DESIGN GOOGLE MAPS (Navigation system)

NFR

- High Availability
- Good Accessibility
- Not too slow

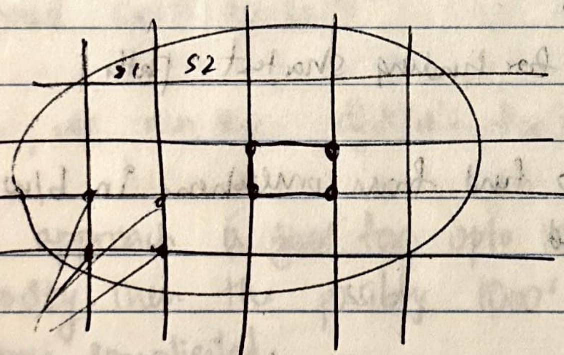
FR

- Identify Roads, Routes
- Distance & ETA b/w 2 points
- plugable

* Billion MAU
* 5 million companies use it (eg:- uber, ola etc.)

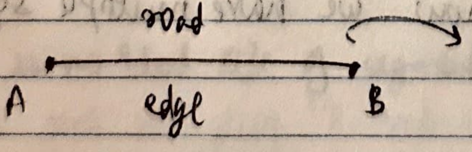
say there is one city, and each city divided into segments.

Each segment will have an identifier.



These co-ordinates are nothing but (lat, long).

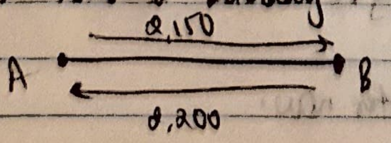
We model the road network like a graph.



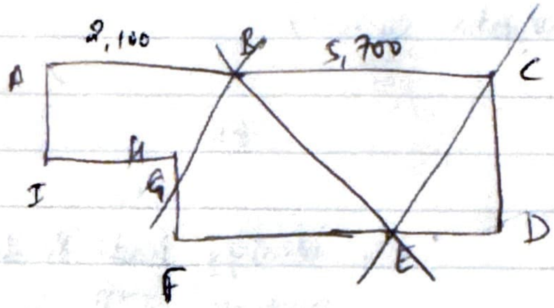
2km, 150ms ↙ say to 150s to cross the road.

↓ each road will have weights

This road is actually a directed graph



for different traffic pattern etc.
and maybe say the road is just one-way.



say this is a graph of all the roads in a segment.

Say we want to go from A → C

A – B – C

A – B – E – C

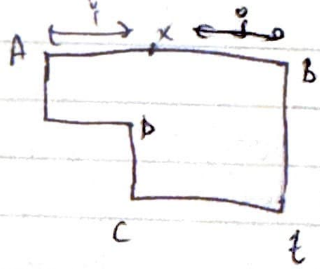
A – I – H – G – F – E – C

A – J – H – G – B – C

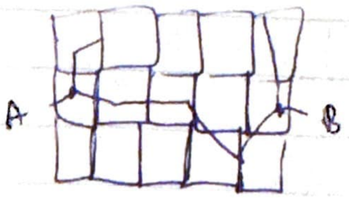
all possible distances in the graph.

Floyd-warshall algorithm for finding shortest path.

Now, say if we want to find from somewhere in b/w, some random point in the road



say now we have multiple segments



given lat, long, we will be able to calculate aerial distance b/w A and B

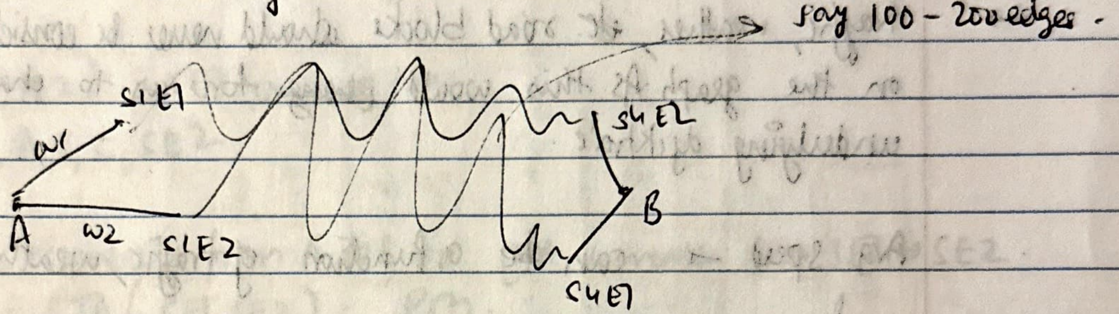
say its 10km aerial distance for now.

And there can be a million ways to reach A → B.

→ so, discuss and choose a segment buffer, meaning say

→ I choose buffer as 20, meaning 20 segments up, down, left, right. Meaning we can consider only 20 segments in and around and choose a shortest path accordingly.

look at exits from segments



→ Connect exits to exits.

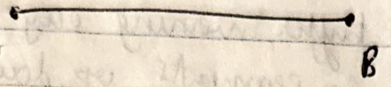
→ Now, we run the djiktra's for shortest path.

→ This approach is good for upto 10km. But say we want to travel intracity. Then the possibly 1000's of segments. Then djiktra's becomes complicated.

→ So, maybe we can create a megasegment. Any we can divide the counties into mega-segments.

→ There is no ideal size of segments (or) mega-segments.

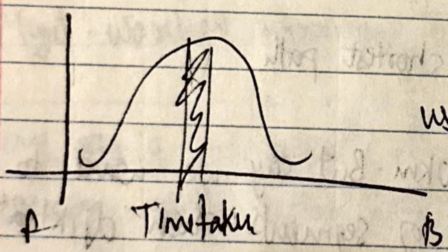
→ Even bigger routes (or) distances we would need more abstractions and more bigger segments.



How to come up with weights on an edge?
What all to consider.

→ traffic, weather, ~~etc.~~ road blocks should never be considered as weights on the graph. As this would ~~chang~~ force us to change the underlying djikstra's.

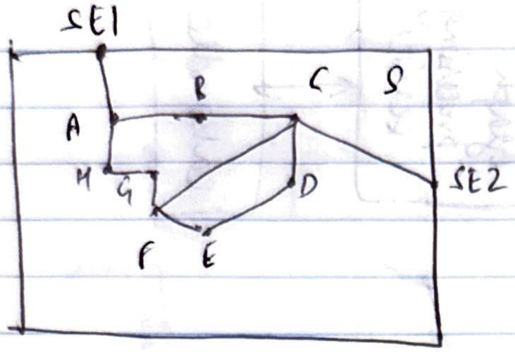
Avg. speed → can be a function of traffic, weather etc.
↳ so we can consider this as a weight.



usually normally distributed.
(organic data).

Traffic	Weather
low	Bad
med ↓	Good
high	

Then based on traffic data received from 3rd party, we keep updating the weights recursively till exit t2. So we keep updating ETA on all roads.



Now, say SE1 & SE2 are exits of the segment.

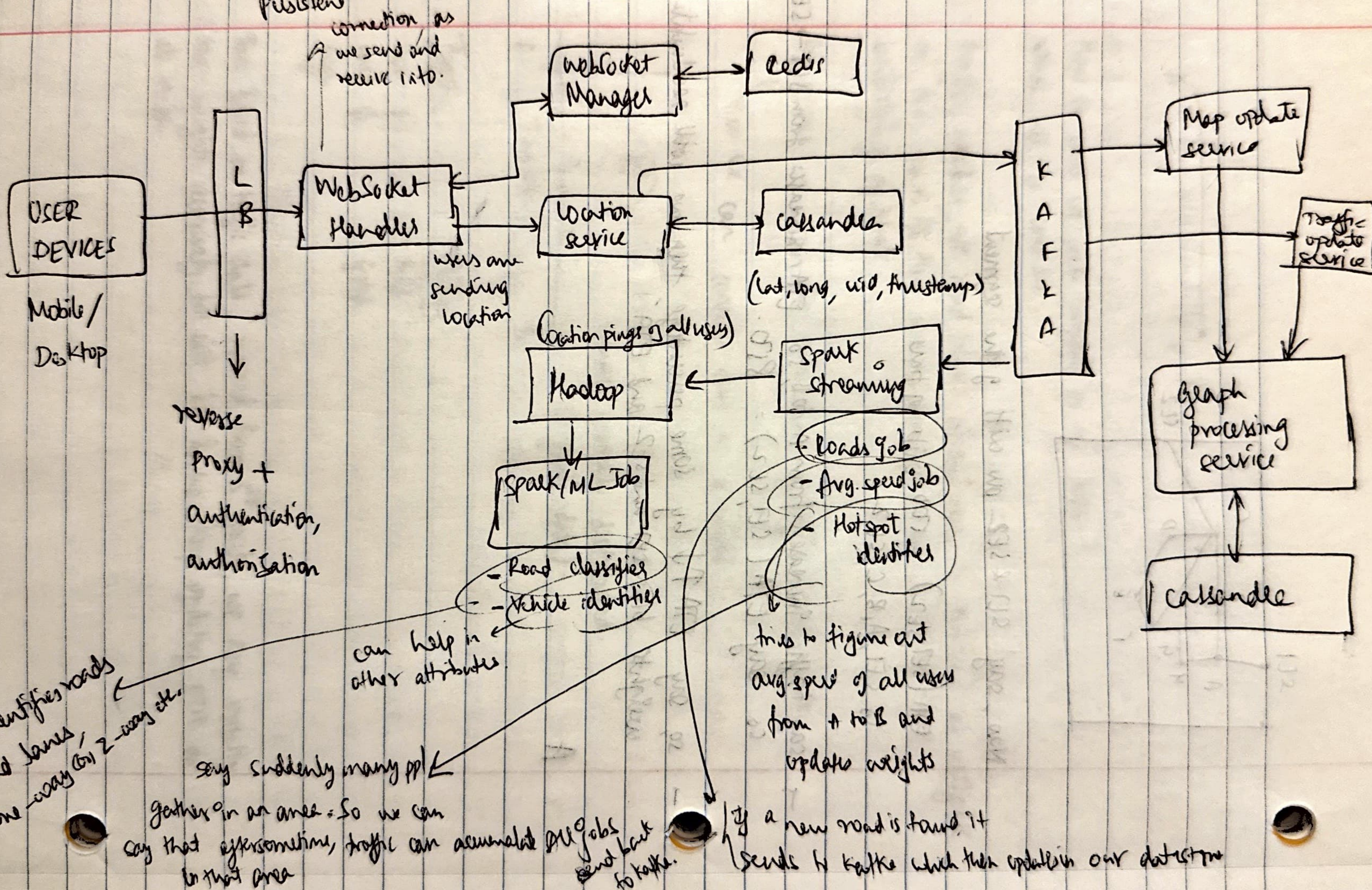
ETA (SE1, SE2) = 500 units of time
↳ SE1, A, B, C, SE2

→ say traffic increases from A to B, so ETA increases from SE1 → SE2.
so say ETA (SE1, SE2) = 850.

→ so, say ETA ↑↓ by some percentage then we will recalculate the weights b/w SE1 and SE2 and ETA.

A

ARCHITECTURE



Persistent connection as we send and receive info.

users are sending location

reverse proxy + authentication, authorisation

can help in other attributes

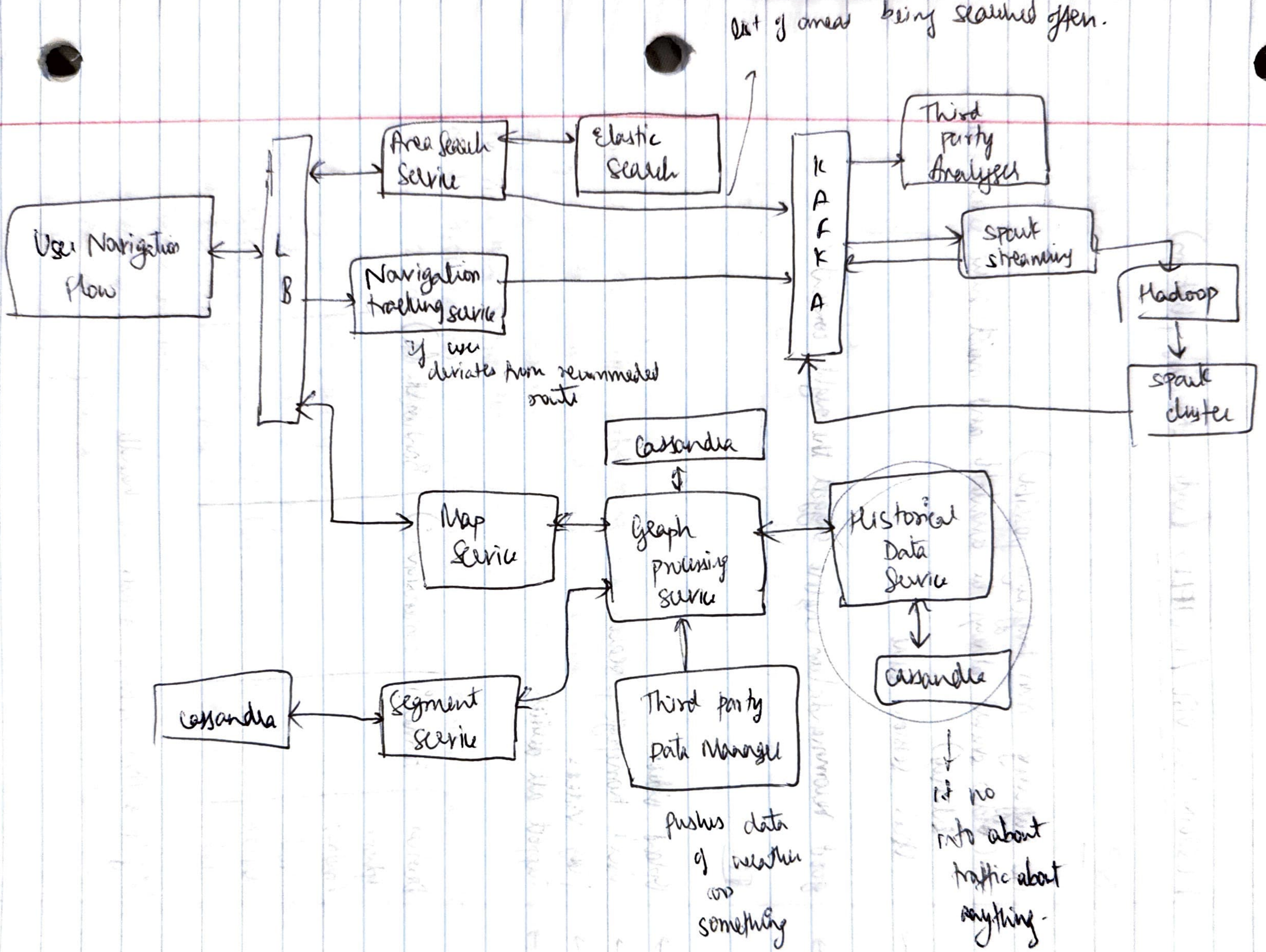
Identifies roads and lanes, one-way (or) 2-way etc.

tries to figure out avg. speed of all users from A to B and updates weights

say suddenly many ppl gather in an area. So we can say that after sometime, traffic can accumulate in that area

All jobs send back to kafka.

If a new road is found it sends to Kafka which then updates in our datastore


list of areas being searched often.
User Navigation Flow
L B
Area Search Service
Elastic Search
Navigation tracking service
If we deviates from recommended route
K A F K A
Third party Analyser
Spark streaming
Hadoop
Spark cluster
Cassandra
Map Service
Graph processing service
Historical Data Service
Cassandra
Cassandra
Segment service
Third party Data Manager
pushes data of weather or something
if no info about traffic about anything.