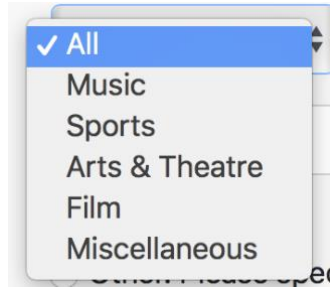
#### 3.1.1 Design

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

There are four 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. This input field should **support autocomplete** which is explained in section 3.1.2. Please note that the **user does not necessarily chooses what suggested by the autocomplete**. Initially, please show the **placeholder** shown in the picture.
2. **Category**: The **default** value of “Category” is “**All**”, which covers all of the “types” provided by the *Ticketmaster API*. The other options are shown in Figure 2.
3. **Distance**: This is the radius of the area within which the search is performed. The center of the area is specified in the “From” field. There are two units: “miles” and “kilometers”. The **default** value is **10** miles.
4. **From**: The **center** of the **area** within which the search is performed. The user can choose between their **current** location or a **different** location. 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

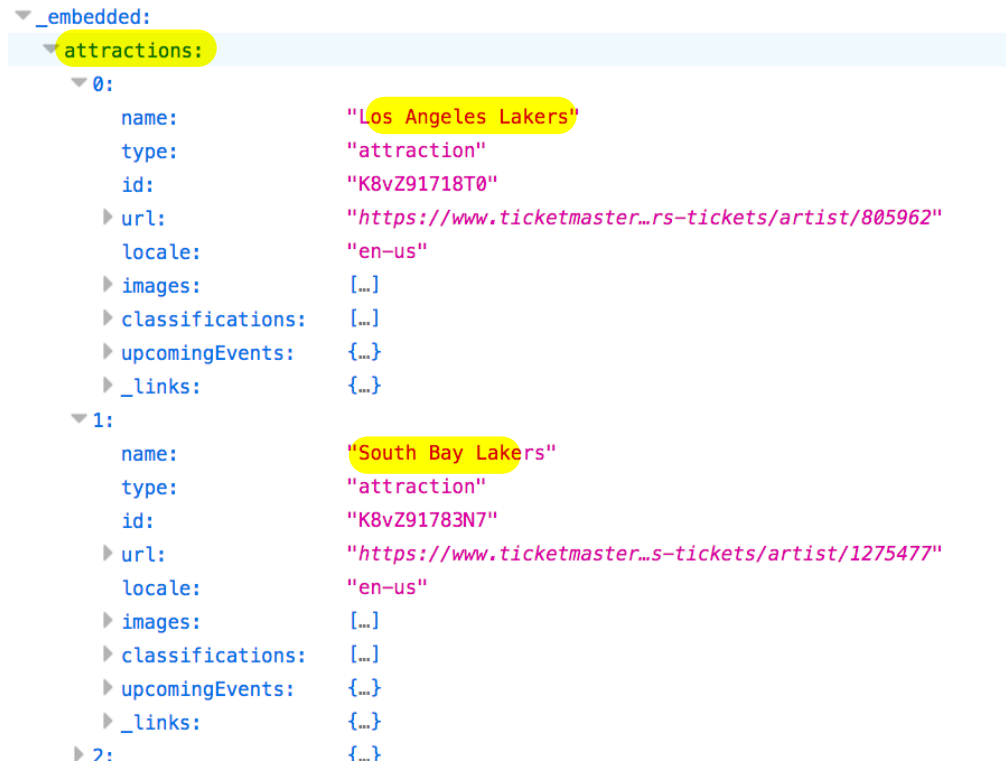
Autocomplete is implemented by using the **suggestion service** provided by **Ticketmaster**. Please go to this page to learn more about this service:

<https://developer.ticketmaster.com/products-and-docs/apis/discovery-api/v2/#find-suggest-10-v2>

An example of an HTTP request to the *Ticketmaster API* Get Suggestion that searches for the keyword “laker” is shown below:

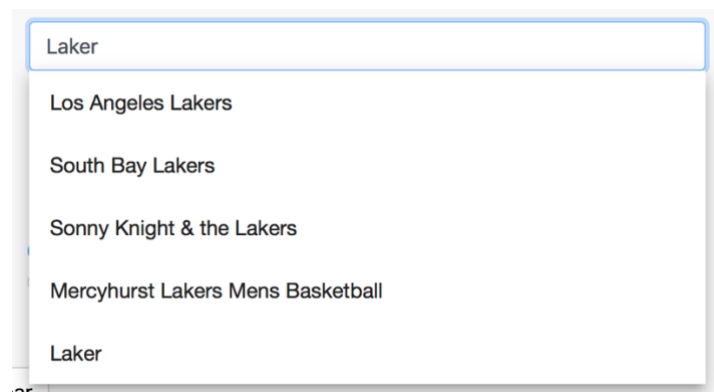
```
https://app.ticketmaster.com/discovery/v2/suggest?apikey=YOUR_API_KEY&keyword=laker
```

The response is a JSON object shown in Figure 3.



**Figure 3** Autocomplete JSON Response

You should use “Attractions” only with the name field to implement the autocomplete. You must use **Angular Material** to implement the Autocomplete. (See section 6.4)



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

Entertainment Event Ticket Search

Keyword \*   
Please enter a keyword.

Category

Distance

From \* ☐ Current location  
☒ Other. Please specify:  
  
Please enter a location.

**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>

The “Search” button should be disabled before the user location is obtained.

### 3.1.5 Search Execution

Once the validation is successful and the user clicks on “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 Ticketmaster API to get the events 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 Ticketmaster API service and display the result in the webpage.

The *Ticketmaster API Event Search Service* is documented here:

<https://developer.ticketmaster.com/products-and-docs/apis/discovery-api/v2/#search-events-v2>

If your application does **not** have the **geohash Code** of latitude and longitude of the event that user specifies, the **Node.js** script should first **use** the **input address** to **get** the **geocoding** via **Google Maps Geocoding API**. The *Google Maps Geocoding API* is documented here:

<https://developers.google.com/maps/documentation/geocoding/start>

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

The **Node.js script** should **pass** the **JSON object returned** by the **Event 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 six columns: # (Index number), Date, Event, Category, Venue Info and Favorite. Events should be displayed by **ascending order of “date”** column.

Results

Favorites

Details >

#	Date	Event	Category	Venue Info	Favorite
1	2018-10-20	Los Angeles Lakers vs. Houston ...	Basketball-Sports	STAPLES Center	<input type="checkbox"/>
2	2018-10-22	Los Angeles Lakers vs. San Antonio ...	Basketball-Sports	STAPLES Center	<input type="checkbox"/>
3	2018-10-25	Los Angeles Lakers vs. Denver ...	Basketball-Sports	STAPLES Center	<input type="checkbox"/>
4	2018-10-31	Los Angeles Lakers vs. Dallas ...	Basketball-Sports	STAPLES Center	<input type="checkbox"/>
5	2018-11-04	Los Angeles Lakers vs. Toronto ...	Basketball-Sports	STAPLES Center	<input type="checkbox"/>
6	2018-11-07	Los Angeles Lakers vs. Minnesota ...	Basketball-Sports	STAPLES Center	<input type="checkbox"/>
7	2018-11-11	Los Angeles Lakers vs. Atlanta ...	Basketball-Sports	STAPLES Center	<input type="checkbox"/>
8	2018-11-14	Los Angeles Lakers vs. Portland ...	Basketball-Sports	STAPLES Center	<input type="checkbox"/>
9	2018-11-23	Los Angeles Lakers vs. Utah Jazz	Basketball-Sports	STAPLES Center	<input type="checkbox"/>
10	2018-11-25	Los Angeles Lakers vs. Orlando ...	Basketball-Sports	STAPLES Center	<input type="checkbox"/>
11	2018-11-29	Los Angeles Lakers vs. Indiana ...	Basketball-Sports	STAPLES Center	<input type="checkbox"/>
12	2018-11-30	Los Angeles Lakers vs. Dallas ...	Basketball-Sports	STAPLES Center	<input type="checkbox"/>
13	2018-12-02	Los Angeles Lakers vs. Phoenix Suns	Basketball-Sports	STAPLES Center	<input type="checkbox"/>
14	2018-12-05	Los Angeles Lakers vs. San Antonio ...	Basketball-Sports	STAPLES Center	<input type="checkbox"/>
15	2018-12-10	Los Angeles Lakers vs. Miami Heat	Basketball-Sports	STAPLES Center	<input type="checkbox"/>
16	2018-12-21	Los Angeles Lakers vs. New Orleans ...	Basketball-Sports	STAPLES Center	<input type="checkbox"/>
17	2018-12-23	Los Angeles Lakers vs. Memphis ...	Basketball-Sports	STAPLES Center	<input type="checkbox"/>
18	2018-12-28	Los Angeles Lakers vs. LA Clippers	Basketball-Sports	STAPLES Center	<input type="checkbox"/>
19	2018-12-30	Los Angeles Lakers vs. Sacramento ...	Basketball-Sports	STAPLES Center	<input type="checkbox"/>
20	2019-01-02	Los Angeles Lakers vs. Oklahoma ...	Basketball-Sports	STAPLES Center	<input type="checkbox"/>

**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
Date	The value of the “localDate” attribute that is part of “events” object.
Event	The value of the “name” attribute that is part of the “events” object.



Category	The value of the “genre” and “segment” attributes that are part of the “classifications” object.
Venue Info	The value of the “name” attribute that is part of the “venue” object.

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

The “#” column starts from 1. The “event” column might not be long enough to show the entire name of the event, using “...” to avoid starting a new row. The “event” column is clickable to trigger the detail search for the corresponding event. The “Favorite” column contains a button that can add the event to, or remove the event from, the favorites list. If an event is on the list, the star is full (yellow). Otherwise, the star is empty (white).

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

1. Judge whether the event 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 event name (Figure 7).

#	Date	Event	Category	Venue Info	Favorite
1	2018-10-20	Los Angeles Lakers vs. Houston ...	Basketball-Sports	STAPLES Center	★
2	2018-10-22	Los Angeles Lakers vs. Houston Rockets	Basketball-Sports	STAPLES Center	☆

**Figure 7** An Example of the tooltip for event 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. If you use the bootstrap tooltip, the style will be a slightly different, but it is fine. (See section 6.3 and 6.4)

### 3.2.3 Details Button and Highlighting

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

<div>Results Favorites</div> <div>Details &gt;</div>					
#	Date	Event	Category	Venue Info	Favorite
1	2018-10-20	Los Angeles Lakers vs. Houston ...	Basketball-Sports	STAPLES Center	☆

**Figure 8 Highlight Selected Event**

### 3.3 Details

Once an **event name** in the “Event” column is **clicked**, your webpage should search for the details of that event.

Above the tabs in the details view, you should display the whole **name** of the event, a **button** that allows you to go **back** to the previous list, a **Twitter** button and a **favorites** button.

#### 3.3.1 Info Tab

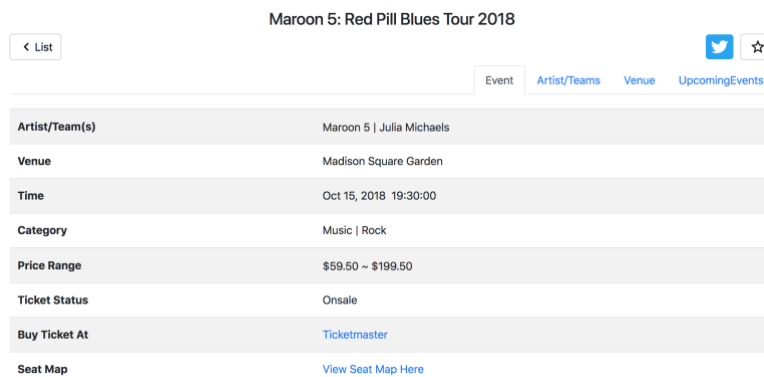
A table containing the detailed info of the event is displayed in this tab. The table has the following fields if they are available in the detail search results:

Fields in the table	Corresponding response object fields
Artist/Team(s)	<i>The value of the “name” attribute that is part of the “attractions” object, segmented by “   ”</i>
Venue	<i>The value of the “name” attribute that is part of the “venue” object.</i>
Time	<i>The value of the “localDate” and “localTime” attributes that is part of “dates” object.</i>
Category	<i>The value of the “genre”, “segment” attributes that are part of the “classifications” object, segmented by “   ”.</i>
Price Range	<i>The value of the “min” and “max” attributes that are part of the “priceRanges” object, combined by “ ~ ”.</i>
Ticket Status	<i>The value of the “status” attribute that is part of the “dates” object.</i>
Buy Ticket At	<i>The value of the “url” attribute.</i>
Seat Map	<i>The value of the “staticUrl” attribute that is part of the “seatmap” object.</i>

**Table 2:** Mapping the result from Event Detail API into HTML table

The value of “Buy Ticket At” is a links, that supposed to open the specific URL in a new tab. When click on “**Seat Map**”, a **modal** should be popped up with the seat map image. See video. Please note that when you try to get the **current time** of the **event** to display, you should use the **Moment.js library** or **Angular Time Pipe** to do the **conversion** into the **format “Oct 12, 2018 19:30:00”**. For Price Range, please format the number with **\$xxx.xx**.

If a **field** is **not** available, please **don’t display that row**.



Maroon 5: Red Pill Blues Tour 2018	
< List	Event Artist/Teams Venue UpcomingEvents
Artist/Team(s)	Maroon 5   Julia Michaels
Venue	Madison Square Garden
Time	Oct 15, 2018 19:30:00
Category	Music   Rock
Price Range	\$59.50 ~ \$199.50
Ticket Status	Onsale
Buy Ticket At	<a href="#">Ticketmaster</a>
Seat Map	<a href="#">View Seat Map Here</a>

**Figure 8** An Example of Event Detail Tab

### 3.3.2 Artist/Team(s) Tab

This section splits into two parts. The first part is the **music-related artist** info using **Spotify APIs**, and the second part is **artist/team(s) photos** using **Google custom search API**. **If an artist is not related to music**, then it will only **show the second part**. You can **decide** whether the artist is related to music by using the **event category**.

#### 3.3.2.1 Music-Related Artist/Team(s) Detail Table

The Spotify API is documented at:

<https://developer.spotify.com/documentation/web-api/>

After you **register** your **Spotify API**, you need to **create** a **project** under “**dashboard**” on the **developer portal of Spotify**. You will get your **client id** and **client secret**.

Please use the **Spotify NodeJS library** that we recommend:

<https://github.com/thelinmichael/spotify-web-api-node>

For the API calls, you need to use the ‘**searchArtists**’ function. Spotify API service does **not** use a **static API key** to make authorizations. It will use your **client id** and **client secret** to **generate** a **Baser Token** that will expire in **60 mins** once generated. So here are some flows you can follow:

1. **Call** ‘**searchArtists**’ **function**.
2. If the function return **success**, it means you have **set** up this **Baser Token** and the token is not expired. In this case you can **return** the **data directly**.
3. If the **function** return **error**, and the **http status code** is **401**, it means you did **not set up** the token **or** your token is **expired**. In this case you need to **call** the ‘**clientCredentialsGrant**’ **function** and then set the return value to ‘**setAccessToken**’. After you set your access token, you can **call** the ‘**searchArtists**’ function again and you will get the correct response.

For the **searchArtists** function, you will only need to provide one parameters: “**keyword**”, please use the **attraction** name to search. You will get the following result:

```
▼ artists:
  href: "https://api.spotify.com/v1/search?query=maroon+5&type=artist&offset=0&limit=2"
  items:
    ▼ 0:
      external_urls:
        spotify: "https://open.spotify.com/artist/04gDigrS5kc9YWfZhwBETP"
      followers:
        href: null
        total: 13428230
      genres:
        0: "pop"
      href: "https://api.spotify.com/v1/artists/04gDigrS5kc9YWfZhwBETP"
      id: "04gDigrS5kc9YWfZhwBETP"
      images: [...]
      name: "Maroon 5"
      popularity: 91
      type: "artist"
      uri: "spotify:artist:04gDigrS5kc9YWfZhwBETP"
    ▼ 1:
      external_urls:
        spotify: "https://open.spotify.com/artist/6SrKlwioiqWe9Sa0YxSMA"
      followers:
        href: null
        total: 3234
      genres: []
      href: "https://api.spotify.com/v1/artists/6SrKlwioiqWe9Sa0YxSMA"
      id: "6SrKlwioiqWe9Sa0YxSMA"
```

**Figure 9** An Example Spotify API Call result

Fields in the artist table	Corresponding response object fields
Name	<i>The value of the “name” of the item</i>
Followers	<i>The value of the “followers.total”.</i>
Popularity	<i>The value of the “popularity”.</i>
Check At	<i>The value of the “external_urls.spotify”</i>

**Table 3:** Mapping the result from Spotify API into HTML table

Event Artist/Teams Venue Upcoming Events	
Maroon 5	
Name	Maroon 5
Followers	13,429,030
Popularity	91
Check At	Spotify

**Figure 10** An Example Spotify API Artist Table

Please note that the Spotify API may **return more** than one **artist** for **each search keyword**. Please **choose** the **item** whose **name** is **equal** to the **attraction name** (**case insensitive**) and return that matched item.

**Format Followers** using **xxx,xxx,xxx**, as shown in Figure 10. For Popularity, since the value will be 0-100, you need to use a **circle progress bar**, which is available in a third-party **library**, to display it.

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>

For **multiple artists**, you should create a **table for each artist**. See the video for accurate explanation.

### 3.3.2.2 Artist/Team(s) **Photos**

The second part is Artist/Teams photos. The **Google Custom Search Engine** is documented at:

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

To retrieve photos about the artist (or team, if event is a game), 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=USC+Trojans&cx=YOUR_SEARCH_ENGINE_ID&imgSize=huge&imgType=news&num=9&searchType=image&key=YOUR_API_KEY
```

Figure 11 shows a sample response.

```

{
  "items": [
    {
      "kind": "customsearch#result",
      "title": "USC Trojans - Wikipedia",
      "htmlTitle": "<b>USC Trojans</b> - Wikipedia",
      "link": "https://upload.wikimedia.usc_trojans_logo.svg.png",
      "displayLink": "en.wikipedia.org",
      "snippet": "USC Trojans - Wikipedia",
      "htmlSnippet": "<b>USC Trojans</b> - Wikipedia",
      "mime": "image/png",
      "image": {
        "contextLink": "https://en.wikipedia.org/wiki/USC_Trojans",
        "height": 1790,
        "width": 1200,
        "byteSize": 11882,
        "thumbnailLink": "https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQUYYTwKp9s7NimbGQT_DqwHuhjssPz-",
        "thumbnailHeight": 150,
        "thumbnailWidth": 101
      }
    },
    {
      "kind": "customsearch#result",
      "title": "USC Trojans Logo Wall Decal | Shop Fathead® for USC Trojans Decor",
      "htmlTitle": "<b>USC Trojans</b> Logo Wall Decal | Shop Fathead® for <b>USC Trojans</b> Decor",
      "link": "https://r.fathead-res.cloudinary.com/image/upload/q_auto/c_pad,w_4225,h_3000/roomplus/",
      "displayLink": "www.fathead.com",
      "snippet": "USC Trojans Logo Wall Decal | Shop Fathead® for USC Trojans Decor",
      "htmlSnippet": "<b>USC Trojans</b> Logo Wall Decal | Shop Fathead® for <b>USC Trojans</b> Decor",
      "mime": "image/jpeg",
      "image": {
        "contextLink": "https://www.fathead.com/college/usc-trojans/usc-trojans-logo-wall-decal/",
        "height": 3000,
        "width": 4225,
        "byteSize": 291252
      }
    }
  ]
}

```

**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 at most 8 photos, which is present in “ <i>link</i> ” attribute.

**Table 4:** 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**.



**Figure 12** Multiple Teams Info Tab

### 3.3.3 Venue Tab

To get the venue info, use the venue name which get from the event detail and call Ticketmaster API search venue call.

A table containing the detailed info of the event venue is displayed in this tab. The table has the following field,s if they are available in the detail search results:

Fields in the info table	Corresponding response object fields
Address	The value of the “line1” attribute that is part of the “address” object.



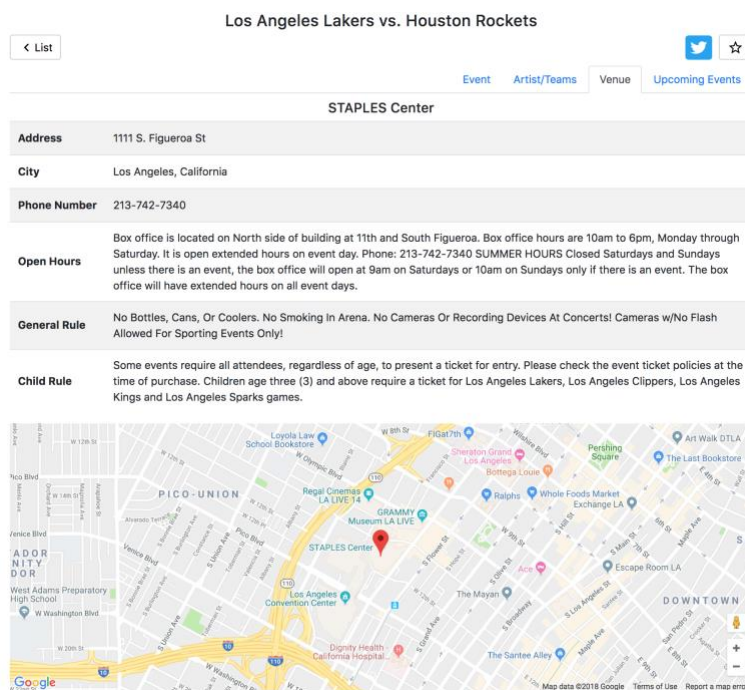
City	The value of the “name” attribute of “city” object and “state” object, connected by a comma.
Phone Number	The value of the “phoneNumberDetail” attribute that is part of the “boxOfficeInfo” object.
Open Hours	The value of the “openHoursDetail” attribute that is part of the “boxOfficeInfo” object.
General Rule	The value of the “generalRule” attribute that is part of the “generalInfo” object.
Child Rule	The value of the “childRule” attribute that is part of the “generalInfo” object.

**Table 5:** Mapping the result from Venue Detail API into HTML table

A **map with marker of venue’s location** should be displayed below the Table. You should use the Google Maps JavaScript Library, documented at: <https://developers.google.com/maps/documentation/javascript/adding-a-google-map> to construct the map.

The usage of this API has been explained in the Homework #6 documents.

A sample of venue tab is shown as Figure 13.



**Figure 13** Venue Tab

### 3.3.4 Upcoming Events Tab

This tab displays the **upcoming events** of the **same venue** using the **Songkick API**. The usage of the Songkick APIs is explained in detail in section 5.2.

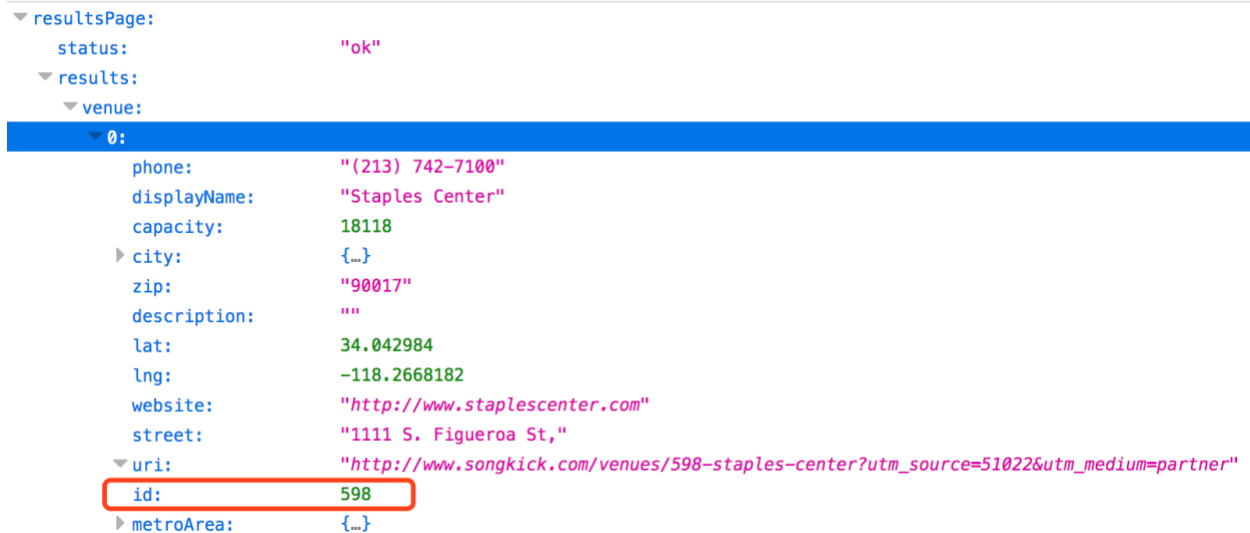
First you need to use the **search venue API** of **Songkick** to get the **id** of the **venue**. Use the **venue name** you get **from** the **Ticketmaster** API to **search** the **id**.

<https://www.songkick.com/developer/venue-search>

The API call will be:

```
https://api.songkick.com/api/3.0/search/venues.json?query={venue_name}&apikey={your_api_key}
```

The sample API response will be:



**Figure 14** Sample API response for Songkick Search Venue

Use the first element of the venue array and get its id. Then **call** the **second Songkick API** for **venue upcoming events** at:

<https://www.songkick.com/developer/upcoming-events-for-venue>

The API call will be:

```
https://api.songkick.com/api/3.0/venues/{venue_id}/calendar.json?apikey={your_api_key}
```

The sample API response will be:



```

    resultsPage:
      status: "ok"
    results:
      event:
        0:
          id: 33922769
          displayName: "Drake with Migos and Roy Woods at Staples Center (October 13, 2018)"
          type: "Concert"
          uri: "http://www.songkick.com/concerts/33922769--drake-at-staples-center?utm_source=510226utm_medium=partner"
          status: "ok"
          popularity: 0.857006
          start:
            date: "2018-10-13"
            datetime: "2018-10-13T19:00:00-0700"
            time: "19:00:00"
          performance:
            0:
              id: 65180879
              displayName: "Drake"
              billing: "headline"
              billingIndex: 1
              artist:
                id: 556955
                displayName: "Drake"
                uri: "http://www.songkick.com/_510226utm_medium=partner"
                identifier: [...]
              1: {}
              2: {}
            ageRestriction: null
            flaggedAsEnded: false
            venue: {}
            location: {}

```

**Figure 15** Sample API response for Songkick Upcoming Events

Each event has a **name**, **artist**, **time**, and **type** and is displayed as a card (see Figure 16(a)). The **event name** is **clickable**. Once it is clicked, a **new page should be opened** and take the user to the **event page on Songkick**. The **time** should be displayed in the **format** “Oct 12, 2018 19:00:00”.

By **default**, upcoming events are **displayed** in the **default order** (the order in which the reviews are returned by the API). There are **two dropdowns** in this tab. The first one allows the user to **sort** the events in several **different categories**: Default, Event Name, Time, Artist, and Type. 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.

By default, **only 5 upcoming events** are **displayed**, like shown in Figure 16(a). After **clicking** the “**Show More**” button, all upcoming events in the returned JSON should be displayed, and the button changes to “**Show Less**”, like in Figure 16(b). After clicking “Show Less” **only the top 5** upcoming events in **givedn sorting** order should remain.

Please note there is an **animation** of the “**Show More/Less**”, and it must be implemented in **AngularJS/Angular 2+**. See the video on this.

Fields in the info table	Corresponding response object fields
Display Name	The value of the “ <i>displayName</i> ” attribute. Add hyperlink whose value is “ <i>uri</i> ”
Artist	The value of the “ <i>name</i> ” attribute of first element of “ <i>performance</i> ”.
Date Time	<i>The value of the “date” and “time” attribute that is part of the “start” object.</i>
Type	<i>The value of the “type” attribute.</i>

**Table 6:** Mapping the result from Upcoming Event Detail API into HTML card

< List

Twitter

Star

EventArtist/TeamsVenueUpcoming Events

Default

Ascending

Julia Michaels at Madison Square Garden (October 14, 2018)  
Artist: Julia Michaels Oct 14, 2018 19:30:00  
Type: Concert

Maroon 5 at Madison Square Garden (October 14, 2018)  
Artist: Maroon 5 Oct 14, 2018  
Type: Concert

Julia Michaels at Madison Square Garden (October 15, 2018)  
Artist: Julia Michaels Oct 15, 2018 19:30:00  
Type: Concert

Maroon 5 at Madison Square Garden (October 15, 2018)  
Artist: Maroon 5 Oct 15, 2018  
Type: Concert

Elton John at Madison Square Garden (October 18, 2018)  
Artist: Elton John Oct 18, 2018  
Type: Concert

Show More

**Figure 16(a)** Show More Upcoming Events

Elton John at Staples Center (January 25, 2019)  
Artist: Elton John Jan 25, 2019  
Type: Concert

Kelly Clarkson with Kelsea Ballerini and Brynn Cartelli at Staples Center (January 26, 2019)  
Artist: Kelly Clarkson Jan 26, 2019 18:00:00  
Type: Concert

Elton John at Staples Center (January 30, 2019)  
Artist: Elton John Jan 30, 2019  
Type: Concert

Pat Braxton at Staples Center (February 10, 2019)  
Artist: Pat Braxton Feb 10, 2019 16:00:00  
Type: Concert

Eric Church at Staples Center (May 17, 2019)  
Artist: Eric Church May 17, 2019 19:00:00  
Type: Concert

Eric Church at Staples Center (May 18, 2019)  
Artist: Eric Church May 18, 2019 19:00:00  
Type: Concert

Shawn Mendes at Staples Center (July 5, 2019)  
Artist: Shawn Mendes Jul 5, 2019 18:30:00  
Type: Concert

Maddie and Tae at Staples Center (September 12, 2019)  
Artist: Maddie and Tae Sep 12, 2019 18:00:00  
Type: Concert

Show Less

**Figure 16(b)** Show Less Upcoming Events

### 3.3.5 List Button, Favorites Button and Twitter Button

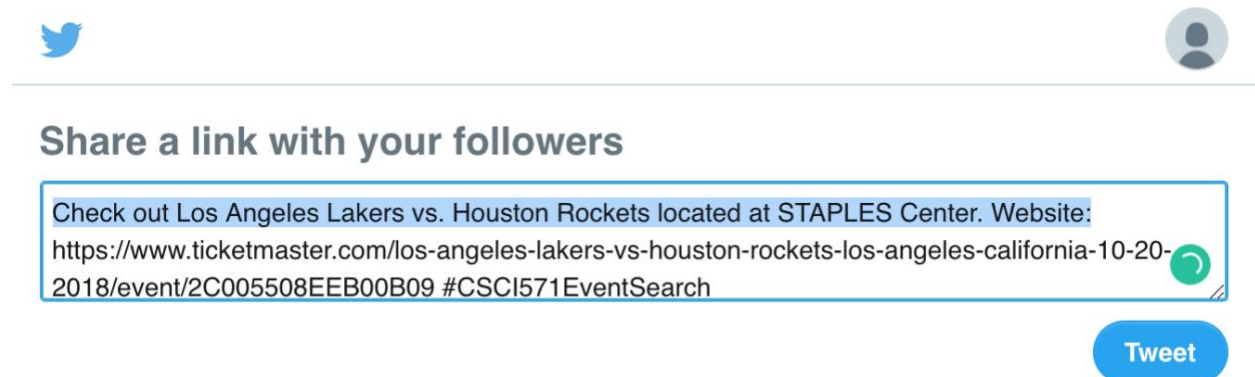
Once the **List** button is clicked, your webpage should go back to the previous list view, whether it's the result list or the favorite list.



**Figure 17** List Button

The **Favorites** button works the same way as the ones in the result list.

The **Twitter** button allows the user to **compose** a **Tweet** and **post** it to **Twitter**. Once the button is clicked, a **new dialog** should be opened and display the **default Tweet content** in this **format**: “Check out **EVENT** located at **VENUE**. Website: **URL** **#CSCI571EventSearch**”. Replace **EVENT** and **VENUE** with the real event name and venue name. Replace **URL** with the event’s URL on Ticketmaster website. For example



**Figure 18** Tweets

Adding the Twitter button to your webpage is very easy. You can visit these two pages to learn how to use Twitter Web Intents:

<https://dev.twitter.com/web/intents>

<https://dev.twitter.com/web/tweet-button/web-intent>

The **favorites** button should be **disabled until** the **content** of the **event tab** and **venue tab** are **available**.



**Figure 19** Favorite and Twitter Buttons

### 3.4 Favorites Tab

The favorites tab is very **similar** to the **results** tab: the events 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 events details by clicking on the event name in the “event” column.





The major differences between these two tabs are: the event information displayed in the favorites tab is saved in and **loaded from the local storage** of the **browser**; the buttons in the “Favorite” column of the favorites tab is only used to remove an event from the list and has a **trash** can **icon instead** of a **star icon**; the events in the favorites tab are sorted in the order they are added to the favorites list.

Please note if a user **closes** and **re-opens the browser**, its **favorite list** will **still** be **there**. If there are **no** favorites, please show ‘**No Records**’. (see Figure 22)

Results

Favorites

Details >

#	Date	Event	Category	Venue Info	Favorite
1	2018-10-07	<a href="#">UCLA Men's Soccer v Oregon State ...</a>	Soccer-Sports	Wallis Annenberg Stadium	
2	2018-10-09	<a href="#">Shannon and the Clams</a>	Other-Music	Regent Theater-CA	
3	2018-10-10	<a href="#">Hozier</a>	Rock-Music	The Wiltern	
4	2018-10-11	<a href="#">King Tut Exhibit</a>	Miscellaneous-Arts & Theatre	California Science Center	

### Figure 20 Favorite List

### 3.5 Error Messages

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

Entertainment Event Ticket Search

Keyword \*

Lakers

Category

All

Distance

10

Miles

From \*

☒ Current location

☐ Other. Please specify:

Search

Clear

Results

Favorites

Failed to get search results.

**Figure 21** Error Message

### 3.6 No Records

Whenever the search receives no records, an appropriate message should be displayed. Initially when there are no items on the favorites list, you should also show a message (see Figure 22).



The screenshot shows the 'Entertainment Event Ticket Search' form. The 'Keyword' field contains 'usc'. The 'Category' is set to 'All', 'Distance' is '10 Miles', and 'From' is 'Current location'. Below the form are 'Search' and 'Clear' buttons. Underneath the form, there are 'Results' and 'Favorites' buttons. The main heading is 'Colorado Buffaloes Football at USC Trojans Football'. To the left is a '< List' button, and to the right are social media icons for Twitter and a star icon. Below these are tabs for 'Event', 'Artist/Teams', 'Venue', and 'UpcomingEvents'. A yellow banner at the bottom states 'No records.'

**Figure 21** No Records on Upcoming Events

The screenshot shows the 'Entertainment Event Ticket Search' form. The 'Keyword' field contains 'Enter Artist, Team or Event Name (eg. Lakers)'. The 'Category' is set to 'All', 'Distance' is '10 Miles', and 'From' is 'Current location'. Below the form are 'Search' and 'Clear' buttons. Underneath the form, there are 'Results' and 'Favorites' buttons. A yellow banner at the bottom states 'No records.'

**Figure 22** No Records on Favorite List

## 4.7 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.

The desktop view of the 'Entertainment Event Ticket Search' form includes the following elements:

- Keyword \***: A text input field containing 'Lakers'.
- Category**: A dropdown menu set to 'All'.
- Distance**: A text input field with '10' and a unit dropdown menu set to 'Miles'.
- From \***: Radio buttons for 'Current location' (selected) and 'Other. Please specify:' (with an empty text input field below it).
- Buttons**: A blue 'Search' button and a 'Clear' button with a reset icon.
- Navigation**: 'Results' and 'Favorites' buttons at the bottom.

Figure 23

### 3.8 Animation

1. Whenever the view switches between results/favorites and details, there should be a sliding effect.
2. Whenever the user clicks Show More/Less buttons on upcoming events, there should be a fade-in/fade-out effect. Please check out the video to see the effect. **These two animations 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.

The three mobile snapshots show the form's responsive design:

- Left Snapshot**: The full form is visible, with fields for Keyword, Category, Distance, and From. The 'From' section has 'Current location' selected.
- Middle Snapshot**: A close-up of the 'From' section, showing the 'Current location' radio button selected and the 'Other. Please specify:' option with its input field.
- Right Snapshot**: The form with red validation boxes and messages. The 'Keyword' field has a message 'Please enter a keyword.' and the 'From' section has a message 'Please enter a location.' when 'Other. Please specify:' is selected.



### Entertainment Event Ticket Search

Keyword \*

- Los Angeles Lakers
- South Bay Lakers
- Sonny Knight & the Lakers
- Mercyhurst Lakers Mens Basketball

Done

q w e r t y u i o p  
a s d f g h j k l  
z x c v b n m  
123 space return

From \*

☒ Current location  
☐ Other. Please specify:

Search Clear

Results Favorites

Progress bar

Results Favorites

Details >

#	Date	Event
1	2018-10-20	Los Angeles Lakers vs. Houston Rockets
2	2018-10-22	Los Angeles Lakers vs. San Antonio Spurs
3	2018-10-25	Los Angeles Lakers vs. Denver Nuggets
4	2018-10-31	Los Angeles Lakers vs. Dallas Mavericks
5	2018-11-04	Los Angeles Lakers vs. Toronto Raptors
6	2018-11-07	Los Angeles Lakers vs. Minnesota Timberwolves
7	2018-11-11	Los Angeles Lakers vs. Atlanta Hawks
8	2018-11-14	Los Angeles Lakers vs. Portland Trail Blazers

Results Favorites

### Los Angeles Lakers vs. Houston Rockets

< List

Event Artist Venue Upcoming

Artist/Team(s) Los Angeles Lakers | Houston Rockets

Venue STAPLES Center

Time Oct 20, 2018 19:30:00

Category Sports | Basketball

Ticket Status Onsale

Buy Ticket At Ticketmaster

Seat Map View Seat Map Here



### Los Angeles Lakers vs. Houston Rockets

< List

Event Artist Venue Upcoming

#### STAPLES Center

Address 1111 S. Figueroa St

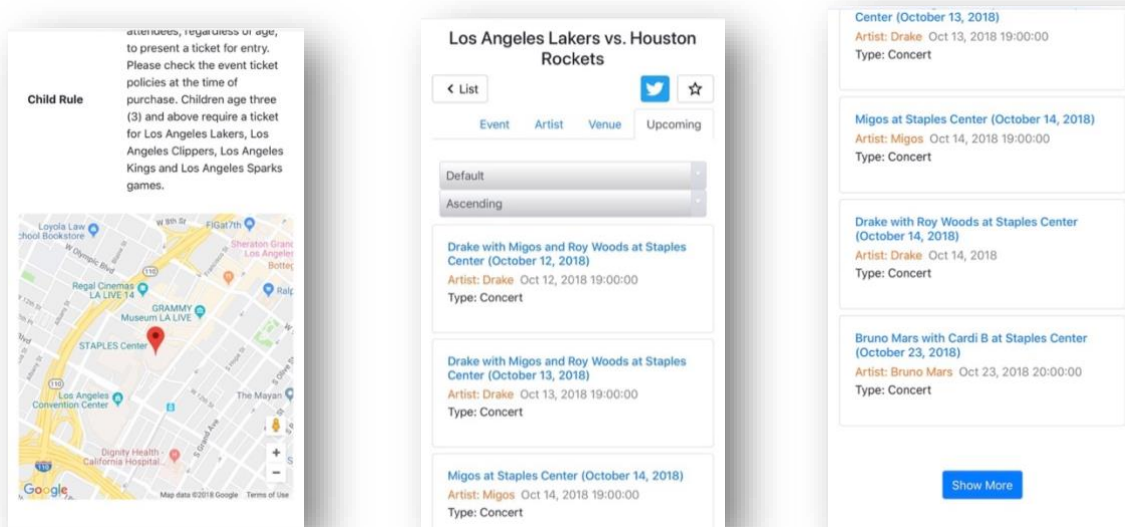
City Los Angeles, California

Phone Number 213-742-7340

Open Hours

Box office is located on North side of building at 11th and South Figueroa. Box office hours are 10am to 6pm, Monday through Saturday. It is open extended hours on event day. Phone: 213-742-7340

SUMMER HOURS Closed Saturdays and Sundays unless there is an event, the box office will open at 9am on Saturdays or 10am on Sundays only if there is an event. The box office will have



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 Spotify API

To use Spotify API, you need first to register a Spotify Account. Then create an application and get your client id and client secret.

<https://developer.spotify.com/dashboard/#>

Please refer this doc and also Spotify NodeJS libraries at:

<https://developer.spotify.com/documentation/web-api/>

<https://github.com/thelinmichael/spotify-web-api-node>

## 4.2 Songkick API

Please apply Songkick APIs **as soon as possible**, as it will take **2 to 7 days** to obtain your API key.  
[https://www.songkick.com/api\\_key\\_requests/new](https://www.songkick.com/api_key_requests/new)

Venue Search:

<https://www.songkick.com/developer/venue-search>

Venue upcoming events:

<https://www.songkick.com/developer/upcoming-events-for-venue>

## 5. Libraries

- **Moment.js** - <http://momentjs.com/> for time conversion
- **Node-geohash** - <https://github.com/sunng87/node-geohash> for geo hash conversion
- **Spotify Web API Node** - <https://github.com/thelinmichael/spotify-web-api-node>
- **Angular Google Maps** - <https://angular-maps.com/> This makes it easier to use Google Maps in Angular

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

## 6. Implementation Hints

### 6.1 Images

The images needed for this homework are available here:

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

### 6.2 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.

### 6.3 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	<a href="https://getbootstrap.com/docs/4.0/components/forms/">https://getbootstrap.com/docs/4.0/components/forms/</a>
Bootstrap Tabs	<a href="https://getbootstrap.com/docs/4.0/components/navs/#tabs">https://getbootstrap.com/docs/4.0/components/navs/#tabs</a>
Bootstrap Progress Bars	<a href="https://getbootstrap.com/docs/4.0/components/progress/">https://getbootstrap.com/docs/4.0/components/progress/</a>
Bootstrap Alerts	<a href="https://getbootstrap.com/docs/4.0/components/alerts/">https://getbootstrap.com/docs/4.0/components/alerts/</a>
Bootstrap Tooltip	<a href="https://getbootstrap.com/docs/4.1/components/tooltips/">https://getbootstrap.com/docs/4.1/components/tooltips/</a>
Bootstrap Cards	<a href="https://getbootstrap.com/docs/4.0/components/card/">https://getbootstrap.com/docs/4.0/components/card/</a>

## 6.4 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>

## 6.5 Material Icon

Icons for the search button, clear button, left arrow, right arrow, star, star border and trash can be viewed here:

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

<https://material.io/tools/icons/>

## 6.6 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/searchEvents?keyword=xxx>

AWS - <http://example.elasticbeanstalk.com/searchEvents?keyword=xxx>

Azure – <http://example.azurewebsites.net/searchEvents?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.

## 6.7 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.

## 6.8 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 6.6 above). Please refer to the grading guidelines for details.

## 6.9 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](http://www.w3schools.com/html/html5_webstorage.asp)

## 7. 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 6.6).

Also, **submit all** your files (**HTML, JS, CSS, TS**) electronically to the **GitHub Classroom** repository so that can be compared to all other students' code. **Don't include library** or any **images** that we provided or that are included in any library or any code generated by the tools.

**\*\*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.