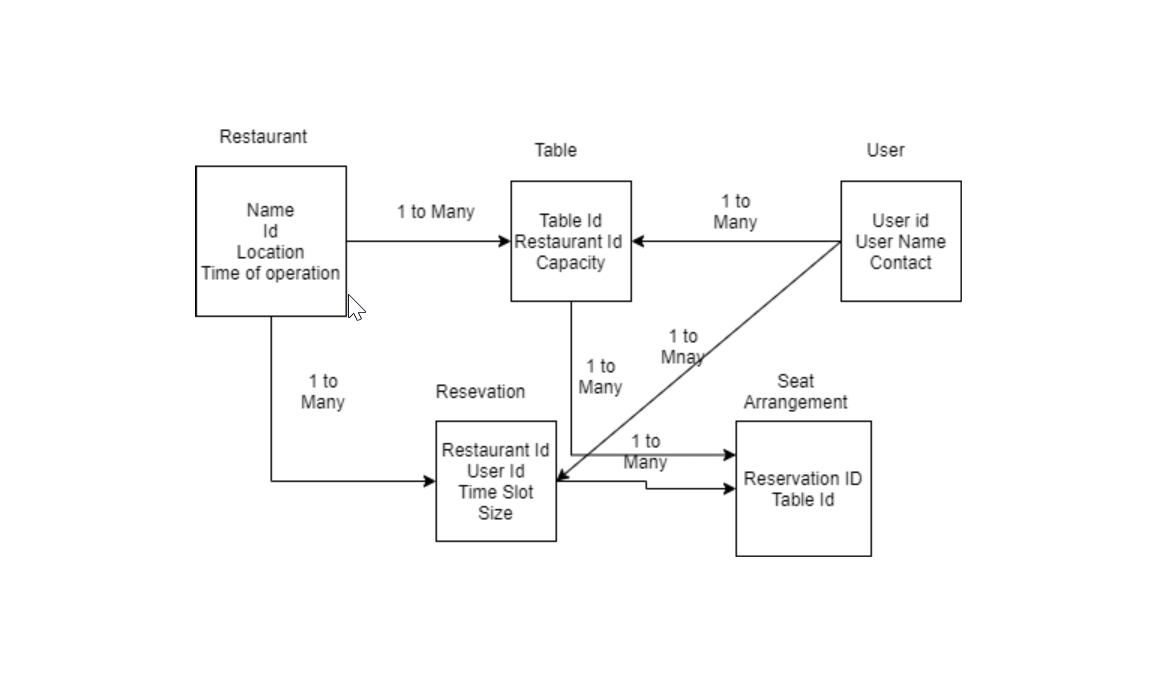
Table booking app

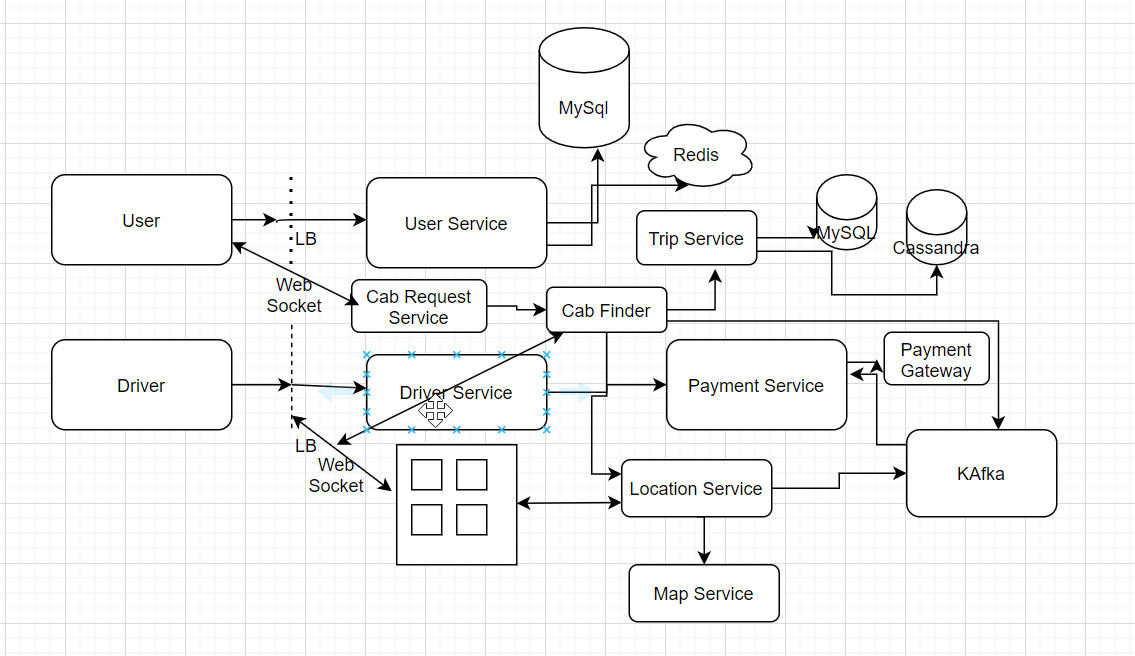


RestaurantService :

1. List<Availablity> getAvailability(Location, Time, BookingSize) (Availablity – Restaurant, timeslot)
2. Restaurant makeReservation(Restaurant, User, Timeslot, Size) throws Exception
3. Void cancelReservation(Reservation)

User Service – User CRUD

Cab Booking System Design



User Service – user CRUD operation , Connect with MySql db & Redis for cache

Cab Request Service – take request from User & connect to Cab Finder.

1. bookCab(cab\_type,userLocation,destination, UserDetails,paymentType)

Cab Finder – Connect with Location service & list of driver near this

1. findCab(cab\_type,cab\_location)

Driver Service – driver CRUD

PaymentService – listen trip completion event

Map Service – convert User & Cab location, calculate shortest distance

Trip Service – store trip related data

**Data Model**

User – id, name, currentlocation, paymentMode, Mobile

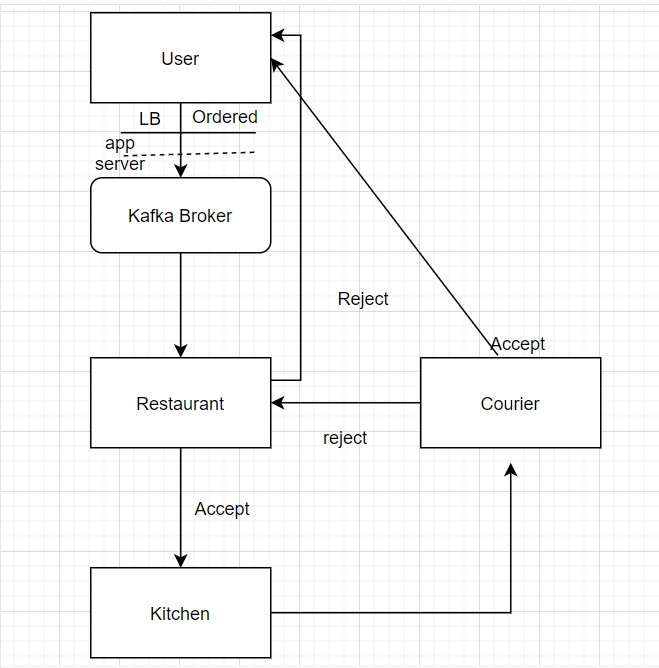
Cab – number , cab\_tpe, location

Driver – id, name, cab\_number, location

Cab- Driver (1-1)

User-cab – 1-1

Food App Design:



user -> order placed -> accept/reject -> resturent -> prepare order -> food ready for pickup -> send message to courier -> accept/reject -> pickup order -> delivered

user (/order)-> load balancer -> app server -> kafka queue(partition as per location) -> order processor -> notification service (sending notification to resturent) -> update order table -> resturent -> dispatch -> send notification to drivers

user - id, name, location, mobile

restaurant - id , name, location, operation of hours, rating

order - id, rest id, cart id, state, driver id

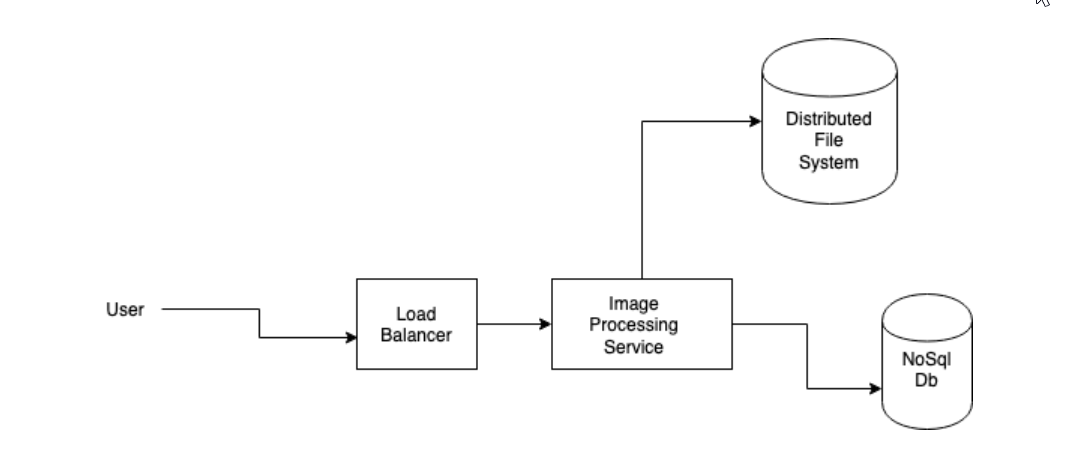
cart - id, value

courier - id, name

payment – payment id, orderId, amount, mode,paymentDate

User Service, Restaurant Service, Order Service, Kitchen Service, Courier Service, Payment Service

**Photo Uploader App :-**



Distributed files system - that will store the actual images. For example Hadoop, Google’s file system (proprietary), S3 e.t.c

Nosql - searchable store. Key are the tags. Value is path for image. For example Elastic or Solr. Uploading is practically adding to Elastic a json, having tags in indexable field and Hadoop’s path in other field.

Now we will be supporting two API endpoints for user.

POST endpoint /save/image, which get the image and a list of tags along with it and we will be storing the image and insert the image url into all the json where the tag is our key.

GET endpoint /get/images, which will take a tag keyword and will return back with the list of image urls present in the tag.

The diagram only contains one node of our Image Processing Service but we scale it up to n numbers and load balancer would take care of the concurrent requests coming in from the users.

Data Model :-

photo - type, id, name, size - valid/invalid type, size should be less then 10 mb

user – id, name, mobile