Project Details

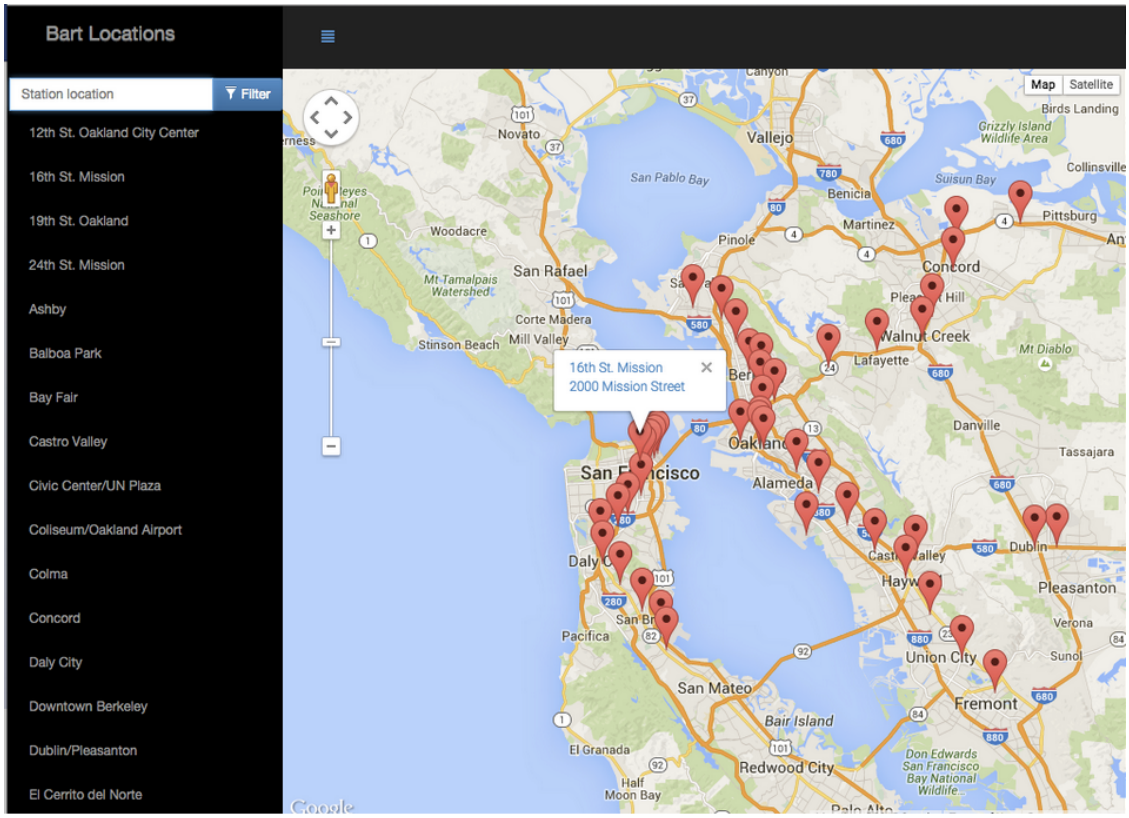
How will I complete this project?

1. Review our course **JavaScript Design Patterns** and check out the **Neighborhood Map project rubric**.
2. Download the **Knockout framework**. Knockout must be used to handle the list, filter, and any other information on the page that is subject to changing state. Things that should not be handled by Knockout: anything the Maps API is used for, creating markers, tracking click events on markers, making the map, refreshing the map. **Note 1**: Tracking click events on list items *should* be handled with Knockout. **Note 2:** Creating your markers as a part of your ViewModel is allowed (and recommended). Creating them as Knockout observables is not.
3. Asynchrony and Error Handling. Note that all data APIs used in the project should **load asynchronously** and errors should be handled gracefully. In case of error (e.g. in a situation where a third party API does not return the expected result) we expect your webpage to do one of the following: A message is displayed notifying the user that the data can't be loaded, **OR** There are no negative repercussions to the UI. **Note:** Please note that we expect students to handle errors if the browser has trouble initially reaching the 3rd-party site as well. For example, imagine a user is using your Neighborhood Map, but her firewall prevents her from accessing the Instagram servers. Here is a reference article on **how to block websites** with the hosts file. It is important to handle errors to give users a consistent and good experience with the webpage. Read **this blogpost** to learn more. Some JavaScript libraries provide special methods to handle errors. For example: refer to .fail() method discussed **here** if you use jQuery's ajax() method. We strongly encourage you to explore ways to handle errors in the library you are using to make API calls.
4. Write code required to add a full-screen map to your page using the **Google Maps API**. For sake of efficiency, the map API should be called only once.
5. If you are prompted to do so, you may want to get a **Google Maps API key**, and include it as the value of the `key` parameter when loading the Google Maps API in **index.html**: `<script src="http://maps.googleapis.com/maps/api/js?libraries=places&key=[YOUR_API_KEY]"></script>` You may have some initial concerns with placing your API key directly within your JavaScript source files, but rest assured this is perfectly safe. All client-side code must be downloaded by the client; therefore, the client must download this API key - it is not intended to be secret. Google has security measures in place to ensure your key is not abused. **It is not technically possible to make anything secret on the client-side.**
6. Write code required to display map markers identifying at least 5 locations that you are interested in within this neighborhood. Your app should display those locations by default when the page is loaded.
7. Implement a list view of the set of locations defined in step 5.
8. Provide a filter option that uses an input field to filter both the list view and the map markers displayed by default on load. The list view and the markers should update accordingly in real

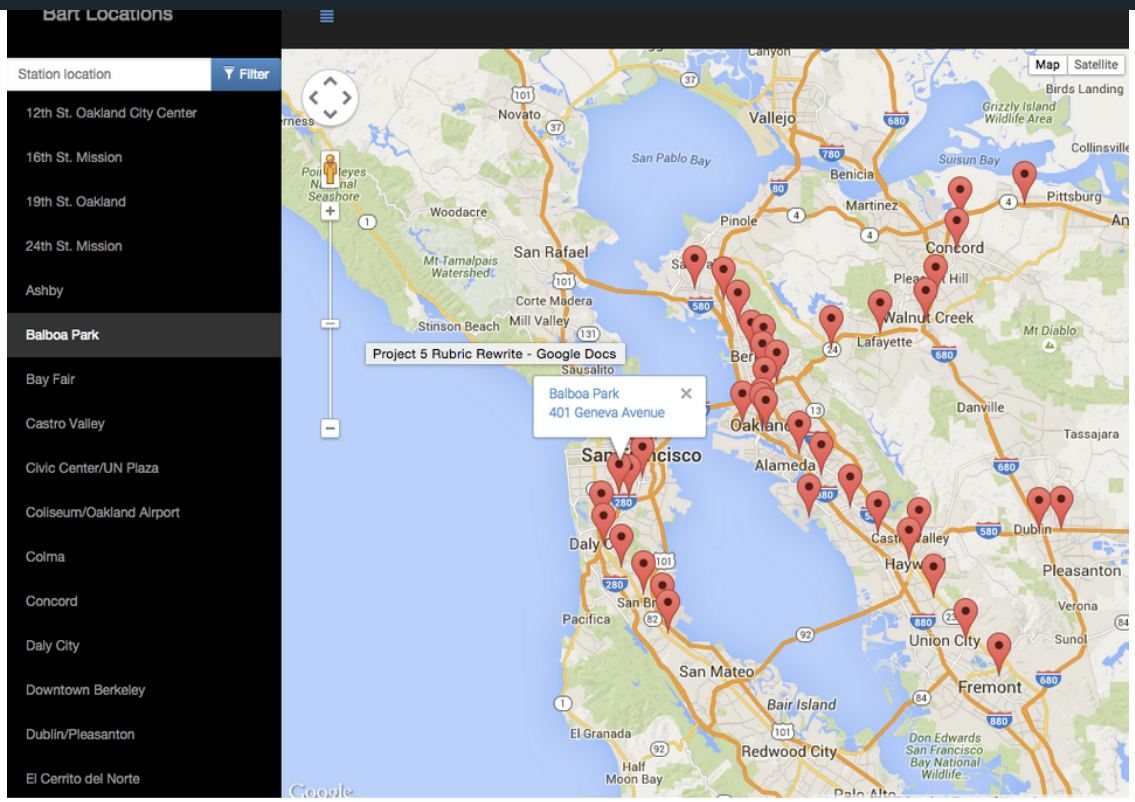
Project Details

9. Add functionality using third-party APIs to provide information when a map marker or list view entry is clicked (ex: Yelp reviews, Wikipedia, Flickr images, etc). Note that StreetView and Places don't count as an additional 3rd party API because they are libraries included in the Google Maps API. If you need a refresher on making AJAX requests to third-party servers, check out our **Intro to AJAX** course. Please provide attribution to the data sources/APIs you use. For example if you are using Foursquare, indicate somewhere in your interface and in your README that you used Foursquare's API.
10. Add functionality to animate a map marker when either the list item associated with it or the map marker itself is selected.
11. Add functionality to open an infoWindow with the information described in step 9 (you can also populate a DOM element with this info, but you should still open an infoWindow, even with minimal info, like location name) when either a location is selected from the list view or its map marker is selected directly.
12. The app's interface should be intuitive to use. For example, the input text area to filter locations should be easy to locate. It should be easy to understand what set of locations is being filtered. Selecting a location via list item or map marker should cause the map marker to bounce or in some other way animate to indicate that the location has been selected and associated info window should open above the map marker with additional information.

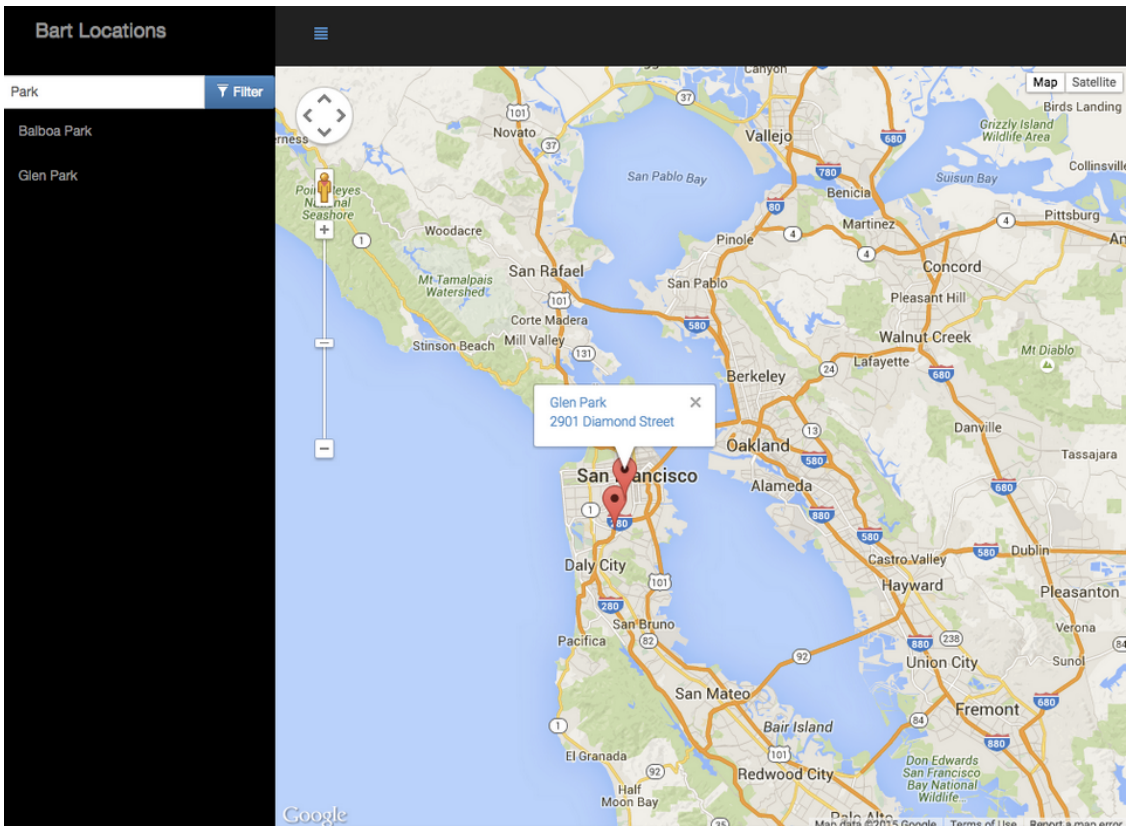
Example: BART Locations San Francisco



Project Details

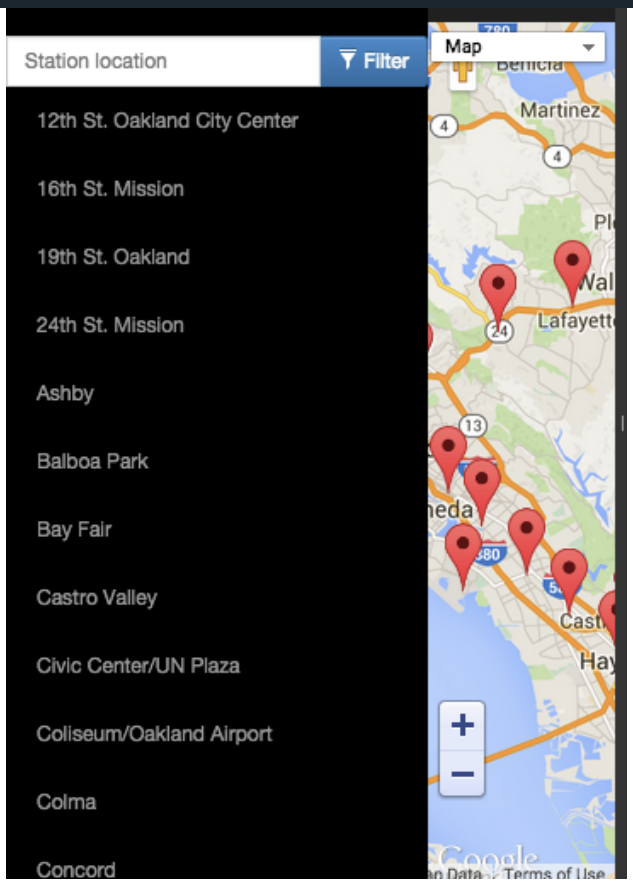


Clicking a name in the list view should open the information window for the associated marker.



The list of locations should be filterable with a text input or dropdown menu. Filtering the list also filters the markers on the map.

Project Details



The web app should be mobile responsive - notice the hamburger menu icon used to hide the list on small screens (this is just one possible mobile implementation).

Helpful Resources

None of these are required, but they may be helpful.

- Foursquare API
- MediaWikiAPI for Wikipedia
- Google Maps Street View Service
- Google Maps
- Project 5 Overview WebCast
- Knockout JS Tutorials
- Todo MVC Knockout Example

NEXT