

Homework 8: Ajax, JSON and Responsive Design

Travel and Entertainment Search
(Bootstrap/Angular/AJAX/JSON/jQuery /Cloud Exercise)

1. Objectives

- Get familiar with the AJAX and JSON technologies
- Use a combination of HTML5, Bootstrap, JQuery, Angular, and PHP/Node.js
- Get hands-on experience of Google Cloud App Engine and Amazon Web Services
- Get hands-on experience of using Bootstrap to enhance the user experience
- Learn to use popular APIs such as Google Places APIs and Yelp 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://cs-server.usc.edu:45678/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://cs-server.usc.edu:45678/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). See the class slides at:

<http://cs-server.usc.edu:45678/slides/Responsive.pdf>

Although Bootstrap 4 is recommended, you can use any version you want.

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 PHP visit this page:

<https://cloud.google.com/appengine/docs/php/>

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

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

2.5 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.

For this homework, either AngularJS, Angular 2, Angular 4 or Angular 5 can be used.

To learn more about AngularJS visit this page:

<https://angularjs.org/>

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

<https://angular.io/>

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.

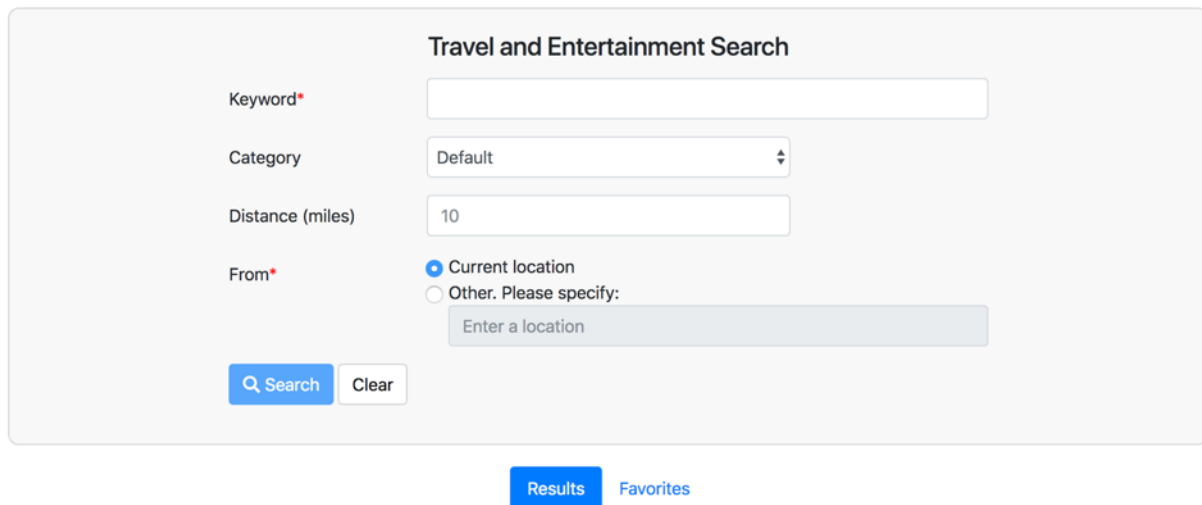
To learn more about Express.js, visit <http://expressjs.com/>

In this document when you see “PHP/Node.js” it means that you can either use a PHP script or a Node.js script; when you see “jQuery/Angular” it means that you can either use a jQuery or Angular function; and when you see GAE/AWS it means that you can either use Google App Engine or Amazon Web Services.

3. High Level Description

In this exercise you will create a webpage that allows users to search for places using the Google Places API and display the results on the same page below the form. Once the user clicks on a button to search for place details, your webpage should display several tabs which contain an info table, photos of the place, map and route search form and reviews respectively. Your webpage should also support adding places to and removing places from favorites list and posting place 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 "Travel and Entertainment Search". It contains the following fields and controls:

- Keyword***: A text input field.
- Category**: A dropdown menu with "Default" selected.
- Distance (miles)**: A text input field with "10" entered.
- From***: Two radio buttons. The first is "Current location" (selected). The second is "Other. Please specify:" followed by a text input field containing the placeholder "Enter a location".
- Search Buttons**: A blue button with a magnifying glass icon and the text "Search", and a white button with the text "Clear".
- Navigation Buttons**: Two blue buttons at the bottom, "Results" and "Favorites".

Figure 1

You can choose to use either PHP or Node.js for the server-side script. If you choose to use Node.js, you will be able to get **2 extra credits** as long as you set up your AWS/GCP **properly**. This will be explained in section 7.5.

4. Implementation

4.1 Search Form

4.1.1 Design

You must replicate the search form displayed in Figure 1 using a **Bootstrap form**. The form fields are basically the same as in HW#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 4.1.3 for details of validation.
2. **Category:** The default value of “Category” is “Default”, which covers all of the “types” provided by the *Google Places API*. The other options are shown in Figure 2.
3. **Distance (miles):** This is the radius of the area within which the search is performed. The center of the area is specified in the “From” field. 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 4.1.3 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. This input field should support autocomplete which is explained in section 4.1.2. Please note that the user does not necessarily chooses a place suggested by the autocomplete.

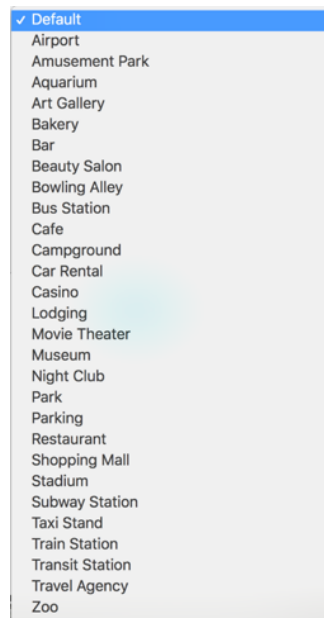


Figure 2

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 4.1.3 for details of validation. Please refer to section 4.1.4 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.

4.1.2 Autocomplete

Autocomplete is implemented by using the autocomplete service provided by Google. Please go to this page to learn more about adding autocomplete to your webpage:

https://developers.google.com/maps/documentation/javascript/places-autocomplete#add_autocomplete

You should learn from the examples and use the client-side Google Maps JavaScript library provided by Google to implement the autocomplete.

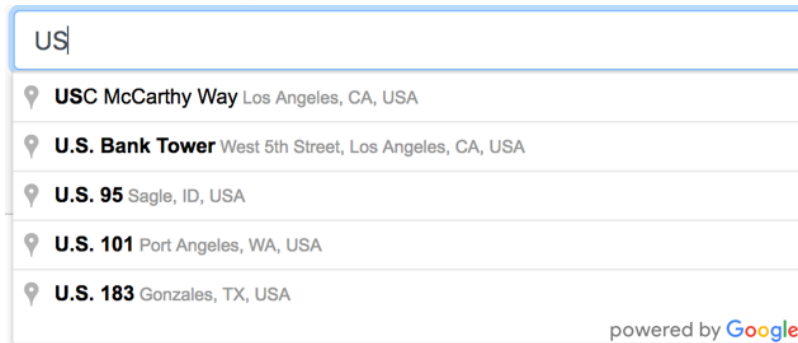


Figure 3

4.1.3 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 4. 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.

A screenshot of a web form titled "Travel and Entertainment Search". The form has several fields: "Keyword*" with a red border and a red error message "Please enter a keyword." below it; "Category" with a dropdown menu showing "Default"; "Distance (miles)" with a text input containing "10"; and "From*" with two radio buttons. The first radio button is "Current location" and is unselected. The second radio button is "Other. Please specify:" and is selected. Below the second radio button is a text input with the placeholder "Enter a location", which has a red border and a red error message "Please enter a location." below it. At the bottom left of the form are two buttons: "Search" (blue) and "Clear" (white).

Figure 4

4.1.4 Obtaining User Location

As in HW#6, you should obtain the current user location using one of the geolocation APIs. An example call to *ip-api.com* looks like:

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

The response is a JSON object shown in Figure 5.

```
as: "AS20001 Time Warner Cable Internet LLC"
city: "Los Angeles"
country: "United States"
countryCode: "US"
isp: "Time Warner Cable"
lat: 34.0266
lon: -118.2831
org: "Time Warner Cable"
query: "104.32.172.65"
region: "CA"
regionName: "California"
status: "success"
timezone: "America/Los_Angeles"
zip: "90007"
```

Figure 5

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

4.1.5 Search Execution

Once the validation is successful and the user clicks on “Search” button, your application should make an AJAX call to the PHP/Node.js script hosted on GAE/AWS. The PHP/Node.js script on GAE/AWS will then make a request to Google Places API to get the places information. This will be explained in the next section.

4.2 Results Tab

4.2.1 Results Table

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

The *Google Places API* “Nearby Search” Service is documented here:

<https://developers.google.com/places/web-service/search#PlaceSearchRequests>

If your application does not have the latitude and longitude of the place the user specifies, the PHP/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 HW#6 documents.

The PHP/Node.js script should pass the JSON object returned by the Nearby 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), Category, Name, Address, Favorite and Details.

Results Favorites

Details >

#	Category	Name	Address	Favorite	Details
1		Mountain Mike's Pizza	22942 Ridge Rte Dr, Lake Forest		
2		Pizza Hut	34119 Pacific Coast Hwy, Dana Point		
3		Domino's Pizza	27131 Aliso Creek Rd, Aliso Viejo		
4		Domino's Pizza	30242 Crown Valley Pkwy, Ste B1B, Laguna Niguel		
5		Pizza Hut	25481 Alicia Pkwy, Laguna Hills		
6		Domino's Pizza	1100 S Coast Hwy Ste 205, Laguna Beach		
7		Pizza Hut	30065 Alicia Pkwy C, Laguna Niguel		
8		Domino's Pizza	1501 Corporate Dr, Ladera Ranch		
9		Pizza Hut	27642 Antonio Pkwy, Ladera Ranch		
10		Domino's Pizza	32211 Camino Capistrano Ste E102, San Juan Capistrano		
11		Pizza Hut	32095 Camino Capistrano, San Juan Capistrano		
12		Blaze Pizza	4255 Campus Dr A120, Irvine		
13		Blaze Pizza	1004 The Shops Blvd, Mission Viejo		
14		Pizza Hut Express	25535 Los Alisos Blvd, Mission Viejo		
15		Pizza Hut Express	26792 Portola Pkwy, Foothill Ranch		
16		Pizza Hut Express	990 Avenida Vista Hermosa, San Clemente		
17		Round Table Pizza	27932 La Paz Rd J, Laguna Niguel		
18		Pizza Lounge	29971 Alicia Parkway, Laguna Niguel		
19		Oggi's Sports Brewhouse Pizza	24042 Alicia Pkwy, Mission Viejo		
20		Ballpark Pizza Team	28813 Los Alisos Blvd, Mission Viejo		

Next

Figure 6 An Example of a Valid Search result

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

HTML Table Column	API service response
Category	The value of the “ <i>icon</i> ” attribute that is part of the “ <i>results</i> ” object.
Name	The value of the “ <i>name</i> ” attribute that is part of the “ <i>results</i> ” object.
Address	The value of the “ <i>vicinity</i> ” attribute that is part of the “ <i>results</i> ” object.

Table 1: Mapping the result from Graph API into HTML table

The “#” column starts from 1. The “Favorite” column contains a button that can add the place to/remove the place from favorites list. If a place is on the list, the star is full. Otherwise, the star is empty. The “Details” column will trigger the detail search for the corresponding place.

4.2.2 Pagination

The results table should display up to 20 places. If there are more places, in the JSON object returned from the “Nearby Search”, there will be a “next_page_token” field, which should be used to get next page results. This is shown in Figure 7. You should read this page to learn how to use this token to get additional results:

<https://developers.google.com/places/web-service/search#PlaceSearchPaging>

Whenever there are additional results, your webpage should have a “Next” pagination button. When there is a previous page, your webpage should have a “Previous” pagination button. An example is shown in Figure 8.




























```
{
  "html_attributions": [],
  "next_page_token": "CqQCHAEAAIa_jJ3YqPvpezNT6M15S3yD3LyXU2Kt9C_am7TbeblHh5gb4IVEqQwtCHKMSTUxRMq1E14cevNYa1HADjafgQvga6djkY896jLRKR1sS5zxc2qYR-gWdNI87NMptnu6weVGzEw1VfsGzYi8ySUpTQMLEdp1CWj0ThrP7tyLRNbAHQsxfKtovT0AtDUC7WSK3pubs7sH98bw8acKc6AntXA5pxGN3G64WX9U5IaieOI7TiQpiLBvfrmrMnSPIMYFqH-apF40t25S5ie2CsrAuxEaoZKGuH0ZEu7CfzUW3020xZ2XmX2TBhG4jeYTxkRb-6EZs-w4TG3VYEi3eyqbNTlmyiYFpF_aqSancPLPhbZPN0MbJD2Ar5aJI18R105iRIQp-VQ0atuHCwczQk9p2dT2RoIdgnSSTru0ZVCeYiFZVeeiXg0VXc",
  "results": [{}],
  "status": "OK",
  "startLocation": {
    "lat": "34.0272467",
    "lng": "-118.2906746"
  }
}
```

Figure 7 Next page token

Results

Favorites

Details >

#	Category	Name	Address	Favorite	Details
1		Blaze Pizza	1091 Newport Center Dr, Newport Beach		
2		Blaze Pizza	225 W Avenida Vista Hermosa Suite A, San Clemente		
3		Blaze Pizza	27231 La Paz Rd, Laguna Niguel		
4		Domino's Pizza	2272 Michelson Dr, Irvine		
5		zpizza	2549 Eastbluff Dr A, Newport Beach		
6		Fresh Brothers	17655 Harvard Ave d, Irvine		
7		Chuck E. Cheese's	26562 Towne Centre Dr, Foothill Ranch		
8		Chuck E. Cheese's	26538-H Moulton Pkwy, Laguna Hills		
9		Johnny's Real New York Pizza	2756 East Coast Hwy, Corona Del Mar		
10		Pieology Pizzeria R&D (Aliso Viejo)	26661 Aliso Creek Rd, Aliso Viejo		
11		New York's Upper Crust Pizza	5613 Alton Pkwy #212, Irvine		
12		Johnny's Real New York Pizza	1320 Bison Ave, Newport Beach		
13		Round Table Pizza	32525 Street of the Golden Lantern, Dana Point		
14		Domino's Pizza	6622 Irvine Center Dr, Irvine		
15		Round Table Pizza	25290 Marguerite Pkwy D, Mission Viejo		
16		Round Table Pizza	612 Camino De Los Mares, San Clemente		
17		Ameci Pizza & Pasta	18068 Culver Dr, Irvine		
18		Pizza Hut	415 E Avenida Pico K, San Clemente		
19		Gina's Pizza	4533 Campus Dr, Irvine		
20		Domino's Pizza	1502 N El Camino Real, San Clemente		

Previous

Next




Figure 8 Pagination buttons

4.2.3 Details Button and Highlighting

The “Details>” button above the results table is initially disabled. It will be enabled once a place details search is triggered. If this button is enabled and clicked, the page will go to details tabs. After a place details search is performed, the corresponding place row in the results table should be highlighted to indicate the place whose details are displayed in the details tabs (Figure 9).

Results Favorites

Details >

#	Category	Name	Address	Favorite	Details
1		Blaze Pizza	1091 Newport Center Dr, Newport Beach		
2		Blaze Pizza	225 W Avenida Vista Hermosa Suite A, San Clemente		
3		Blaze Pizza	27231 La Paz Rd, Laguna Niguel		
4		Domino's Pizza	2272 Michelson Dr, Irvine		
5		zpizza	2549 Eastbluff Dr A, Newport Beach		
6		Fresh Brothers	17655 Harvard Ave d, Irvine		
7		Chuck E. Cheese's	26562 Towne Centre Dr, Foothill Ranch		
8		Chuck E. Cheese's	26538-H Moulton Pkwy, Laguna Hills		
9		Johnny's Real New York Pizza	2756 East Coast Hwy, Corona Del Mar		
10		Pieology Pizzeria R&D (Aliso Viejo)	26661 Aliso Creek Rd, Aliso Viejo		
11		New York's Upper Crust Pizza	5613 Alton Pkwy #212, Irvine		
12		Johnny's Real New York Pizza	1320 Bison Ave, Newport Beach		
13		Round Table Pizza	32525 Street of the Golden Lantern, Dana Point		
14		Domino's Pizza	6622 Irvine Center Dr, Irvine		
15		Round Table Pizza	25290 Marguerite Pkwy D, Mission Viejo		
16		Round Table Pizza	612 Camino De Los Mares, San Clemente		
17		Ameci Pizza & Pasta	18068 Culver Dr, Irvine		
18		Pizza Hut	415 E Avenida Pico K, San Clemente		
19		Gina's Pizza	4533 Campus Dr, Irvine		
20		Domino's Pizza	1502 N El Camino Real, San Clemente		

Previous Next

Figure 9

4.3 Details

Once a button in the “Details” column is clicked, your webpage should search for the details of that place. You should use the client-side library that Google provides to get place details. You can find examples of how to use this library on this page:

https://developers.google.com/maps/documentation/javascript/places#place_details

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

4.3.1 Info Tab

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

Fields in the info table	Corresponding response object fields
Address	<i>formatted_address</i>
Phone Number	<i>international_phone_number</i>
Price Level	<i>price_level</i> ; must be converted into "\$" notation
Rating	<i>rating</i> ; must be represented by stars precisely (e.g. a rating of 4.3 and a rating of 4.5 should be represented differently)
Google Page	<i>url</i>
Website	<i>website</i>
Hours	<i>open_now</i> and <i>weekday_text</i> in <i>opening_hours</i> ; must be converted into text values of "Open now: <u>TODAY'S HOURS</u> " and "Closed" from Boolean values. Replace TODAY'S HOURS with the actual opening hours of today.

There should also be a "link" with text "Daily open hours" in the "Hours" field. Once it is clicked, it should open a new dialog displaying the open hours of every day with the current day's hours at the top in bold font. Please note that when you try to get the current day of the week to display "Hours" and "Daily open hours", you should use the place's local time. Therefore, you also need value of "utc_offset" field in the response object to figure out the local time. You can use Moment.js library to do the conversion.

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

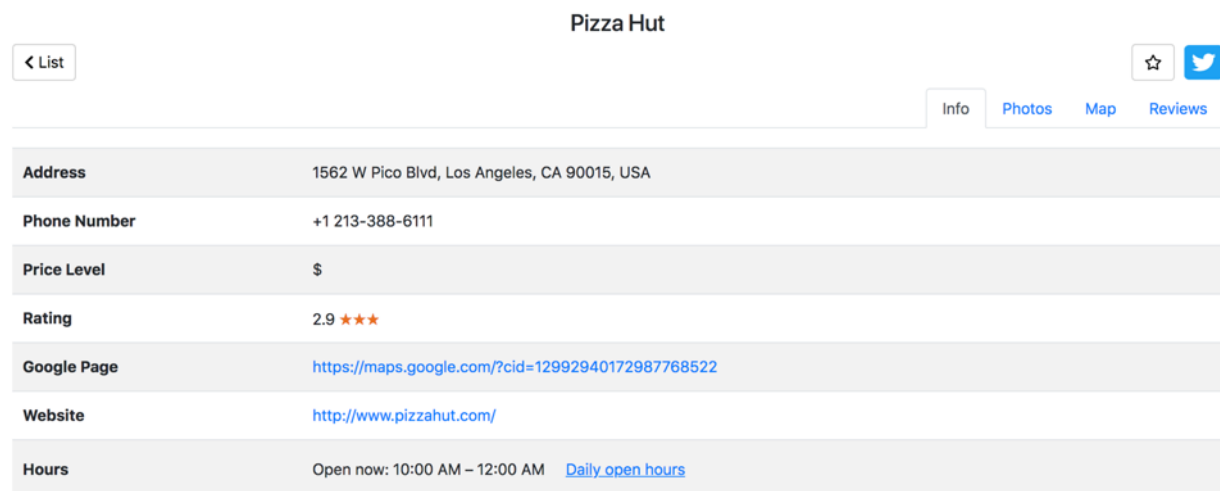


Figure 10

4.3.2 Photos Tab

Using the Google library, you can get the URLs of the photos very easily. You can visit this page to learn how to get the URLs of photos:

https://developers.google.com/maps/documentation/javascript/places#places_photos

You should display the photos in four columns and arrange them in the same manner as in Figure 11 (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.

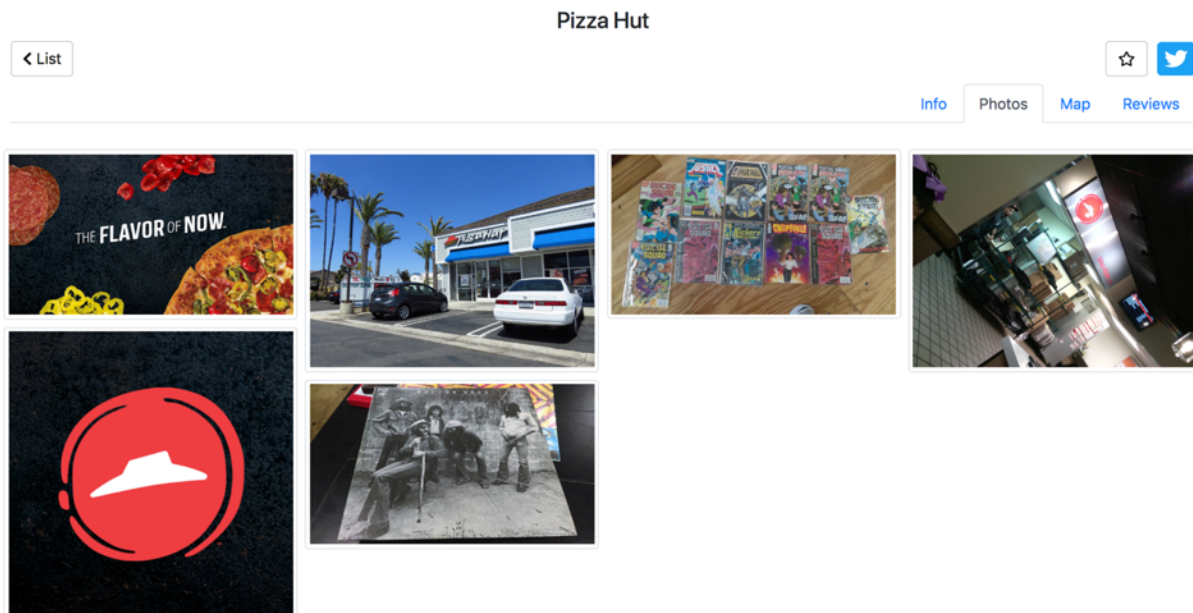


Figure 11

4.3.3 Map Tab

4.3.3.1 Direction Search Form

This form has three input fields and a button:

1. **From:** The starting point goes into this field. If the user uses "Current location" in the initial search form, the default value of this field should be "Your location" which means the user location. Otherwise, the default value is the location specified by the user. The field should support autocomplete in the same way as is described in section 4.1.2. The user can enter "My location" to use the user's current location as the starting point.
2. **To:** This is a read-only field which contains the name and the address of the target place whose details are being displayed.
3. **Travel Mode:** This is a dropdown list containing four travel modes: Driving, Bicycling, Transit, and Walking. The default value is Driving.
4. **Get Directions:** This button is disabled if "From" field is empty or contains only spaces.

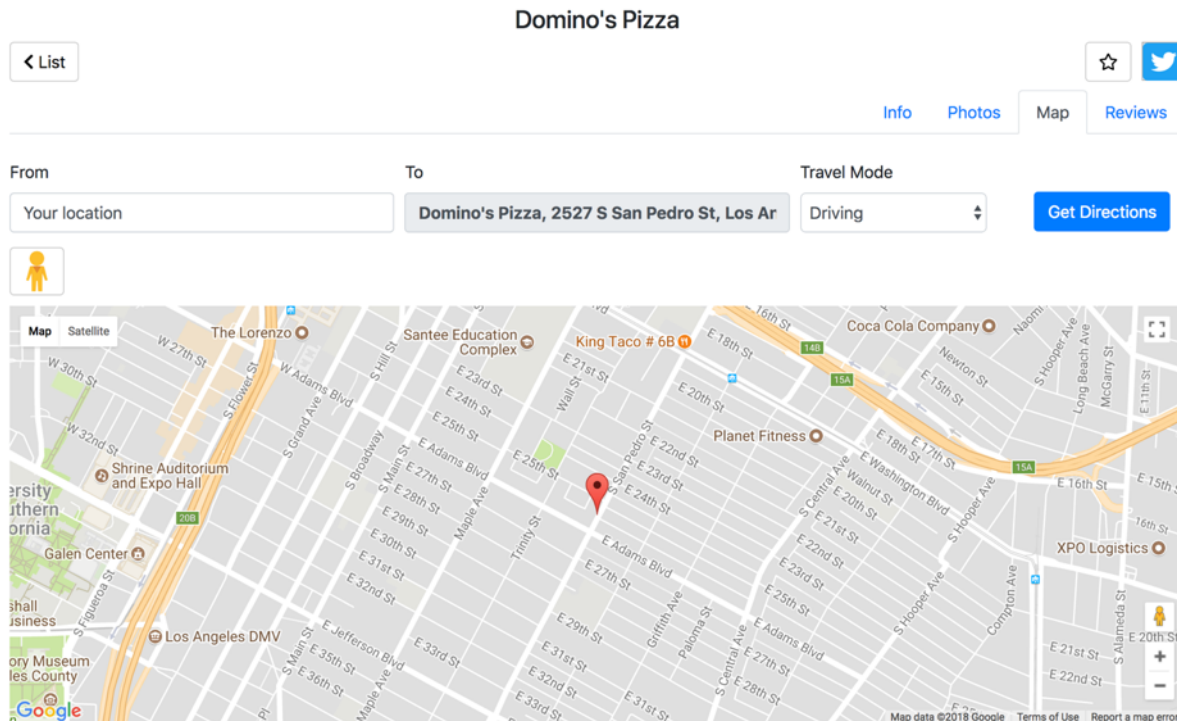


Figure 12

4.3.3.2 Google Map, Street View and Routes

Below the form, there is a Google Map centered at the target place with a marker on it by default. There is a button between the map and the form which toggles the Google Street View. When the street view is on, it's displayed above the map.

Once the "Get Directions" button is enabled and clicked, your application should get the route(s) from the starting point to the destination in the specified travel mode. The detailed travel instructions should be displayed below the map. You should search for all the feasible routes and if there are multiple routes, you should list all the options below the map. The route selected by the user should be displayed on the map and the travel instructions should be listed below.

You can visit this page to learn to get directions using the Google library:

<https://developers.google.com/maps/documentation/javascript/directions>

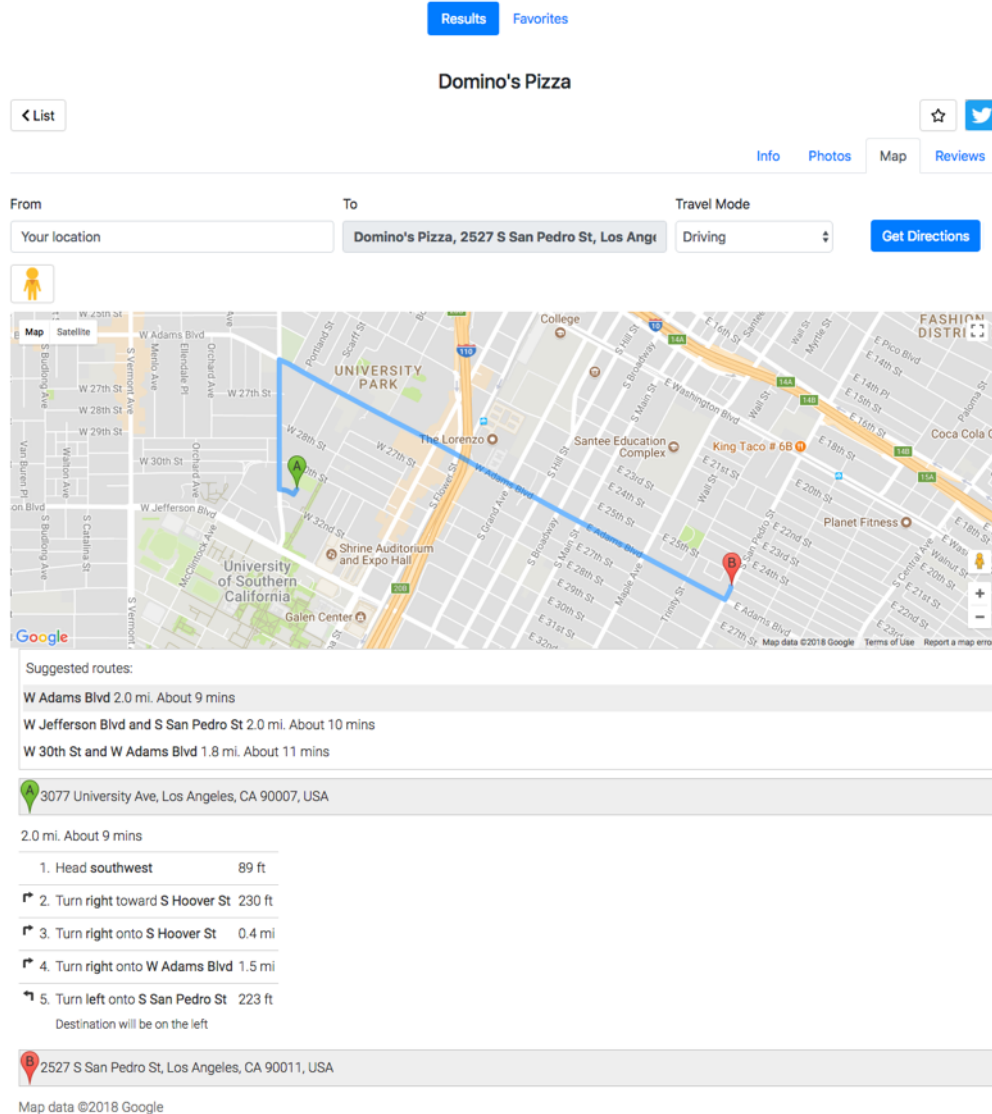


Figure 13

4.3.4 Reviews Tab

This tab displays the Google reviews and Yelp reviews of the place. By default, Google reviews 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 switch between Google reviews and Yelp reviews. The second one allows the user to sort the reviews in several different ways: Default Order, Highest Rating, Lowest Rating, Most Recent and Least Recent.

Each of reviews has its author name, author image, time, rating, and content. The author name and image are clickable. Once they are clicked, a new page should be opened and go to the author's page for Google reviews or the review page for Yelp reviews. The time should be displayed in the format "2018-03-08 14:03:40". The rating should be represented by the stars.

The usage of Yelp APIs to get Yelp reviews is explained in detail in section 5.

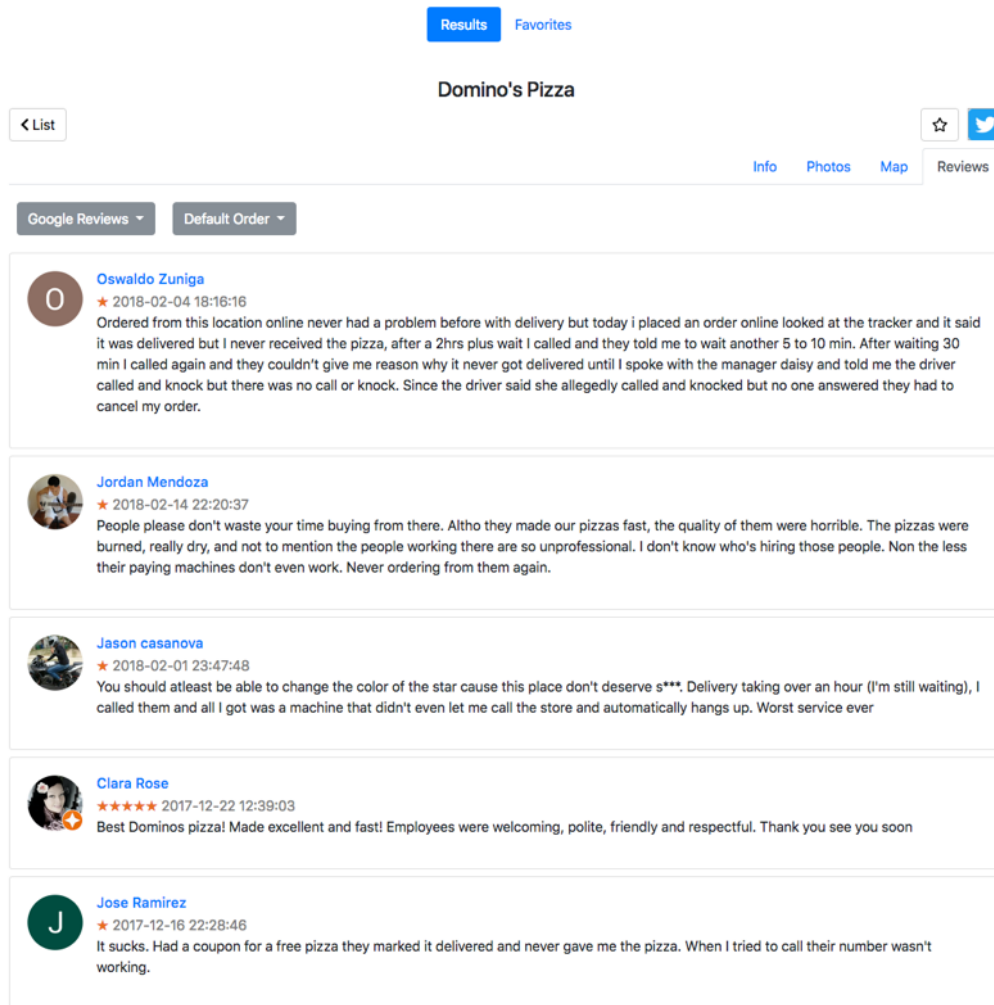


Figure 14

4.3.5 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 15

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

NAME located at ADDRESS. Website: URL #TravelAndEntertainmentSearch”. Replace NAME and ADDRESS with the real name and address. Replace URL with the website’s URL or the Google Page’s URL if the place doesn’t have a website.

Adding 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 and the Twitter button should be disabled until the content of the info tab is available.



Figure 16

4.4 Favorites Tab

The favorites tab is very similar to the results tab: the places 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 place details by clicking on the buttons in the “Details” column; pagination is supported with each page containing up to 20 records.

The major differences between these two tabs are: the place 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 a place from the list and has a trash can icon instead of a star icon; the places in the favorites tab are sorted in the order they are added to the favorites list.



















<div>Results Favorites</div>					
Details >					
#	Category	Name	Address	Favorite	Details
1		Pizza Hut	1562 W Pico Blvd, Los Angeles, CA 90015, USA		
2		Vinny's Pizza	310 E Grand Ave, El Segundo		
3		Domino's Pizza	740 S Olive St, Los Angeles		
4		Pizza Hut	1014 W Martin Luther King Jr Blvd, Los Angeles		
5		Domino's Pizza	5401 S Figueroa St, Los Angeles		
6		Pizza Hut	1555 S Western Ave, Los Angeles		

Figure 17

4.5 Error Messages

If for any reason an error occurs whether it's places search or details search, an appropriate error message should be displayed.

The screenshot shows a search form titled "Travel and Entertainment Search". The form has the following fields and controls:

- Keyword***: A text input field containing "USC".
- Category**: A dropdown menu set to "Default".
- Distance (miles)**: A text input field containing "10".
- From***: A radio button group with "Current location" selected and "Other. Please specify:" as an option. Below "Other" is a text input field containing "Enter a location".
- Buttons**: A blue "Search" button with a magnifying glass icon and a white "Clear" button.

Below the form are two buttons: "Results" (highlighted in blue) and "Favorites".

At the bottom of the page, a red error message bar displays the text: "Failed to get search results."

Figure 18

4.6 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 same search form as in Figure 18, but with the following changes:

- Keyword***: The text input field now contains "blahblahblahlah".

Below the form are the same "Results" and "Favorites" buttons.

At the bottom of the page, a yellow message bar displays the text: "No records."

Figure 19

Keyword*

Category

Default

Distance (miles)

From*

☒ Current location
 ☐ Other. Please specify:

Enter a location

Search

Clear

Results Favorites

Pacific Coast Youth Football and Cheer

< List

Info Photos Map Reviews

No records.

Figure 20

Keyword*

Category

Default

Distance (miles)

From*

☒ Current location
 ☐ Other. Please specify:

Enter a location

Search

Clear

Results Favorites

Pacific Coast Youth Football and Cheer

< List

Info Photos Map Reviews

Google Reviews

Default Order

No records.

Figure 21

The screenshot shows a web form titled "Travel and Entertainment Search". It contains the following fields and controls:

- Keyword***: A text input field.
- Category**: A dropdown menu with "Default" selected.
- Distance (miles)**: A text input field with "10" entered.
- From***: A section with two radio buttons: "Current location" (selected) and "Other. Please specify:". Below the "Other" option is a text input field with the placeholder "Enter a location".
- Search Controls**: A blue button with a magnifying glass icon and the text "Search", and a white button with the text "Clear".
- Navigation**: Two blue buttons labeled "Results" and "Favorites".

Below the form, a yellow banner displays the message "No records.".

Figure 22

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 the feature. You can find hints about the bootstrap components in the Hints section.

This screenshot shows the same "Travel and Entertainment Search" form as Figure 22, but with the following differences:

- The **Keyword*** field now contains the text "USC".
- The **Search Controls** (blue "Search" button and white "Clear" button) are now visible below the form fields.
- At the bottom of the page, a horizontal progress bar is displayed. It consists of a blue segment on the left and a light gray segment on the right, indicating that data is being fetched.

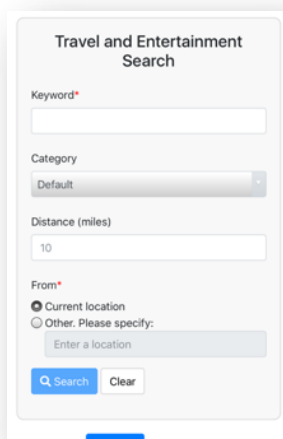
Figure 23

4.8 Animation

Whenever the view switches between results/favorites and details, there should be a sliding effect. Whenever the reviews are switched between Google reviews and Yelp reviews, there should be a fade-in effect. Please check out the video to see effect. **The animation must be implemented with Angular.**

4.9 Responsive Design

The following are snapshots of the webpage opened with Safari on iPhone 6 Plus. See the video for more details.



Travel and Entertainment Search

Keyword*

Category

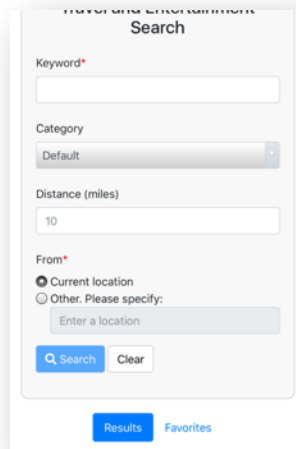
Distance (miles)

From*

Current location

Other. Please specify:

Search Clear



Search

Keyword*

Category

Distance (miles)

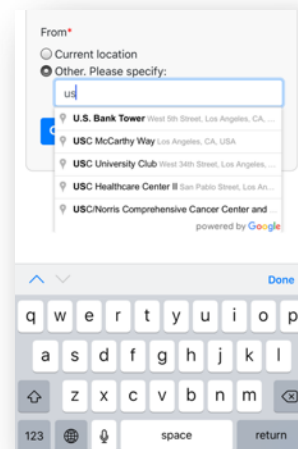
From*

Current location

Other. Please specify:

Search Clear

Results Favorites



From*

Current location

Other. Please specify:

usd

U.S. Bank Tower

USC McCarthy Way

USC University Club

USC Healthcare Center II

USC Norris Comprehensive Cancer Center and

powered by Google

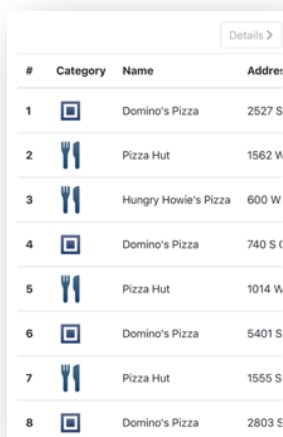
Done

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



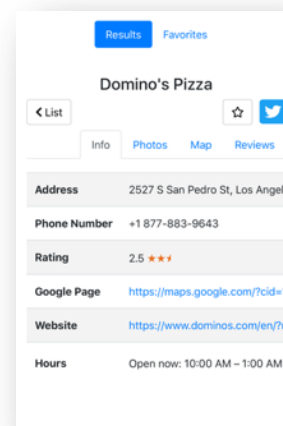
#	Category	Name	Address
1		Domino's Pizza	2527 S
2		Pizza Hut	1562 W
3		Hungry Howie's Pizza	600 W
4		Domino's Pizza	740 S C
5		Pizza Hut	1014 W
6		Domino's Pizza	5401 S
7		Pizza Hut	1555 S
8		Domino's Pizza	2803 S

Details >



12		Pizza Time Theatre	5089 F
13		Jino's Pizza & Deli	1122 C
14		Prime Pizza	141 S C
15		Pizza Man	627 E C
16		Blaze Pizza	1111 S F
17		Blaze Pizza	3335 E
18		Blaze Pizza	4114 S
19		Blaze Pizza	110 S F
20		Blaze Pizza	4809 F

Next



Results Favorites

Domino's Pizza

< List

Info Photos Map Reviews

Address 2527 S San Pedro St, Los Angeles

Phone Number +1 877-883-9643

Rating 2.5 ★★

Google Page <https://maps.google.com/?cid=1>

Website <https://www.dominos.com/en/tr>

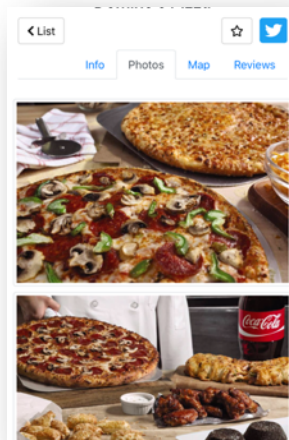
Hours Open now: 10:00 AM - 1:00 AM

Results
Favorites

Open hours

Tuesday	10:00 AM – 1:00 AM
Wednesday	10:00 AM – 1:00 AM
Thursday	10:00 AM – 1:00 AM
Friday	10:00 AM – 2:00 AM
Saturday	10:00 AM – 2:00 AM
Sunday	10:00 AM – 1:00 AM
Monday	10:00 AM – 1:00 AM

Close



List
Info
Photos
Map
Reviews

From
Your location

To
Domino's Pizza, 2527 S San Pedro St, Los Angi

Travel Mode
Driving

Get Directions

Info
Photos
Map
Reviews

Google Reviews
Default Order

0

Oswaldo Zuniga
★ 2018-02-04 18:16:16
Ordered from this location online never had a problem before with delivery but today i placed an order online looked at the tracker and it said it was delivered but i never received the pizza, after a 2hrs plus wait i called and they told me to wait another 5 to 10 min. After waiting 30 min i called again and they couldn't give me reason why it never got delivered until i spoke with the manager daisy and told me the driver called and knock but there was no call or knock. Since the driver said she allegedly called and knocked but no one answered they had to cancel my order.

Suggested routes:
W Adams Blvd 2.0 mi. About 9 mins
W Jefferson Blvd and S San Pedro St 2.0 mi. About 10 mins
W 30th St and W Adams Blvd 1.8 mi. About 11 mins

3077 University Ave, Los Angeles, CA 90007, USA
2.0 mi. About 9 mins
1. Head southwest 89 ft
2. Turn right toward S Hoover St 230 ft

Jason casanova
★ 2018-02-01 23:47:48
You should atleast be able to change the color of the star cause this place don't deserve s***. Delivery taking over an hour (I'm still waiting). I called them and all i got was a machine that didn't even let me call the store and automatically hangs up. Worst service ever

Clara Rose
★★★★★ 2017-12-22 12:39:03
Best Dominos pizza! Made excellent and fast! Employees were welcoming, polite, friendly and respectful. Thank you see you soon

Jose Ramirez
★ 2017-12-16 22:28:46

Current location
Other. Please specify:
Enter a location

Search
Clear

Results
Favorites

Details

#	Category	Name	Address
1		Domino's Pizza	2527 S S
2		Pizza Hut	1562 W
3		Hungry Howie's Pizza	600 W C

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

- The search form should display each component in a vertical way on smaller screens.
- All tables can be scrolled horizontally.
- The photos should be aligned vertically on smaller screens.
- Animation must work on mobile devices.

You must watch the video carefully to see what the page look 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 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.

5. Yelp API Documentation

To use Yelp APIs, you need to first create an App:

https://www.yelp.com/developers/v3/manage_app

You will need to use two APIs:

- Business Match: https://www.yelp.com/developers/documentation/v3/business_match
- Business Reviews: https://www.yelp.com/developers/documentation/v3/business_reviews

Since Business Match API is still in Beta stage, you need to join developer beta on this page:

https://www.yelp.com/developers/v3/manage_app

You **must** make requests to these two APIs from server side in your PHP/Node.js script because you shouldn't expose your API key.

You should first use the place information you get from Google services to make a request to Yelp Business Match **Best Match**: <https://api.yelp.com/v3/businesses/matches/best>. This API will return one best match if there is any match. You should check if this is the same place you are search for. If this is, you should use the business id you obtain to make request a request to <https://api.yelp.com/v3/businesses/{id}> to get the Yelp reviews and returns them to your front end.

6. Libraries

- **Moment.js** – <http://momentjs.com/> for time conversion
- **Yelp libraries** – <https://github.com/Yelp/yelp-fusion> There are several Yelp PHP and Node.js libraries that make it easier to make requests to Yelp APIs
- **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.

7. Implementation Hints

7.1 Images

The images needed for this homework are available here:

<http://cs-server.usc.edu:45678/hw/hw8/images/Map.png>

<http://cs-server.usc.edu:45678/hw/hw8/images/Pegman.png>

<http://cs-server.usc.edu:45678/hw/hw8/images/Twitter.png>

7.2 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.

7.3 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/>

7.4 Google App Engine/Amazon Web Services

You should use the domain name of the GAE/AWS service you created in HW#7 to make the request.

For example, if your GAE/AWS server domain is called

example.appspot.com/example.elasticbeanstalk.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/places?keyword=xxx>

AWS - <http://example.elasticbeanstalk.com/places?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 HW#8, you should also provide a link like this in the table where you have all your links to homework. When your grader clicks on this additional link, a valid link should return a JSON object with data.

7.5 Deploy Node.js application on GAE/AWS

If your backend is implemented with Node.js, when you deploy HW#8 to AWS or GAE, you should select Nginx as your proxy server, which should be the default option. If you select any proxy server other than Nginx, you will NOT get the 2 (two) extra credits.

7.6 AJAX call

You should send the request to the PHP/Node.js script 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 PHP/Node.js script code is running in the “cloud” on Google GAE/AWS. Please refer to the grading guideline for details.

7.7 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. So, 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

8. Files to Submit

In your course homework page, you should update the HW#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 service where the AJAX call is made with sample parameter values (i.e. a valid query, with keyword, location, etc.). Also, submit all your files (HTML, JS, CSS, PHP) electronically to the csci571 account so that they can be graded and compared to all other students’ code. Don’t include any images that we provided or that are included in any library.

****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.