

A decorative graphic on the left side of the slide consisting of two overlapping parallelograms. The front one is blue and the back one is a light green. They are positioned diagonally, with the blue one partially covering the green one.

City Transportation Analysis

Emile & Warren



The issue at hand

The availability of public transportation in New York City can often seem unbalanced, especially if you look at the outer boroughs like Brooklyn and Queens.

Even beyond service issues and delays, there are just some areas of the city that are very “dry” in terms of transportation, whether it be a lack of train stops, a lack of bus stops, or a lack of both.

This is a clear ethics issue, as there is an inequity for transportation, which can ultimately negatively affect the opportunities that citizens living in “dry” transportation neighborhoods.

How can we build an app/tool that will help us illustrate the differences in public transportation availability, and perhaps tell us where we need to improve infrastructure?



Our “solution”/tool to use for analysis

Using the [Transitland](#) API, we knew we would be able to obtain some data on the current state of public transportation in the city.

By discovering/analyzing areas in the city where we can confirm that transportation services/availability are inadequate, we can further illustrate the issue that we’ve brought to the table.

Our program uses the API to take an input combination of longitude and latitude as well as a radius around the coordinate (in meters), and then tells the user what other types of transportation are available in that vicinity.

Hopefully, it will be able to demonstrate that Brooklyn and Queens in particular lacks good transportation services.



How it currently* works

The program, at its core, takes a set of coordinates and a radius to search around them then gives all the public transit stops near that location. This program makes extensive use of the Transitland API, detailed [here](#)

The program uses 5 files to work

- `data_builder.py`: this Python file builds local JSON files that help the program run while minimizing requests made to the API
- `routes.json`: this JSON file stores all of the route data needed for the program to work, including all of the stops that make up a route.
- `download_log.json`: this JSON file keeps track of what route data has already been downloaded so that duplicate data is not added to `routes.json` and unnecessary API requests are not made
- `pathfinder.py`: this Python file searches through `routes.json` to find transit stops that are within a specified distance of a given location
- `main.py`: this Python file is what is actually used to run the program and takes user input.



How it currently* works

The program essentially follows this process

1. Get data from the user about what coordinates to search and what search radius to use
2. Use the Transitland API to get what routes are within range of those coordinates. This API request does not return what stops are nearby that location
 - a. Routes are things like the A train, B44 bus, SI Ferry, etc., so this can tell you that the A train is near you but not what stop on the A line is near you
3. Check download_logs.json to see if the data for those routes have already been downloaded to routes.json. If all relevant route data has been downloaded, skip to step 5
4. For each route that routes.json doesn't have data for, make a request to the Transitland API to download that route data (including all the stops in the route) to routes.json and add that those routes have been downloaded to download_log.json
5. Traverse through the entirety of routes.json to find all the stops that are within the specified search radius of the given coordinates
 - a. If multiple stops in the same route are within the search radius, it will only include the closest one
6. Print out all the stops near the given coordinates along with their distance and associated route
 - a. In some cases the program will also make a CSV for the results of the program



Justifications for program design

The free version of the Transitland API has limitations, including how many times it can be requested per month. Thus storing data locally was decided as a way to somewhat circumvent this.

At the same time though, it was unrealistic to download all the transportation data required for the program beforehand. Downloading the data as needed made for a good middle ground.

Before route data is saved to routes.json, it is stored in a temporary dictionary first. This is so that any errors that may arise from API requests occur before writing to the file to minimize chances of the files becoming corrupted.



Code showcase (live)

- https://github.com/EmileJB/ethics-final/blob/main/data_builder.py
- https://github.com/EmileJB/ethics-final/blob/main/download_log.json
- <https://github.com/EmileJB/ethics-final/blob/main/main.py>
- <https://github.com/EmileJB/ethics-final/blob/main/pathfinder.py>
- <https://raw.githubusercontent.com/EmileJB/ethics-final/main/routes.json>

Sample result

```
Command Prompt
C:\Users\oldsp\Downloads\APITestingBackUp>py main.py
This program can tell you the public transit stops near a location in NY (and potentially more in the future)
Enter the coordinates below (If you're not sure of the coordinates try using this link: https://www.latlong.net/convert-address-to-lat-long.html)
Enter the latitude: 40.768742
Enter the longitude: -73.965179
Enter how far you're willing to travel to a public transit stop (in meters). Keep in mind if you would theoretically be walking, riding a bike or car over, or getting dropped off
Enter the distance: 400
Downloading Route Data for r-dr5ru-m66
r-dr5ru-m66 Download Successful
Downloading Route Data for r-dr5ru-m72
r-dr5ru-m72 Download Successful

##### The transit stops closest to your location are: #####

Stop: MADISON AV/E 69 ST
Coordinates: 40.770075, -73.966558
188.3282645350457 meters away from you
Part of the M2 Washington Heights - East Village route

Stop: MADISON AV/E 69 ST
Coordinates: 40.770075, -73.966558
188.3282645350457 meters away from you
Part of the M3 Fort George - East Village route

Stop: LEXINGTON AV/E 68 ST
Coordinates: 40.767713, -73.964248
138.69436877925938 meters away from you
Part of the M101 East Village - Fort George route

Stop: LEXINGTON AV/E 70 ST
Coordinates: 40.769067, -73.96332
161.04842909487138 meters away from you
Part of the BxM1 Riverdale - East Midtown route

Stop: LEXINGTON AV/E 68 ST
Coordinates: 40.767713, -73.964248
138.69436877925938 meters away from you
Part of the M102 Harlem - East Village route

Stop: LEXINGTON AV/E 68 ST
Coordinates: 40.767713, -73.964248
138.69436877925938 meters away from you
Part of the M103 East Harlem - City Hall route

Stop: 68 St-Hunter College
Coordinates: 40.768141, -73.96387
129.1063054918885 meters away from you
Part of the 4 Lexington Avenue Express route
```

Note that we use a radius of 400 meters for our tests. We got this number based on research that shows that 400 meters is about the average amount that people are willing to walk.
(https://safety.fhwa.dot.gov/ped_bike/ped_transit/ped_transguide/ch4.cfm)



How we decided on points to use

We needed to come up with different coordinate points to use that are relevant to our issue.

So, we jumped onto data.ny.gov to find relevant datasets related to transportation that might be useful to analyze with our program.

The following are a few different sources that we drew data/inspiration from for the purposes of our project.




Bike requests in building

One dataset that seemed relevant to our topic was one called [“Bikes in Buildings Requests”](#).

It’s exactly as the title suggests – data of people that requested to have their bicycles in their apartment building. It consists of a lot of different types of data, such as information on the tenant, information on the area, building information, parking information, etc. It does actually have latitude and longitude as well, but using those coordinates for every single apartment building in the dataset would be unreasonable.

This dataset made a lot of sense for us to incorporate into our project, as bike riders (especially after the pandemic), are often people who do it as an alternate way to commute in the city perhaps because the public transportation situation in their area is not good.



```
bikes_in_buildings = client.get("scjj-6yaf", select = 'noofbicyclerequested, ownerzipcode', limit = 50000) #not sure if all of these are necessary

#find the areas with the most requests
#print(bikes_in_buildings)

zip_codes = {}

for entry in bikes_in_buildings:
    if entry['ownerzipcode'] not in zip_codes: #traverse through entire list of entries and add zip codes to dictionary while counting the entries for each zip code
        zip_codes[entry['ownerzipcode']] = int(entry['noofbicyclerequested'])
    else:
        zip_codes[entry['ownerzipcode']] += int(entry['noofbicyclerequested'])

# print(zip_codes) #this is a dictionary with the zip codes and the number of requests for each one
sorted_zip_codes = dict( sorted(zip_codes.items(), key=operator.itemgetter(1), reverse=True)) #sorts by descending values. found the solution online: https://www.w3resou

print('Dictionary in descending order by value : ', sorted_zip_codes)
```

The only pieces of data we needed to look at were the number of bicycles requested, and the owner's zip code. I then made a library for each zip code, and began adding(counting) the number of requests that corresponded with each zip code.

```
Dictionary in descending order by value : {'10017': 2566, '10019': 1579, '10022': 1316, '10036': 963, '10001': 581, '10010': 405, '10003': 397, '11201': 388, '10020': 353, '10281': 294, '10007': 252, '10005': 230, '10018': 220, '10038': 211, '10006': 208, '10004': 187, '10011': 178, '10170': 171, '10168': 163, '10012': 153, '11241': 133, '10016': 129, '10175': 116, '10120': 103, '10158': 100, '10178': 98, '10013': 97, '10014': 93, '10153': 80, '10105': 73, '10107': 64, '10055': 60, '10174': 60, '11245': 53, '10041': 53, '10282': 49, '10166': 41, '10111': 41, '10106': 33, '10171': 29, '11101': 29, '10167': 28, '10121': 25, '10177': 25, '10285': 25, '10151': 20, '10172': 19, '10176': 18, '10458': 16, '10103': 15, '10065': 13, '10154': 12, '11238': 10, '33131': 10, '10068': 10, '10112': 9, '10027': 9, '10032': 7, '10461': 6, '10035': 5, '11211': 5, '10039': 5, '2458': 5, '60606': 5, '10165': 4, '11802': 4, '10705': 3, '12180': 3, '2210': 2, '6103': 2, '10040': 2, '11038': 2, '10152': 2, '10271': 2, '10681': 2, '10037': 1, '10110': 1, '10109': 1, '11225': 1, '10128': 1, '10025': 1, '11424': 1, '10009': 1, '10088': 1, '10220': 1, '10023': 1, '10286': 1, '10074': 1, '11019': 1, '10021': 1, '10279': 1, '10115': 1, '10301': 1, '10080': 1}
```

We only paid attention to the zipcodes with requests greater than or equal to 50.

Using <https://www.freemaptools.com/convert-us-zip-code-to-lat-lng.htm>:

10017(M): 40.75252,-73.97307
10022(M): 40.75835,-73.96800
10019(M): 40.76719,-73.99181
10036(M): 40.75948,-73.98987
10170(M): 40.75286,-73.97610
10020(M): 40.75891,-73.97902
10001(M): 40.75080,-73.99612
10281(M): 40.70580,-74.01860
10010(M): 40.73863,-73.98294
11201(BK): 40.69460,-73.99064
10012(M): 40.72557,-73.99821
10003(M): 40.73139,-73.98840
10018(M): 40.75526,-73.99668
10168(M): 40.75240,-73.98300
10006(M): 40.70793,-74.01333
10016(M): 40.74459,-73.97809
10038(M): 40.70879,-74.00322
10005(M): 40.70617,-74.00860
11241(BK): 40.69320,-73.99120
10107(M): 40.76590,-73.98140
10004(M): 40.69538,-74.02653
10007(M): 40.71422,-74.00817
10178(M): 40.74790,-73.97930
10120(M): 40.74870,-73.98620
10175(M): 40.75640,-73.98270
10011(M): 40.74406,-74.00459
10177(M): 40.75521,-73.97608
10041(M): 40.70312,-74.01002
10153(M): 40.76374,-73.97268
10167(M): 40.75515,-73.97497
10105(M): 40.76440,-73.97800
10013(M): 40.72092,-74.00888

Surprisingly, or maybe not so surprisingly, the vast majority of the zip codes that we were able to derive from were actually in Manhattan.

There are obviously going to be tons of public transportation opportunities in Manhattan, so it left us wondering why there were so many bikes in buildings requests in these areas.

Some potential reasons for this:

- Less requests for bikes in Brooklyn and Queens as more residents there are in homes, not apartments
- Those in Manhattan have better occupations, and thus more disposable income to purchase bikes
- Manhattan buildings have many more residents than the average building in Brooklyn, which means there are going to be way more requests in those buildings

We ended up using the zip code data anyways with our program to see if we could learn anything else from it.

Interborough Express line

The Interborough Express line is a decades old idea that was recently revived by Governor Kathy Hochul.

The idea is to create an avenue of transportation that connects Brooklyn and Queens. It would open up the possibility of Brooklyn and Queens residents to access many more subway lines.

IBX proposed route neighborhoods:

(coordinates by Google)

Sunset Park: 40.6527, -74.0093

Borough Park: 40.6350, -73.9921

Kensington: 40.7934, -73.7221

Midwood: 40.6204, -73.9600

Flatbush: 40.6415, -73.9594

New Lots: 40.6618, -73.8931

Brownsville: 40.6652, -73.9125

East New York: 40.6591, -73.8759

Bushwick: 40.6958, -73.9171

Ridgewood: 40.7044, -73.9018

Middle Village: 40.7174, -73.8743

Maspeth: 40.7294, -73.9066

Elmhurst: 40.7394, -73.8779

Jackson Heights: 40.7557, -73.8831

Clearly, there is a need in Brooklyn and Queens for more access to transportation.

We took the neighborhoods that the interborough express line would service and got the coordinates for each and put it in our program.





Bus Service Delivered

Another interesting dataset we found was “[MTA Bus Service Delivered](#)”.

The dataset doesn't include a whole lot of information. It calculates “service delivered” by taking the number of scheduled buses in a day for a particular route_id, and then the actual number of buses on that given day, and dividing to get a percentage.

What we can derive from this dataset are route_id's, which are the MTA bus routes. It can tell us where the MTA is failing to meet expectations in terms of bus services.

```
bus_service_delivered = client.get("2e6s-9gpm", select = 'borough, route_id, service_delivered', limit = 50000)

bad_service = []

for entry in bus_service_delivered:
    if float(entry['service_delivered']) < 0.60 and entry['route_id'] not in bad_service: #if the service delivered is less than 60% and the route id is not already in the list of bad service, add it to the list
        bad_service.append(entry['route_id'])

bad_service = [entry for entry in bad_service if not entry.startswith('S')] #remove all the routes related to Staten island because they are outliers (sorry SI)
print(bad_service)
```

We chose to consider bus routes with less than 60% service delivered as “buses with bad service”. Note that the dataset takes data daily, meaning that it’s not as if these routes are always only providing 60% of scheduled service, but that there were times where the route did get to that point.

Either way, it describes some bus routes that we can safely say provide inconsistent/inadequate services.

Output:

```
bus routes with bad service: ['M100', 'Q56', 'Q54', 'Q24', 'Q64', 'Q58', 'Q110', 'Q55', 'Q112', 'Q59', 'QM17', 'BM3']
```

We decided not to include Staten Island buses, because they were outliers that skewed the data results. As we predicted, Queens and Brooklyn were prevalent (particularly Queens).

#####

The transit stops closest to your location are:

Stop: GRAND ST/BEDFORD AV
Coordinates: 40.714667, -73.961579
0.0 meters away from you
Part of the Q59 Williamsburg - Rego Park route

Stop: BEDFORD AV/GRAND ST
Coordinates: 40.714721, -73.961019
47.69674270970089 meters away from you
Part of the B62 Downtown Brooklyn - Long Island City route

#####

The transit stops closest to your location are:

Stop: BROADWAY/KENT AV
Coordinates: 40.710842, -73.968417
0.0 meters away from you
Part of the Q59 Williamsburg - Rego Park route

Stop: BROADWAY/KENT AV
Coordinates: 40.710842, -73.968417
0.0 meters away from you
Part of the B32 Williamsburg - Long Island City route

#####

The transit stops closest to your location are:

Stop: BROADWAY/WYTHE AV
Coordinates: 40.710487, -73.966188
0.0 meters away from you
Part of the Q59 Williamsburg - Rego Park route

Stop: BROADWAY/WYTHE AV
Coordinates: 40.710487, -73.966188
0.0 meters away from you
Part of the B32 Williamsburg - Long Island City route

#####

Downloading Route Data for r-dr5rt-b43
r-dr5rt-b43 Download Successful

The transit stops closest to your location are:

Stop: GRAND ST/GRAHAM AV
Coordinates: 40.711655, -73.943117
0.0 meters away from you
Part of the Q59 Williamsburg - Rego Park route

Stop: GRAND ST/GRAHAM AV

Q59 to QUEENS BLVD 62 DRIVE via GRAND

- BROADWAY/ROEBLING ST
- BROADWAY/BEDFORD AV
- BROADWAY/WYTHE AV
- BROADWAY/KENT AV
- KENT AV/S 6 ST
- KENT AV/S 3 ST
- KENT AV/S 1 ST
- GRAND ST/WYTHE AV
- GRAND ST/BEDFORD AV
- ROEBLING ST/GRAND ST
- METROPOLITAN AV/HAVEMEYER ST
- METROPOLITAN AV/RODNEY ST
- GRAND ST/UNION AV
- GRAND ST/LORIMER ST
- GRAND ST/GRAHAM AV
- GRAND ST/BUSHWICK AV
- GRAND ST/WATERBURY ST
- GRAND ST/MORGAN AV
- GRAND ST/METRO BRIDGE
- GARDNER AV/GRAND ST
- GRAND ST/GRAND ST BRIDGE
- GRAND AV/47 ST
- GRAND AV/PAGE PL
- GRAND AV/54 ST
- GRAND AV/56 ST
- GRAND AV/RUST ST
- GRAND AV/59 ST
- GRAND AV/58 AV
- GRAND AV/61 ST
- GRAND AV/64 ST
- GRAND AV/66 ST
- GRAND AV/QUEENS MIDTOWN EP
- GRAND AV/69 PL
- GRAND AV/71 ST 🚶 < 1 stop away, ~12 passengers on vehicle
- GRAND AV/73 ST
- GRAND AV/74 ST
- GRAND AV/79 ST
- GRAND AV/84 ST 🚶 approaching, ~15 passengers on vehicle
- GRAND AV/HASPEL ST
- GRAND AV/VAN HORN ST
- GRAND AV/QUEENS BL
- QUEENS BL/55 AV
- QUEENS BL/56 AV
- QUEENS BL/WOODHAVEN BL
- HOR HARDING SER RD S/QUEENS BL
- JUNCTION BL/HOR HARDING SR RD S

Each bus route has a lot of data points, since there are so many stops along the way.

For example, the Q59 bus line has about 46 different stops on it on it's own.

We ran code to observe the data for each individual bus stop along each of the routes. They were then put in a CSV file, just like the others.

Findings

The data that we derived from our tests were put in csv files that were put in our GitHub repository.

Emile found a cool way to visualize this data through Google's "My Maps".

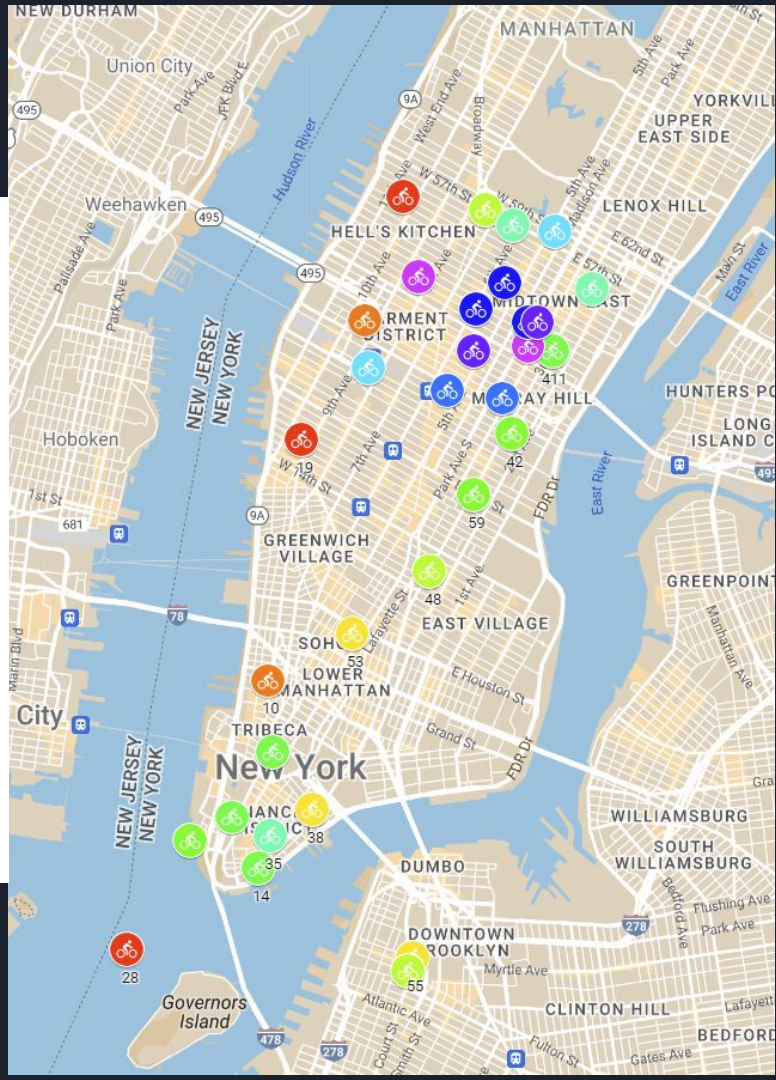
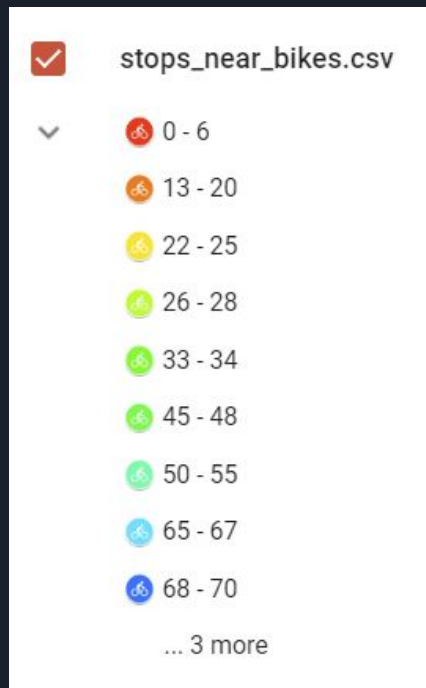
Take a look!

Map 1:

https://www.google.com/maps/d/viewer?mid=1CWre-j9GC4_ytR5KWOxKWtZ-hrWCM5M&ll=40.727877636912936%2C-73.874315&z=11

Map 2:

<https://www.google.com/maps/d/viewer?mid=1E4lJ8kPh4rQKeCokaq3s-AiKZ6Y7z4g&ll=40.74654441000801%2C-73.86569999999999&z=11>





1-1

2-2

3-3

4-4

5-5

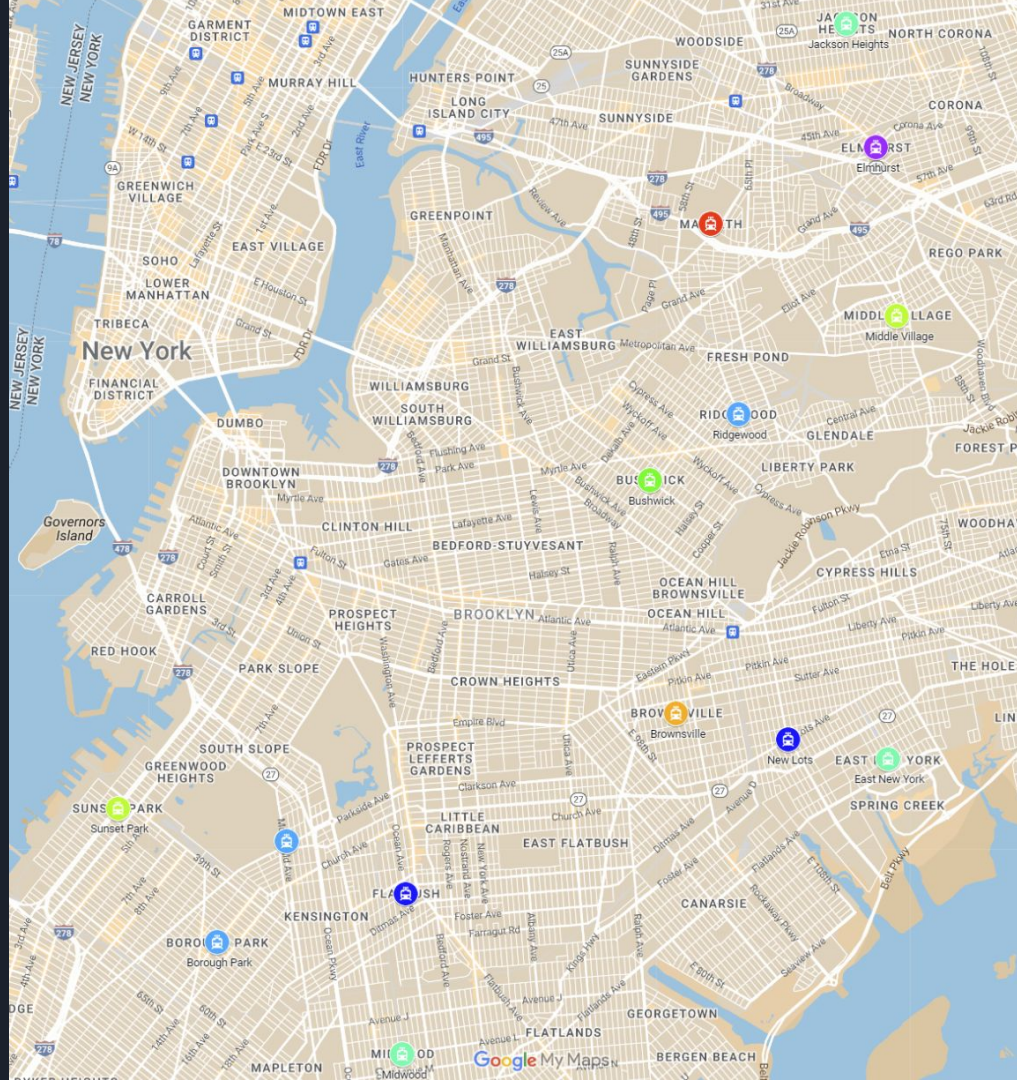
6-6

7-7

8-8



AllBuses.csv





Q59.csv



1 - 2



3 - 3



4 - 4



5 - 6



7 - 7



8 - 8



10 - 13

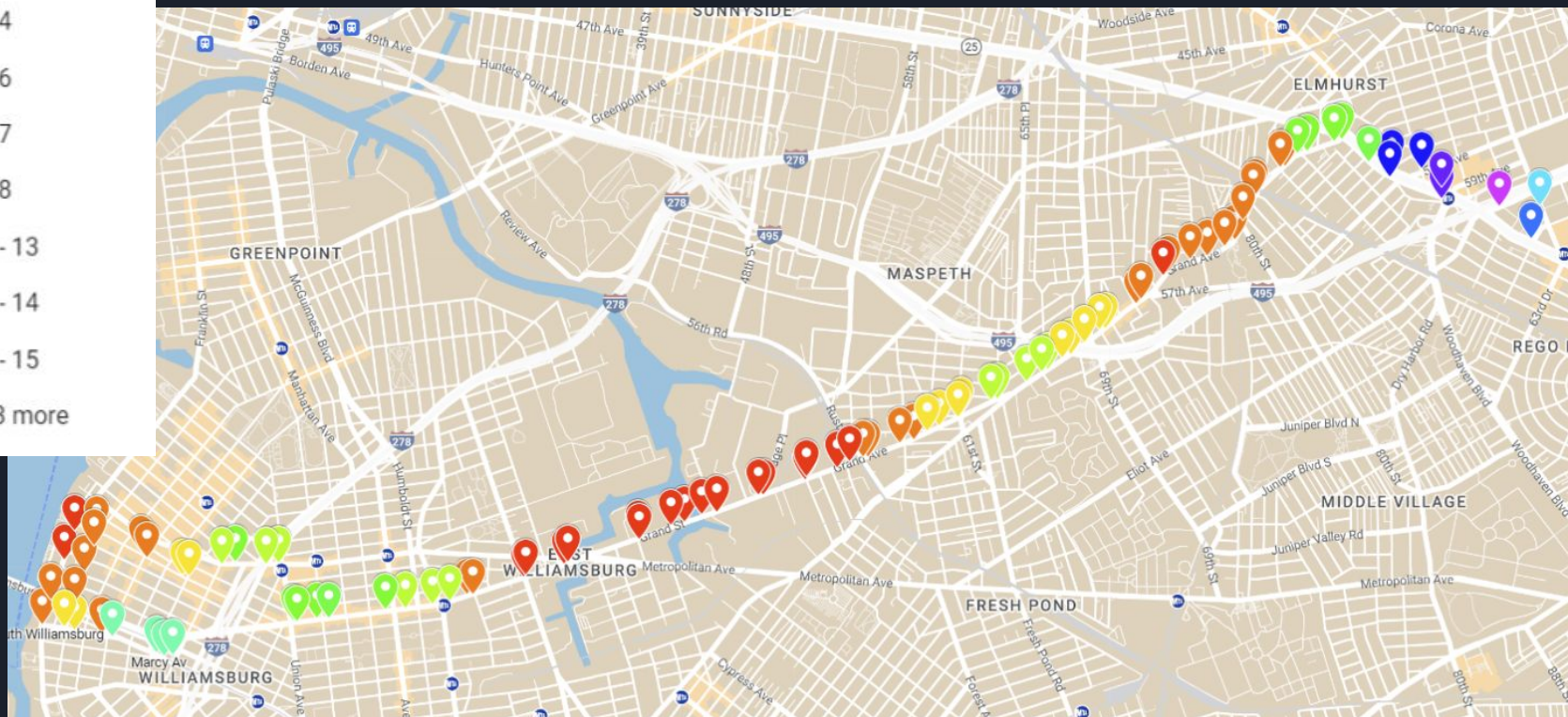


14 - 14



15 - 15

... 3 more





Q110.csv



1 - 4

5 - 5

6 - 6

7 - 8

9 - 9

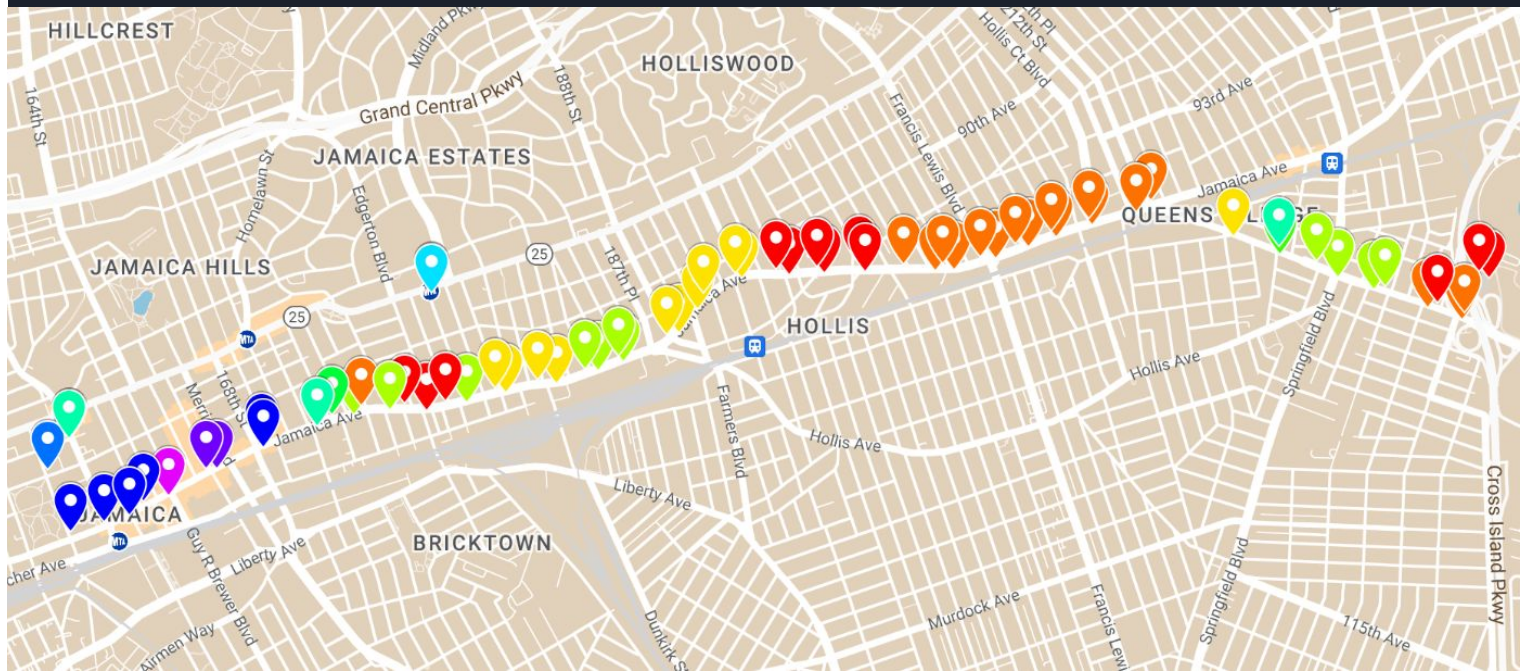
10 - 10

12 - 18

19 - 19

21 - 21

... 3 more





1 - 5



6 - 9



10 - 14



15 - 18



19 - 23



25 - 31



32 - 37

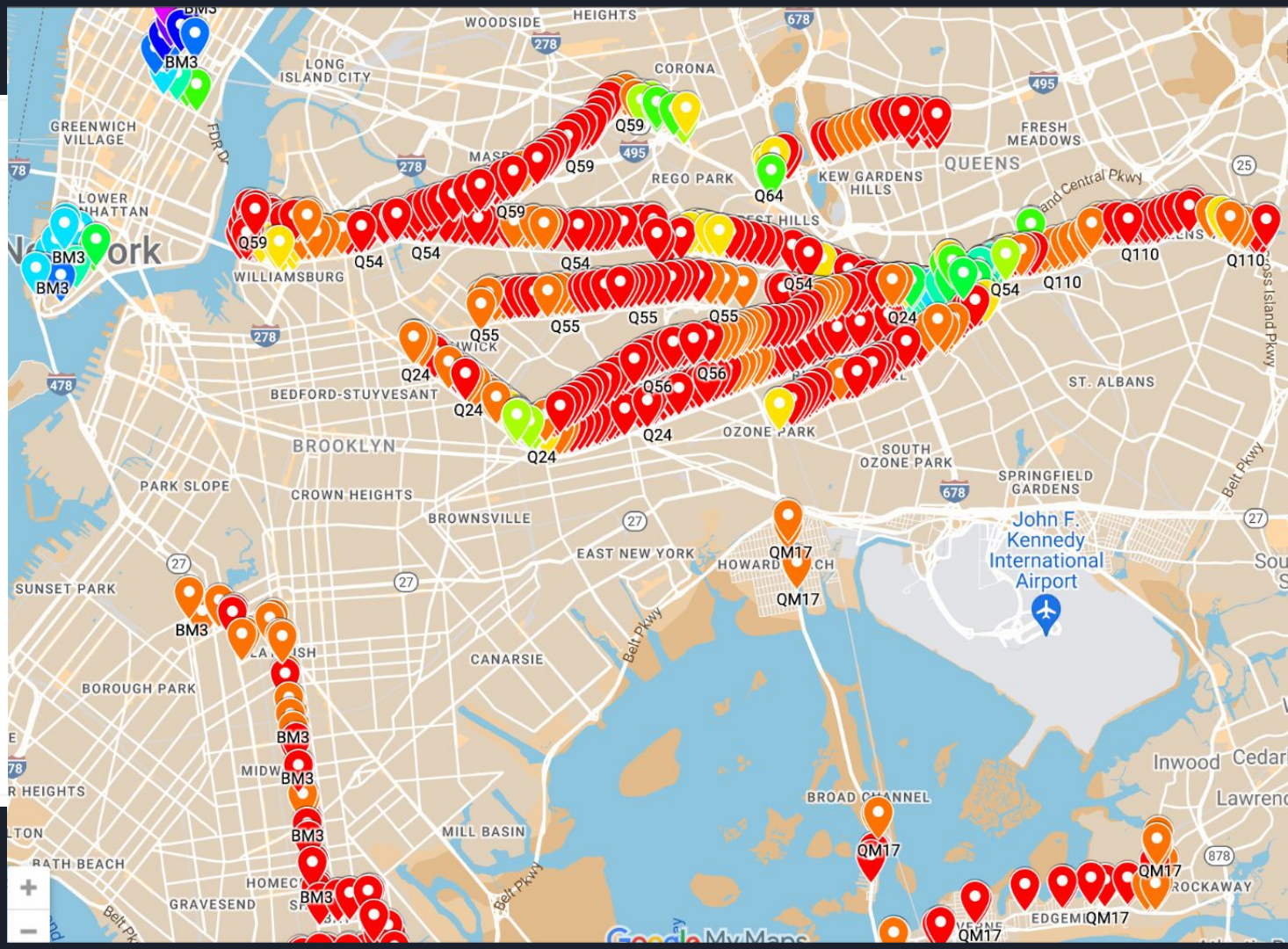


40 - 45



47 - 58

... 4 more





Conclusions

The tests that we ran seemed to corroborate with our hypothesis. From the bike requests in building test, we saw that many locations in Manhattan had a variety of transportation already.

When it came to Brooklyn and Queens, however, the situation was much worse. The IBX map of our .csv results shows that many of the intended stops along the proposed IBX route do in fact have a lack of transportation options. (This supports the proposal to build the IBX!!!)

When looking at some of the bus stops data, we can see that at many times, the singular bus route has stops along the way where there are very, very few alternate modes of transportation. It describes how difficult it is to make transfers in the outer boroughs, meaning that service is not as adequate as it can be.

All this is to say that more methods of transportation would be a great benefit, especially in the outer boroughs. Our program is able to locate dry areas that could use a new bus/train route, or perhaps another bus rerouted to go through that area.



Other use cases/potential for more

- Instead of just bus stops, we can also use the program to analyze subway stops.
- While not implemented, the program has the infrastructure to support getting information about getting from one location to another using public transportation
- Making the program more user friendly in terms of ease of input
- Adding more information to output like if the stop is wheelchair accessible or what routes are part of the same stop/station
- Factoring in the times and/or schedules certain routes are available
- While we fixed some bugs affecting performance, it could be more time efficient



References/Resources

<https://www.transit.land/>

https://safety.fhwa.dot.gov/ped_bike/ped_transit/ped_transguide/ch4.cfm

<https://dev.socrata.com/foundry/data.cityofnewyork.us/scjj-6yaf>

<https://dev.socrata.com/foundry/data.ny.gov/2e6s-9gpm>

<https://new.mta.info/project/interborough-express>

<https://mymaps.google.com/>

<https://www.latlong.net/convert-address-to-lat-long.html>