

# 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 Google Cloud App Engine.
- Learn to use popular APIs such as Ticketmaster APIs, Spotify APIs, Google Maps APIs, Google Customized Search APIs and Twitter APIs.

### 2. Background

#### 2.1 AJAX and JSON

AJAX (Asynchronous JavaScript + XML) incorporates several technologies:

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

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

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

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

#### 2.2 Bootstrap

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

<http://csci571.com/slides/Responsive.pdf>

#### 2.3 Google App Engine (GAE)

Google App Engine applications are easy to create, easy to maintain, and easy to scale as your traffic and data storage needs change. With App Engine, there are no servers to maintain. You simply upload your application and it's ready to go. App Engine applications automatically scale

based on incoming traffic. Load balancing, micro services, authorization, SQL and noSQL databases, memcache, traffic splitting, logging, search, versioning, roll out and roll backs, and security scanning are all supported natively and are highly customizable.

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

<https://cloud.google.com/appengine/docs/flexible/Node.js/>

## 2.4 Angular

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

For this homework, Angular 8+ (Angular 8, 9 or 10) can be used, but Angular 10 is recommended. Please note Angular 8+ will need familiarity with Typescript and component-based programming. **AngularJS will not be an acceptable choice.**

To learn more about Angular 8+, visit this page:

<https://angular.io/>

## 2.5 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/>

## 2.6 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.

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

The image shows a web form titled "Entertainment Event Ticket Search". It has the following fields and controls:

- Keyword \***: A text input field with a placeholder "Enter Artist, Team or Event Name (eg. Lakers)".
- Category**: A dropdown menu currently showing "All".
- Distance**: A text input field showing "10" and a unit dropdown menu showing "Miles".
- From \***: Two radio buttons. The first is "Current location" (selected). The second is "Other, Please specify:" followed by a text input field.
- Buttons**: A blue "Search" button with a magnifying glass icon and a "Clear" button with a reset icon.
- Tabs**: Below the form are two tabs: "Results" (active) and "Favorites".

**Figure 1** Initial Search Form

### 3.1 Search Form

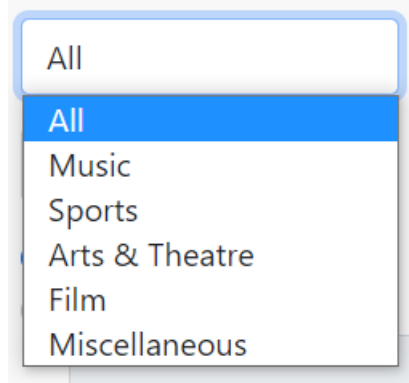
#### 3.1.1 Design

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

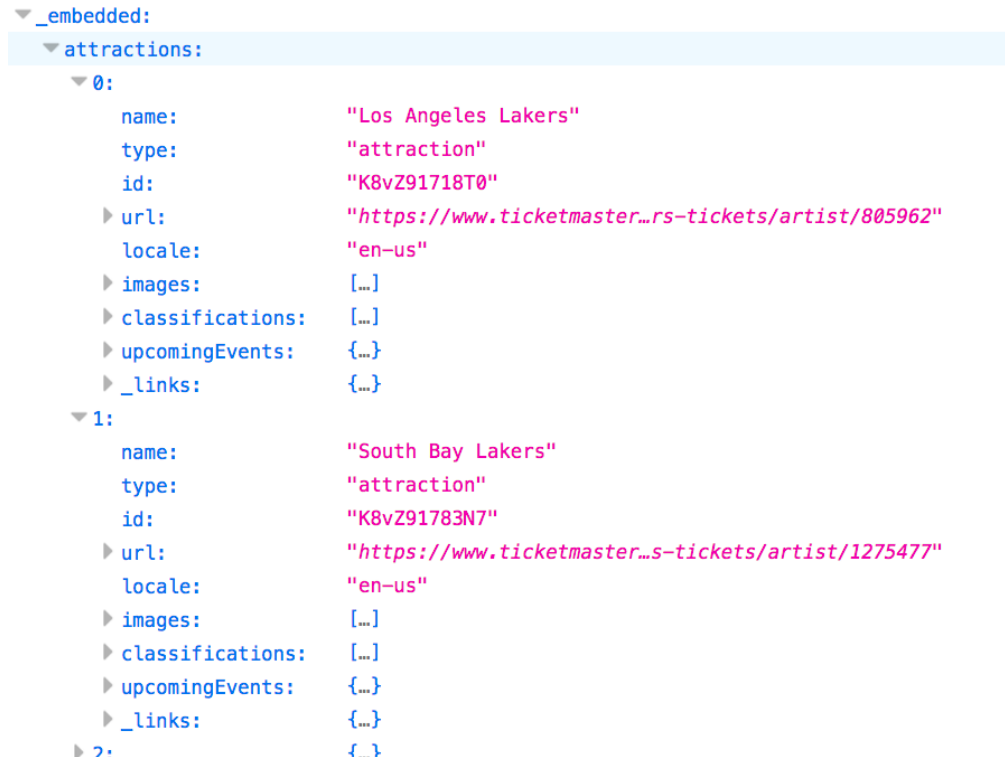
Autocomplete is implemented by using the suggestion service provided by Ticketmaster. Please go to this page to learn more about this service:

<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:

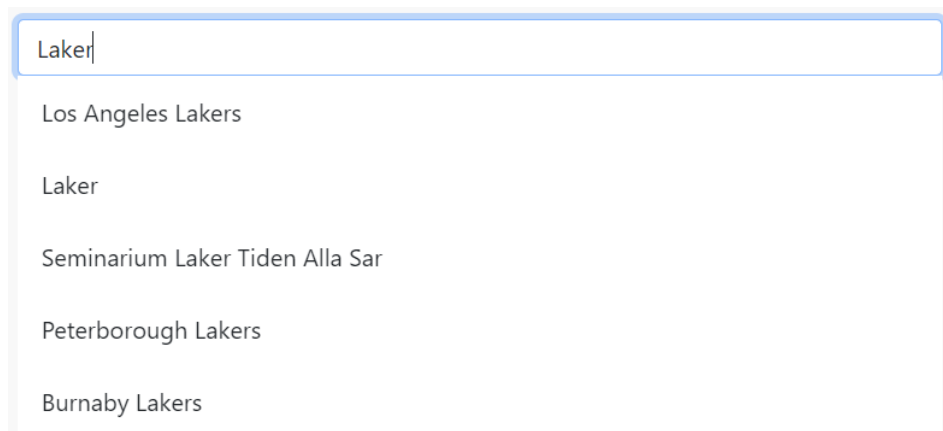
```
https://app.ticketmaster.com/discovery/v2/suggest?apikey=YOUR_API_KEY&keyword=laker
```

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.

**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.

[https://ipinfo.io/?token=YOUR\\_TOKEN\\_NUMBER](https://ipinfo.io/?token=YOUR_TOKEN_NUMBER)

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. The Node.js script on GAE 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	2021-08-14	<a href="#">Los Angeles Rams vs. Los Angeles Ch</a>	NFL   Football   Sports   Team   Group	SoFi Stadium	☆
2	2021-08-21	<a href="#">Los Angeles Rams vs. Las Vegas Raid</a>	NFL   Football   Sports   Team   Group	SoFi Stadium	☆
3	2021-09-12	<a href="#">Los Angeles Rams vs. Chicago Bears</a>	NFL   Football   Sports   Team   Group	SoFi Stadium	☆
4	2021-09-26	<a href="#">Los Angeles Rams vs. Tampa Bay Bucc</a>	NFL   Football   Sports   Team   Group	SoFi Stadium	☆
5	2021-10-03	<a href="#">Los Angeles Rams vs. Arizona Cardin</a>	NFL   Football   Sports   Team   Group	SoFi Stadium	☆
6	2021-10-24	<a href="#">Los Angeles Rams vs. Detroit Lions</a>	NFL   Football   Sports   Team   Group	SoFi Stadium	☆
7	2021-11-07	<a href="#">Los Angeles Rams vs. Tennessee Tita</a>	NFL   Football   Sports   Team   Group	SoFi Stadium	☆
8	2021-12-05	<a href="#">Los Angeles Rams vs. Jacksonville J</a>	NFL   Football   Sports   Team   Group	SoFi Stadium	☆
9	2021-12-19	<a href="#">Los Angeles Rams vs. Seattle Seahaw</a>	NFL   Football   Sports   Team   Group	SoFi Stadium	☆
10	2022-01-09	<a href="#">Los Angeles Rams vs. San Francisco</a>	NFL   Football   Sports   Team   Group	SoFi Stadium	☆

**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	2021-08-14	<a href="#">Los Angeles Rams vs. Los Angeles Ch</a> <div>Los Angeles Rams vs. Los Angeles Chargers</div>	NFL   Football   Sports   Team   Group	SoFi Stadium	☆
2	2021-08-21	<a href="#">Los Angeles Rams vs. Las Vegas Raid</a>	NFL   Football   Sports   Team   Group	SoFi Stadium	☆

**Figure 7** An Example of the tooltip for event name

For tooltip component, if you use entirely Angular8+, 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)

### 3.2.2 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

## 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:

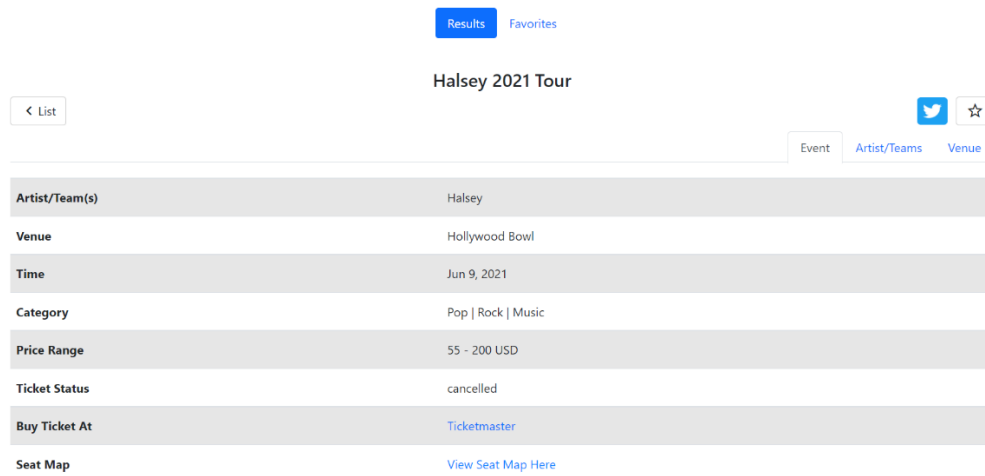
Fields in the table	Corresponding response object fields
Artist/Team(s)	<i>The value of the “name” attribute that is part of the “attractions” object, segmented by “   ”</i>
Venue	<i>The value of the “name” attribute that is part of the “venue” object.</i>
Time	<i>The value of the “localDate” attribute 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 Angular date Pipe to do the conversion into the format “Oct 12, 2018”.

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



Halsey 2021 Tour	
Artist/Team(s)	Halsey
Venue	Hollywood Bowl
Time	Jun 9, 2021
Category	Pop   Rock   Music
Price Range	\$5 - 200 USD
Ticket Status	cancelled
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 tab shows the music-related artist info using Spotify APIs. Please see figure 10.

#### 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/6SrKlwioiqWe9Sa00YxSma"
      followers:
        href: null
        total: 3234
      genres: []
      href: "https://api.spotify.com/v1/artists/6SrKlwioiqWe9Sa00YxSma"
      id: "6SrKlwioiqWe9Sa00YxSma"

```

**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

The screenshot shows the Spotify web interface for the artist Maroon 5. At the top, there are tabs for 'Results' and 'Favorites'. Below the tabs, the artist's name 'Maroon 5' is displayed. To the right of the name are social media icons for Twitter and a star icon. Below the name, there are three tabs: 'Event', 'Artist/Teams', and 'Venue'. The main content area shows a table with the following data:

Maroon 5	
Name	Maroon 5
Followers	31,009,899
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 Angular8+ users, please use

<https://github.com/crisbeto/angular-svg-round-progressbar>

### 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 “ <i>line1</i> ” attribute that is part of the “ <i>address</i> ” object.
City	The value of the “ <i>name</i> ” attribute of “ <i>city</i> ” object and “ <i>state</i> ” object, connected by a comma.
Phone Number	<i>The value of the “phoneNumberDetail” attribute that is part of the “boxOfficeInfo” object.</i>
Open Hours	<i>The value of the “openHoursDetail” attribute that is part of the “boxOfficeInfo” object.</i>
General Rule	<i>The value of the “generalRule” attribute that is part of the “generalInfo” object.</i>
Child Rule	<i>The value of the “childRule” attribute that is part of the “generalInfo” object.</i>

**Table 4:** 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.

A sample of venue tab is shown as Figure 11.

**Maroon 5**

< List

Twitter
☆

Event
Artist/Teams
Venue

---

**Address**  
**City**  
**Phone Number**  
**Open Hours**

3939 S. Figueroa St

Los Angeles, California

(213) 519-9900

Monday - Friday, 10:00 am - 6:00 pm. Hours are extended to accommodate the needs of specific events.

---

**General Rule**

We aim to provide an enjoyable, yet exciting environment for ALL fans attending games. In order to achieve that, the following are prohibited: - Abusive, foul or disruptive language or clothing- Alcohol and illegal drugs- All bags, including clear bags, backpacks, and purses, that are larger than 12" x 6" x 12"- Animals (with the exception of service animals)- Any type of marketing collateral such as pamphlets, product samples, etc.- Balloons, beach balls, frisbees, markers, brooms, skateboards, roller blades, skates, etc.- Clothing containing wires, batteries, or other electronic components- Coolers, outside food, beverages (with the exception of a single factory sealed plastic water bottle no more than 20oz), cans, flasks, bottles, etc. (F&B needed for infants or medical reason will be permitted with approval)- Drums & Drumsticks- Explosives, pepper spray, tear gas, etc.- Flags larger than 3' x 4'- Flag/banner poles - Folding/beach chairs- Items containing adhesive ie. Stickers, Decals, etc.- Laptops & tablets- Loud noise-making devices such as air horns, plastic horns, vuvuzelas, whistles, megaphones, etc.- Objects that can be thrown or viewed as projectiles- Professional camera equipment, tripods, GoPros, etc. - Selfie Sticks- Smoking of any kind including E-cigarettes & Vaporizers- Streamers, Confetti, Register tape, etc.- Umbrellas- Unmanned and Remote-Controlled Aircraft Systems- Water guns, toy/replica weapons, squirt bottles, etc.- Weapons of any kind, fireworks, slingshots, smoke/stink bombs, laser pointers, etc.- 2-way radio devices- At the discretion of Banc of California Stadium Management, any other item determined to be prohibited. - Any attempt to bring alcohol into the Stadium will be considered a violation of the Code of Conduct and may result in an ejection or refusal of entry, arrested, or tickets being revoked. All persons and/or their belongings are subject to search. Violators of the rules of any State or City laws will be refused admission, asked to leave, or ejected from Stadium property. Banc of California Stadium cannot safeguard items which are not permitted to be brought into the stadium. Please keep in mind that permitted or prohibited items may vary depending on the event. Banc of California Stadium management will communicate any deviations from the standard house policy as necessary.

---

**Child Rule**

Children two and under will be admitted free but will be required to sit on a ticket holders lap. Children who have had their third birthday, or who require their own seat will require a ticket.

Figure 11 Venue Tab

### 3.3.4 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 12 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*. #CSCI571EventSearch”. Replace EVENT and VENUE with the real event name and venue name. For example



Figure 13 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 14 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 19)

Results

Favorites

Details >


#	Date	Event	Category	Venue Info	Favorite
1	2021-06-09	<a href="#">Halsey 2021 Tour</a>		Hollywood Bowl	

Figure 15 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.

Keyword \*

Category

Distance

From \*

Entertainment Event Ticket Search

Lakers

All

10

Miles

☒ Current location

☐ Other. Please specify:

Search

Clear

Results

Favorites

Failed to get search results.

Figure 16 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 19).

Entertainment Event Ticket Search

Keyword \*

hhhh

Category

All

Distance

10

Miles

From \*

☒ Current location

☐ Other. Please specify:

Search

Clear

Results

Favorites

No records.

Figure 17 No Records on Result List

Results

Favorites

Hollywood Roses (The Ultimate Guns N' Roses Experience)

< List

Event

Artist/Teams

Venue

Twitter

Star

No records.

Figure 18 No Records on Artist Tab

The screenshot shows a search form titled "Entertainment Event Ticket Search". It includes fields for "Keyword \*" (with placeholder text "Enter Artist, Team or Event Name (eg. Lakers)"), "Category" (set to "All"), "Distance" (set to "10" with a "Miles" unit selector), and "From \*" (with radio buttons for "Current location" and "Other. Please specify:"). Below the form are "Search" and "Clear" buttons. At the bottom, there are "Results" and "Favorites" buttons. A yellow banner at the bottom of the form area displays the text "No records."

**Figure 19** No Records on Favorite List

### 3.7 Progress Bars

**Whenever** data is being fetched, a dynamic progress bar must be displayed as shown in Figure 20.

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.

This screenshot is identical to Figure 19, but with the "Keyword" field populated with the text "Halsey". Below the search form, a blue progress bar is shown, consisting of a solid blue segment followed by a dashed blue segment, indicating a loading or progress state.

**Figure 20**

### 3.8 Animation

1. Whenever the view switches between results/favorites and details, there should be a sliding effect.



### 3.9 Responsive Design

The following are snapshots of the webpage opened with Chrome on iPhone 7/8/9 Plus.

The following are six snapshots of the 'Entertainment Event Ticket Search' app on an iPhone, demonstrating its responsive design across different screen sizes and states.

**Snapshot 1 (Top Left):** The search form is displayed. It includes a 'Keyword' field with a placeholder 'Enter Artist, Team or Event Name (eg. La)', a 'Category' dropdown set to 'All', a 'Distance' field set to '10 Miles', and a 'From' section with radio buttons for 'Current location' (selected) and 'Other. Please specify:'. A 'Search' button and a 'Clear' button are at the bottom.

**Snapshot 2 (Top Middle):** The search form is shown with a 'No records.' message displayed in a yellow box at the bottom.

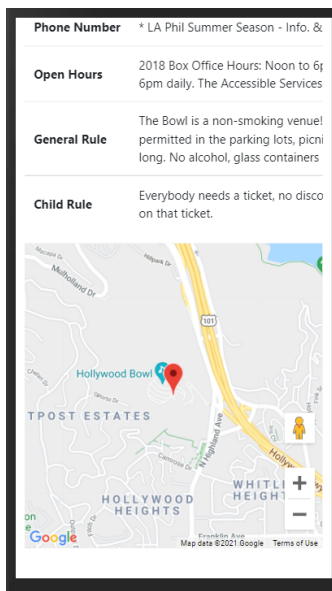
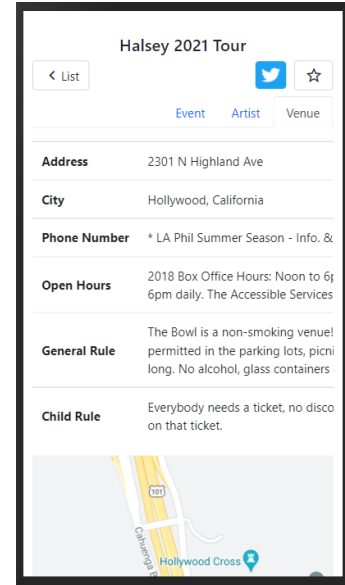
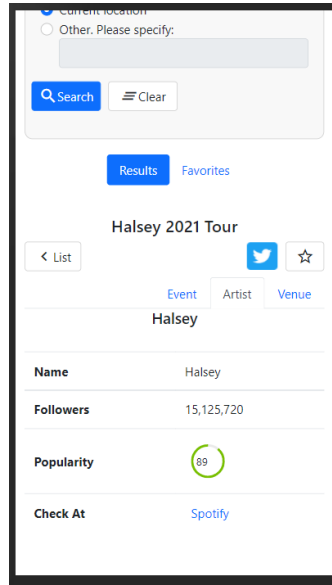
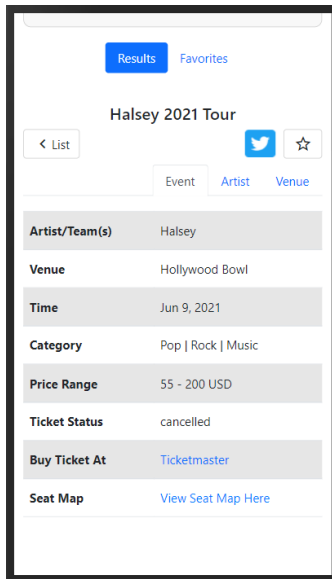
**Snapshot 3 (Top Right):** The search form is shown with a 'Keyword' field containing 'Enter Artist, Team or Event Name (eg. Lakers)' and a 'Please enter a keyword.' error message. The 'From' section also shows a 'Please enter a location.' error message.

**Snapshot 4 (Bottom Left):** The search form is shown with a list of suggestions for the keyword 'Lak': 'Los Angeles Lakers', 'Salt Lake Bees', 'Lake Street Dive', 'Lakeyah', and 'Swan Lake'. The 'From' section is also visible.

**Snapshot 5 (Bottom Middle):** The search form is shown with a 'Distance' field set to '10 Miles' and a 'Results' button.

**Snapshot 6 (Bottom Right):** The 'Results' screen is displayed, showing a list of events. The table below represents the data shown in the app.

#	Date	Event	Category	Venue Info	Favorites
1	2021-08-14	Los Angeles Rams vs. Los Angeles Chargers	NFL   Football   Sports   Team   Group	SoFi Stadium	☆
2	2021-08-21	Los Angeles Rams vs. Las Vegas Raiders	NFL   Football   Sports   Team   Group	SoFi Stadium	☆
3	2021-09-12	Los Angeles Rams vs. Chicago Bears	NFL   Football   Sports   Team   Group	SoFi Stadium	☆
4	2021-09-26	Los Angeles Rams vs. Tampa Bay Buccaneers	NFL   Football   Sports   Team   Group	SoFi Stadium	☆



**Note:** The black border is the device frame.

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”).
- Animations must work 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>

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

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

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

## 6.4 Angular Material

Angular Material (Angular 8+) :<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

You should use the domain name of the GAE service you created in Homework #7 to make the request. For example, if your GAE server domain is called example.appspot.com/, 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>

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

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

## 6.8 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 (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 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 front-end and back-end files (HTML, JS, CSS, TS) electronically as a **SINGLE ZIP FILE** to the DEN D2L dropbox folder 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.