HOMEWORK ASSIGNMENT 8

CSCI 571 – Spring 2023

Abstract

Ajax, JSON, Angular, Bootstrap, RWD, Node.js, and Ticketmaster API

This content is protected and may not be shared, uploaded, or distributed.

Prof. Marco Papa

papa@usc.edu

Homework 8: Ajax, JSON, Angular, Bootstrap, RWD, Node.js, and Ticketmaster API

1. Objectives

1. Get experience with creating backend applications using JavaScript/Node.js on the server side with Express framework.
2. Get experience with using Angular, TypeScript, and Bootstrap on the client side and creating responsive front-end.
3. Get experience with using HttpClientModule of Angular for AJAX.
4. Get experience with Ticketmaster APIs, Spotify APIs, Google Maps APIs, Google Geocoding APIs, Ipinfo APIs, Facebook APIs, and Twitter APIs.
5. Get experience with Cloud Platform (GCP, AWS, Azure).

2. Homework Description Resources

1. Homework Description Document (This document)
2. Grading Guidelines
3. Web Reference Video: YouTube video
4. Mobile Reference Video: YouTube video
5. Piazza

3. General Directions

1. The backend of this homework must be implemented in Javascript using the Node.js Express framework. Refer to Node.js website for installing Node.js and learning how to use it. Have a look at "Getting started" guides in the Express website to learn how to create backend applications using Express. Also, refer to the Node.js Express framework tutorial to learn more about it. The Axios library can be useful to make requests from your Node.js backend to Ticketmaster servers. **Implementing the backend in anything other than Node.js will result in a major point reduction.**
2. The frontend of this homework must be implemented using the Angular framework, for which Angular 8 or above should be used. It requires familiarity with Typescript and component-based programming. Refer to the Angular setup docs for installing Angular and creating Angular projects. The Angular "Tour of Heroes" app tutorial is a very good tutorial to see different Angular concepts in action. **Implementing the frontend in anything other than Angular or React will result in a major point reduction. Students willing to use React, should expect no support from the CSCI 571 staff (i.e., Instructor, TAs and CPs).**

3. You are expected to create a responsive website. For that reason, we require you to use **Bootstrap**, a CSS framework for responsive, mobile-first web development. Bootstrap will save you from the burden of dealing with CSS peculiarities and the website you create will be responsive automatically, if you develop within the framework provided by Bootstrap. Please refer to the Bootstrap docs for reference (especially look at the "Layout" section, and the components that you want to use). Refer to this post to learn how to add Bootstrap to Angular projects. **Failing to use Bootstrap will result in a major point reduction.**
4. The backend and frontend of this homework must be deployed to the cloud. The backend should serve the frontend as well as other endpoints you may define. Please refer to Homework 7 for deploying Node.js applications to GCP/Azure/AWS.
5. You must refer to the homework description document (this document), grading guidelines, the reference videos and instructions on Piazza while developing this homework. All discussions and explanations in Piazza related to this homework are part of the homework description and grading guidelines. Therefore, please review all Piazza threads before submitting the assignment. If there is a conflict between Piazza and this description and/or the grading guidelines, answers/clarifications in Piazza take priority. Piazza posts / responses by the instructors will not increase the functionality of the assignment, and only provide additional specifics on items in this document. **In general, if something is not specified in this document, you should feel free of implementing that functionality in any reasonable fashion**. This is especially true of many "boundary" conditions, which I am sure we will miss.
6. Making calls to the Ticketmaster API from the NodeJS backend can be done using a variety of libraries, such as axios(), node-fetch() and others.
7. The assignment will be graded using the latest version of the Google Chrome browser. Developing this assignment using the latest version of Google Chrome is recommended.

4. System Overview

The system contains three components: 1) browser (frontend), 2) Node.js application (backend) and 3) Ticketmaster servers. You will have to implement both the frontend and the backend. Your backend will include two major functionalities: serving the frontend static files to the browser and responding to the frontend's AJAX requests by fetching data from Ticketmaster servers. You will not directly call the Ticketmaster APIs from the frontend as it requires disclosing a secret API key to the public. The data flow diagram after an AJAX call is shown below in **Figure 1**.

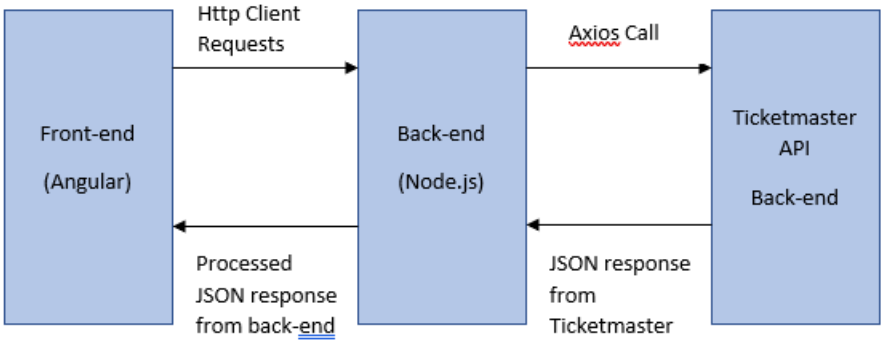


Figure 1: System design overview

NOTE: Setup authorization is required to call Ticketmaster API endpoints and is using request headers with the API key for the app. Refer to the Additional Hints section to see how to add authentication headers. You can re-use the API Key obtained and used in Assignment 6.

Please do not directly call the Ticketmaster API endpoints from your frontend. Calls to other APIs can be made from the frontend.

All requests from the frontend to the backend must be implemented using the HTTP GET method, as you will not be able to send us sample backend endpoint links as a part of your submission, if you use HTTP POST.

5. Description

In this exercise, you are asked to create a web application that allows you to search for event information using the Ticketmaster API, and the results will be displayed in a card in tabular format. The application will also allow users to mark events as "Favorites" and see the list of all events marked as favorites. Also, users can share a post on Facebook and a tweet on Twitter about the events.

All implementation details and requirements will be explained in the following sections.

There are 2 front-end routes/pages for this application:

a) Search Route ['/search'] – It is the default route of this application which is used to search for events and see event details
b) Favorites Route ['/favorites'] – It displays the list of favorite events

Hint: Refer Angular routing tutorial for implementing routes.

5.1 Navbar component

The Navigation bar must be present on top of all the routes of the application as shown in **Figure 2** below. You can use Bootstrap to create a navbar. It consists of the following menu options:

1. Search
2. Favorites

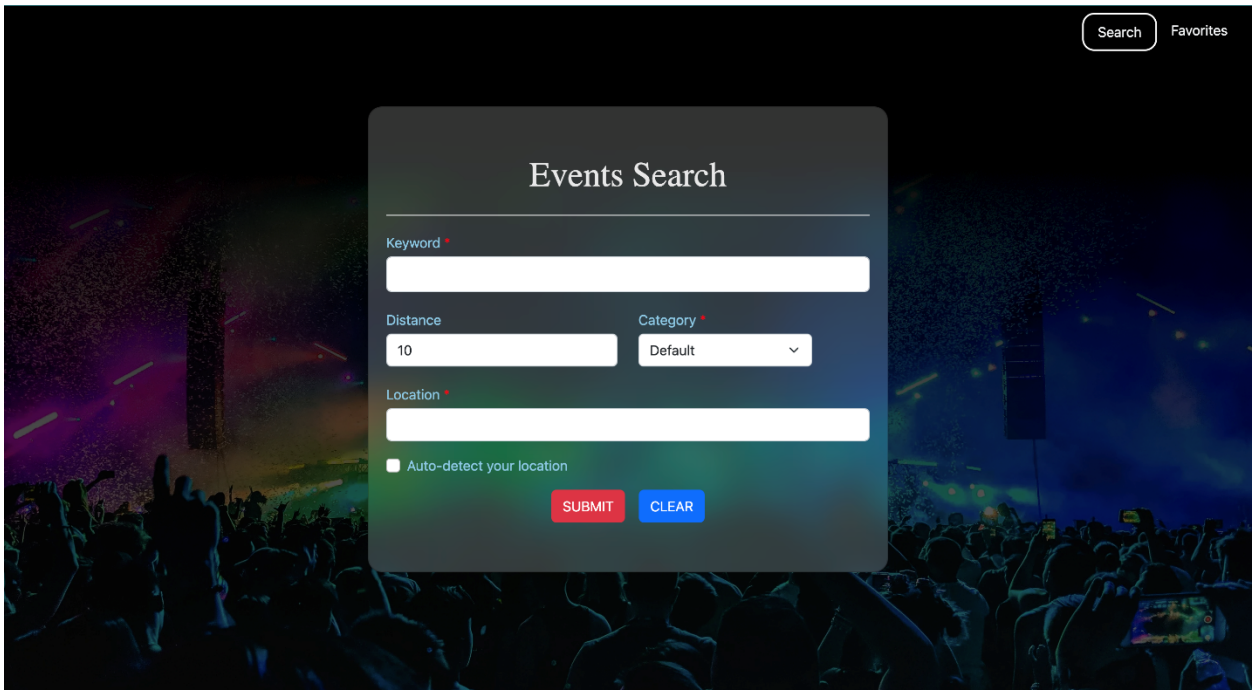


Figure 2: Navbar

5.2 Search Route

The Search route consists of various components:

- Search Form
- Results Table
- Event Details card
- Mark event as “Favorite”

5.2.1 Search Form

The form has 5 components: Keyword, Distance (miles), Category, Location, and a checkbox to auto-detect location as you can see in **Figure 3**. You should use the ipinfo.io API (as you did in Assignment 6) to fetch the user’s geolocation, if the location checkbox is checked. Otherwise,

the user must enter an address location to search. Keyword, Distance and Location should be implemented as text boxes while Category should be implemented as a dropdown.

These are the categories to include in the dropdown:

- Music
- Sports
- Arts & Theatre
- Film
- Miscellaneous

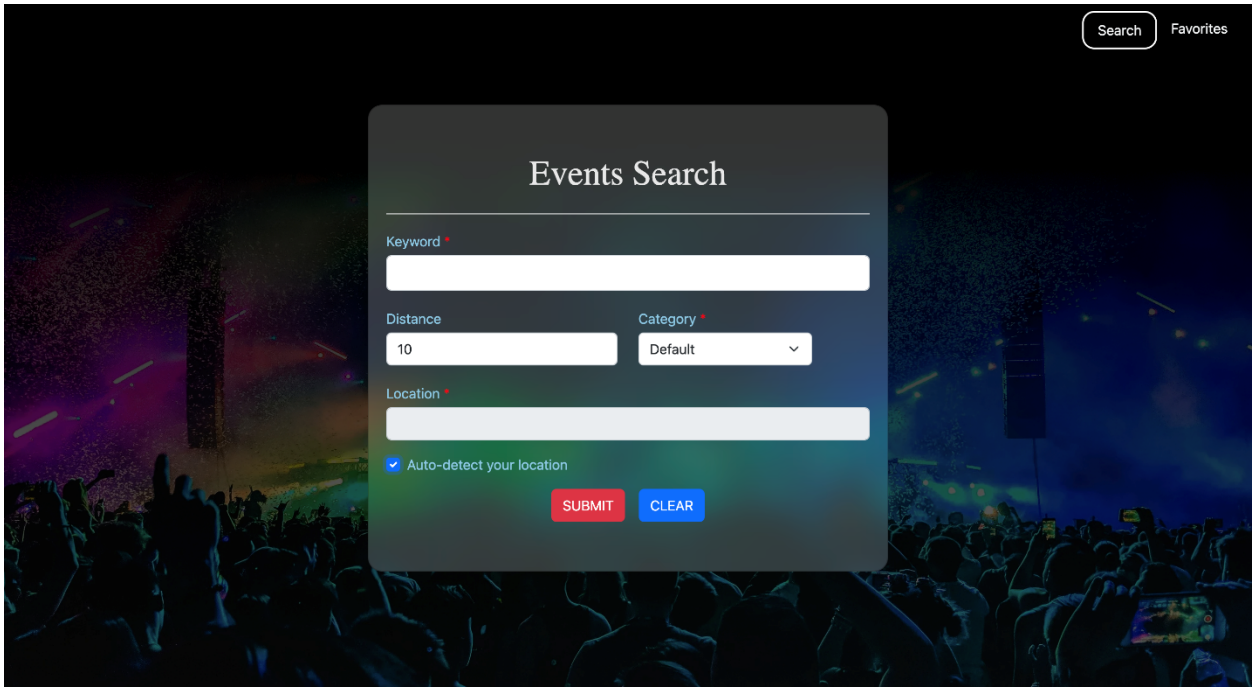


Figure 3: Events search form

5.2.1.1 Autocomplete

The Keyword Input field allows the user to enter a keyword to retrieve results. Based on the user input, the text box should display a list of all the suggestions fetched using a *Ticketmaster suggest API*, as seen in **Figure 4**).

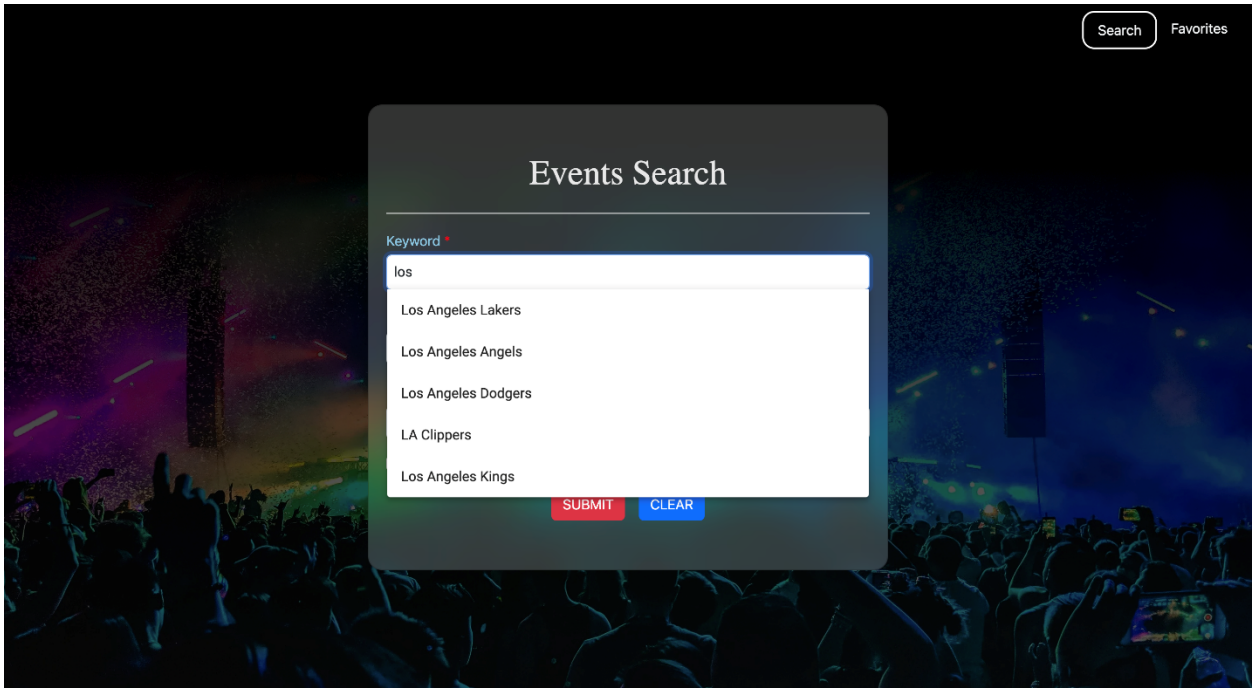


Figure 4: Example of Autocomplete

This is the API endpoint for autocomplete from the *Ticketmaster suggest API*:
GET https://app.ticketmaster.com/discovery/v2/suggest

Please refer to this documentation for the *Ticketmaster Suggest API*:
https://developer.ticketmaster.com/products-and-docs/apis/discovery-api/v2/suggest

This is a sample API call:

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

The response is a JSON object, as shown in **Figure 5**.

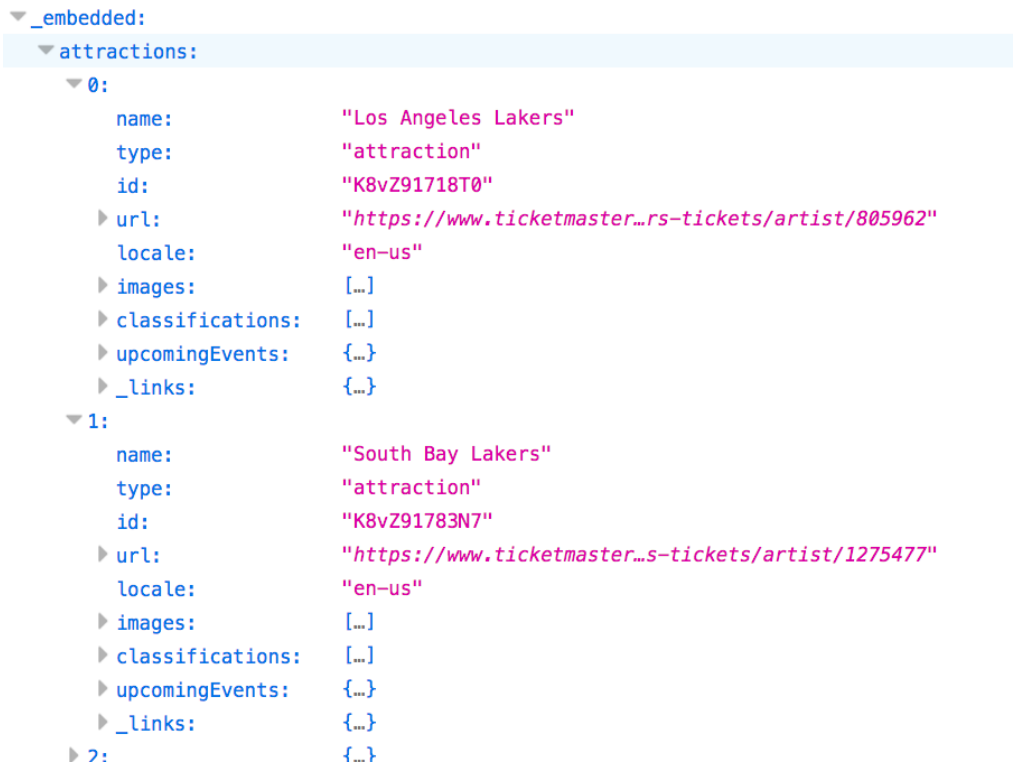


Figure 5: Autocomplete JSON Response

The autocomplete function should be implemented using **Angular Material Autocomplet**e. Please refer to the Angular Material Autocomplete tutorial, for the details.

5.2.1.2 Location Field

If the "**Auto-detect your location**" **checkbox** is checked then the location field should reset the **Location** textbox to blank and disable the field as shown in **Figure 7**. When the **Auto-detect checkbox** is not checked, the user needs to enter the location address. Use the *Google Maps Geocoding API* to get latitude and longitude of the location that the user entered and pass that latitude / longitude values in the Ticketmaster's event search API. This behavior is pretty much the same as in Assignment 6.

The *Google Maps Geocoding API* is documented here:
https://developers.google.com/maps/documentation/geocoding/start

The *Google Maps Geocoding API* expects two parameters:

1. **address**: The location that you want to geocode, in the format used by the national postal service of the country concerned. Additional address elements such as event names and unit, suite or floor numbers should be avoided.

2. **key**: Your Google application's API key. This key identifies your application for purposes of quota management.

An example of an HTTP request to the *Google Maps Geocoding API*, when the location address is "University of Southern California, CA" is shown below:

https://maps.googleapis.com/maps/api/geocode/json?address=University+of+Southern+California+CA&key=YOUR_API_KEY

The response includes the latitude (lat) and longitude (lng) of the address

Figure 6: Example of JSON object from Google Maps Geocoding

Figure 6 shows an example of the JSON object returned by the *Google Maps Geocoding API* web service response.

The latitude (lat) and longitude (lng) of the location are used when constructing a RESTful web service URL to retrieve matching search results.

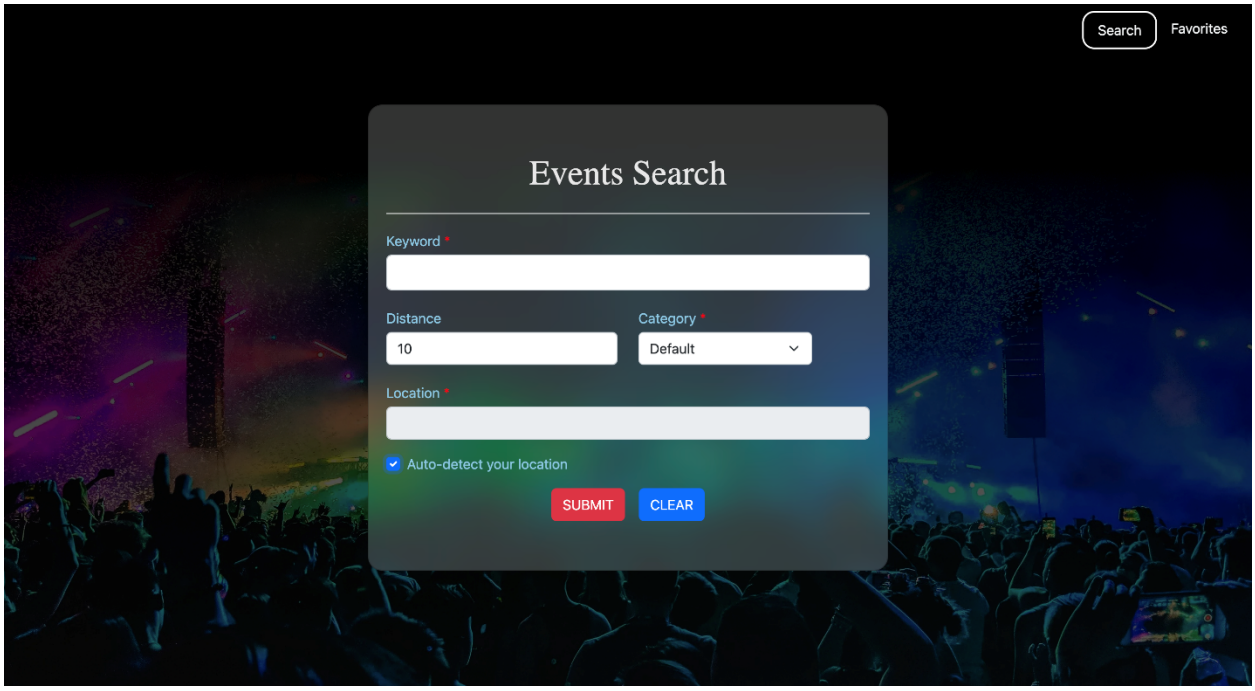


Figure 7: Example of auto-detect location selection

5.2.1.3 Submit button

Clicking the **SUBMIT button** performs a search using the given event *keyword*, the *distance* filter from the given location and the *category* of events. An example of valid input is shown in **Figure 9**. Once the user has provided valid input, your frontend should send a request to your backend NodeJS server with the form inputs. You must use GET to transfer the form data to your web server (**do not use POST**, as you would be unable to provide a sample link to your cloud services). The backend code will extract the form inputs and make an axios() HTTP call, using the inputs to invoke the *Ticketmaster API* event information service . You need to use the backend to make all the Ticketmaster API calls. You can use other NodeJS HTTP client libraries or promises in lieu of axios().

If the user clicks on the SUBMIT button without providing a value in the "Keyword", "Category" or "Location" fields or checking the location checkbox, you should show an error "tooltip" that indicates which field is missing. An example is shown in **Figure 8**.

Using XMLHttpRequest for anything other than calling your own "cloud" backend will lead to a 4-point penalty. Do not call the Ticketmaster API or directly from the front-end. You may use XMLHttpRequest to call the *Ipinfo API* service and the Google Geocoding API directly from the front-end JavaScript, as in Assignment 6.

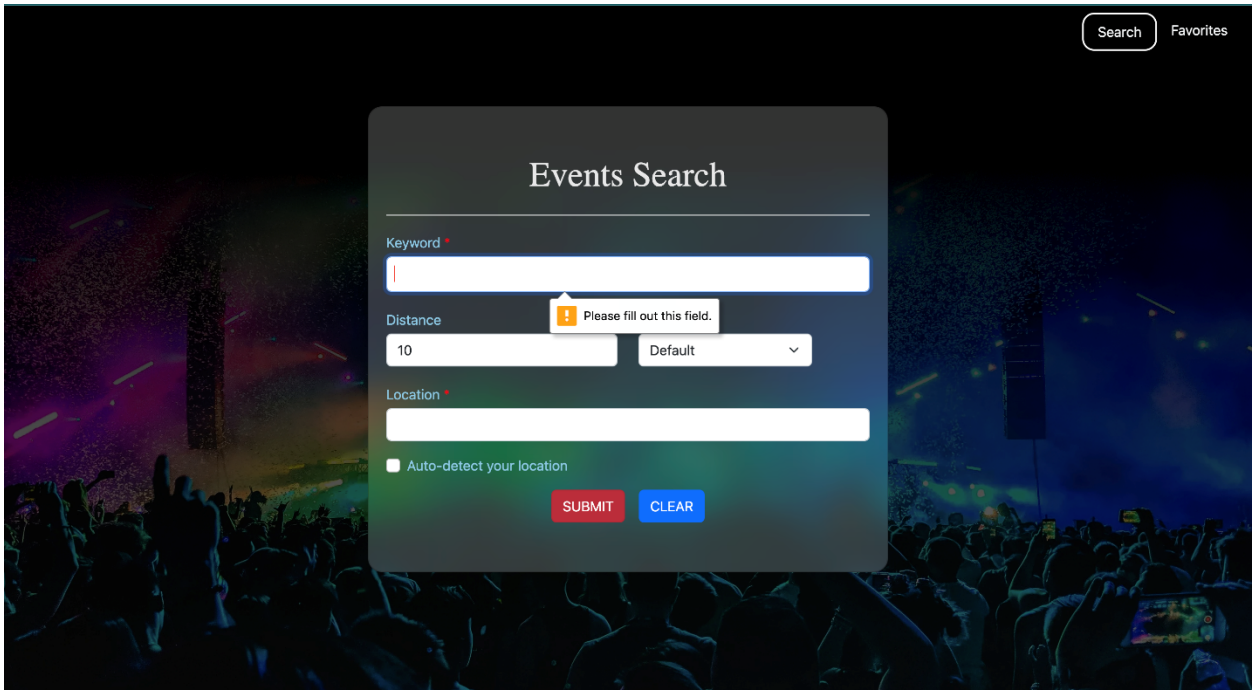


Figure 8: Example of form validation

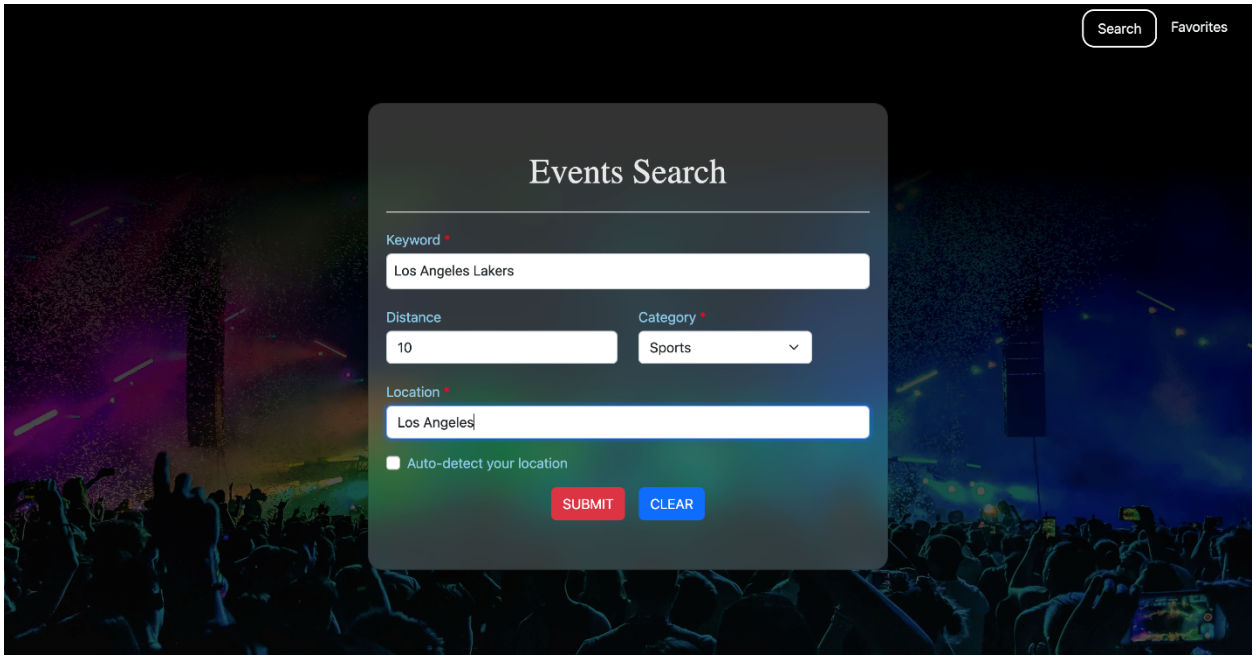


Figure 9: Example of valid input

5.2.1.4 Clear Button

The **CLEAR button** must clear the result area (below the search area) and set all the form fields to the default values in the search area. The **"clear"** operation must be done using a JavaScript function.

5.2.2 Results Table

The response received from the backend after clicking the SUBMIT button will be parsed and displayed in a table as shown in **Figure 10**. The **results table** consists of 5 columns:

1) Date/Time
2) Icon
3) Event
4) Genre
5) Venue

The results table must display a maximum of **20 search results**. Each row of the results table is such that, when clicking anywhere in the row, you should display a card showing more details about the event, which will be discussed in the next section.

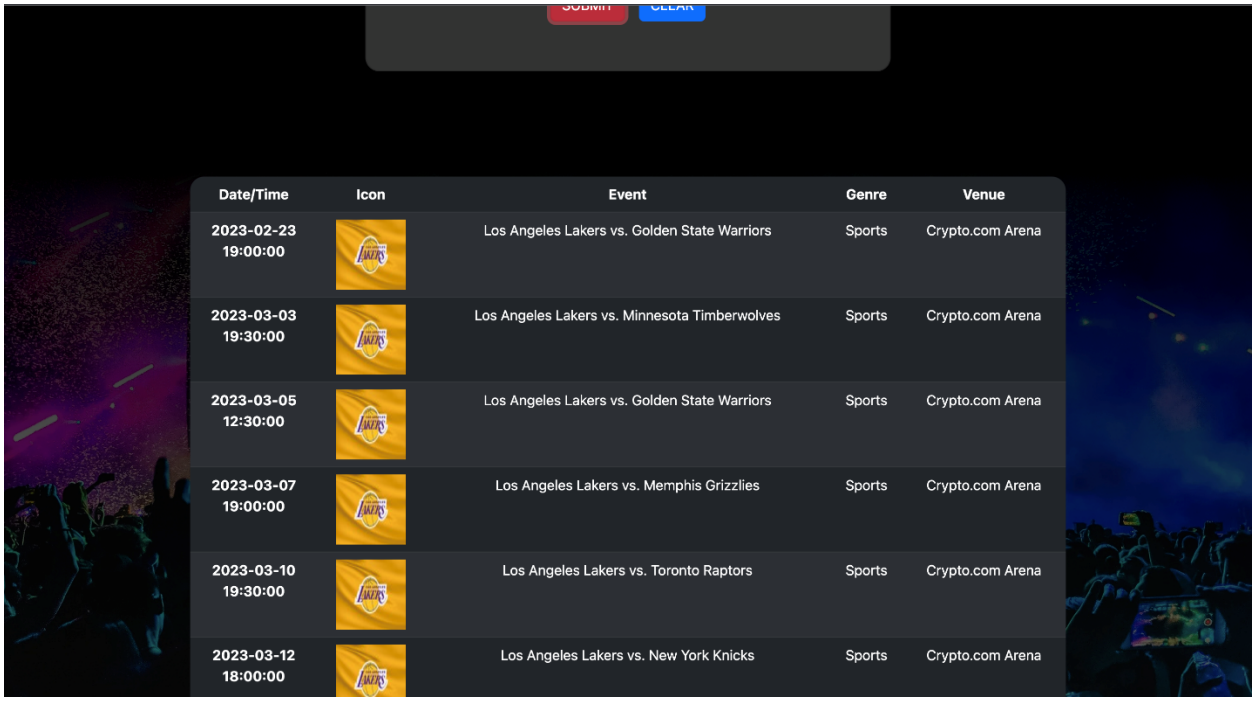


Figure 10: Example of results table

If results are not found, display "No results available", as shown in **Figure 11**.

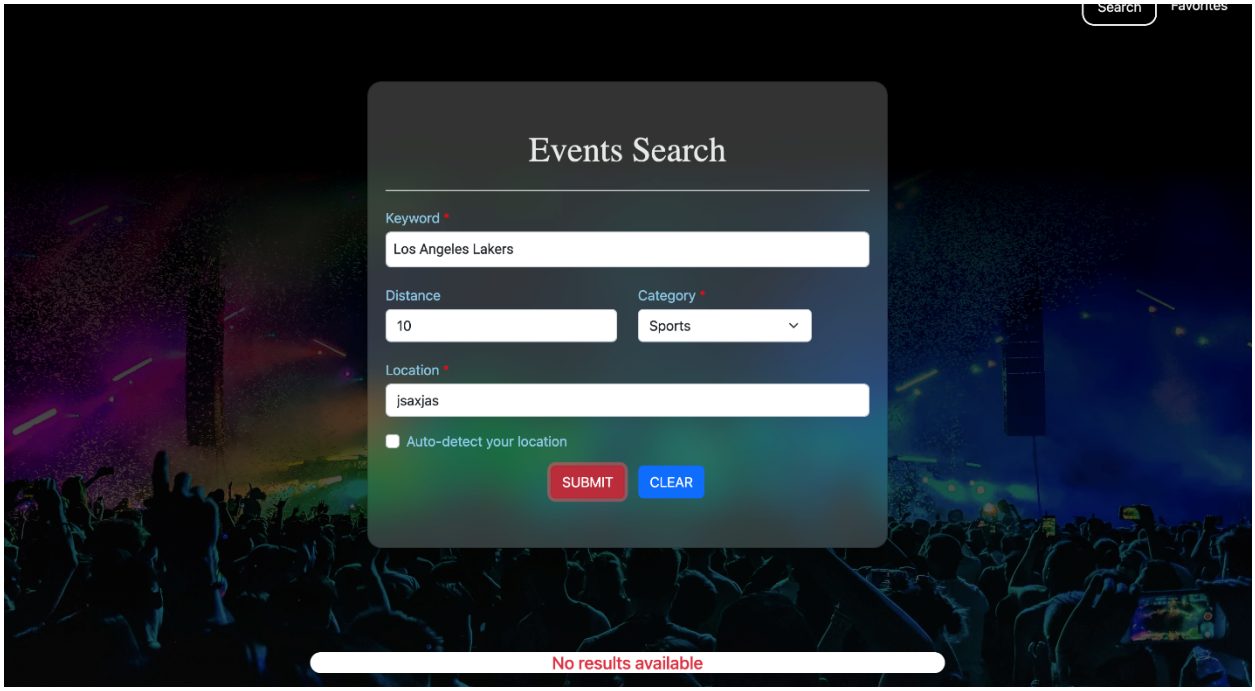


Figure 11: Example of no search result found

The API endpoint for search is:
GET https://app.ticketmaster.com/discovery/v2/events

Refer to the search documentation at:

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

API Sample:

```
https://app.ticketmaster.com/discovery/v2/events.json?apikey=
YOUR_API_KEY&keyword=University+of+Southern+California&segmentId=KZFzni
wnSyZfZ7v7nE &radius=10&unit=miles&geoPoint=9q5cs
```

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 Angular to parse the JSON object and display the results in a tabular format. A sample output is shown in **Figure 9**. The displayed table includes five columns: Date/time, Icon, Event, Genre and Venue Info. Events should be displayed by **ascending order of "Date/Time"** column.

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*" and "*localTime"* attributes that is part of "*events*" object
Icon	The value of the "*images*" attribute that is part of the "*events*" object.
Event	The value of the "*name*" attribute that is part of the "*events*" object.
Genre	The value of the "segment" attribute that is part of the "*events*" object.
Venue	The value of the "*name*" attribute that is part of the "*venue*" object inside "*events*" object.

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

5.2.3 Details Card

The **Details card** consists of Back button, Event name, Favorite button and **three tabs** as follows:

1) Events
2) Artists/Teams
3) Venue

Please refer to the Angular Material tab tutorial for implementing angular tabs. In the search results, if the user clicks on the event row, a card should be displayed with detailed results of the event, replacing the search result table (see reference video for behavior). The page should request the detailed information using the *Event Details API* and *Venue Search API*, documented at:

https://developer.ticketmaster.com/products-and-docs/apis/discovery-api/v2/#event-details-v2

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

To retrieve event details, the request needs two parameters (output should be JSON):

- ***id:*** ID of the event
- ***apikey:*** Your application's API key. This key identifies your application for purposes of quota management.

5.2.3.1 “Events” tab

The **Events details** tab shows a table containing the detailed info of the event such as Date, Artists/Team, Venue, Genres, Price Ranges, Ticket Status, a link to Buy Tickets, Facebook icon, Twitter icon and a Seat map. See **Figure 13**.

- *Date* – displays “*localDate*” and “*localTime”*
- *Artists/Team* – displays artists/team *“name”* segmented by “ | ”
- *Venue* – displays “venue name”
- *Genres* – displays genre in the order of "segment", "genre", "subGenre", "type", "subType"
- *Price Ranges* – displays value of “*min*” and *“max” prices* object, combined with “ -”
- *Ticket Status* – in the event details card is color coded. The convention used is follows:
    - On sale: Green
    - Off sale: Red
    - Canceled: Black
    - Postponed: Orange
    - Rescheduled: Orange

Figure 12 displays the desired appearance of the respective status of the tickets.

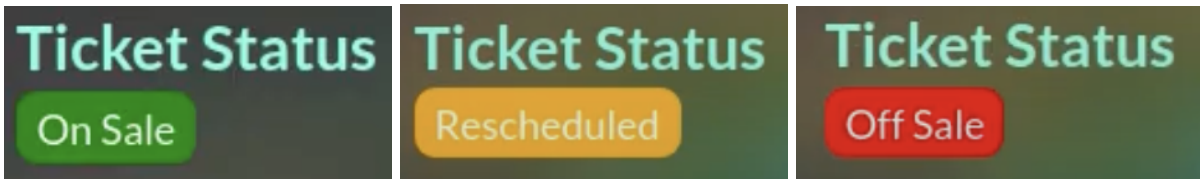


Figure 12: Color coded card for the “Ticket Status” field

- *Buy Ticket At* – Under “Buy Ticket At”, there should be a “Ticketmaster” link, a page to buy tickets online which should open in a new page.
- *Seat Map* – displays an image of the seat map
- *Facebook icon* – On clicking the icon, create a post in a new tab containing the event’s Ticketmaster link as shown in **Figure 16**
- *Twitter icon* – On clicking the icon, create a tweet in a new tab containing *Check <event_name> on Ticketmaster*, followed by event’s Ticketmaster link as shown in **Figure 17**

NOTE: If any of the above fields is not available, please don’t display that attribute in the detail card.

Figure 13: Example of Events Details card

Back button –

Clicking on the **Back button** on the top left corner of the Event details card, should navigate back to the search result table.

Favorite icon –

On clicking the **Favorite icon** on the top header row of the details card, the fields "Date", "Event", "Category" and "Venue" of that event are stored in local storage which is explained in **Section 9.9**. See **Figure 13**.

On clicking the **Favorite icon**, display a 'Event Added to Favorites!' alert as shown in **Figure 14** and change the heart icon to **a red color-filled heart icon** as shown in **Figure 15**. Once an event is marked as favorite, if we revisit the details of that event, it should display the "red" color-filled heart icon instead of a "white" heart, until the event is removed from the "favorites" page or if the **Favorite** icon is clicked again, which should display a 'Removed from Favorites!' alert and change the icon to display **white heart icon**.

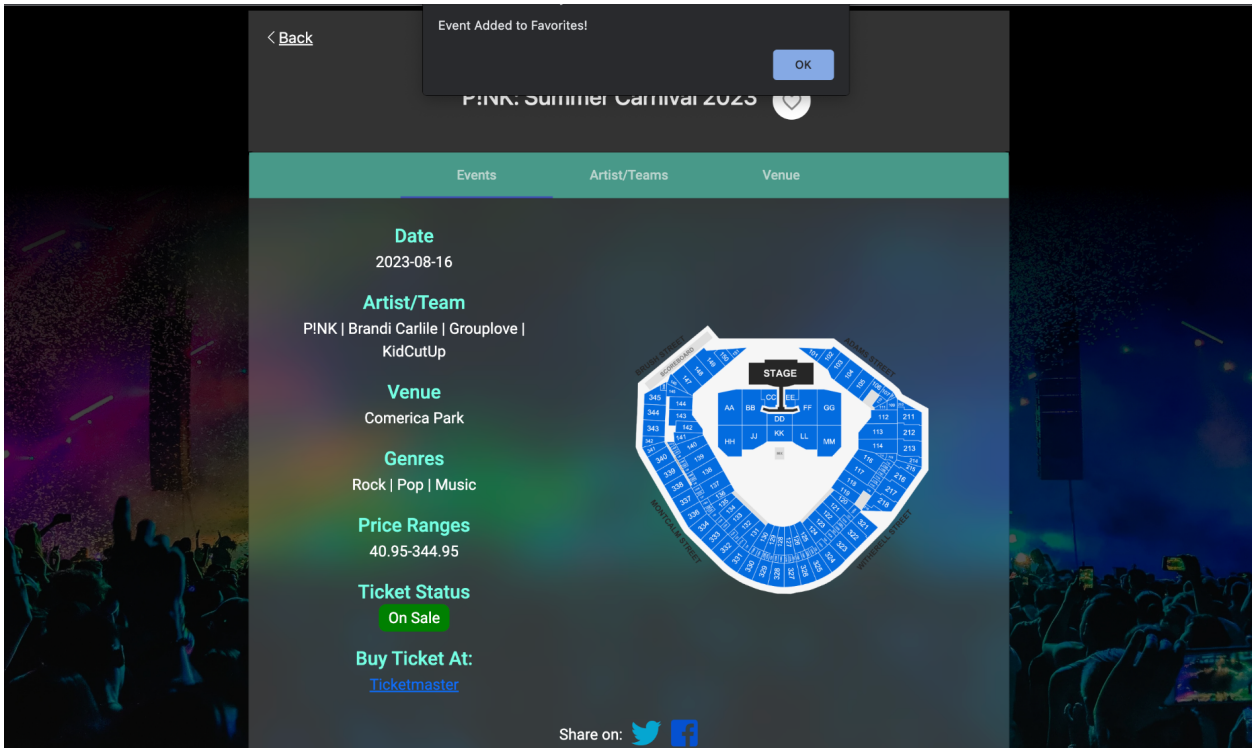


Figure 14: Example of Event added to Favorites.

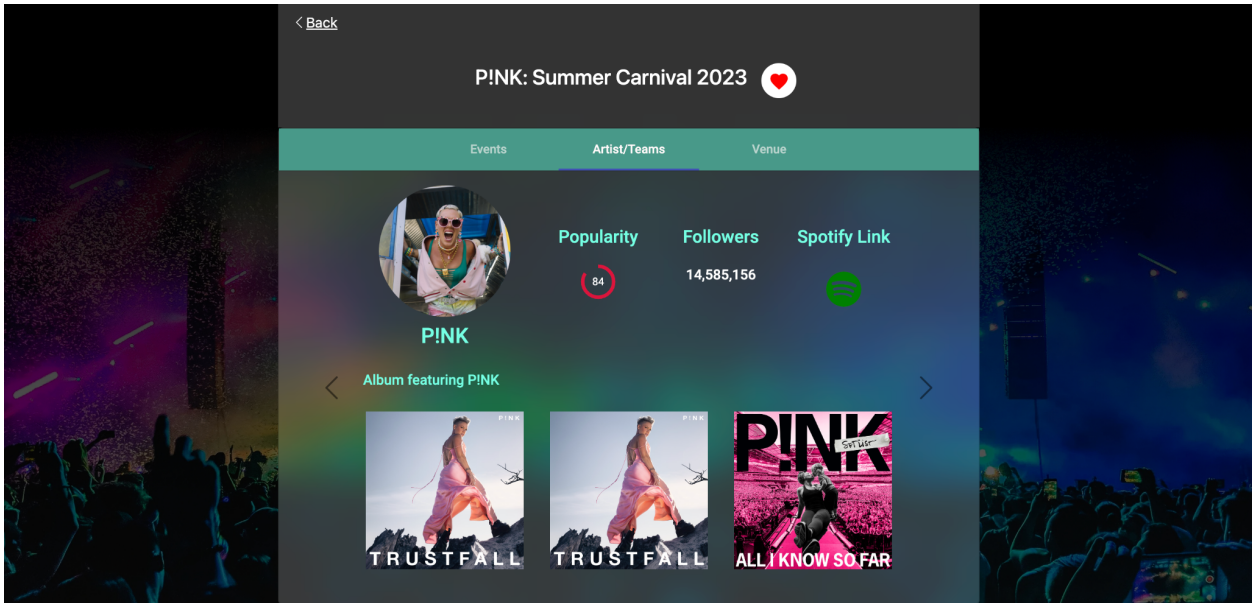


Figure 15: Example of Favorite heart icon turning red after event marked as Favorite.

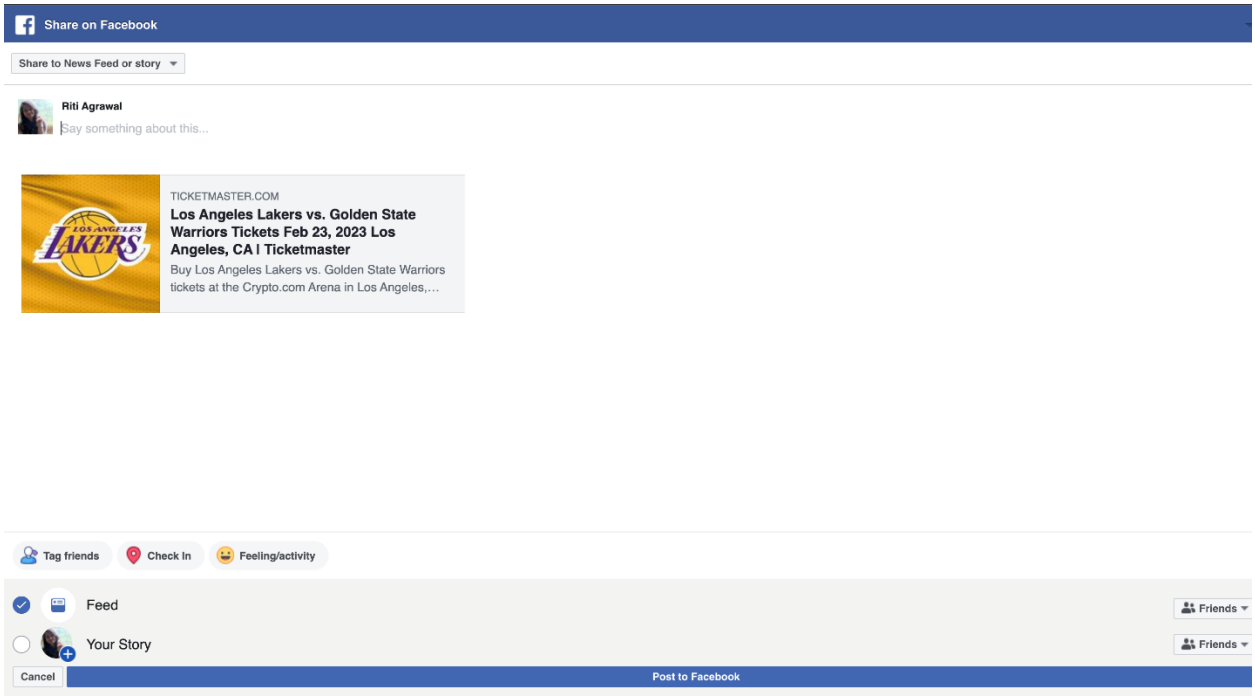


Figure 16: Example of Facebook post

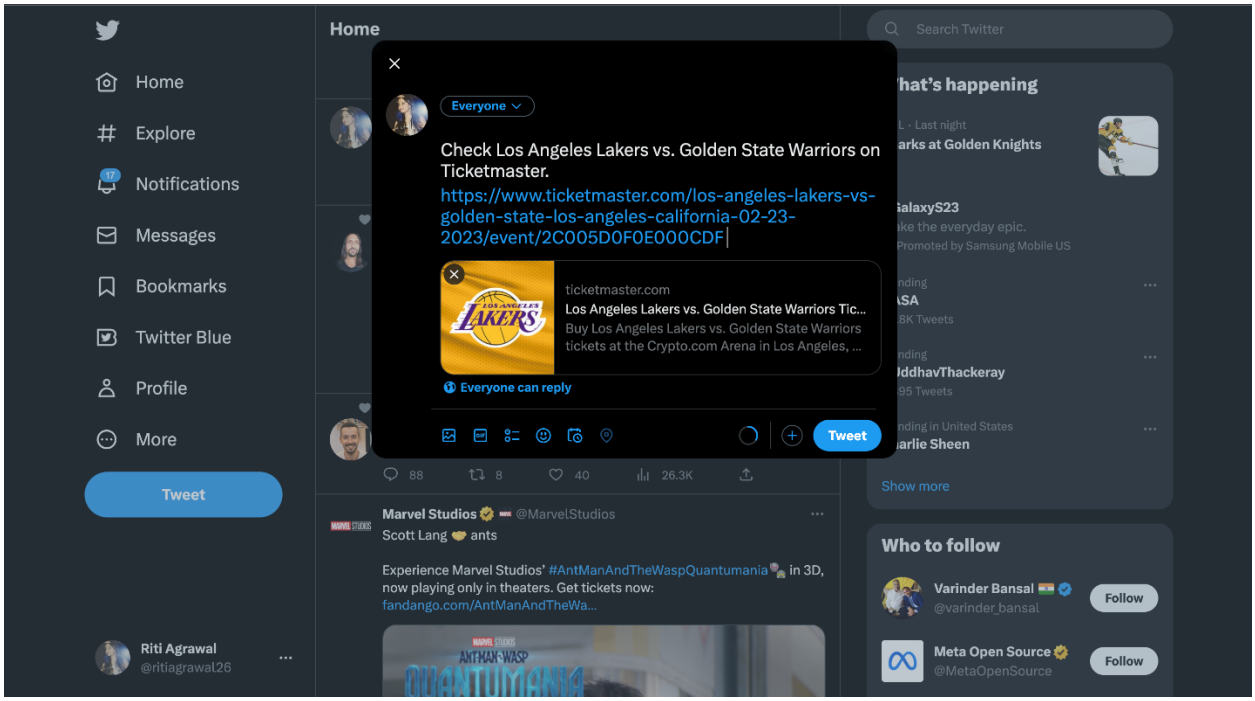


Figure 17: Example of tweet

5.2.3.2 “Artists/Team” tab

The **Artists/Team** tab will show information about the artist, like popularity, followers, Spotify link, and images of the artist/team’s Top 3 album cover.

Since the artist information will be available on the Spotify API only if it is a music related event, you are supposed to show the artist/teams tab in the info page only if the event is a Music related event. In other events this tab should display "No music related artist details to show", as shown in **Figure 21.**

The *Spotify API* is documented at:
https://developer.spotify.com/documentation/web-api/

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

We recommend you use the wrapper/client for the *Spotify Web API* available here:
https://github.com/thelinmichael/spotify-web-api-node
The “Authorization” section of this documentation explains how to use client ID and client secret to obtain the ‘Access Token’.”

For the API calls, you need to use the ‘searchArtists’ function. The Spotify API service does not use a static API key to make authorizations. It will use your client id and client secret to generate an Access Token. The Access Token can later be used to authorize an API object and gain access to the Spotify API service. So here is a workflow you can follow:

1. Call the ‘searchArtists’ function.
2. If the function returns success, it means you have set up this Access Token and the token is not expired. In this case you can return the data directly.
3. If the function returns an 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 parameter: “keyword”, please use the attraction name to search. You will get the following result:

Figure 18: An Example Spotify API Call result

Fields in the artist table	Corresponding response object fields
Name	*The value of the "name" of the item*
Followers	*The value of the "follwers.total"*
Popularity	*The value of the "popularity"*
Spotify Link	*The value of the "external_urls.spotify"*

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

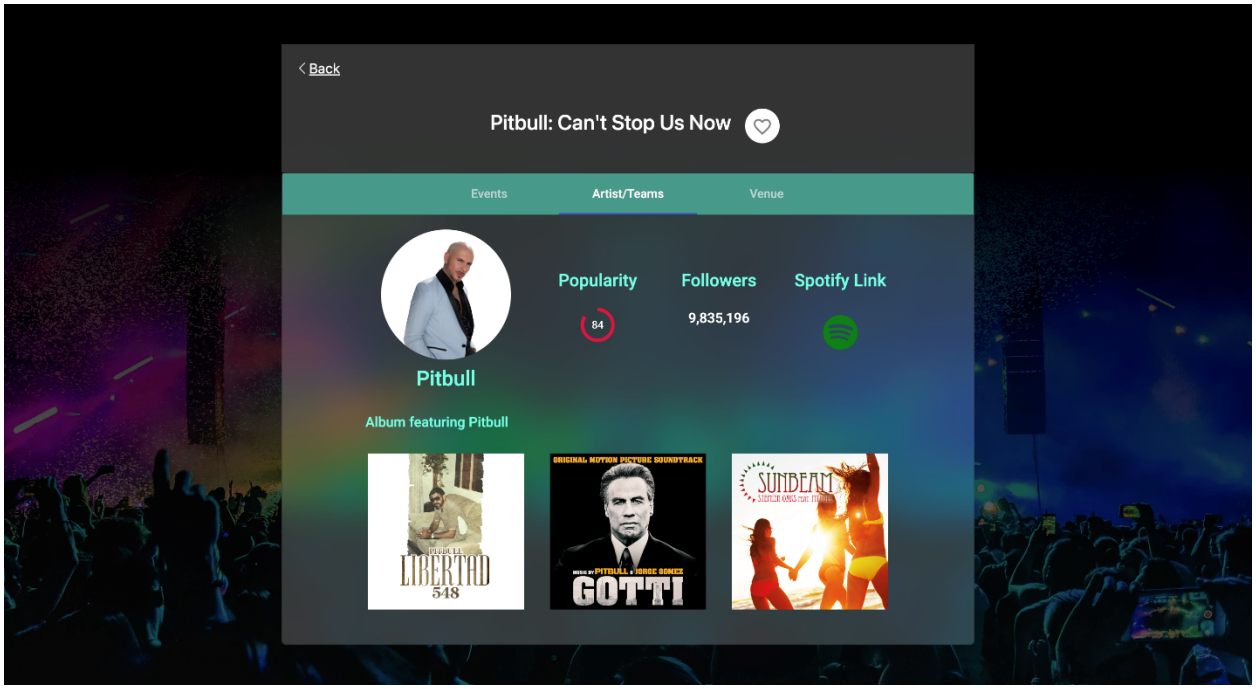


Figure 19: An Example Artists/Teams tab details

NOTE: If none of the artists of the events are musicRelated (Can be checked by using musicRelated property under artist), then the Artist/Teams tab should show "No music related artist details to show". See **Figure 20** and **Figure 21** for a case where the event is not related to music.

Figure 20: An Example of an event outside "Music" genre

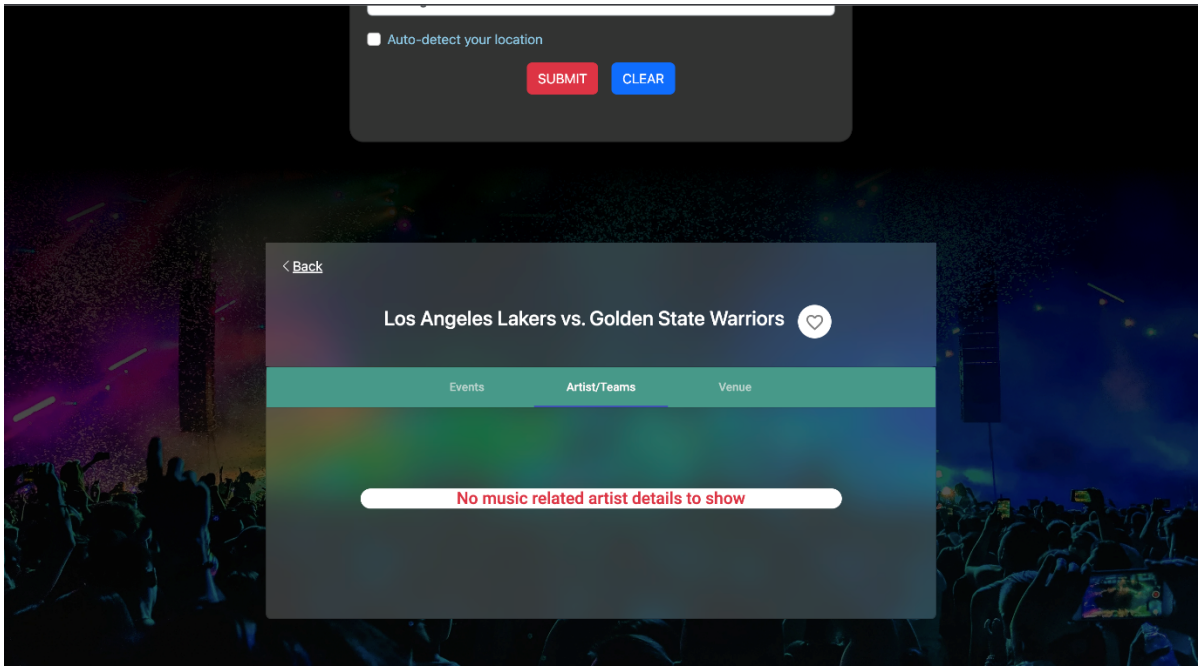


Figure 21: An Example Artists/Teams tab for an event that is not music-related

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

Here is the hint for the progress spinner :
https://material.angular.io/components/progress-spinner/overview

For multiple artists, you should use a "carousel" to view all the details table for each artist as a slide. See the video for accurate behavior. On clicking the left or right arrow, the carousel should display slides containing the artist's detail in a rotating queue fashion. See **Figure 22**

Please refer to this tutorial for the bootstrap carousel effect.

NOTE: If there is only 1 music related artist, then the carousel should not show left and right arrows (as shown in **Figure 19**).

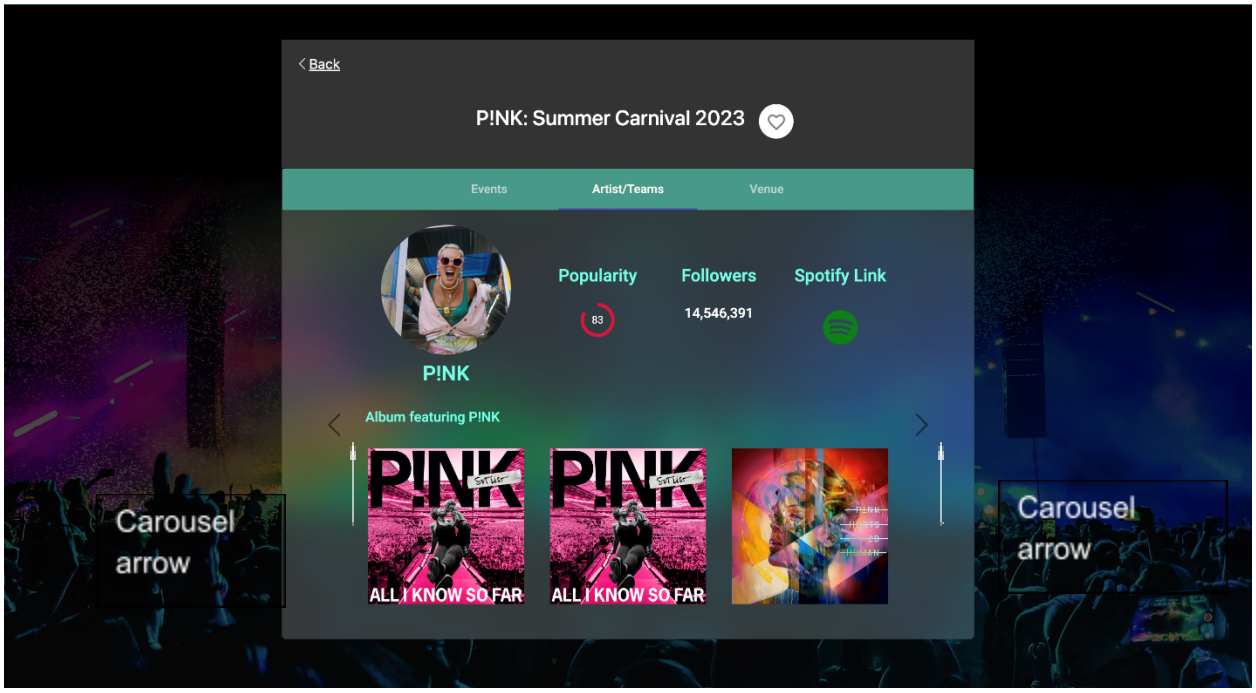


Figure 22: Example of carousel in case of multiple artists details to display

5.2.3.3 Venue tab

The **Venue tab** should display a table with Name, Address, Phone Number, Open Hours, General Rule, and Child rule. See **Figure 23**

To get the venue info, use the venue name which get from the event detail and call *Ticketmaster Venue Search API* documented below:
https://developer.ticketmaster.com/products-and-docs/apis/discovery-api/v2/#search-venues-v2

A table containing the detailed info of the event venue is displayed in this tab (see **Table 3**). The table has the following **six** (6) fields 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 "childlRule" attribute that is part of the "generalInfo" object.*

Table 3: Mapping the result from Venue Detail API into HTML table

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

NOTE: The Address field in Venue should be a concatenation of address, city, and state fields of the venueDetail object, separated by commas. In case the API returns a result, which has one or more missing fields out of the above six, skip displaying that field in the table.

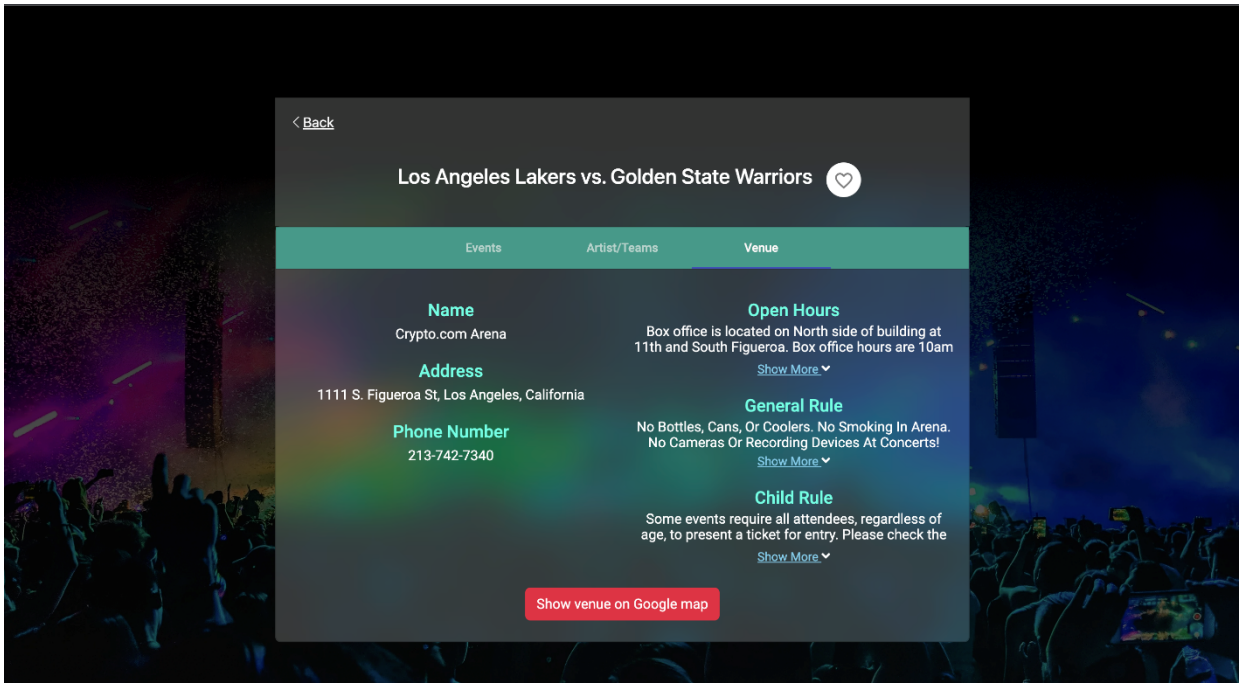


Figure 23: Example of an event having all the six fields under Venue tab

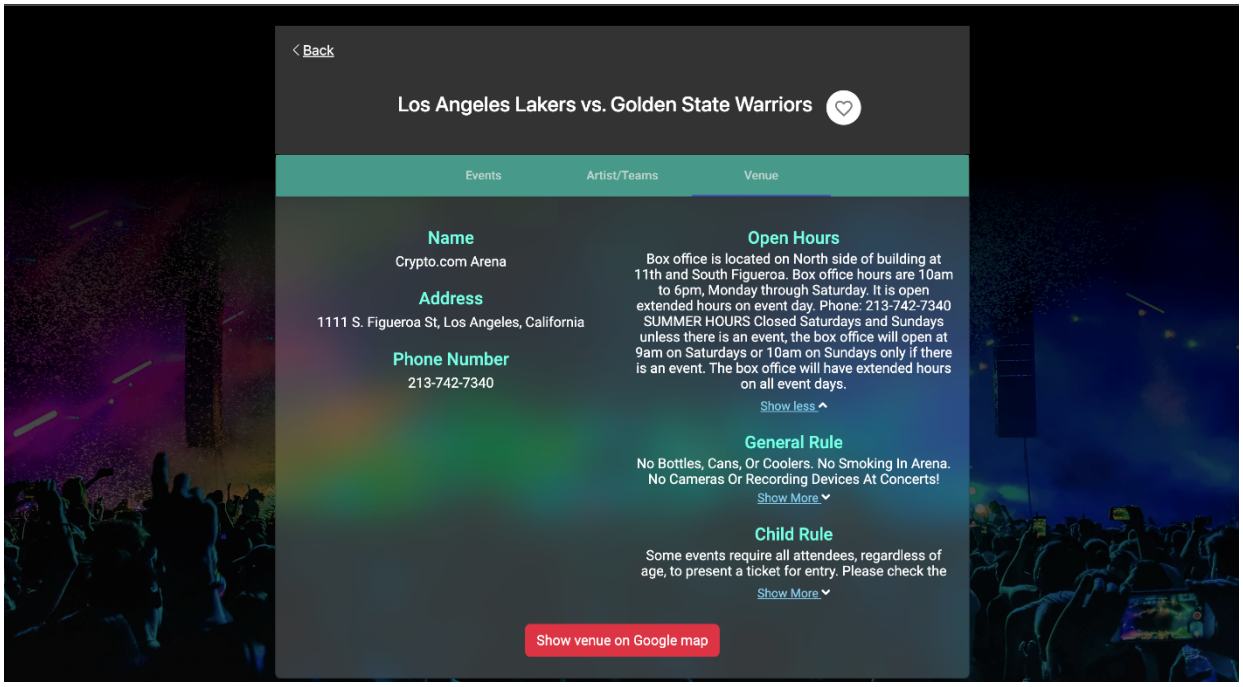


Figure 24: Example of an extended row of detail

The fields that have lengthy details to display should be wrapped under the "Show More" arrow button, as shown in **Figure 23**, which on clicking expands to display full details and the arrow button turns to "Show Less" as it can bee seen in **Figure 24.** (See the reference video for exact behavior of "Show More" and "Show Less" arrow buttons)

At the bottom of the Venue tab section there is a button labeled "**Show venue on Google map**". It is used to open a Google map with a marker of the venue location, in a modal, on top of the Venue tab. You can **only** use the '@angular/google-maps' package to load/display Google Maps. The google maps modal should have a "**Close**" button in the bottom-left to close the modal and show the underlying **Venue tab**.

Please refer to this tutorial for adding Google Maps to Angular. See **Figure 25**.

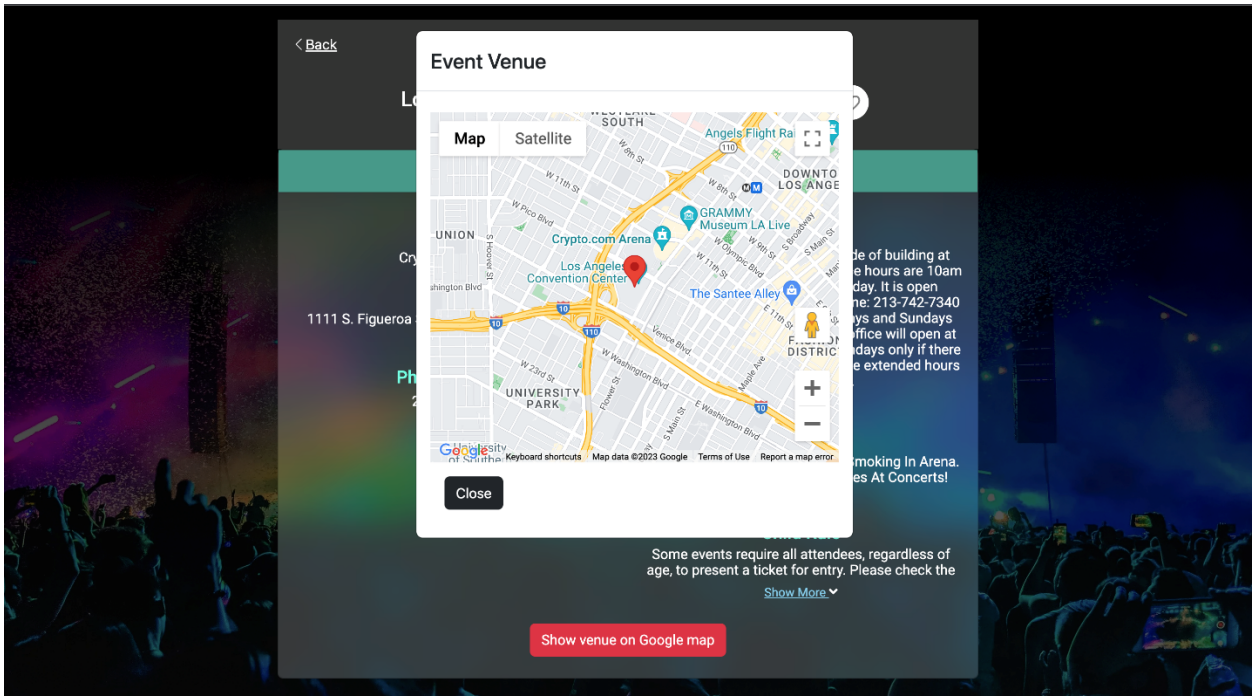


Figure 25: Example of Map location modal

5.3 Favorites Route

The Favorites route displays a list of events marked as "Favorites". It fetches all the favorite events from the HTML5 "localstorage" of the browser and displays them in a table of 5 columns. The titles of the 6 columns should be: **#**, **Date**, **Event**, **Category**, **Venue**. See **Figure 26**.

For each record, you should display a Trash Icon. On clicking the Trash icon, remove the event from Favorites and display a 'Removed from Favorites!' alert. Also turn the Favorite-marked red-heart icon on the Event Details card to **white-heart** again. See **Figure 27.**

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 favorites list will still be there. If no favorites are available, display 'No favorite events to show' message. See **Figure 28.**

Refer to the HTML Web Storage tutorial for implementing Local Storage.

Note: All the CSS should be matched as shown in Images/Video.

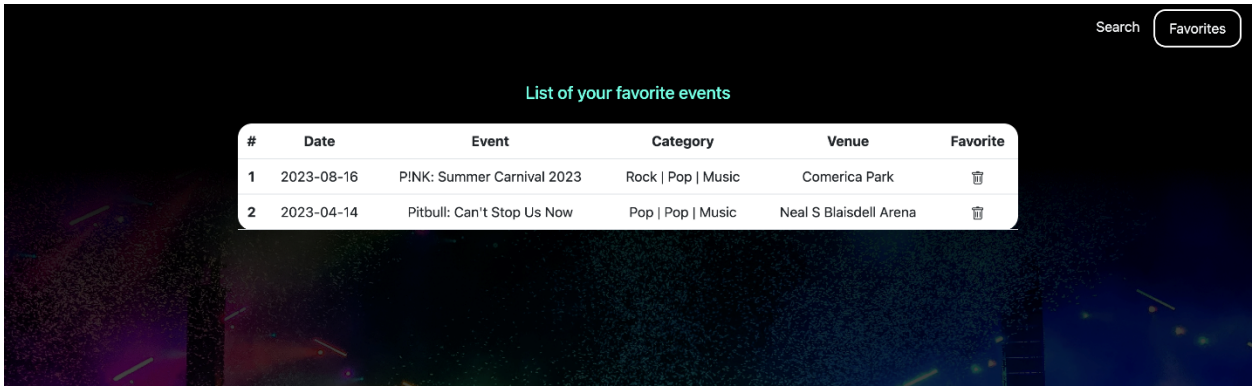


Figure 26: Example of list of Favorite events

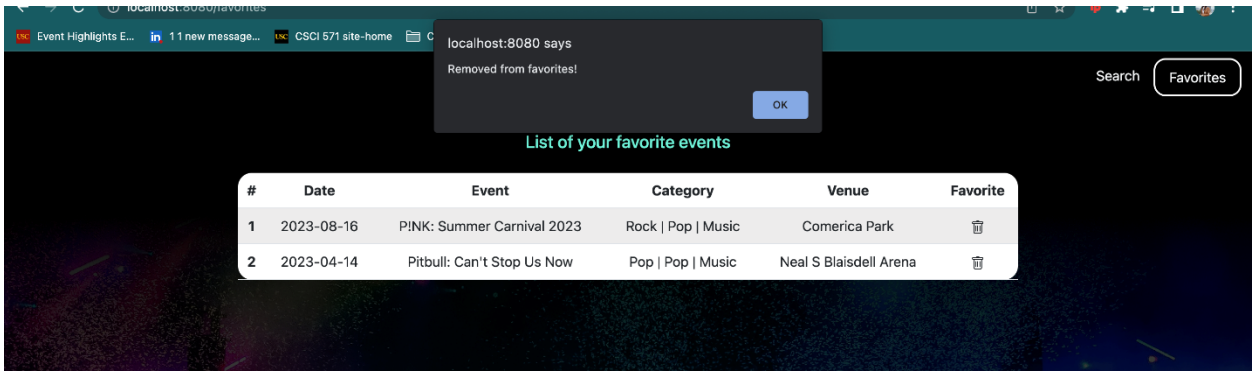


Figure 27: Example of alert message when event removed from the list of Favorites.

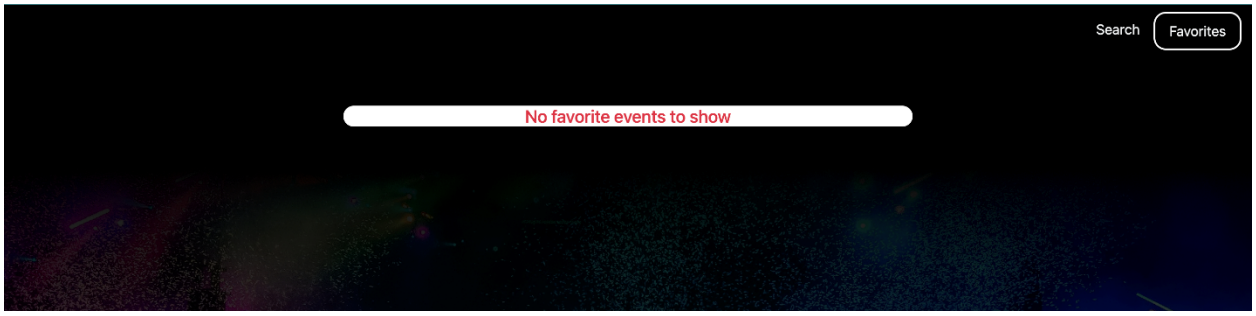
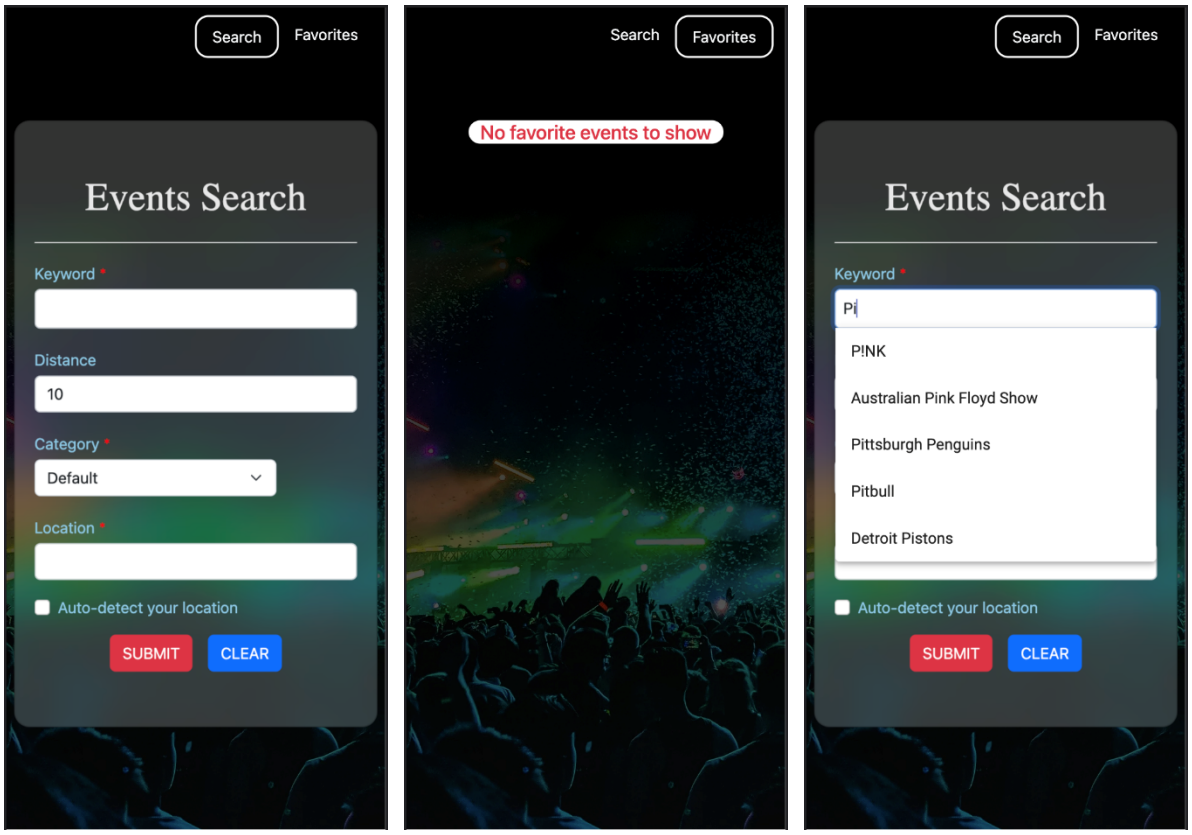
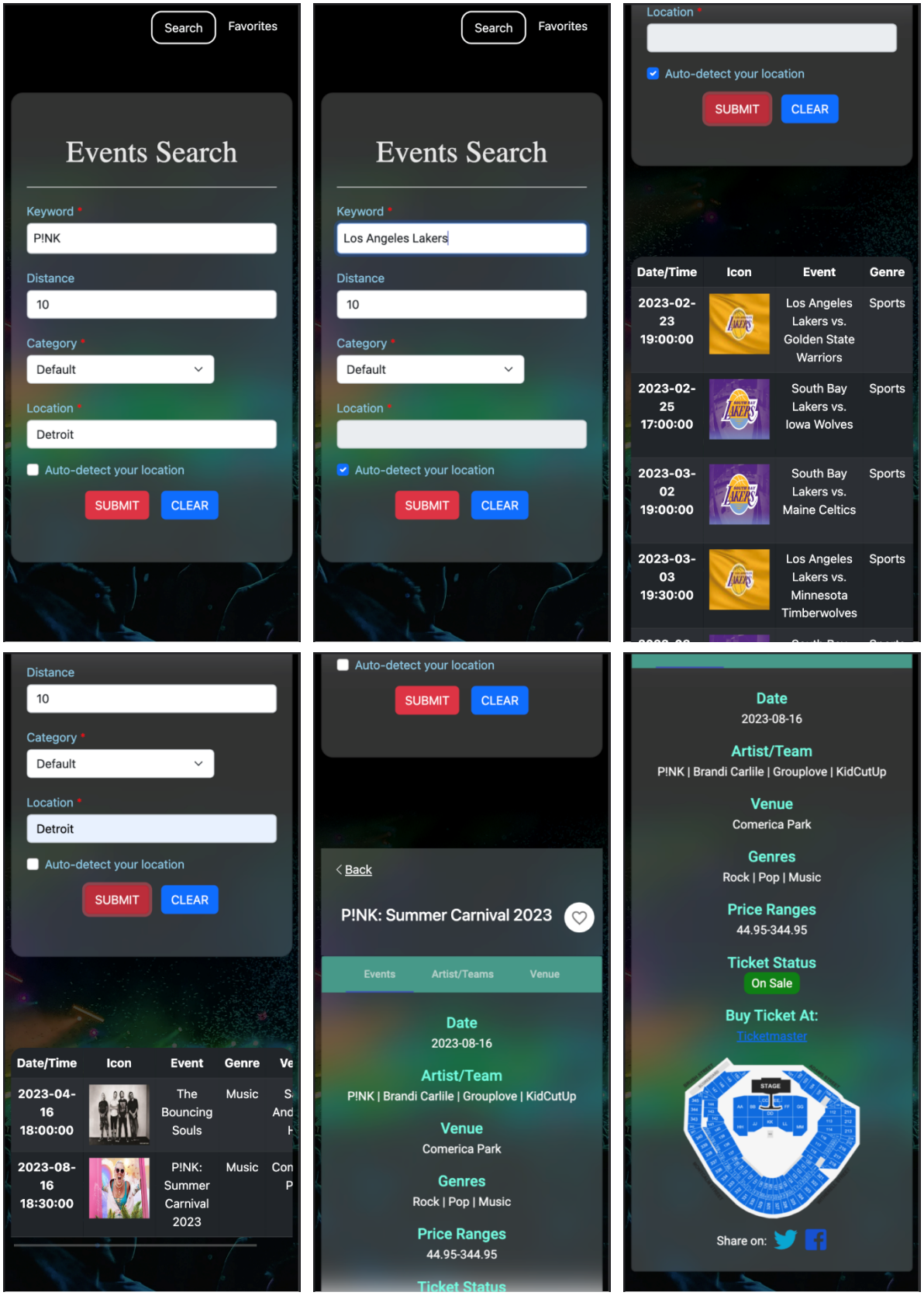


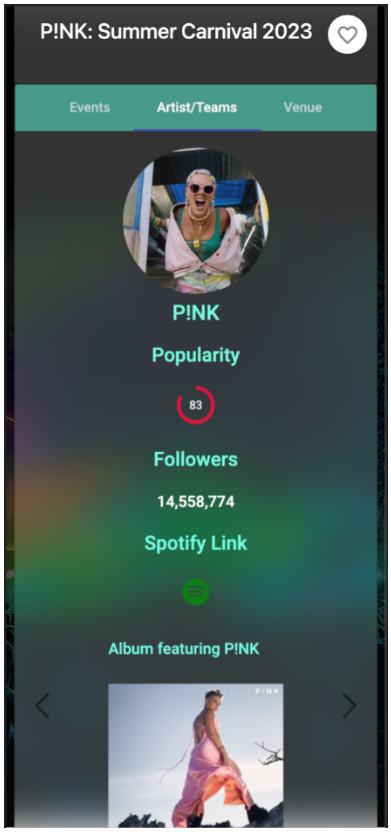
Figure 28: Example of empty list of Favorites.

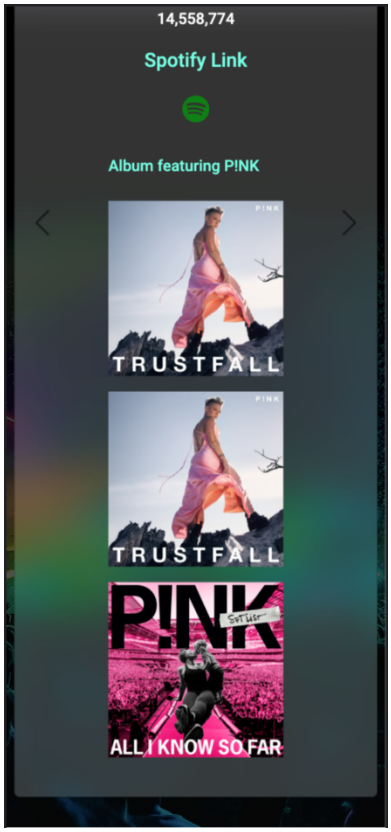
6. Responsive Design

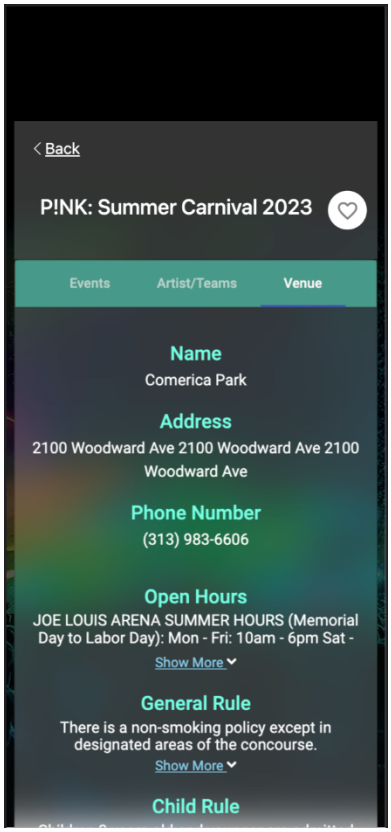
The webpage you develop must be responsive. One easy way to test responsive behavior is to use Google Chrome Responsive Design Mode in the Developer Console. Here are some snapshots for an **iPhone 12 Pro** using the Google Chrome Responsive Design Mode. All functions should work on mobile devices.

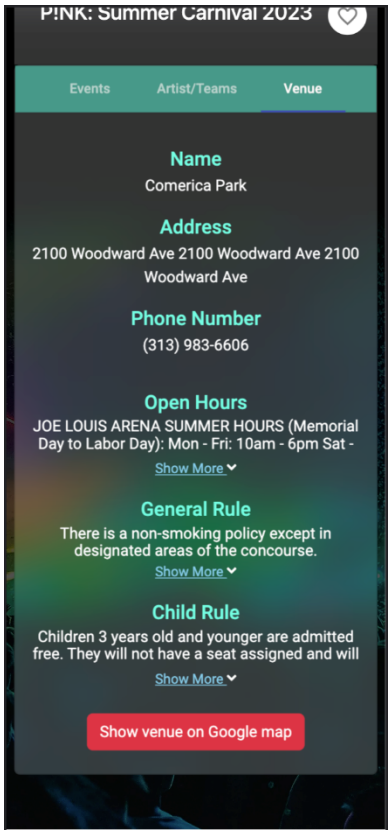


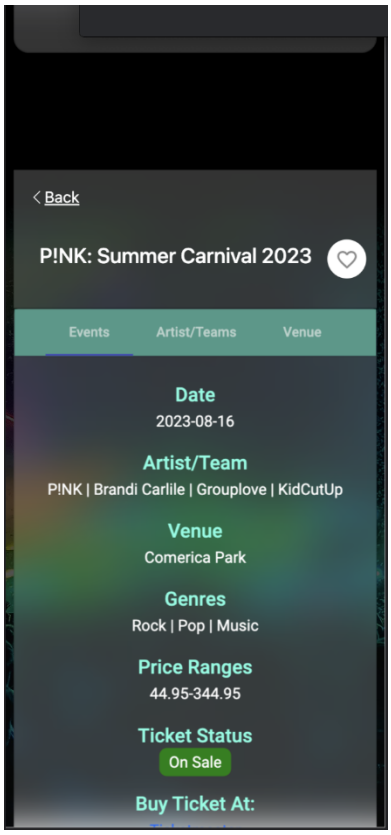

Search
Favorites
Events Search
Keyword *
P!NK
Distance
10
Category *
Default
Location *
Detroit
Auto-detect your location
SUBMIT
CLEAR
Search
Favorites
Events Search
Keyword *
Los Angeles Lakers
Distance
10
Category *
Default
Location *
Auto-detect your location
SUBMIT
CLEAR
Location *
Auto-detect your location
SUBMIT
CLEAR
Date/Time
Icon
Event
Genre
2023-02-23 19:00:00
Los Angeles Lakers vs. Golden State Warriors
Sports
2023-02-25 17:00:00
South Bay Lakers vs. Iowa Wolves
Sports
2023-03-02 19:00:00
South Bay Lakers vs. Maine Celtics
Sports
2023-03-03 19:30:00
Los Angeles Lakers vs. Minnesota Timberwolves
Sports
Distance
10
Category *
Default
Location *
Detroit
Auto-detect your location
SUBMIT
CLEAR
Date/Time
Icon
Event
Genre
2023-04-16 18:00:00
The Bouncing Souls
Music
2023-08-16 18:30:00
P!NK: Summer Carnival 2023
Music
Auto-detect your location
SUBMIT
CLEAR
Back
P!NK: Summer Carnival 2023
Events
Artist/Teams
Venue
Date
2023-08-16
Artist/Team
P!NK | Brandi Carlile | Grouplove | KidCutUp
Venue
Comerica Park
Genres
Rock | Pop | Music
Price Ranges
44.95-344.95
Ticket Status
Date
2023-08-16
Artist/Team
P!NK | Brandi Carlile | Grouplove | KidCutUp
Venue
Comerica Park
Genres
Rock | Pop | Music
Price Ranges
44.95-344.95
Ticket Status
On Sale
Buy Ticket At:
Ticketmaster
STAGE
Share on:

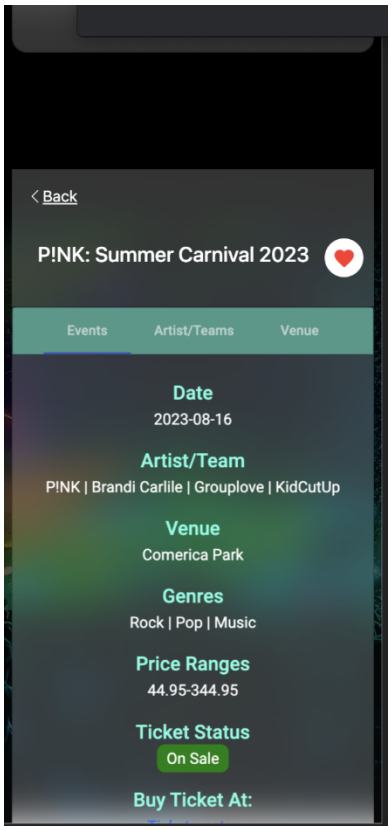

P!NK: Summer Carnival 2023
Events
Artist/Teams
Venue
P!NK
Popularity
83
Followers
14,558,774
Spotify Link
Album featuring P!NK

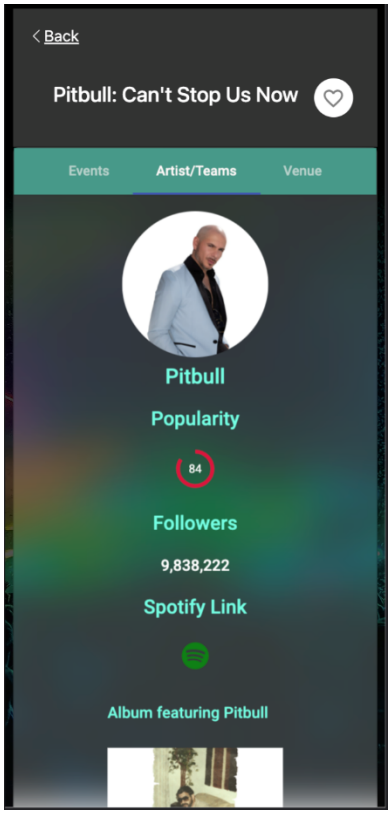

14,558,774
Spotify Link
Album featuring P!NK
TRUSTFALL
TRUSTFALL
P!NK
ALL I KNOW SO FAR

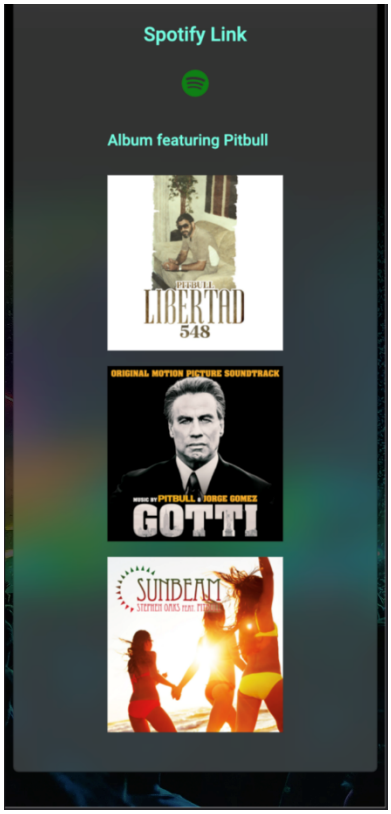

Back
P!NK: Summer Carnival 2023
Events
Artist/Teams
Venue
Name
Comerica Park
Address
2100 Woodward Ave 2100 Woodward Ave 2100 Woodward Ave
Phone Number
(313) 983-6606
Open Hours
JOE LOUIS ARENA SUMMER HOURS (Memorial Day to Labor Day): Mon - Fri: 10am - 6pm Sat -
Show More
General Rule
There is a non-smoking policy except in designated areas of the concourse.
Show More
Child Rule

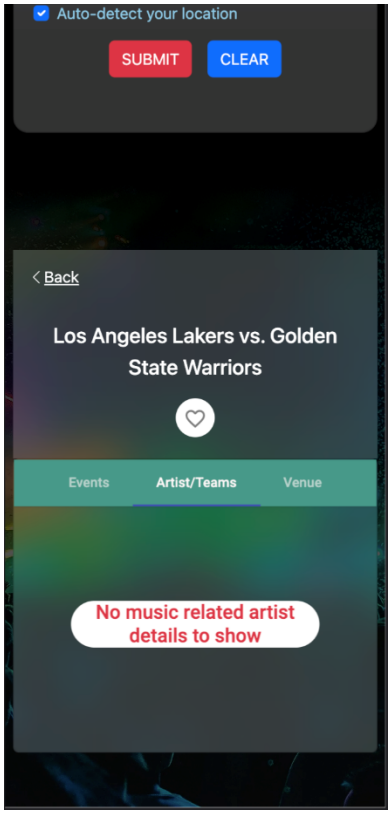

P!NK: Summer Carnival 2023
Events
Artist/Teams
Venue
Name
Comerica Park
Address
2100 Woodward Ave 2100 Woodward Ave 2100 Woodward Ave
Phone Number
(313) 983-6606
Open Hours
JOE LOUIS ARENA SUMMER HOURS (Memorial Day to Labor Day): Mon - Fri: 10am - 6pm Sat -
Show More
General Rule
There is a non-smoking policy except in designated areas of the concourse.
Show More
Child Rule
Children 3 years old and younger are admitted free. They will not have a seat assigned and will
Show More
Show venue on Google map

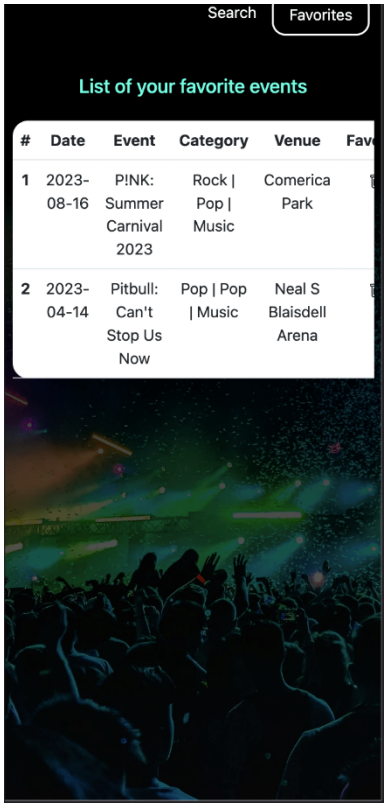

Back
P!NK: Summer Carnival 2023
Events
Artist/Teams
Venue
Date
2023-08-16
Artist/Team
P!NK | Brandi Carlile | Grouplove | KidCutUp
Venue
Comerica Park
Genres
Rock | Pop | Music
Price Ranges
44.95-344.95
Ticket Status
On Sale
Buy Ticket At:

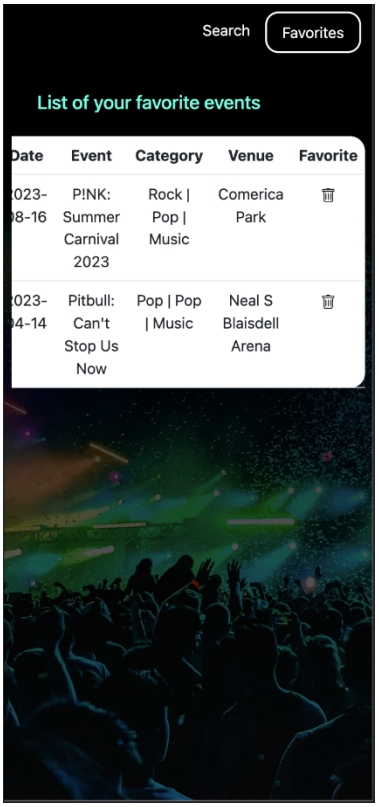

Back
P!NK: Summer Carnival 2023
Events
Artist/Teams
Venue
Date
2023-08-16
Artist/Team
P!NK | Brandi Carlile | Grouplove | KidCutUp
Venue
Comerica Park
Genres
Rock | Pop | Music
Price Ranges
44.95-344.95
Ticket Status
On Sale
Buy Ticket At:

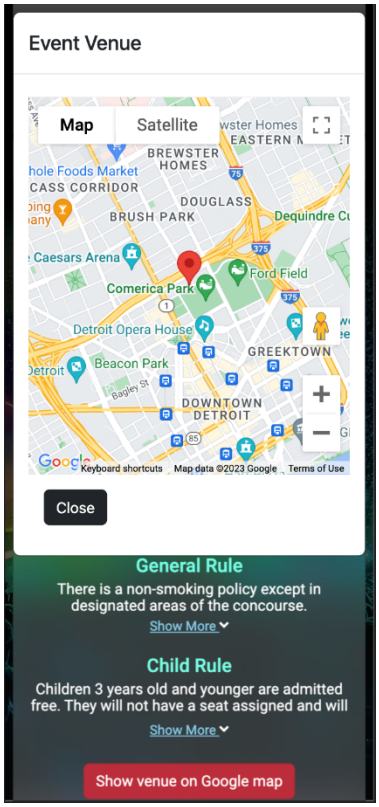

Back
Pitbull: Can't Stop Us Now
Events
Artist/Teams
Venue
Pitbull
Popularity
84
Followers
9,838,222
Spotify Link
Album featuring Pitbull

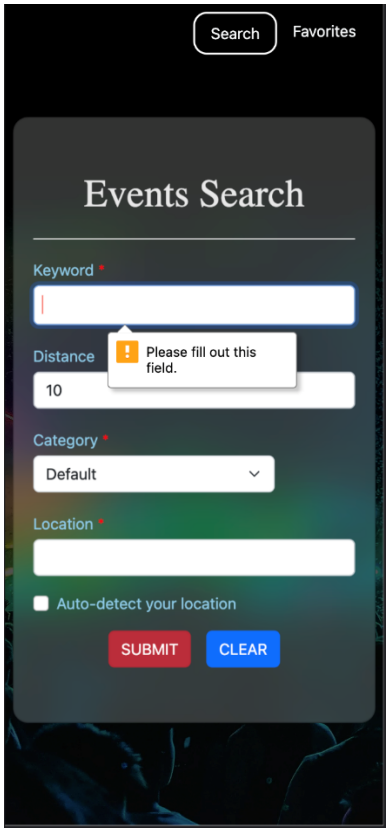

Spotify Link
Album featuring Pitbull
LIBERTAD
548
ORIGINAL MOTION PICTURE SOUNDTRACK
GOT'TI
SUNBEAM

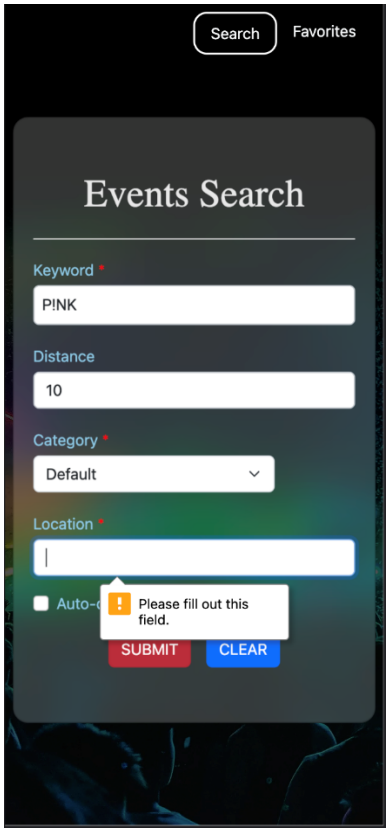

Auto-detect your location
SUBMIT
CLEAR
Back
Los Angeles Lakers vs. Golden State Warriors
Events
Artist/Teams
Venue
No music related artist details to show


Search
Favorites
List of your favorite events
Date Event Category Venue Fav
1 2023-08-16 P!NK: Summer Carnival 2023 Rock | Pop | Music Comerica Park
2 2023-04-14 Pitbull: Can't Stop Us Now Pop | Pop | Music Neal S Blaisdell Arena


Search
Favorites
List of your favorite events
Date Event Category Venue Favorite
2023-08-16 P!NK: Summer Carnival 2023 Rock | Pop | Music Comerica Park
2023-04-14 Pitbull: Can't Stop Us Now Pop | Pop | Music Neal S Blaisdell Arena


Event Venue
Map
Satellite
BREWSTER HOMES
DOUGLASS
BRUSH PARK
Caesars Arena
Comerica Park
Ford Field
Detroit Opera House
Beacon Park
GREEKTOWN
DOWNTOWN DETROIT
Close
General Rule
There is a non-smoking policy except in designated areas of the concourse.
Show More
Child Rule
Children 3 years old and younger are admitted free. They will not have a seat assigned and will
Show More
Show venue on Google map


Search
Favorites
Events Search
Keyword *
Please fill out this field.
Distance
10
Category *
Default
Location *
Auto-detect your location
SUBMIT
CLEAR


Search
Favorites
Events Search
Keyword *
P!NK
Distance
10
Category *
Default
Location *
Please fill out this field.
SUBMIT
CLEAR

7. API Documentation

7.1 Spotify API

To use the *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 to the documentation doc and also the Spotify NodeJS libraries, documented at:

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

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

8. Libraries

- **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.

9. Implementation Hints

9.1 Images

The images needed for this homework are available on D2L.

9.2 Get started with the Bootstrap Library

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

https://getbootstrap.com/docs/4.6/getting-started/introduction/.

You need to import the necessary CSS file and JS file provided by Bootstrap. We recommend using Bootstrap, 4.0, 4.6 or 5.0.

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

Change the version # (4.0 above) to get other versions of components. Bootstrap 5.0 uses a different architecture, so you will have to find compatible components.

9.4 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

9.5 Material Icons

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

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

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

9.6 Google App Engine/Amazon Web Series/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**. When your grader clicks on this additional link, the response should return a JSON object with appropriate data.

9.7 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. In most cases, this selection is hidden.

9.8 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 9.6 above). Please refer to the grading guidelines for details.

9.9 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

10. Assignment Submission

In your course Table of Assignments page (on GitHub Pages), update the Homework 8 link to refer to your deployed website. Also, provide an additional link to one of your cloud query entry points, below the homepage link. Your project must be hosted on GCP, AWS or Azure. Graders will verify that these links are indeed pointing to one of the listed cloud platforms. Refer to Homework 7 to deploy node.js applications on GCP/AWS/AZURE.

Also, submit your source code files to DEN D2L as a ZIP archive. Include your frontend and backend source code, plus any additional files needed to build your app (e.g., yaml file). The timestamp of your last submission will be used to verify if you used any grace days. Make sure you don't upload the `node_modules` in the zip file and also make sure 'package.json' is added in the zip file.

11. Notes

- You can use **React** in lieu of **Angular**, but the look of the app must be the same, or penalties will be assessed. Also note that there will be no support on Piazza for React.
- You must use Bootstrap for Assignment 8. If Bootstrap is not used, there will be a point penalty assessed.
- The appearance of the webpage should be like the reference videos/screenshots as much as possible.

12. Additional References

- Conditional rendering in Angular: https://angular.io/api/common/NgIf
- Types of form handling in Angular: https://blog.angular-university.io/introduction-to-angular-2-forms-template-driven-vs-model-driven/Use Authentication Headers: put the API key inside of the request header as "**Authorization: Bearer <YOUR API KEY>**" and make requests against the API.
- https://chanchad.medium.com/how-to-setup-angular-10-and-express-js-application-for-developing-it-simultaneously-ca42e009c1ea
- https://youtube.com/playlist?list=PLHPSxQDpPvUmhqzw7K6R0zBXW6YJkV5zz

****IMPORTANT**:**

- All discussions and explanations in Piazza related to this homework are part of the homework description and grading guidelines. So please review all Piazza threads, before finishing the assignment. If there is a "clarification" on Piazza that conflicts with this description and/or the grading guidelines, **Piazza always rules**. In most cases, the

clarification is related to an error in the description, or missing information from the description, but **no additional functionality**.

- You can use jQuery for Homework 8, but its use is not required.
- You can use FasS like *Google Cloud Functions*, *Amazon Lambda* and *Azure Functions*, in lieu of building a monolithic application.
- You **should not call any of the Ticketmaster APIs directly from Javascript**, bypassing the NodeJS proxy. Implementing any one of them in JavaScript instead of Node will result in a **4-point penalty.** Other APIs can be called from JavaScript.
- You may call the Google Maps Geocoding API directly from JavaScript.
- **APPEARANCE OF CARD VIEW and TABLE should be as similar as possible to the reference video.** See Reference Video -
  Web Version: https://youtu.be/8FV4jEmK1T0
  Mobile version: https://youtu.be/A0nkmA4Tnp8