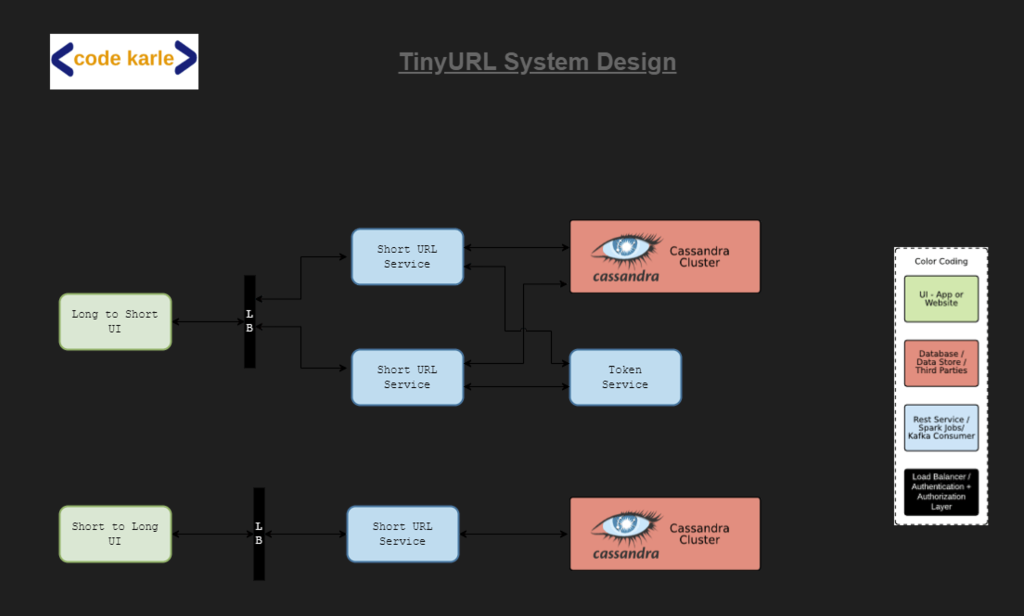
Section 2 :- System design case studies : Practical real world application

                                                    2.1 URL shortner system design



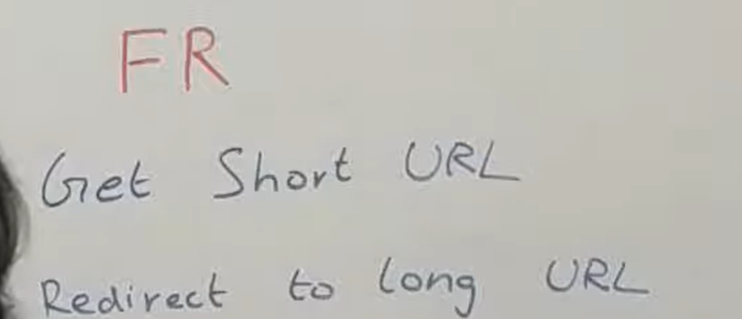
Text

Description automatically generated

**Requirement : How** do we design  a URL shortening service , something very similar to tinyurl.com

Let start with functional and non functional requirement

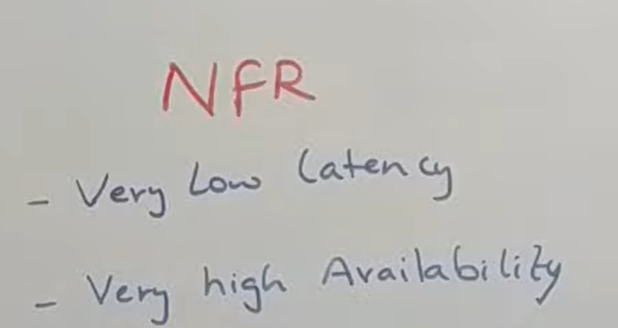
**FR**



When someone give a long url we need to return a short url

When someone hit short url then we need to redirect to long URL

**NFR**



What does low latency mean in system design?

Low latency describes **a computer network that is optimized to process a very high volume of data messages with minimal delay (latency)**.

The service need to be highly available and it should work at**very low latency**

**What should be the length of short url ?**

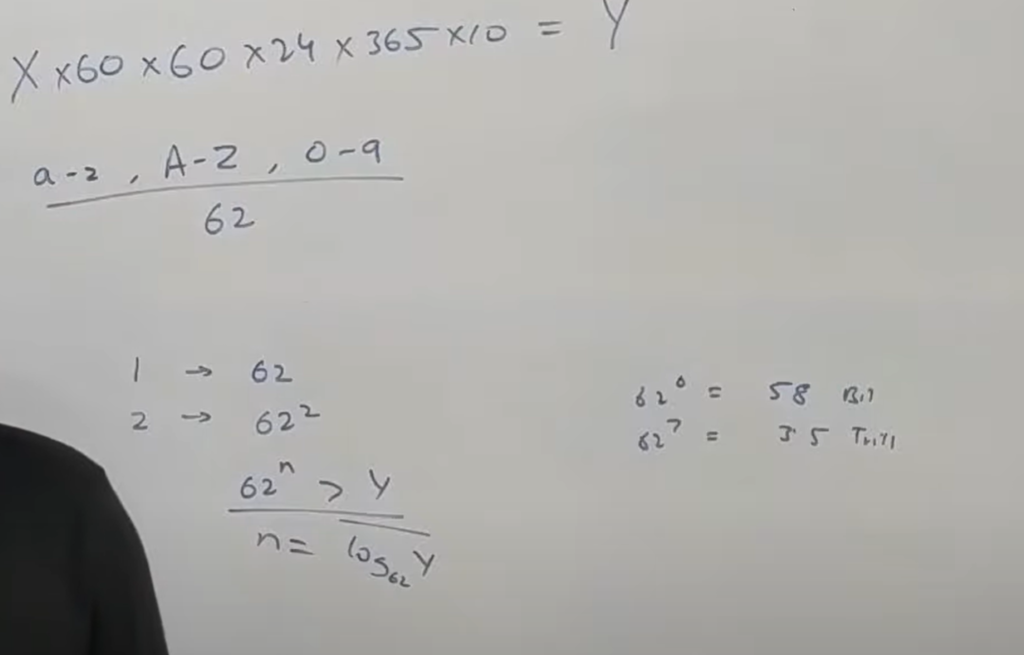
**Ans :** that depend upon the scale that we need to build this system for

But if we are building google app or facebook app then we should think what should be the length of short url

What characters allowed in short url

Basically [A-Z][a-z]numbers

X request is coming for 10 years



problem

**If two instance of a service generate same  short url  for two request then we have a problem. In technical terms it is called collision but it is a problem because we cannot have one short url pointing two long url.**

**Solution:**

 Predictable  way to generate a short url so that there would be no collision

One very simple way to do this**is using one of the features of redis**

**Redis :**

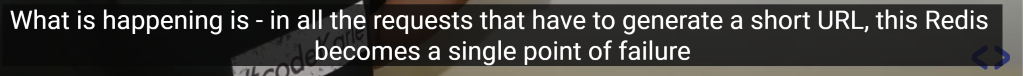
make sure it will always return a unique number

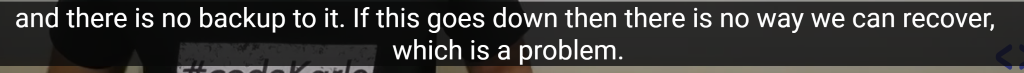
Each time it get a request it will increment the number and response back.

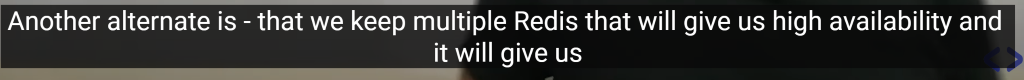
problem

That work fine but every instance of service**now will connect to redis** and now redis will under huge amount of load.

We should never make a single point if failure

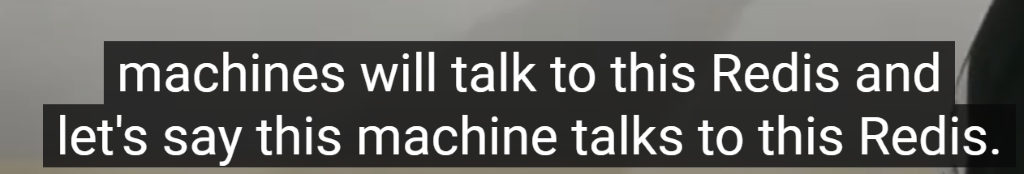


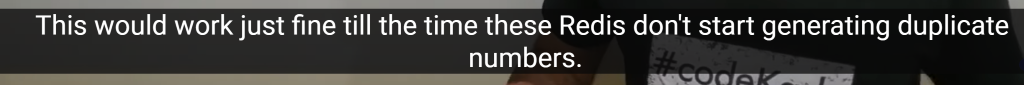


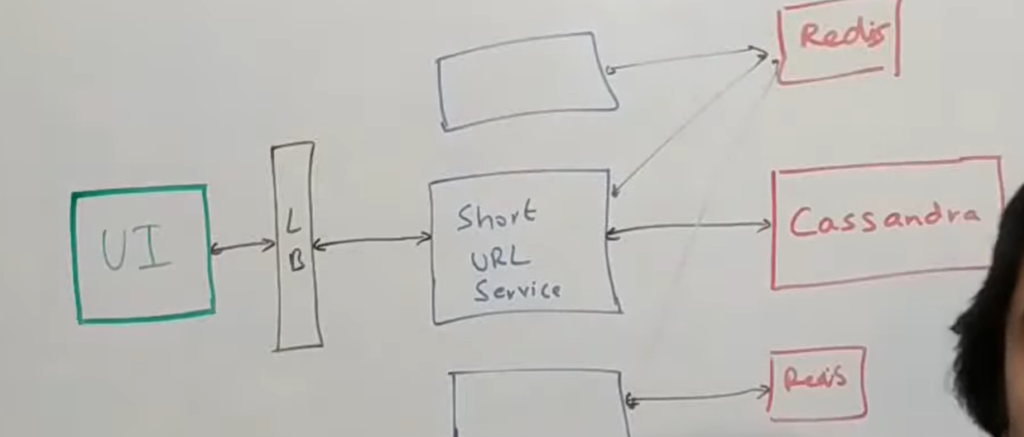


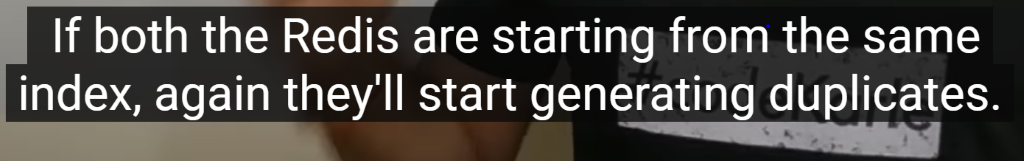
Graphical user interface, text

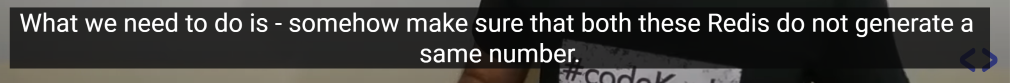
Description automatically generated







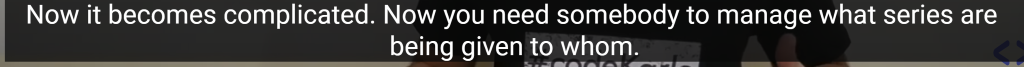


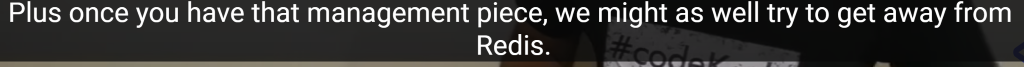


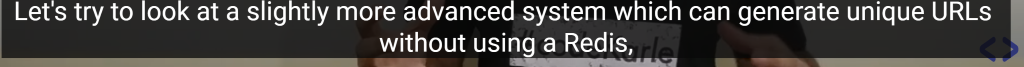
Text

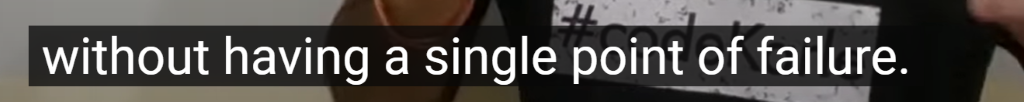
Description automatically generated with low confidence

That would work fine but what if we have to add third redis

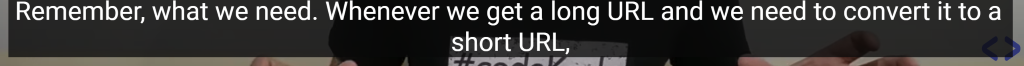


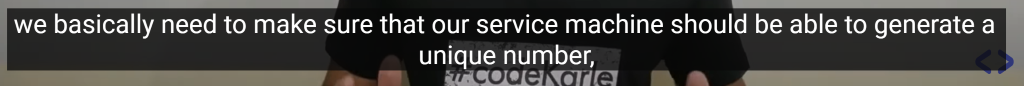




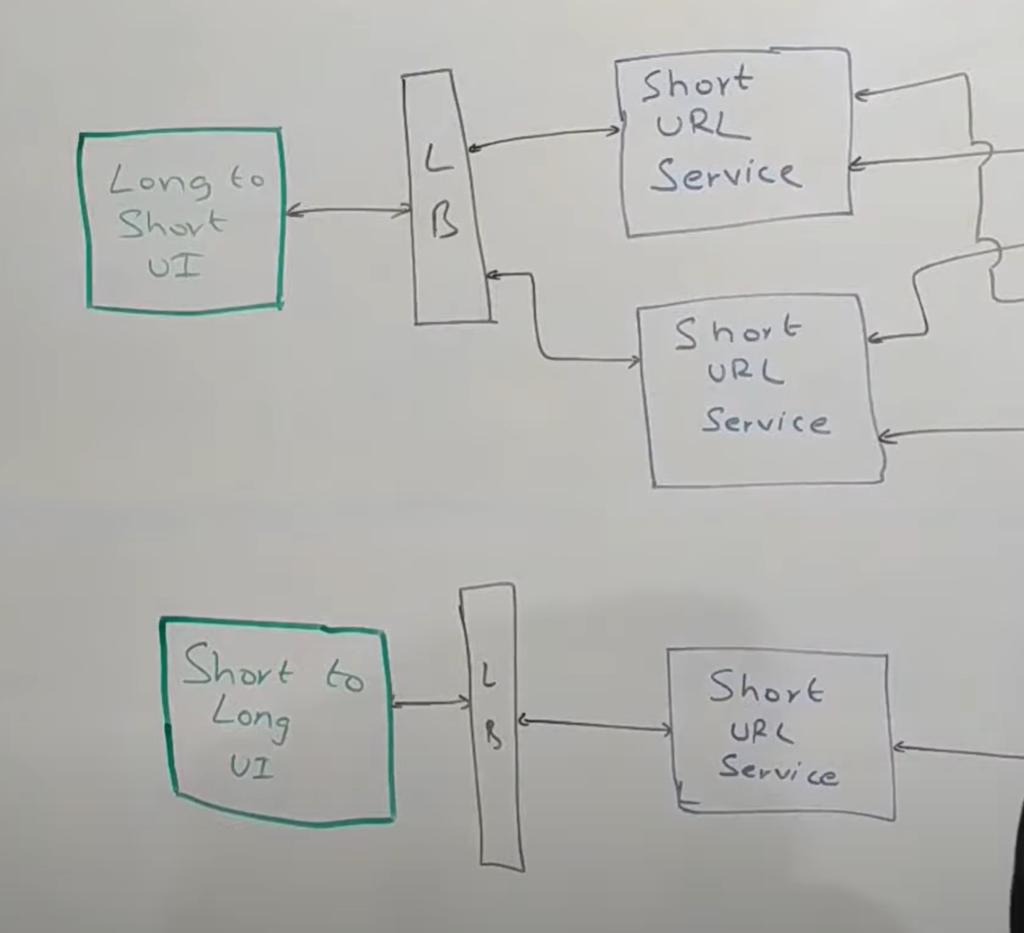


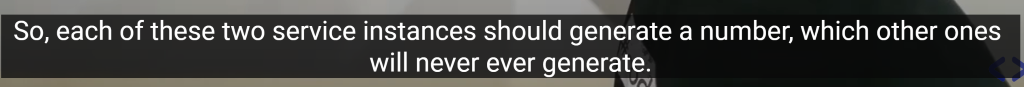


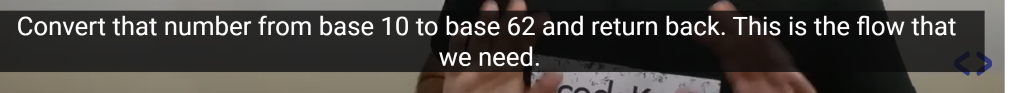


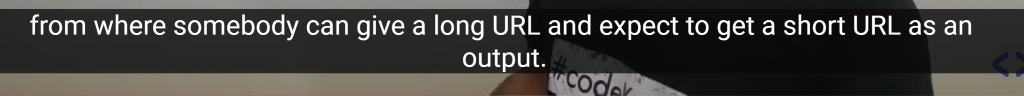


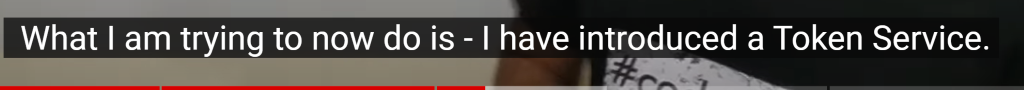


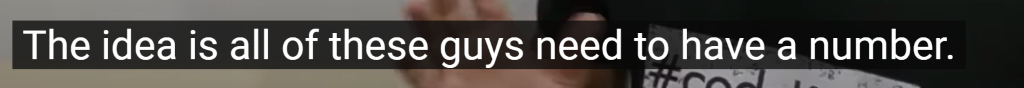


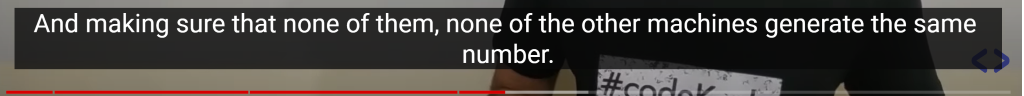


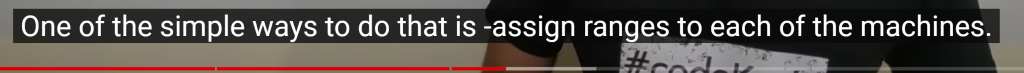


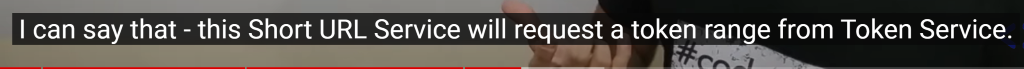


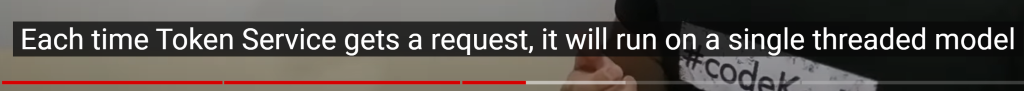


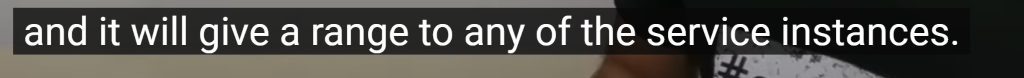


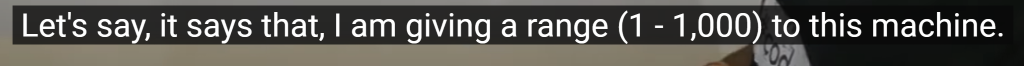


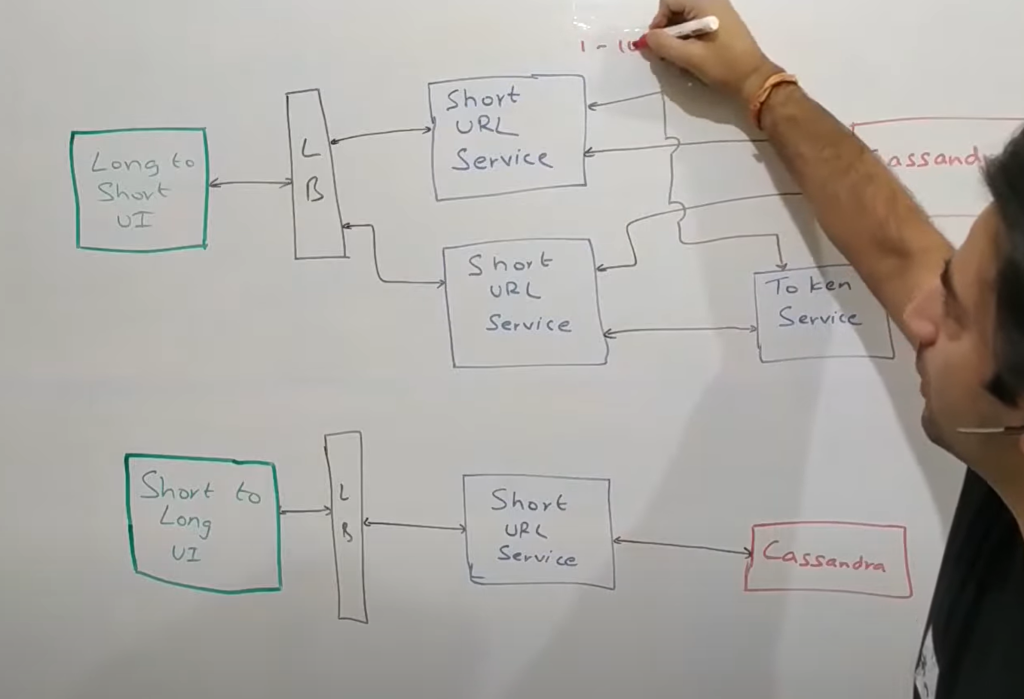




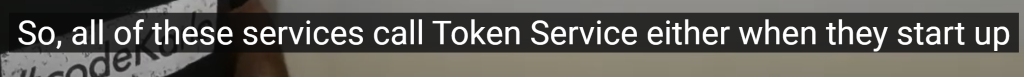


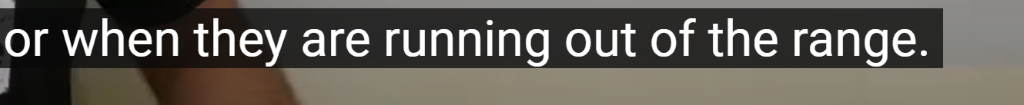


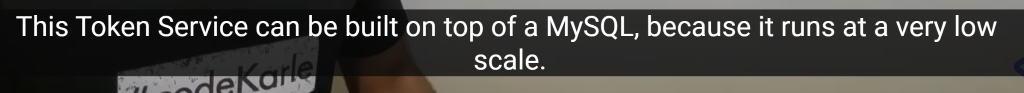


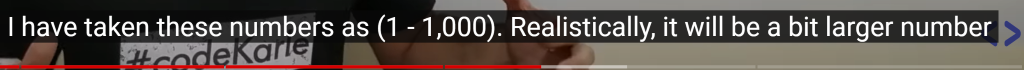


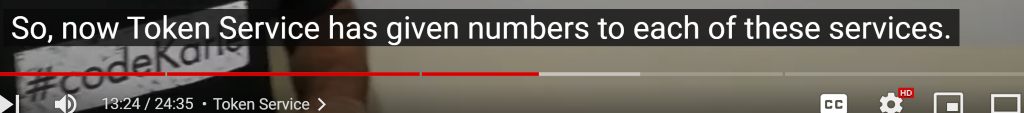




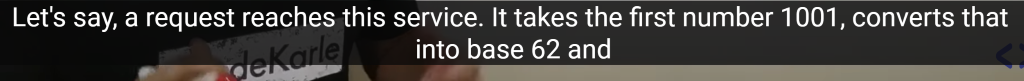




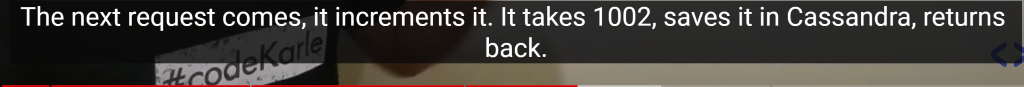


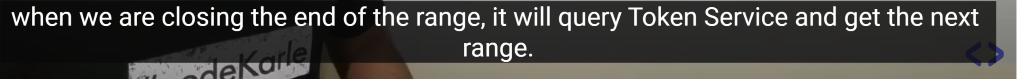


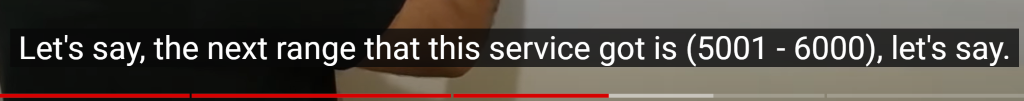
Now the flow is very simple

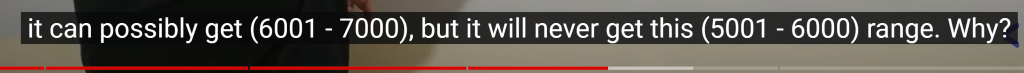






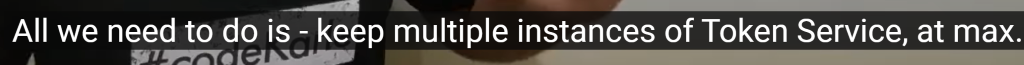


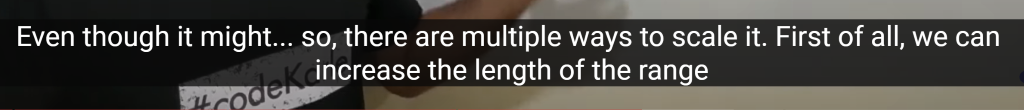


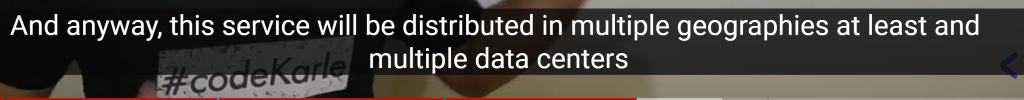


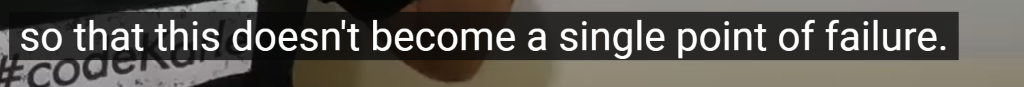
================

Now let say there is massive amount of traffic

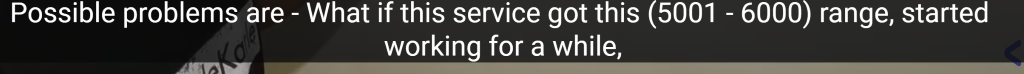


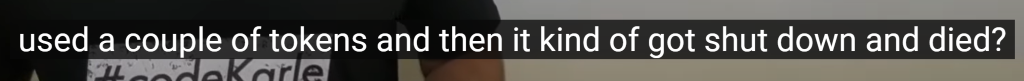


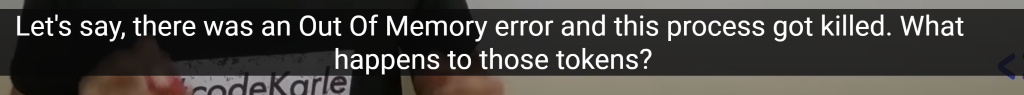


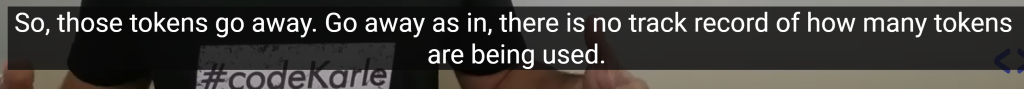


**Now there are some problem in this**

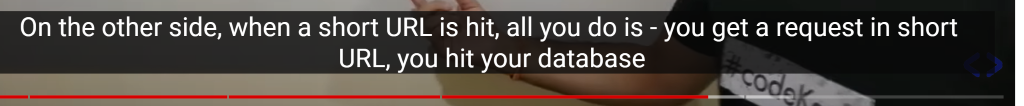


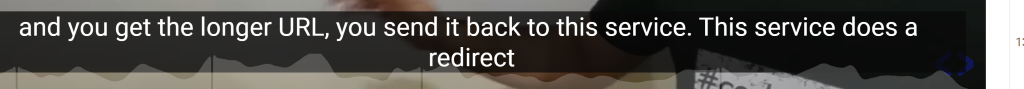


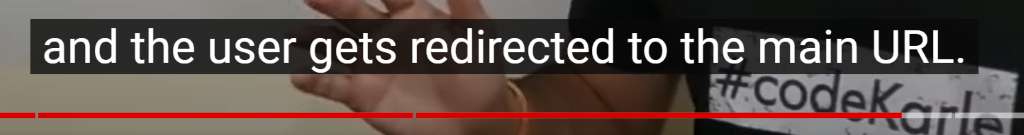


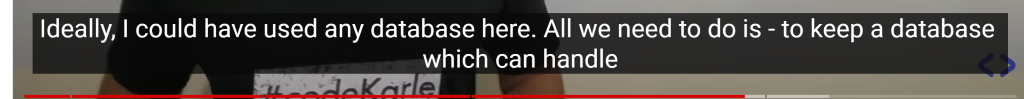


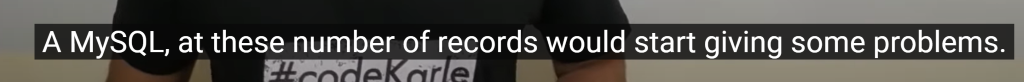
So this is basically a long URL to  short URL path

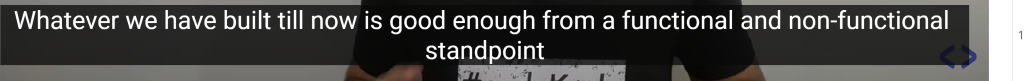




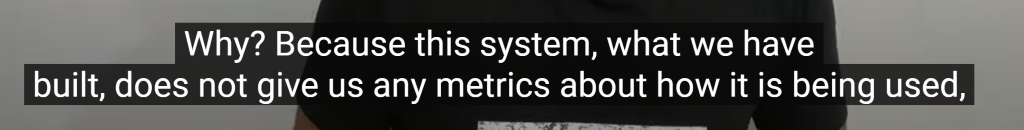


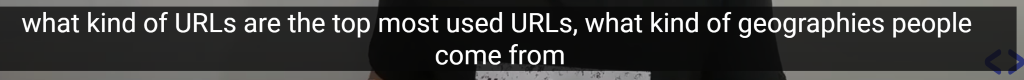


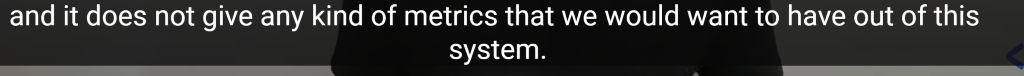




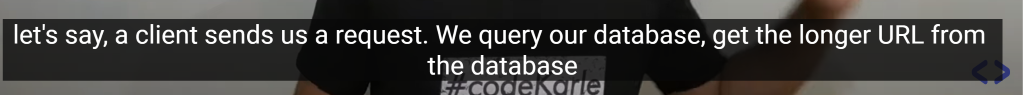
But it is good enough definitely not why

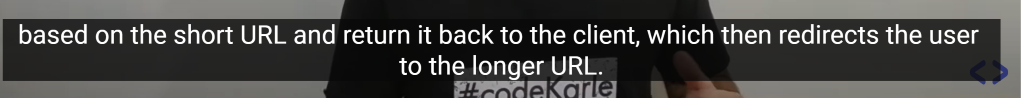


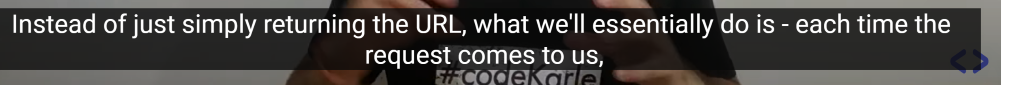


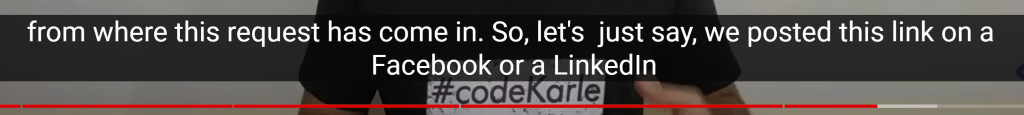


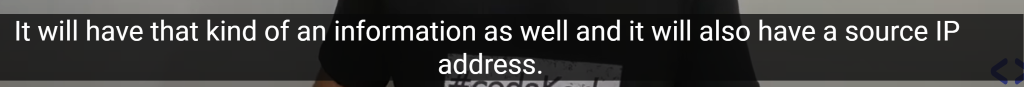
===



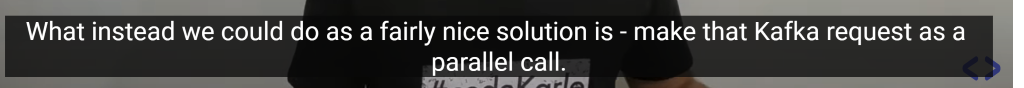










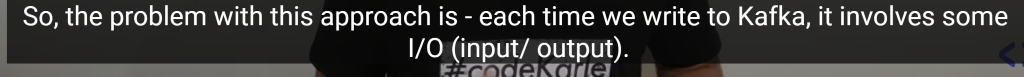


What is the problem in doing asynchronous thing

There is a potential possibility kafka write could fail for whatever reason

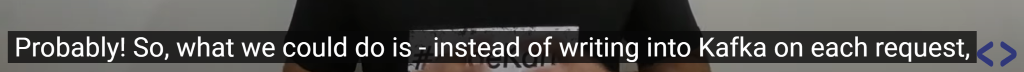
And you return back to use so u miss certain analytics

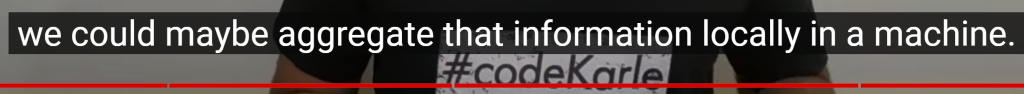
But can we improve it even further

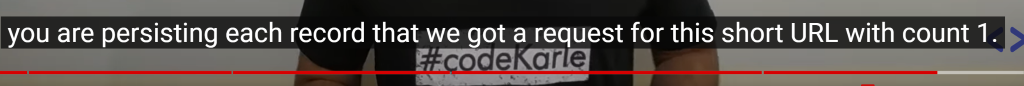




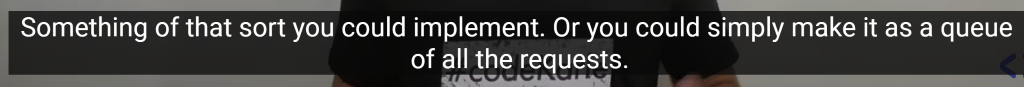
Can we avoid all those thing on each call

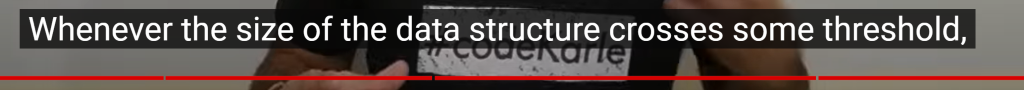


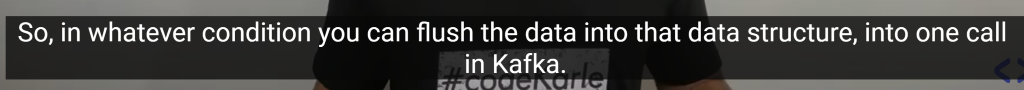


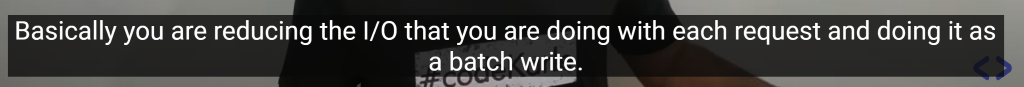


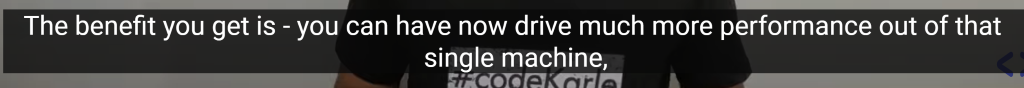


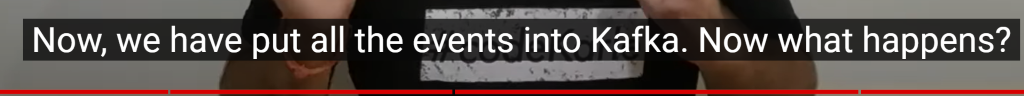


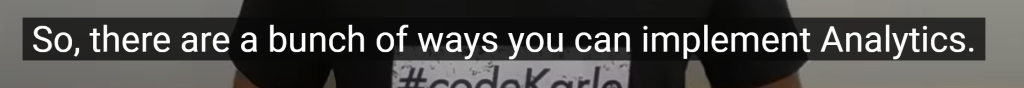




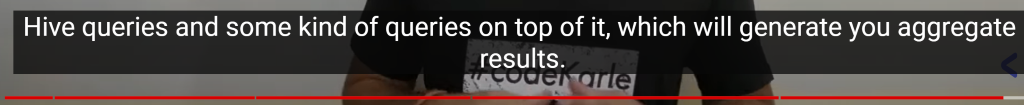








One very simple approach we can dump all of the data in hadoop and  and then build  some



An URL shortener is a website that reduces the length of your URL (Uniform Resource Locator). The idea is **to minimize the web page address into something that's easier to remember and track**. There are many URL shorteners on the market today, including Bit.ly, Goog.