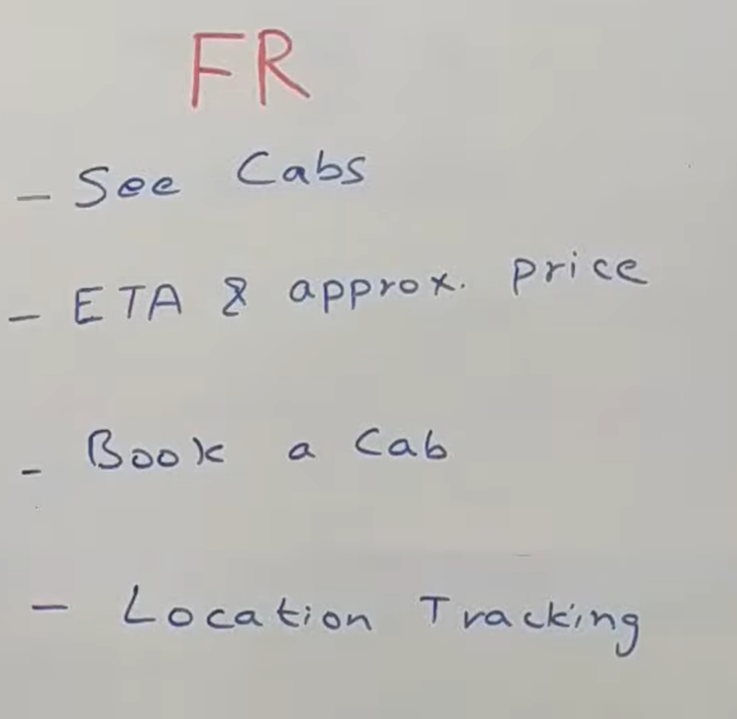
                                   2.6 Uber/ Ola  System Design

Requirement :

How  to design  a cab booking system  like uber or ola .

Let start with functional and non functional requirement that system should support.



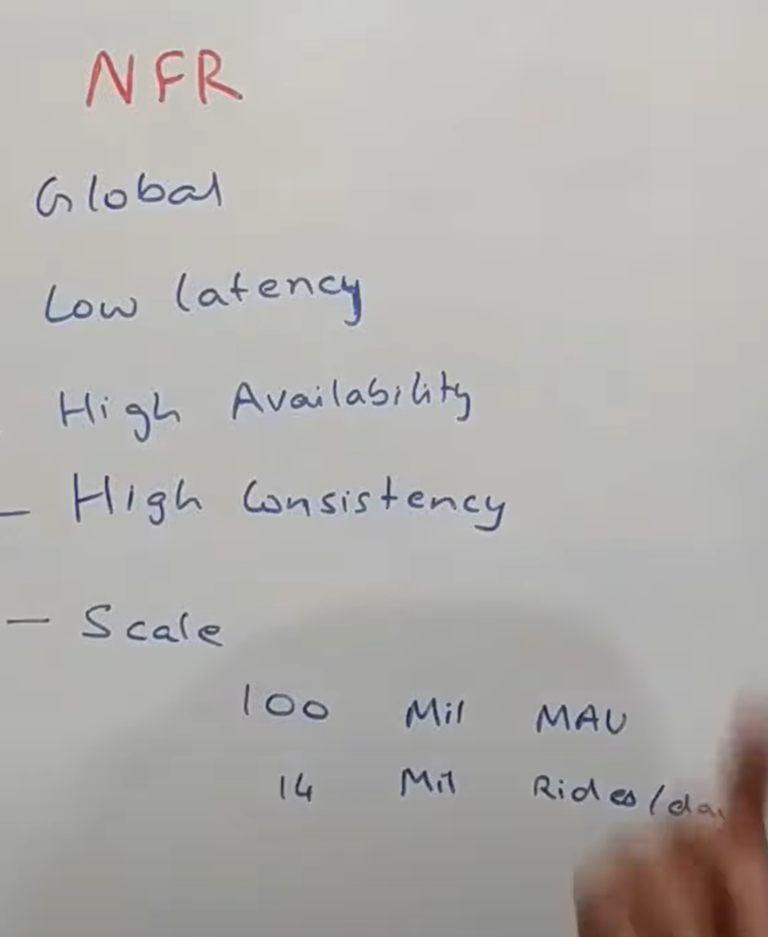
So the very first thing as a customer  when we open app we could able to see what cab around us so that is see cab feature in your vicinity( in the  area that is close to a place).

The next thing is if we want to book a cab from point a to point b we  should know how much time it will take to travel and we should also get approximate price for this travel.

We should able to book a cab.

There should be location tracking at what place driver is  at what point in time for various reason.

**NFR**



The platform should be**global** it should be accessible  to people of all countries.

It need to work at fairly **low latency**. It should be fast.

**Availability**should be very high. The system should not go down. It would cause a lot of problem to people who are stuck somewhere.

It should have**high consistency**.

High availability and high consistency might seems like to violate **CAP theorem** which says that : assuming all the systems in the world are distributed nowadays  so out of the availability and consistency you can just get one . The idea is : certain components of this need to be very highly available  and certain other component need to be very high consistent not both at the same time.

**From a scale standpoint**  this system should scale to a very good number.

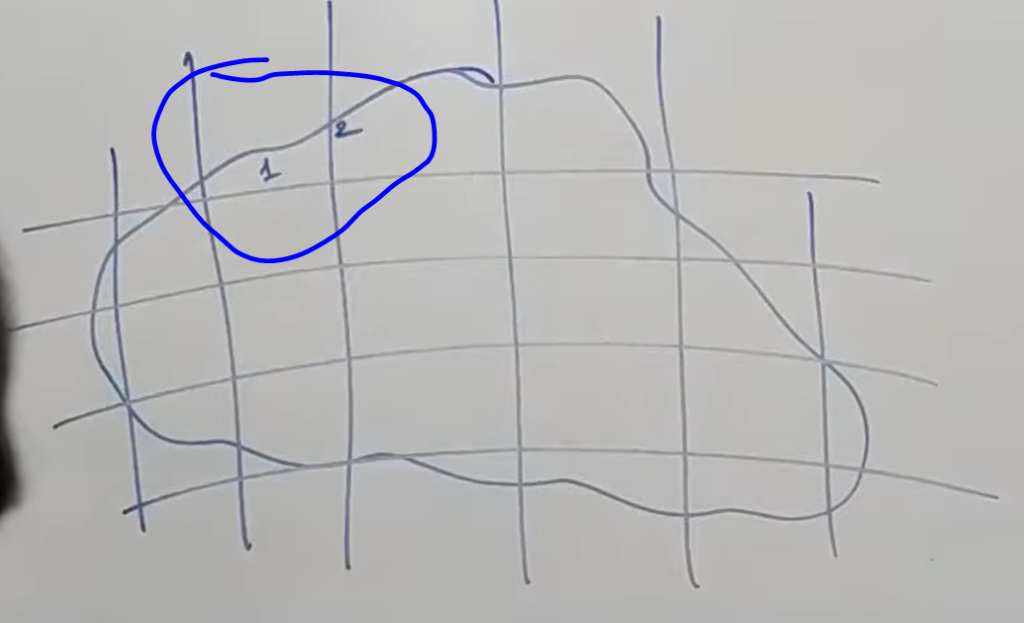
There are roughly 100 million active users that uses uber on a monthly basic. These are unique users. Roughly does 14 million rides per day so with that thought in mind let try to look at a system  that can scale up to these number with these criterion.

**Now the main problem that companies** like to solve when u have **customer location**who want to book a cab. You try to find out some few driver who are very near to this location  and then using some logic  try to come up with best driver who is suited to do this trip for this customer.

**So the problem then becomes how to find these 2 or 3 closest driver to this customer**

**Th**ere are multiple ways to do that we will go over one of the ways which uses a concept called **segment and mapping segment** basically. This is not a industry standard term.

Now let say we have a city. The idea is we basically divide it into rectangular segment. So we divide city into multiple pieces and we say that this is probably  your segment id 1



City is normally divided into more segment what we see here.

Now the idea is given certain coordinates of the segment boundary and given certain coordinates of a cab. We should be able to figure out  which segment does a cab belong in.

Now the problem look trivial and it not difficult also.

So think like a standard coordinate system. This is (0,0),(0,1),(1,0) and (1,1). Now if a point lies somewhere in b/w  let say (0.4,0.5). You should be able to mathematically say  this point lies within this boundary .



A very similar logic we will try to use when we try to assign a particular segment to a cab.

Also keep in mind cab are continuously moving  and their location are continuously  changing

So we will try to make sure  that we get continuous ping from all the cabs and then keep track of which segment do they belong in. When the cab driver change the segment then this info

Would be calculated at runtime as when we are getting ping from the cab. So we have something called a **maps service : this maps service** will do a couple of thing

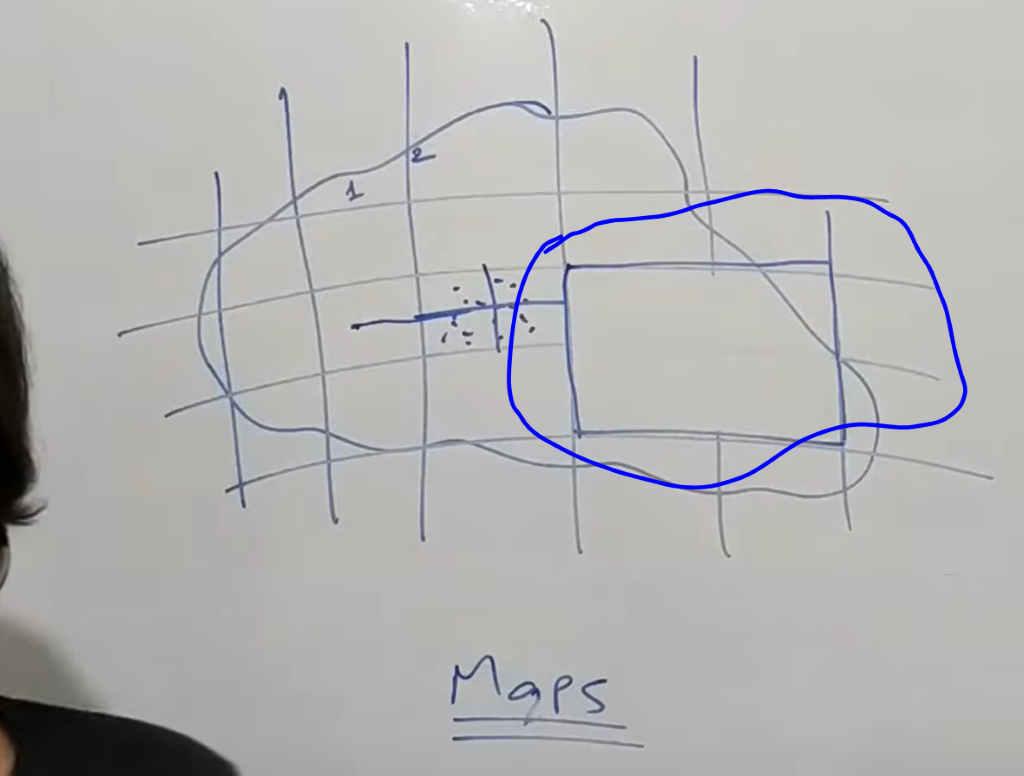
1:- it will be responsible for dividing the city into these segment and take care of these segment.

2:- the other thing it will do is : given a lat long of a cab and given a lat(latitude) long of customer tell which segment user belong to at this point in time.

**This service will also calculate**  ETA from point a to point b  and the route from point a to point b and thus even the distance. But we will abstract out . We will not go into details  and how that ETA and distance piece is implemented. We will see this into google maps

**There is one more** thing this service does let say there is huge amount of traffic or huge amount of cab driver in this particular segment and it is getting unmanageable so the idea of segment is :- it should be small set of driver that are in the segment. So this service will take care of dividing the segment into multiple parts . It could do it into 4 parts. It could divide it into 6 parts. That logic resides within map service.

Now let say there is very less traffic  in some other locality so it could also decide  to merge a couple of segment into one segment  and say this whole thing is now one segment.



So all the segment management remains within this service.

**Now lets look   at the overall architecture  and how individual user and driver get connected to the system**



**All** the users get connected to thi**s user app** which talks to a load balancer  which talks to something called a **User Service.** now this user service is the repository of all the user  info.

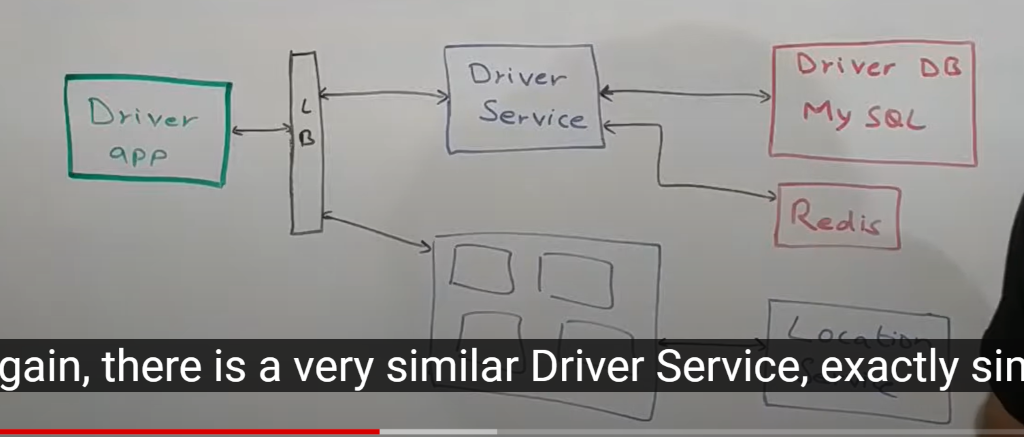
Plus it is also a proxy that connect to other service  to get any info that user want. So for example if a user want to see their profile , update their profile all the apis to do that are powered by user service. If somebody want to fetch user info  any other service for example then all the api for that are powered by  this service.

Let say if a user want to see their trip  then this user service  will talk to trip service  fetch all the trips for that user  and send it back to the user so that the responsibility of user service.

From the **d/b standpoint** it sit on the top of mysql cluster which store all the user info within that and it also uses **redis**for caching the same info.  so let say  a GET api to get a user info is called it first queries the redis if it has info it return from there and if it doesn't have  info then it queries a mysql slave fetch that info  stored it into redis and then return back to whoever was calling.

The next flow is user call is basically **when they try to book a cab.** The whole screen  that the customer kind of  goes through when they are trying to book a cab is powered by this **cab request service : esse**ntially what it does is :- it basically makes a web socket connection  with the user app which display them a few cabs onto their UI which are around them. Also it places a request with something called a **cab finder** . We will go over this in the next section. Whenever cab finder response back with a cab  this cab request service talk to user app basically send them a response through this web socket connection saying that the cab is booked  and these are the details and whatever is required. **This is approx. Major user flow**

**Now quickly look at driver flow**



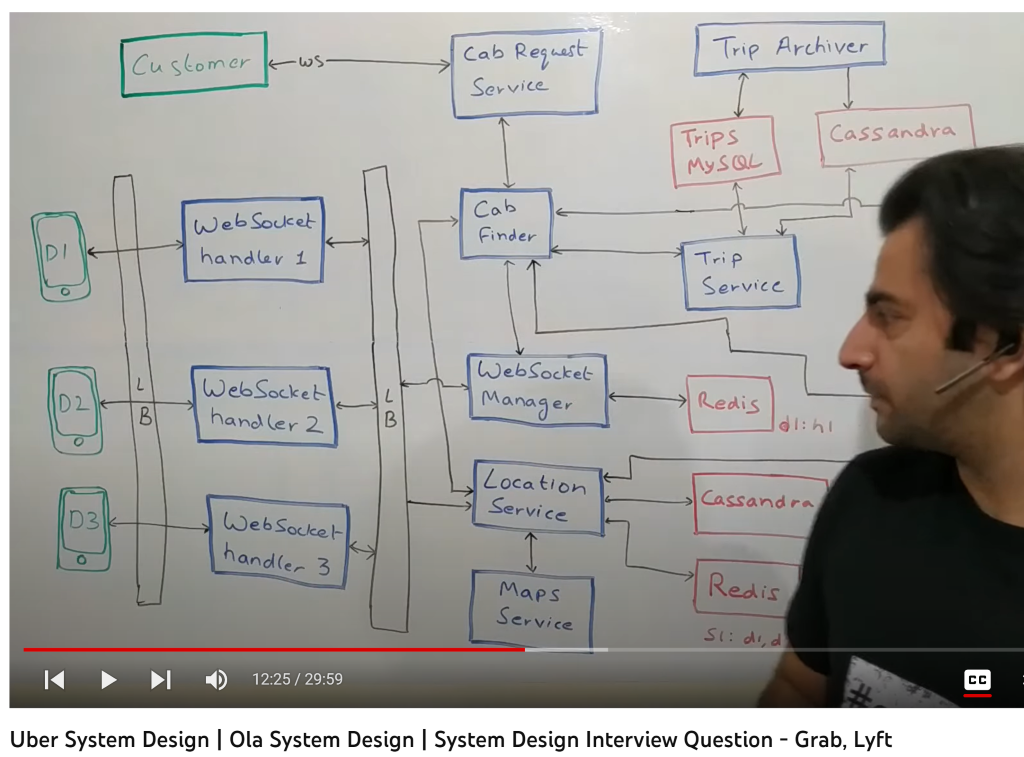
A driver basically talk via a driver app  and again there is**driver service** exactly similar to the user service. But for driver again all the APIs for getting , updating  all the driver info is powered via this. If a driver want to see  their payment info for example  their payment history this driver **service will expose an api** which the driver app will call  and driver service will internally call a **payment service** to get that info and response back. It could call **trip service to** get trips of all the driver  and all the UI data get powered by this service. This service sit on the top of another **mysql**which has driver info and it uses **redis**for caching in exactly the same way as **User Service** does. This driver app will also talk to something called a **location service** through a **series of server** again maintain a web socket connection with these server. And as when driver is moving through the city every 5 second or 10 second their location is send out  to this **location service** which queries then**map service** that we talked about earlier  to find out which segment does this driver belong to.

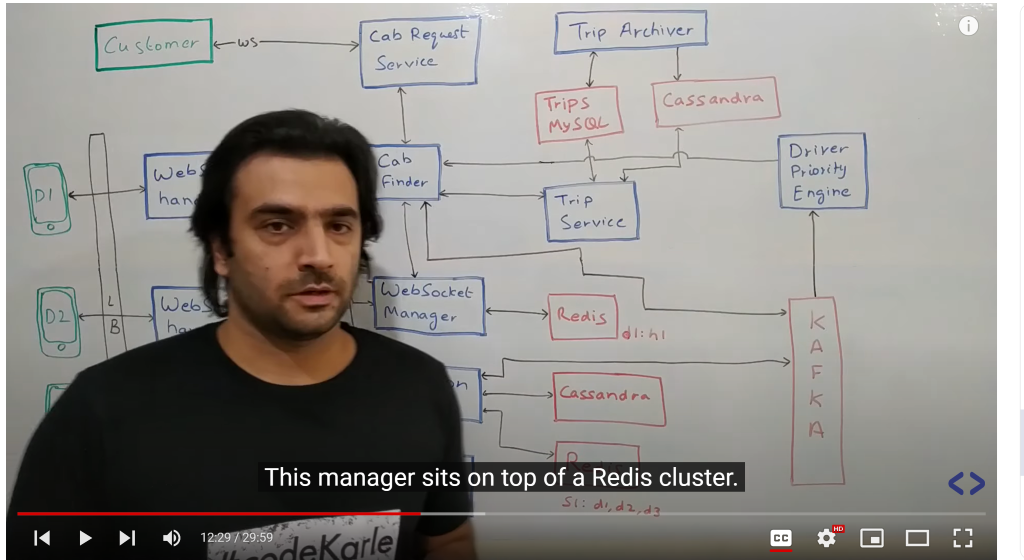
**And when the customer places this cab request**  customer segment is calculated , driver segment is calculated  and they are mixed and match by the **cab finder and** a couple of other component to give the best suited driver for a particular trip.

**Now lets look how does a customer and driver come together**

Diagram

Description automatically generated





**So** all the active driver in the system who are online right now ready for trips and all are mentioned as D1, D2and d3. All those driver each of them is connected to one server  through a web socket connection. And those server are called as  webSocket handler 1,  webSocket handler 2 and  webSocket handler 3. **now why we do need websocket here.**

**So** we always need a connection b/w a driver and a service  for lot of reasons. One of the very first thing driver continuously send location ping to the backend  telling about the location. Now if each time they start creating a new connection  that kind of heavy operation so we will have this connection live. Also at time the server might want to talk to driver so let say if a trip is assigned to driver we need to inform the driver so we can reuse  the same connection to talk to driver  and tell them this is the trip info that u have to do right now so for all of that there are **websocket handler server.** in the real world  there will be hundred of such server  who are interfacing with all the driver which are throughout the world geographically split.

Now let say somebody in the system identifies that trip is being  given to a driver  and to reach out to the driver they first need to know that which out of these  hundred of  webSocket handler do I need to talk to  for that there is something called a **websocket manager. Now** this websocket manager is another distributed service which manages the fact that which server is connected to what all driver. So let say d3 is a new driver that has come online  and through the load balancer it connected to  webSocket handler 3. so this handler will inform the websocket manager saying now I have got connected to d3 if there is anything for d3 inform me. And this manager will store in its database. Noe let say this connection got broken and d3 is offline right now and again this handler will inform the manager  that d3 is now offline do now reach out to me for any communication of d3.

Now this manager sit on the top of**redis cluster**. This redis  would not just be storing data in memory it will also be storing it in a persistent store on disk. And  it will basically store two kind of mapping. One is  saying that the  most frequently use one which driver is connected to which host. Similarly there will be  an entry for each driver  in the system saying  which driver\_id is connected to what host\_id. It will also have a reverse mapping saying which host id  is connected to what all driver id.

Coming to other thing that websocket  Is used for so these driver/ devices send location ping to our backend let say every 5 second . So every 5 second we get a hit about the location info. All the location related info is managed by something called **location service . It** does a lot of thing. On of the thing happen here  it store the info abt driver location into **cassandra**. Why cassandra  because again there are million of driver across the globe who are sending their location updates every 5 second. So there are lot of updates happening so cassandra should ab able to scale up to that number. That’s the main reason. There are two kind of info that get stored here. One is live location of the driver which is the last known  location. The other thing that is stored is while a driver is doing a trip with a customer we need to know exactly what was the route followed for any auditing purpose or billing purpose. So all of these are basically responsibility of **location service. Now** loacation service also talk to**map service . Map** service is a service that maintain the segment we have created  throughout the city  and throughout the globe so map service not just maintain the segment  it also give us the ETA (estimated time of arrival) which will be the  time taken from point a to point b  and the distance that will be followed  and also the route that should be followed from point a to point b. think of map service as an abstraction we have we will not go into the details of implementation of map service right now. **Design of google map will tell how map service is implemented.**

 But that being said location service as soon as it get a ping from a driver  it basically query map service  and tries to figure out that this lat(longitude) long belong to which segment. It then store it into redis saying this segment has these drivers. This update happens only when a driver segment changes . If he is in the same segment then no changes happen. This basically is use for lot of purpose  so let say we find out a driver in a vicinity will query this service  who all are driver d1,d2,d3 that are basically in segment s1. There is one more thing that map service do. It basically keep a mapping of  which all are of segment surrounding a particular segment which we will come to a while on how it been used.

There is something called **trip service which** is basically source of truth for all the trip info.it sit on top of mysql d/b and a cassandra d/b. It uses mysql  for all the live info basically  info of all the trip that are about to happen in some near future or in progress. Once the trip is completed then it can basically move to cassandra. Now why we don’t store all the info in mysql because over time  this will become a very massive amount of data.plus if it is just for read queries cassandra is also good enough so we don’t really to store in mysql d/b. The main reason for storing in mysql d/b because trip would  have a lot of info. It would have info about the driver,customer , potential start times, end time, about the potential distances about the real values , payment info. Now if u look in table terms  these will be a lot of tables and for each event that comes in against a trip we might need to update a lot of such tables. And there it is very good to have transactional property so that’s the reason will be using mysql for all the trips that will be updated.once the trip is compledted we can move it to cassandra. Now this movement from mysql to cassandra is taken up by this **trip archiver service** which basically is a cron which spawn in every 12 hour and ull the data from mysql and put it into cassandra.

Coming back to trip service . Trip service will expose all the api around trip. So if u want to get a trip by id or if u want to get all the trip  of a particular driver  or all the trip of a user all of those api would be powered by trip service and lets say  if it search by a driver id it will query mysql, it will query cassandra  it will get the result from both of them it will merge them and then return it back whoever was calling so that how it flow would be.

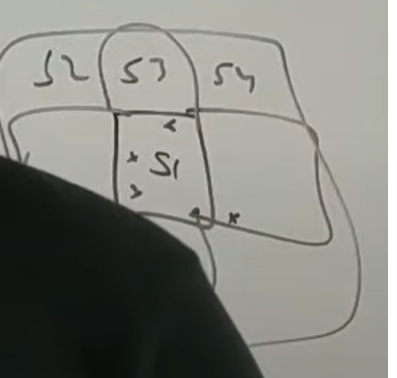
**Now let go into the main flow what happens when actually customer make a request**

**The** customer flow begin at this point where they basically make a request to cab request service through again  an open connection b/t both the parties  the customer and this service .

And basically what the customer say this is my **source lat long .** I need to go to a destination which is identifies by a certain lat long . Get me a cab. But given this request  c**ab request service** queries something called as a **cab finder which** is responsible to come up with that one driver who will do the trip . at the end  of all of it cab finder  will respond back to cab request service  saying  this is your trip id and this is your driver id  go send back it to the customer in a nice form wth all the details  about the driver and all of that and can service would send that. Now cab finder will also put a notification into a kafka whether or not it was able to find a driver. Now lets say it would not able to find a driver  may be because there is a scarcity of driver all of that would go into kafka which can be used for further analytics like for example telling driver that this is a location where there are more customer and less driver.

**Now coming to what cab finder does**

**The** very first thing cab finder would do is basically  it has a source lat long  which is basically identifier of  a particular location of customer. It will first query the **location service** saying get me the segment in which this customer currently is  along with that get me list of driver  that are near this customer. What it does is : it first of all queries map service  with the lat long of customer to get the segment which the customer is in. It then queries surrounding segment I will try to explain why we need surrounding segment



Let say map service  get some 5-10 driver which are close to the customer . **Location service** will return it to **cab finder.** now basically we need to find one driver out of 10 who will do the trip. Now there come something called **modes .** there will be multiple modes in which this cab request could be served. We could say that for certain kind of customer just pick the best driver now let say  if it’s a premium customer  we just pick the best driver out of the lot and assign that. Or if an avg customer we might to do diff thing . We might want to broadcast all the driver and who assign first will do the trip. So all of these modes are basically **cab finder decide**that which mode I want to run it. Given whatever mode it is it need some additional input so if it need the best driver mode  then it might need to stack rank of all the driver. So all of these are basically something that is handled by **driver priority engine .** we will get into the logic what it follows later on.**Cab finder query  driver priority engine** saying I have those driver  for this kind of customer u arrange them and give me back. Then it get list of driver given the mode  and given the list it then tries to identify  one of the driver who will actually do the trip. It then queries web socket manager  and ask **web socket manager**saying  which was the host that actually you know integrating with this particular driver . It will then call web socket handler.

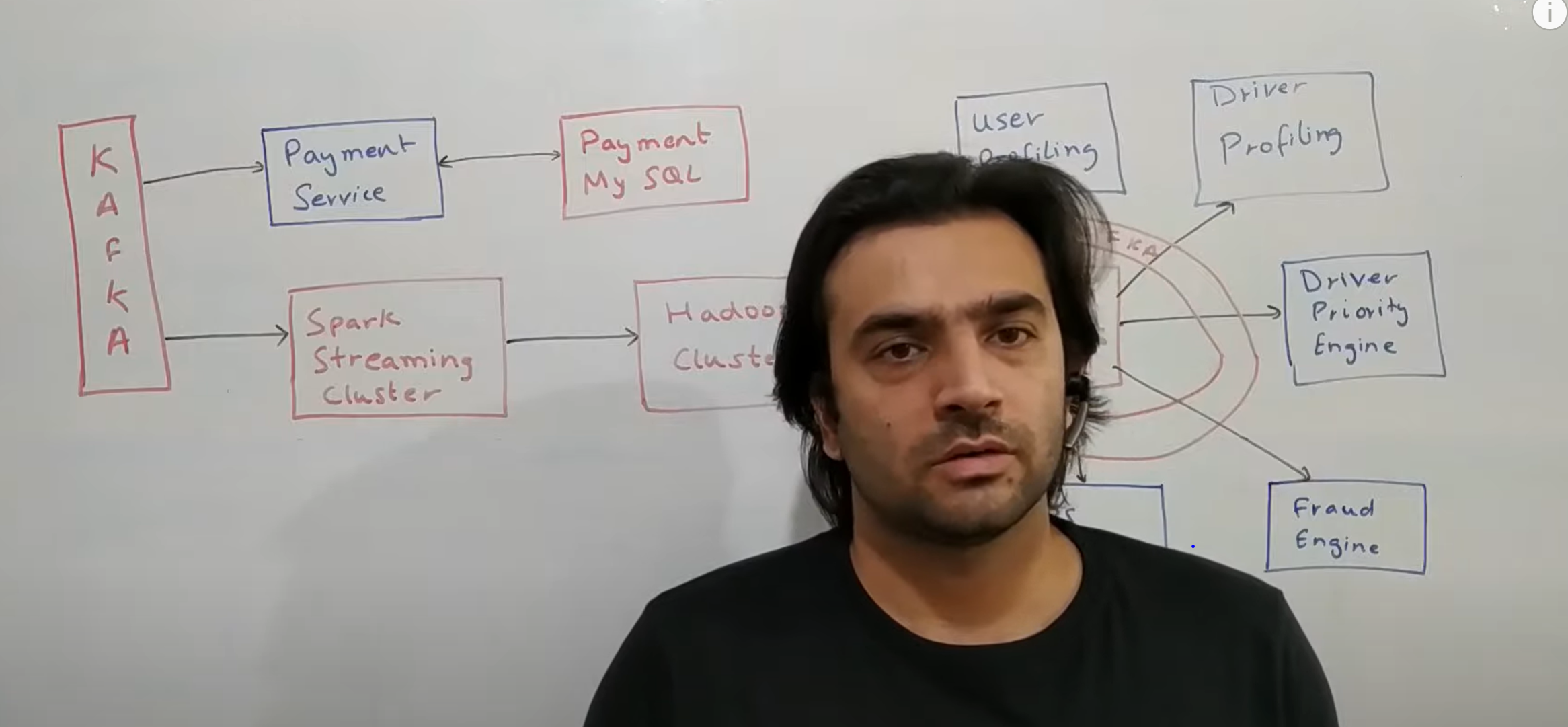
Let say it choose d1 it will then call this handler 1 saying d1 u have the trip go and do it. The same n/f would be send  via cab request service to customer  saying customer u have got a new driver d1. And then regular flow start where driver move to customer location so this is **how we figure out how to assign a driver to a customer.**

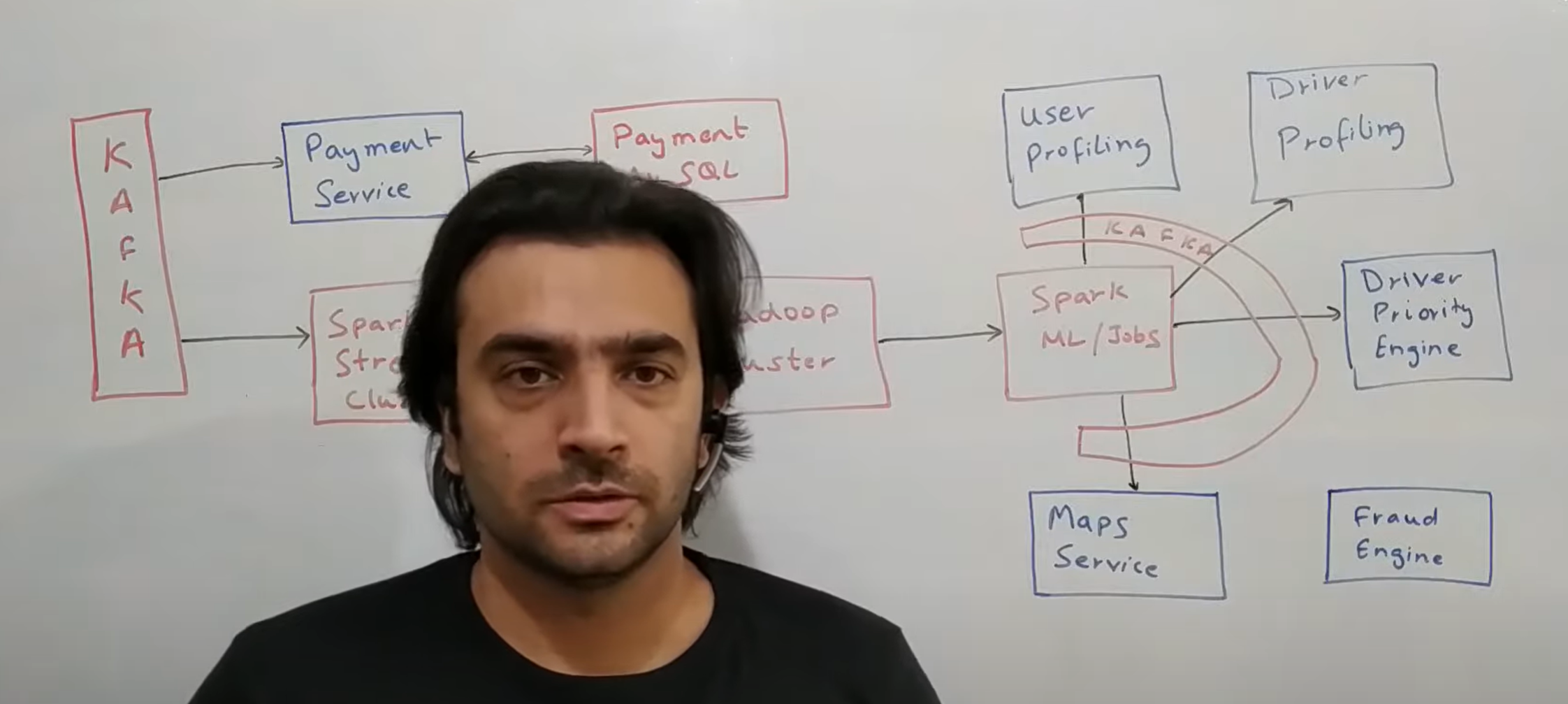
 Now once a driver is assigned so basically what it means trip has to be created and updated

So this will basically update the trip service saying I have created a trip  with this customer with this particular start point and end point and that get persisted into mysql via trip service. So this is basically the booking flow.

**Now as part of booking flow we inserted a lot of events** whether cab finder was able to find a ride for the customer or not and even location service was putting a lot of events into kafka.

**Now lets try to look how do we use those events**





This kafka is getting lot of events like location update events , trip update events , no driver found event s , a lot of thing. Let look at some of the use cases where we can utilize those events to our benefit. One of the very common thing is – whenever a trip is completed , we need to initiate a payment to the driver. That could be aggregated over a few hours or a few days or something of that sort. But we still need to store info about potential payment.there would be a **payment service which**  sit on top of this kafka which would have a kafka consumer which listen to all trip completion event . and as soon as a trip is completed, it would insert a record in its payment my sql d/b, which says that this particular driver did this particular trip\_id for a user with user\_id and with lot of attributes like distance travelled , time taken and all of that. And finally amount of money that need to be paid to this driver. If that need a instantaneous payment, this payment service could talk to a payment gateway to deliver a payment like transact the money. Let say if driver want to see their payment history there would be api that would be running out of this service which will give all the payment transaction info against a particular driver.

**Now let look at some other use cases**

**A heat map** (or heatmap) is **a data visualization technique that shows magnitude of a phenomenon as color in two dimensions**. The variation in color may be by hue or intensity, giving obvious visual cues to the reader about how the phenomenon is clustered or varies over space

**On**  top of this cluster there would be a spark streaming cluster in which some spark streaming jobs would be running . one of the very common thing is to basically create a heat map. If let say from a particular geography getting lot of event saying there is no driver

That’s means that there is a surge of customer in that area and there are very few driver. So we create a heap map within driver app powered by this streaming which kind of show a particular segment or a few areas which are having this kind of scarcity so driver can move to location to get more trips. This is a classic example of streaming kind of application.

What will also do is it will basically put all the events into a Hadoop cluster which can be used for further analytics.on top of this Hadoop cluster we could run a lot of ML jobs or regular spark jobs which will do a lot of things. So the very first thing to do a customer classification into various categories so if a customer takes a trip every day will classify as premium customer . same classification can be done for driver . so there will be user profiling and driver profiling jobs which would ideally be a ml classification model running which will classify those user as premium or regular or driver as premium driver or regular drivers. Based on customer rating or many other attribute we can also create driver priority. Same data could be used as input to map service .

Traffic data and some of the road data will be there into map service