Functional Requirements:
- Find out the business (like restaurent) near me → as of now figure for a fixed radius → if possible extend to → extend if not in that radius.
- business can be added , deleted etc → no need to reflect in real time.
- customer can view the detailed information about a business.

Non-Functional Requirements:
- Low latency → in searching
- high available and scalable systems

Capacity Estmations:
- QPS → Read business
- QPS → CRUD business
- Count of business ? and its growth → would help about the database scale etc.

1. High Level Design 2. Deep diving

api designing:

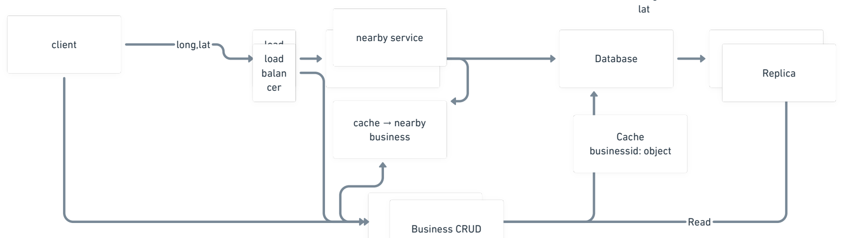
/api/business ? long = '' && lat ='' → list of nearby business objects → which contain list of lat and long of the business

Remember we are not going to focus how the UI will show those long and lat in google map

it will only search form cache → because delay is fine (rem cache only require when database not able to read fast → sql have lots of data or have some 80 | 20 criteria)

Business Table:

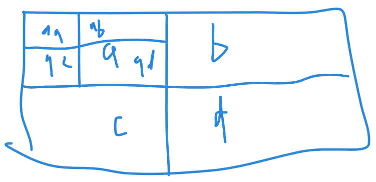
businessId,
metadata,
long
lat



Possible Solutions for finding nearest business
1. do sql query about the business , such that their lat and lon are with in that radiuis
   1. no much efficient → while search → find out the all lat first with logn → again long with logn → multiple and find out the distance, if we some how change these 2d to 1d, that would be efficient to search
2. Change 2 day location → one dimensional → Geohashing.
   1. geo hashing → dividing the whole map → into let day 4 matric → with some value as string 'a' , 'b' , 'c' and 'd'
   2. then those matric → will again divide the grid into 4
   3. this is how → we define upto very good prevision → 12 → within in 10 meters
   4. we can select the grid → according to radius and also bit smaller as well

      business id. → geohash.

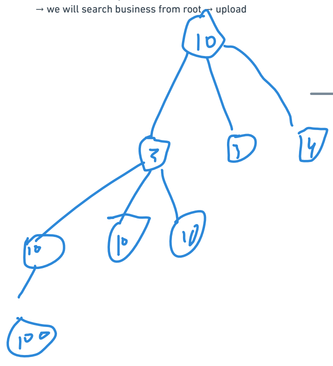
   5. select * where geo hash like '<usergeohash>%' .
   6. since nearest location will have share prefix.



select * from business
where hash like 'a%'

Cons:
- Corner cases → in above case if a is on boundary → b business won't come
- solution → find all business near to current grid and combine them with location

3. Quad Tree index inmemory
 - We are going to divide map in tree → node will have no. of business
 - so like root will have total no. of business
 → will divide → upto leaf will have 100 business
 → we will search business from root → upload



I think redis provide geo hashing also with predefined functions

```
Node →
{
  geohash → it should be long and lat → so that we can get to know the distance from given location to
the node location and only go in nearest node → but what will be long and lat ? would it be centre of
that structure ?  No → it will be top left and bottom right of the rectange.
  list of business Id
}
```

Algo:
- start from root → search until leaf and that location → if more than require restaurents → return
- else go to parent , figure out nearest and sum up and check and return.

Pros:
- if we want more → we can also get that easily

Cons:
- updating is tough → and built also would require bit time → because of data given.
- solution → while update in business → we can crating another tree parallelly and once that complete deprecate this → there would be delay in showing business. But this is not much impactful as per requirements.

Table:

Business → id, hash, or hash: [id1, id2 ...]
but second one is bit problematic to write

For business we would require cache
Also geo has we would require cache