**Write up**

The exercise we did was given an application we wanted to return a list of all available restaurants that are available. We built an extremely basic implementation of this. Let’s address how we can help make this application more feature enriched and also at the same time can build it in such a way that we could scale this application. Let’s divide the topic on same lines.

Features

1. We could make the application better by trying to return not just the list of restaurants but returning the closest restaurants nearby you. The Socrata API has calls like within\_circle and within\_box which we can use my tracking our lat, long and thereby getting closest restaurants near us.
2. We can enhance this capability by adding feature of choosing the type of restaurants that you want to visit e.g.: Mexican, Indian, Chinese. There is optional text which contains some of this information. We can also check to see if there is a way to get hold of ratings or reviews and enrich the user-experience.

Scalability:

1. We well build a containerized application so that we could run multiple copies of it at the same point in time in multiple VM.
2. We can stand an application load balancer to help us route the traffic and allow millions of users use the application without there being downtime.
3. We can cache API connections so that we won’t have to instantiate it for every call. (guava supplier memoize)
4. We can use some sort of caching mechanism especially if there are too many requests at the wee hours of the day where many users are trying to query at the same time. (Simple Redis read through mechanism for starters with LRU policy should be fine).
5. We can build the Application to be fault tolerant and high available by making sure that we use some of orchestration mechanism. Use Kubernetes to achieve this.
6. We can add security layer to see that there is DDOS attack on our application and we can throttle request based on application token keys.
7. We can make our application Asynchronous so that the user experience is not bad.