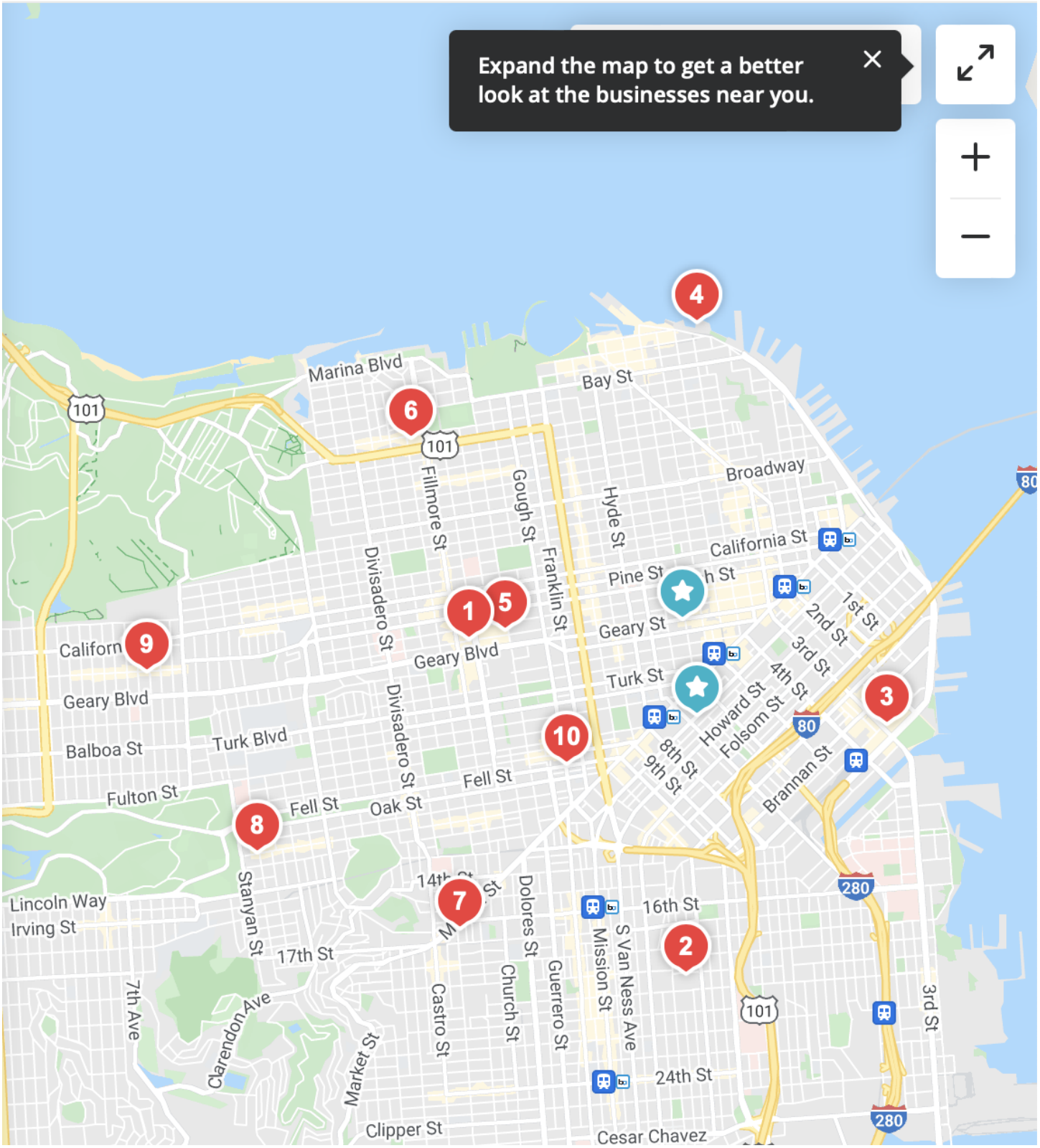
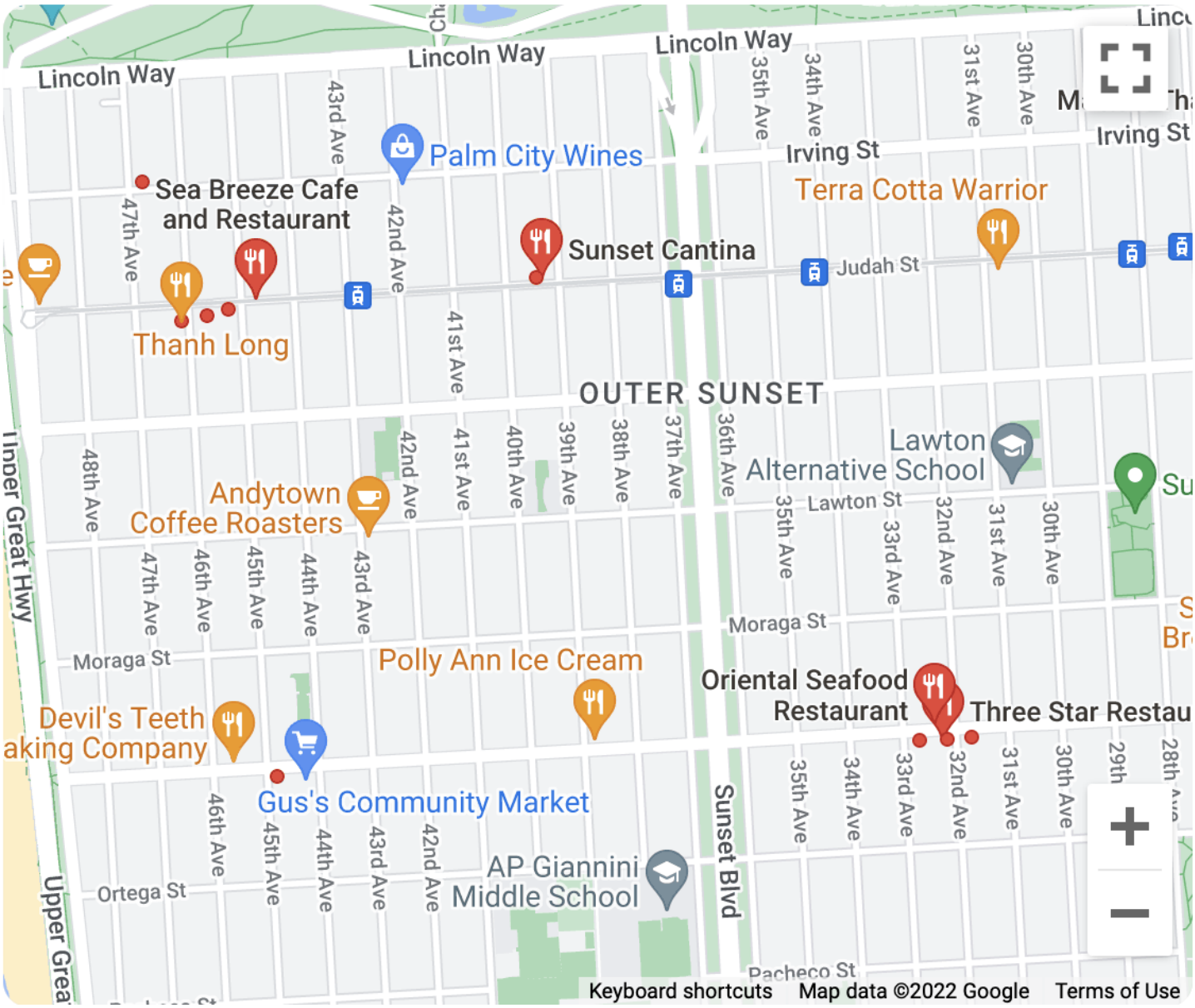
#programming #systemDesign

Real Life Examples

Yelp



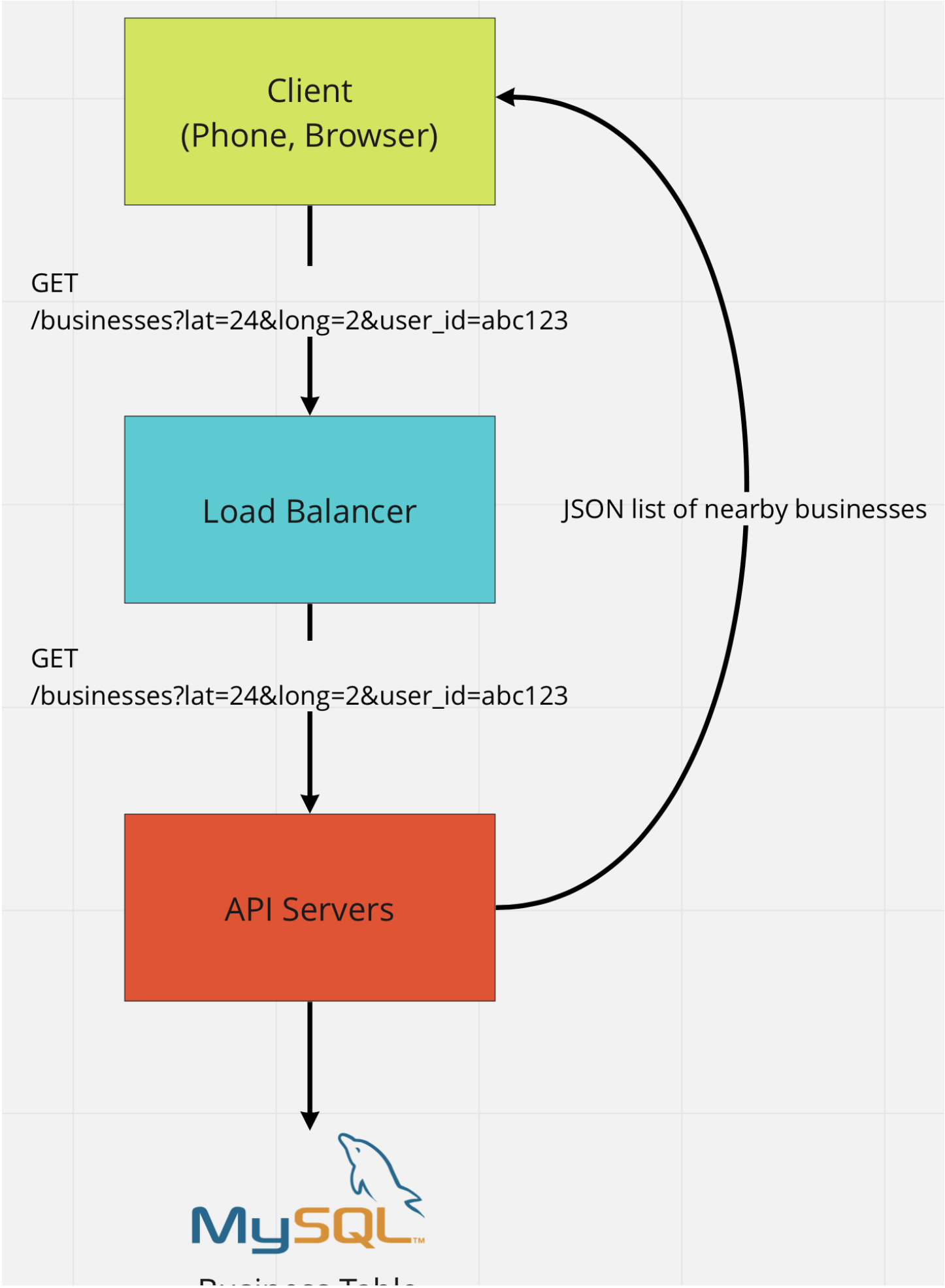
Google



High Level Features

- Given a user's location, return top X points of interest (businesses) near the user
- Filter businesses by some criteria (rating, category, etc)
- Business owners add new businesses

Naive Design


Client
(Phone, Browser)
GET
/businesses?lat=24&long=2&user_id=abc123
Load Balancer
JSON list of nearby businesses
GET
/businesses?lat=24&long=2&user_id=abc123
API Servers
MySQL

Business Table

Database Design

id	name	latitude	longitude
1	Super Duper	12.5	-120
2	McDonalds	10	89
3	Panda Express	5	-60

Example

User Location = (lat, long) = (10, 120)
Find businesses within 5 Miles of user

```
SELECT *
FROM business
WHERE latitude BETWEEN 10 - (:radius) AND 10 + (:radius)
AND longitude BETWEEN 120 - (:radius) AND 120 + (:radius)
```

- This will give you the set of businesses
- Return businesses to frontend

What's Wrong?

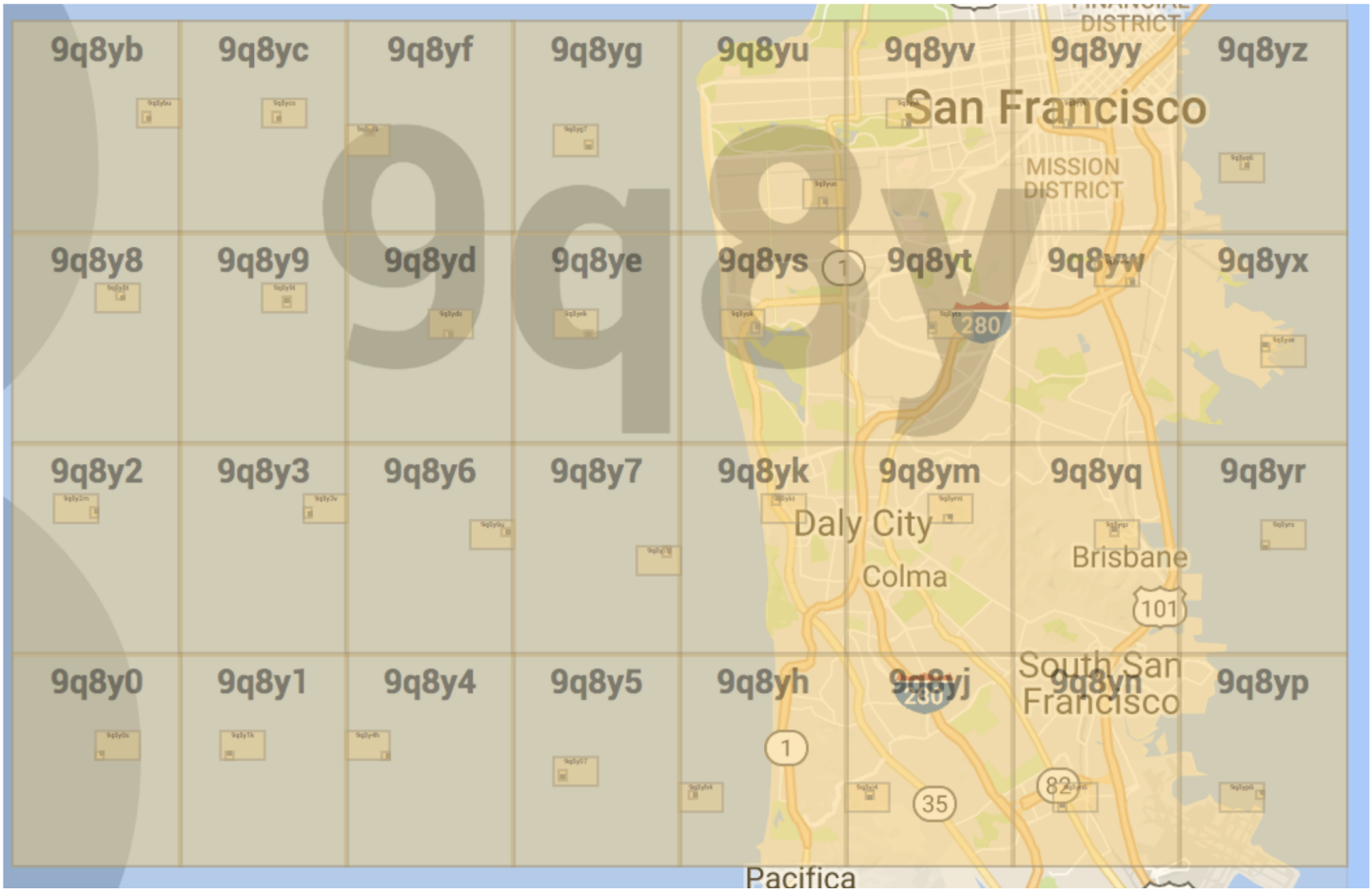
- Query tremendously slow!
- You are scanning almost your whole table
- Even with indexes on lat and long, you will need to scan way more rows then you need.
- Relational databases do not do well with floating point numbers and comparisons
- Doesn't scale with traffic

How can we improve?

- We can use geohashes instead of lat and long for each business

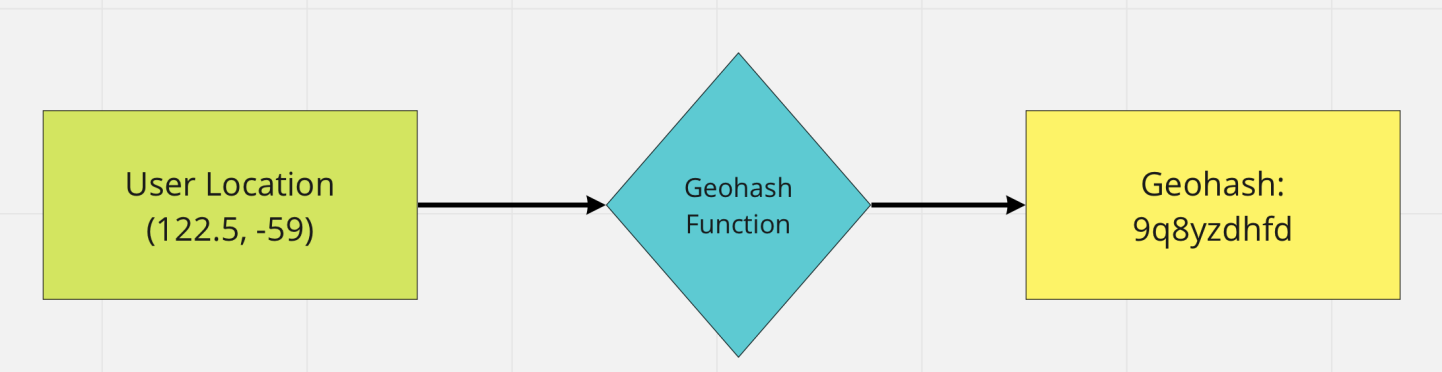
What is Geohash?

geohashing-large-scale-example



Improved Database Design

id	name	geohash
1	Super Duper	9q8yw
2	McDonalds	9q8ym
3	Panda Express	9q8yh



Geohash Length	Area Width in Meters	Area Height in Meters	Area Width in Feet	Area Height in Feet
1	5000000	5000000	16404200	16404200
2	1250000	6250000	4101050	20505249
3	156000	156000	511811	511811
4	39100	19500	128281	63976
5	4890	4890	16043	16043
6	122	610	400	2001
7	153	153	502	502
8	38.2	19.1	125	63
9	4.77	4.77	16	16
10	1.19	0.59	3.9	1.9

Steps:

- Compute User's complete geohash from lat/long
- Decide how many characters to match depending on how close/far you want
- Execute following SQL query

```
SELECT *
FROM business WHERE geohash LIKE "9q8%"
```

Why is this better?

- SQL query performs better

- String comparison much quicker than floating point numbers
- Index on `geohash` for optimization

Can we do better?

- `LIKE` in SQL can still be slow. Ideal would be doing a `=`
- Every request still makes a DB query which can be a bottleneck
- Difficult to scale during peak hours

Solution 1:

- Usually we need only a few miles around the user. For example:
  - 1 mile
  - 5 mile
  - 10 mile
- So we need the following prefixes:
  - Length 6
  - Length 5
  - Length 4

We should never need to match anything more or less than this

So let's just store these lengths directly:

id	name	geohash_6	geohash_5	geohash_4
1	Super Duper	9q8ywa	9q8yw	9q8y
2	McDonalds	9q8ymb	9q8ym	9q8y
3	Panda Express	9q8yhc	9q8yh	9q8y

If you want businesses around 1 mile of the user:

```
SELECT *
FROM business WHERE geohash_6="<first_6_characters_of_users_geohash>"
```

If you want businesses around 5 mile of the user:

```
SELECT *
FROM business WHERE geohash_5="<first_5_characters_of_users_geohash>"
```

If you want businesses around 10 mile of the user:

```
SELECT *
FROM business WHERE geohash_4="<first_4_characters_of_users_geohash>"
```

For best performance:

- Indexes on all 3 geohash columns
- READ should be very quick now
- WRITE will be slower, but it's acceptable given lack of change

Solution 2

Still every request hits the DB, which is slow.

Especially during peak hours DB will be overwhelmed.

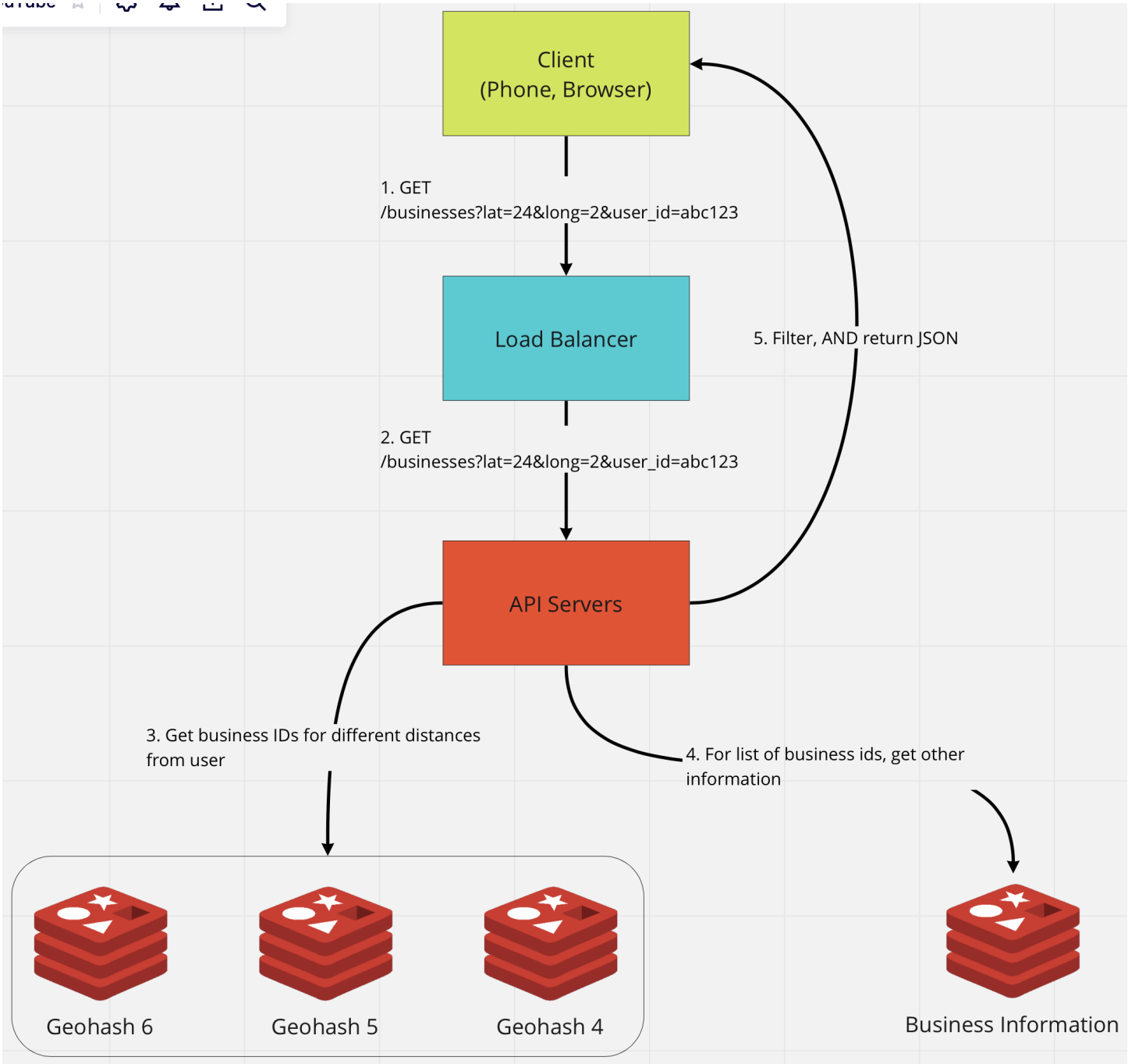
Can we do better?

Add Caches

We can add 4 caches:

- Business information
  - `120: {"name": "Burger", "country": "US"}`
- Geohash Length 6
  - `"9q8ynf": [10, 20, 30 40]`
- Geohash Length 5
  - `"9q8yn": [10, 20, 30 40]`
- Geohash Length 4
  - `"9q8y": [10, 20, 30 40]`

How Does it Look Like Now?



What does the API server do?

1. Compute geohash of user
2. Gets list of businesses from Redis cache using different prefix depending on distance required
3. Get other business details from Business Information cache
4. Does any filtering required by rating, category
5. Returns JSON list of businesses to the frontend to be rendered

What About Adding a New Business?

- Compared to getting businesses, creating businesses should be orders of magnitude less
- Business locations are more or less static
- It's not super important to start returning new business immediately
- Can afford to do things asynchronously