Senior Design 2020-2021

Sidewalk Slope Monitoring System

Task: Web Application

Version: 1.0

Table of Contents

Google Maps API 3

Create Google Cloud Platform account 3

Billing account 3

Activate Google Maps JavaScript API 3

Static, Embed, and JavaScript maps 4

Map types 4

API-specific use cases 6

Use case 1: Add an interactive map to each rover image page 6

Use case 1: Add markers in the map for each rover image using GPS coordinates 6

Use case 2: Add markers that are in close proximity to the user's location. 7

Use case 3: Group markers by proximity 7

API support use cases 8

Use case 1: Return the rover data for images captured in close proximity of the user's location 8

Use case 2: Group the data for the front page map 8

Use case 3: TBD 8

Spatial Queries 9

Mapping Data 11

Basic Components 11

Implementation 12

Google Maps API

Create Google Cloud Platform account

Follow the steps below to complete the account creation process.

Billing account

Enable the free trial that add $300 worth of credit to the billing account. A credit card is required to ensure that the user is not a robot. The credit card will not be auto charged at the end of the trial.

Activate Google Maps JavaScript API

Visit the Google Cloud Platform home. Select the hamburger icon to open the menu. Click **APIs & Services** > **Library**.

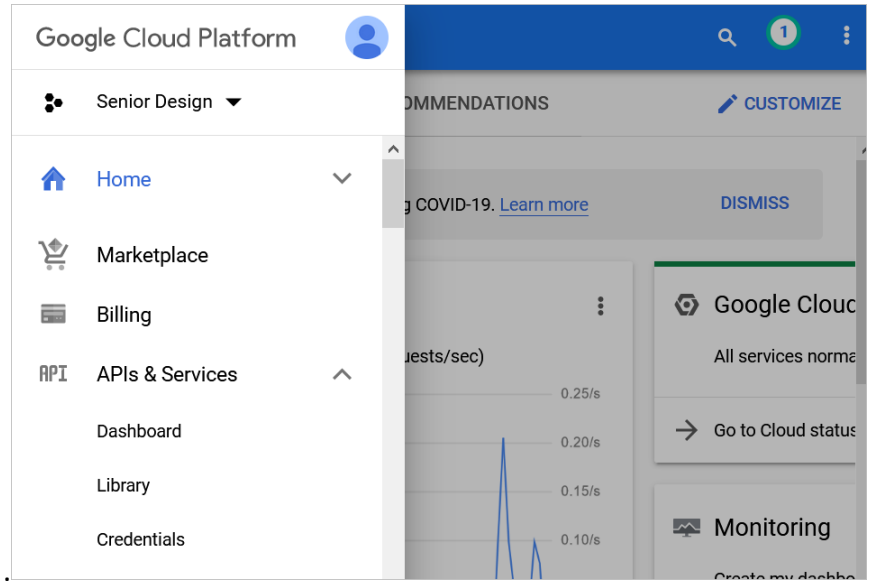


Figure 1 Google Cloud Platform home

Search for “maps” and select **Maps Embed API**, **Maps JavaScript API**, and **Maps Static API**. For each API, click enable. A green checkmark confirms it has been enabled.

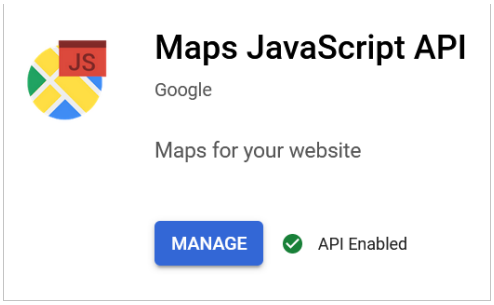


Figure 2 Enable and manage API

Static, Embed, and JavaScript maps

The Google Maps API has various types of maps. For the testing, three types of maps were selected: static, embed, and JavaScript.

Map types

- **Static maps** are images. Users cannot zoom in or out and are limited to the location displayed by the site.
    - `<img src="https://maps.googleapis.com/maps/api/staticmap?center=Brooklyn+Bridge,New+York,NY&zoom=13&size=600x300&maptype=roadmap&key=<apikeyhere>">`
- **Embed maps** are dynamic and can be controlled by the user. Users can zoom in and out, move around in the map, and change to satellite view.
    - `<iframe width="600" height="450" style="border:0" loading="lazy" allowfullscreen src="https://www.google.com/maps/embed/v1/place?q=place_id:EisxMTQ5IFMgQnJvYWR3YXksIExvcyBBbmdlbGVzLCBDQSA5MDAxNSwgVVNBIlESTwo0CjIJK2YbSsnHwoAR90oe2ZB86ZkaHgsQ7sHuoQEaFAoSCZf5vkzGx8KAEReFIrIOAHKcDBD9CCoUChIJvbgjOJrJwoARd0Lt5rv7VRs&key=<apikeyhere>"></iframe>`
- **JavaScript maps** are also dynamic and can be controlled by the user. Users can zoom in and out, move around in the map, and change to satellite view. This map allows for additional manipulation from a back-end perspective. We can use coordinates instead of addresses or place IDs to create a map.
    - `<script async src="https://maps.googleapis.com/maps/api/js?key=<apikeyhere>&callback=initMap"> </script>`

The documentation also provides generators to assist developers in creating their HTML iframes, JavaScript tags, and more. Here is an example of a generator: JavaScript generator.

DOCUMENTATION: https://developers.google.com/maps/documentation/embed/map-generator

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <iframe width="600" height="450" style="border:0" loading="lazy" allowfullscreen
    src="https://www.google.com/maps/embed/v1/place?
    q=place_id:EisxMTQ5IFMgQnJvYWR3YXksIExvcyBBbmdlbGVzLCBDQSA5MDAxNSwgVVNBIlESTwo0CjIJK2YbSsnHwoAR90oe2ZB86ZkaHgsQ7sHuoQEaFAoSCZf5vkzGx8KAEReFIrIOAHKcDBD9CCoUChIJvbgjOJrJwoARd0Lt5rv7VRs&
    key=AIzaSyDA4CKO-V4qOSYxy4DeG8qu2FoBUOh9ZwE"></iframe>

    <img src="https://maps.googleapis.com/maps/api/staticmap?center=Brooklyn+Bridge,New+York,NY&
    zoom=13&size=600x300&maptype=roadmap&key=AIzaSyDA4CKO-V4qOSYxy4DeG8qu2FoBUOh9ZwE">
</body>
</html>
```

Figure 3 Snippet of code using the maps

Figure 4 Static map, limited customization

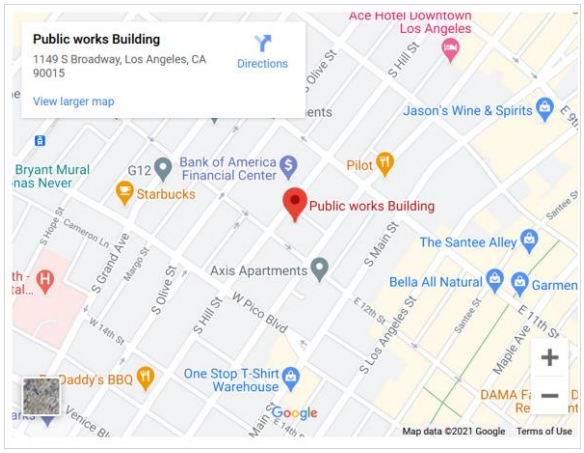


Figure 5 Embed map, zoomed out

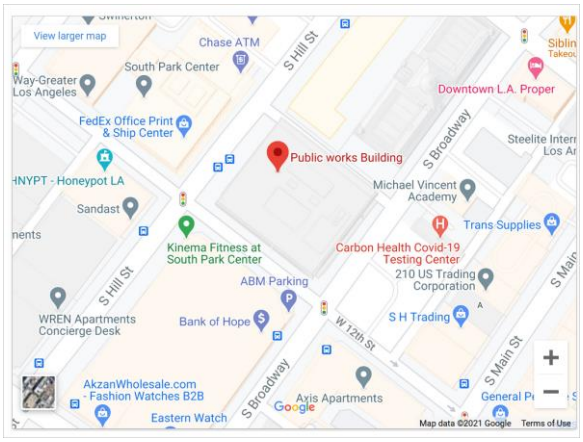


Figure 6 Embed map, zoomed in

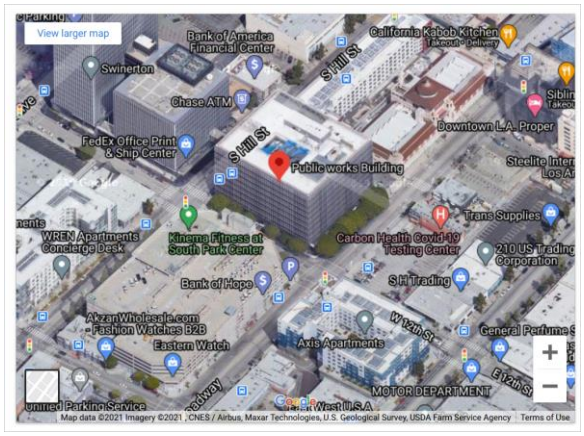


Figure 7 Embed map, using satellite and zoomed in

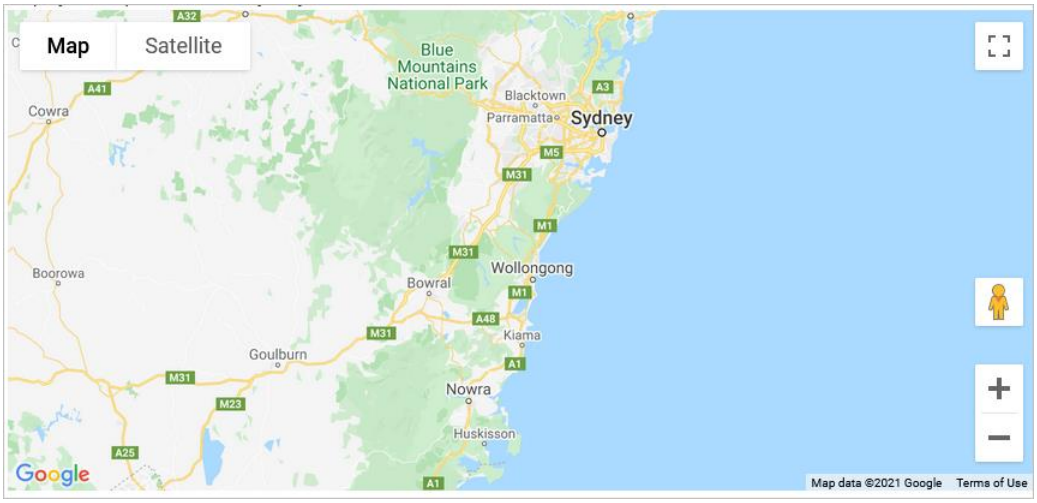


Figure 8 JavaScript map

API-specific use cases

Use case 1: Add an interactive map to each rover image page

A front page may be used to show the entire map of Los Angeles. However, for each rover image page, an interactive map must be embedded into the site.

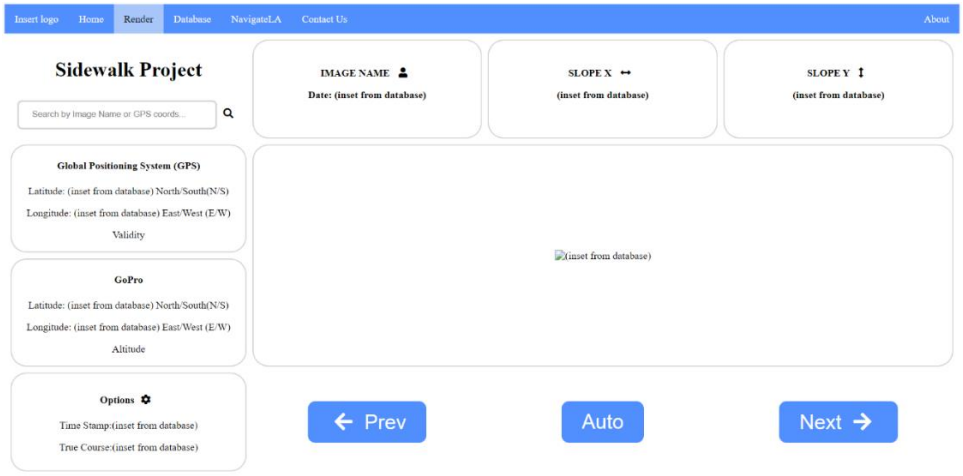


Figure 9 Rover image page example

Use case 1: Add markers in the map for each rover image using GPS coordinates

We can remove the point of interest markers and add markers for sidewalk segments that have images. Depending on how often the GPS coordinates change, we may need to group images. In this case, the user will be in one location in the map but can flip through the images captured at that GPS location.

TO-DO: Observe the difference in GPS coordinates. If the markers overlap due to small differences in GPS coordinates, the images need to be grouped. Otherwise, each marker represents one rover image.

DOCUMENTATION: https://developers.google.com/maps/documentation/javascript/manage-marker-label-collisions

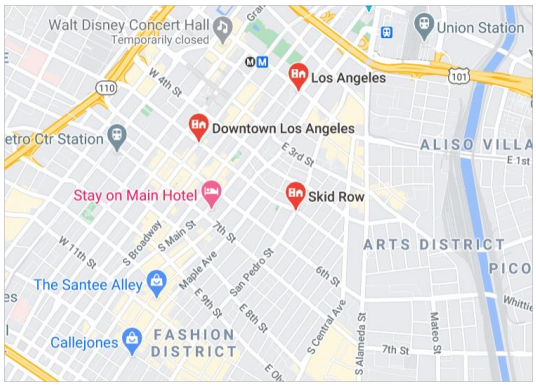


Figure 10 Original POI markers

Figure 11 Mockup of new markers for sidewalk data

Use case 2: Add markers that are in close proximity to the user's location.

Because the map will be interactive, the user can zoom in and out of the map. In order to limit the markers in the map, only rover images captured within a certain distance (e.g. a mile) from the user's location will be shown.

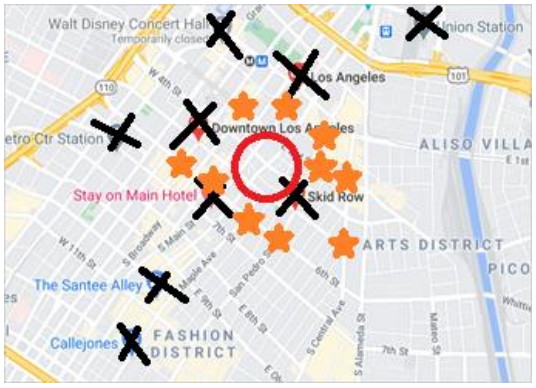


Figure 12 Markers within 1 mile of the user's location

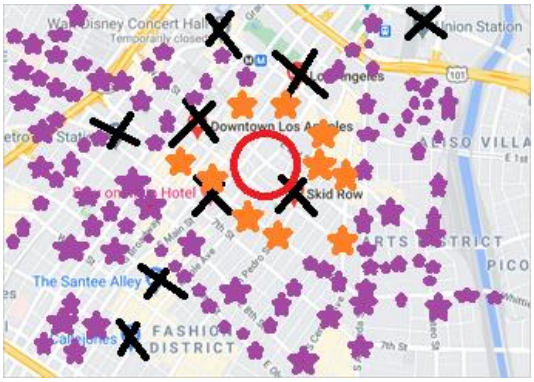


Figure 13 No limit to markers shown

Use case 3: Group markers by proximity

If the user wishes to see the entire map of Los Angeles, the markers can be grouped together. This could be used as the front-page map. This is meant to show a larger picture of the rover data. It will not be embedded in the rover data page.

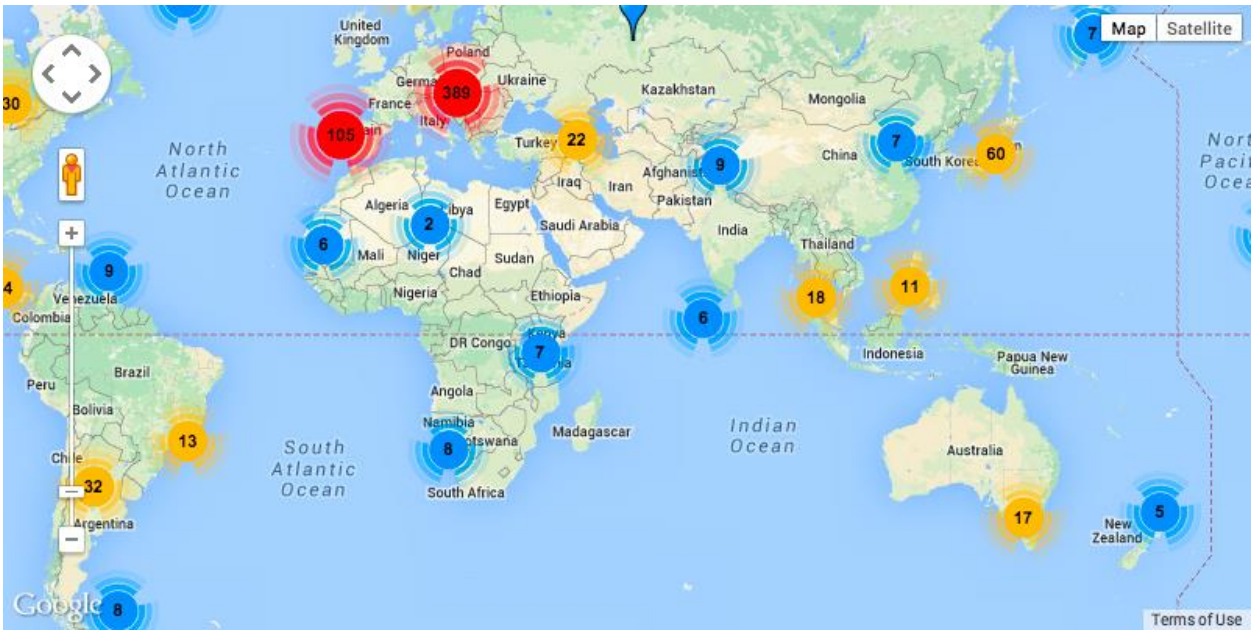


Figure 14 Example of grouping data using markers

EXAMPLE: https://stackoverflow.com/questions/31722630/group-coordinates-by-proximity-to-each-other

API support use cases

Use case 1: Return the rover data for images captured in close proximity of the user's location

See Use case 2: Add markers that are in close proximity to the user's location.

EXAMPLE: https://stackoverflow.com/questions/15759518/find-proximity-distance-gps

Use case 2: Group the data for the front page map

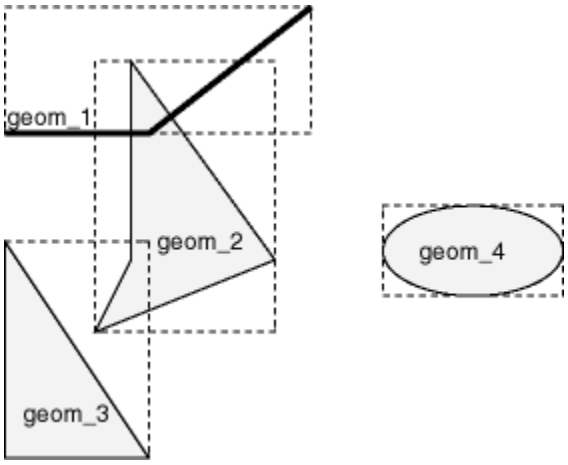
See Use case 3: Group markers by proximity.

Use case 3: TBD

Spatial Queries

*Suggested by Soo

Depending on the query performed, we may be able to pull all objects that lie within a query window [Source].



This is an example where we get all buildings in a region:

Get all the buildings in the KwaZulu region:

```
SELECT a.id, a.name, st_astext(a.the_geom) as point
  FROM building a, region b
    WHERE st_within(a.the_geom, b.the_geom)
    AND b.name = 'KwaZulu';
```

Result:

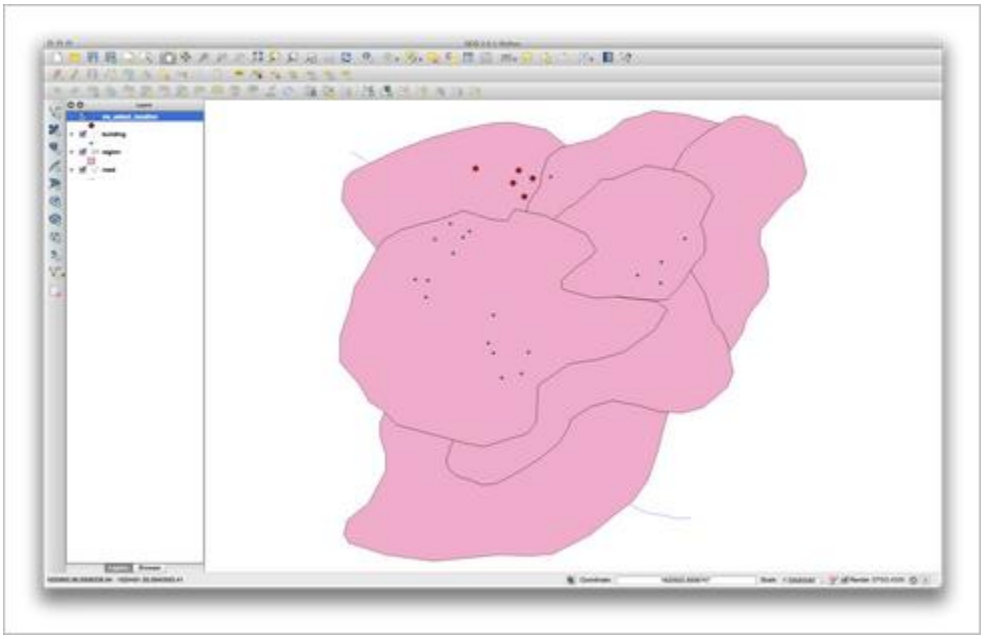
```
 id | name |                   point
----+------+-------------------------------------------
 30 | York | POINT(1622345.23785063 6940490.65844485)
 33 | York | POINT(1622495.65620524 6940403.87862489)
 35 | York | POINT(1622403.09106394 6940212.96302097)
 36 | York | POINT(1622287.38463732 6940357.59605424)
 40 | York | POINT(1621888.19746548 6940508.01440885)
(5 rows)
```

We can also get items that are closest to the given GPS coordinate: [Source]

Ex: Select * from people where distance({gps_coordinates}) < 2;

Requires PostGIS functions in a database

Using the results from the queries, we can leverage software to build our maps for us.



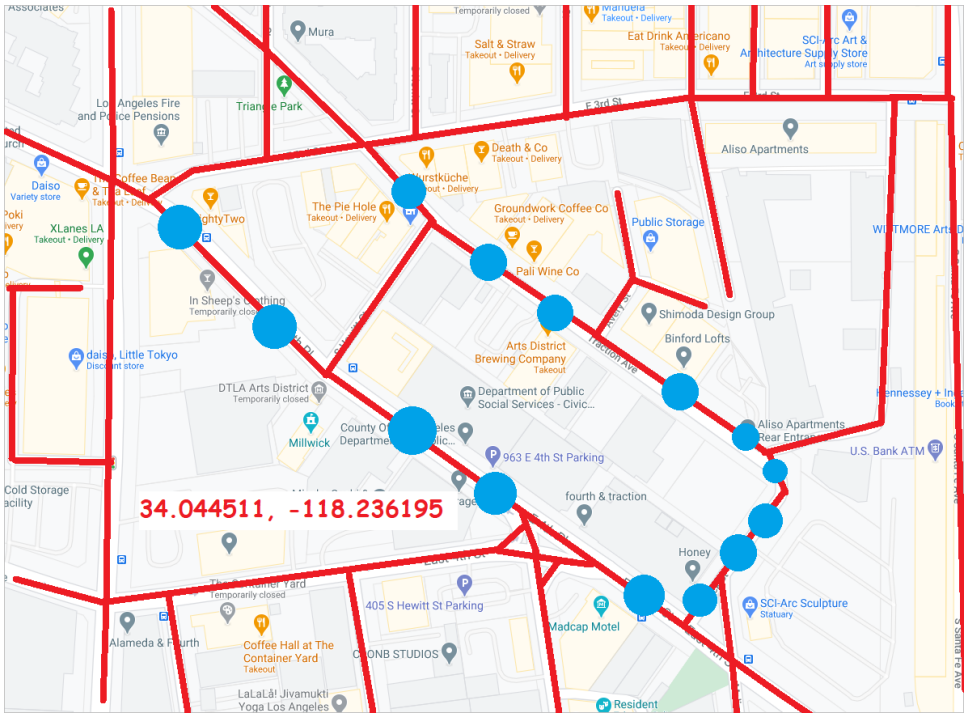
Mapping Data

From the data that we collect in our queries, we would need to have a script that builds a view of that in a map form. The user could select the dot and each dot already has an ID for the images taken at that spot.

For example: The user might enter the GPS coordinates of a particular part of the city. Our query would find the closest points to that location and display the image at that spot using our website. In the site, we might need to add a map feature like the image below. It would display the location at where there are images of the sidewalk taken by the rover.

It may be necessary to display the actual streets. In this case, we can use existing mapping programs (like ArcGIS, QGIS, MapInfo) and embed them to our site. We would just need to adjust any necessary HTML to display the blue points. [Example] [Example]

We can also query Google Map's API and use their map customization [Example] or as recommended by Alexis, Microsoft's Azure Maps [Example].



Basic Components

These basic components were outlined in the scholarly journal, "Usage of Web Mapping Systems and Services for Information Support of Regional Management" [Source]. Only the components that can apply to our system are listed

1. Database system with geospatial data [WIP, database team already has some data in it]
2. Spatial metadata management subsystem aka spatial query functions in the database [implementation needs to be looked into]

3. Web application [Already have most of frontend, backend still needs to be developed which is pretty much this whole thing]
4. Auxiliary services subsystem [all our systems must talk to eachother smoothly and have a standard form of passing data aka JSON, which we've already started doing with our web requests to the database]

Implementation

Now that we've looked into spatial queries and explored possible methods of mapping our data, we will start to see how we can integrate spatial queries with the database and how to start displaying that data in our site.