System Design - URL shortener

https://www.youtube.com/@15minutesystemdesign

Functional requirements:

Functional requirements:

- URL Shortening: Given an original URL and optional custom short key, our service should generate a shorter and unique URL of it

Functional requirements:

- URL Shortening: Given an original URL and optional custom short key, our service should generate a shorter and unique URL of it
- URL Redirection: When user access a short URL, our service should redirect them to original URL

Functional requirements:

- URL Shortening: Given an original URL and optional custom short key, our service should generate a shorter and unique URL of it
- URL Redirection: When user access a short URL, our service should redirect them to original URL
- Expiry: Shorter URL will expire after a standard default timespan

Functional requirements:

- URL Shortening: Given an original URL and optional custom short key, our service should generate a shorter and unique URL of it
- URL Redirection: When user access a short URL, our service should redirect them to original URL
- Expiry: Shorter URL will expire after a standard default timespan

Non Functional requirements:

Functional requirements:

- URL Shortening: Given an original URL and optional custom short key, our service should generate a shorter and unique URL of it
- URL Redirection: When user access a short URL, our service should redirect them to original URL
- Expiry: Shorter URL will expire after a standard default timespan

Non Functional requirements:

- Low Latency

https://www.youtube.com/@15minutesystemdesign

Functional requirements:

- URL Shortening: Given an original URL and optional custom short key, our service should generate a shorter and unique URL of it
- URL Redirection: When user access a short URL, our service should redirect them to original URL
- Expiry: Shorter URL will expire after a standard default timespan

Non Functional requirements:

- Low Latency
- High Availability

https://www.youtube.com/@15minutesystemdesign

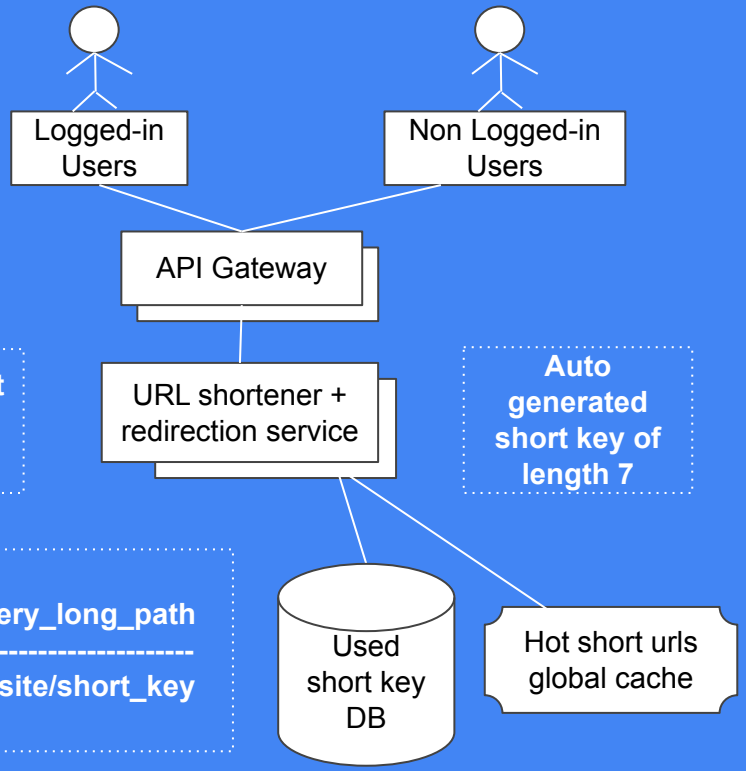
Functional requirements:

- URL Shortening: Given an original URL and optional custom short key, our service should generate a shorter and unique URL of it
- URL Redirection: When user access a short URL, our service should redirect them to original URL
- Expiry: Shorter URL will expire after a standard default timespan

Non Functional requirements:

- Low Latency
- High Availability
- Random Short URLs (Unpredictability)

https://www.youtube.com/@15minutesystemdesign



Custom short key of length 3 to 10

Auto generated short key of length 7

www.any_website/very_long_path

---

www.short_url_website/short_key

Considering Base 62 encoding [0-9] [a-z] [A-Z]
Number of Short key with length 7 = 62^7 = 3.5 Trillion

short_key :
{
long_url: www.any_website/very_long_path
created_by: user_id / anonymous
created_at: current_epoch
expiry_at: current epoch + 2 years
}

https://www.youtube.com/@15minutesystemdesign

API Gateway

URL shortener + redirection service

Custom short key of length 3 to 10

Auto generated short key of length 7

www.any_website/very_long_path

---

www.short_url_website/short_key

Used short key DB

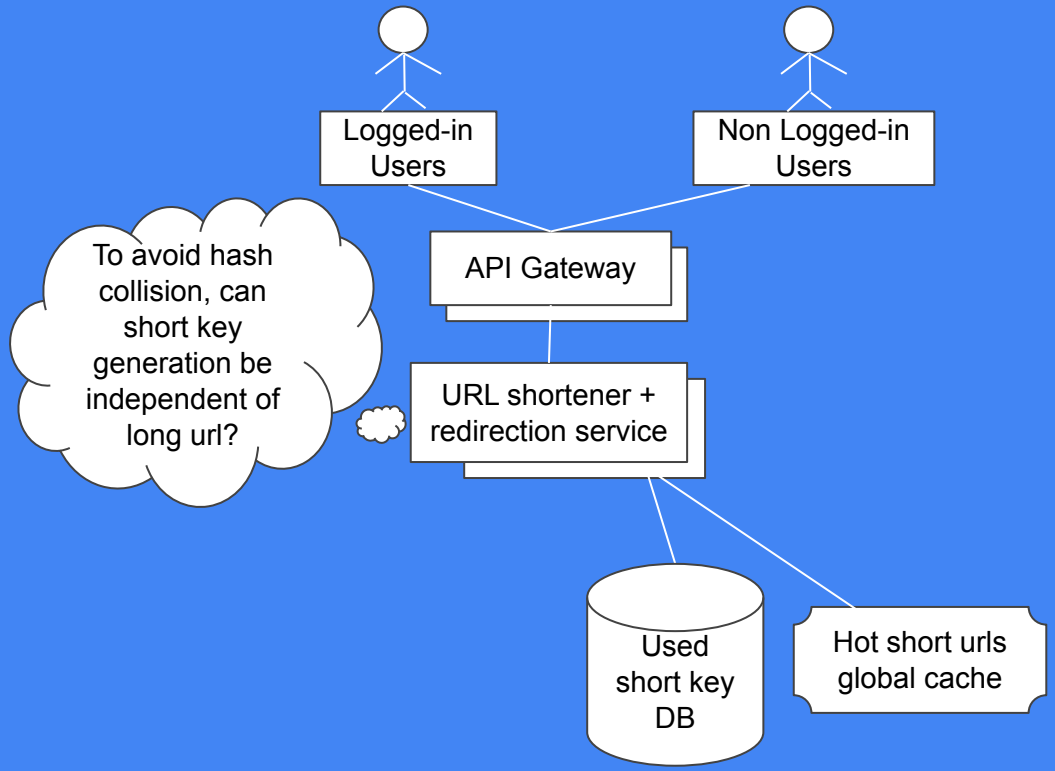
Hot short urls global cache

Hash collision: Generated key should not exist already

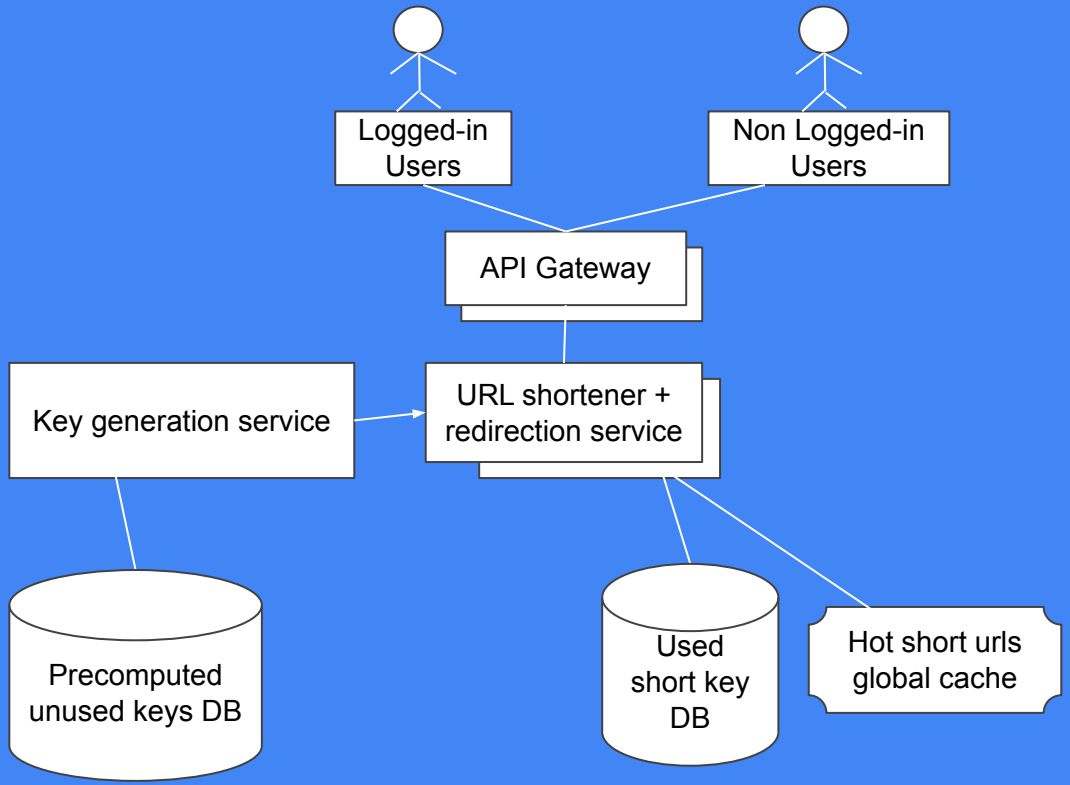
Considering Base 62 encoding [0-9] [a-z] [A-Z]
Number of Short key with length 7 = 62^7 = 3.5 Trillion

short_key :
{
long_url: www.any_website/very_long_path
created_by: user_id / anonymous
created_at: current_epoch
expiry_at: current epoch + 2 years
}

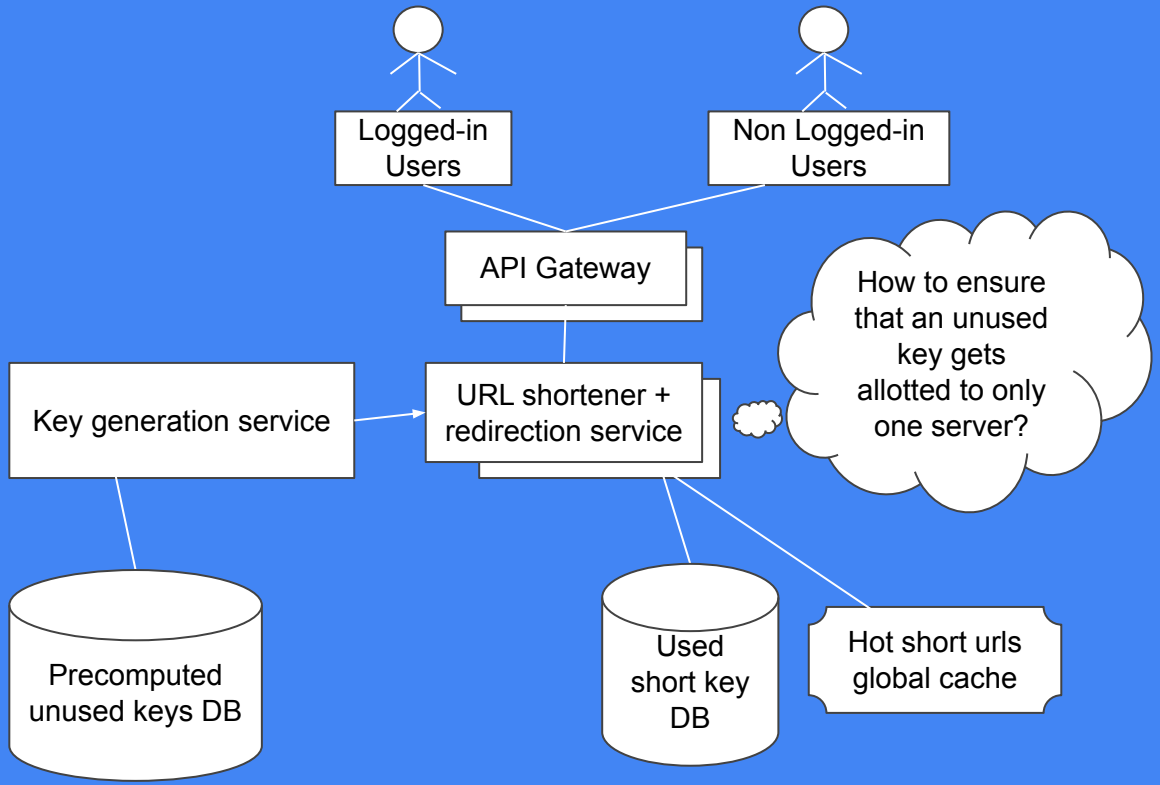
https://www.youtube.com/@15minutesystemdesign



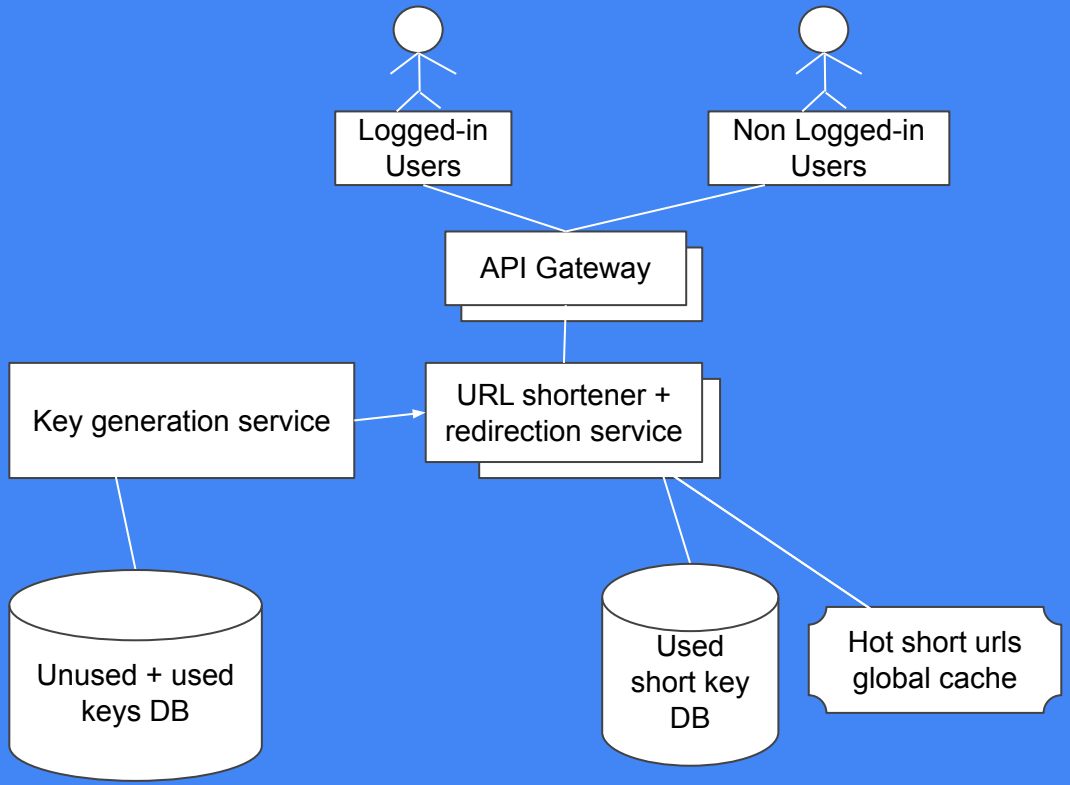
https://www.youtube.com/@15minutesystemdesign



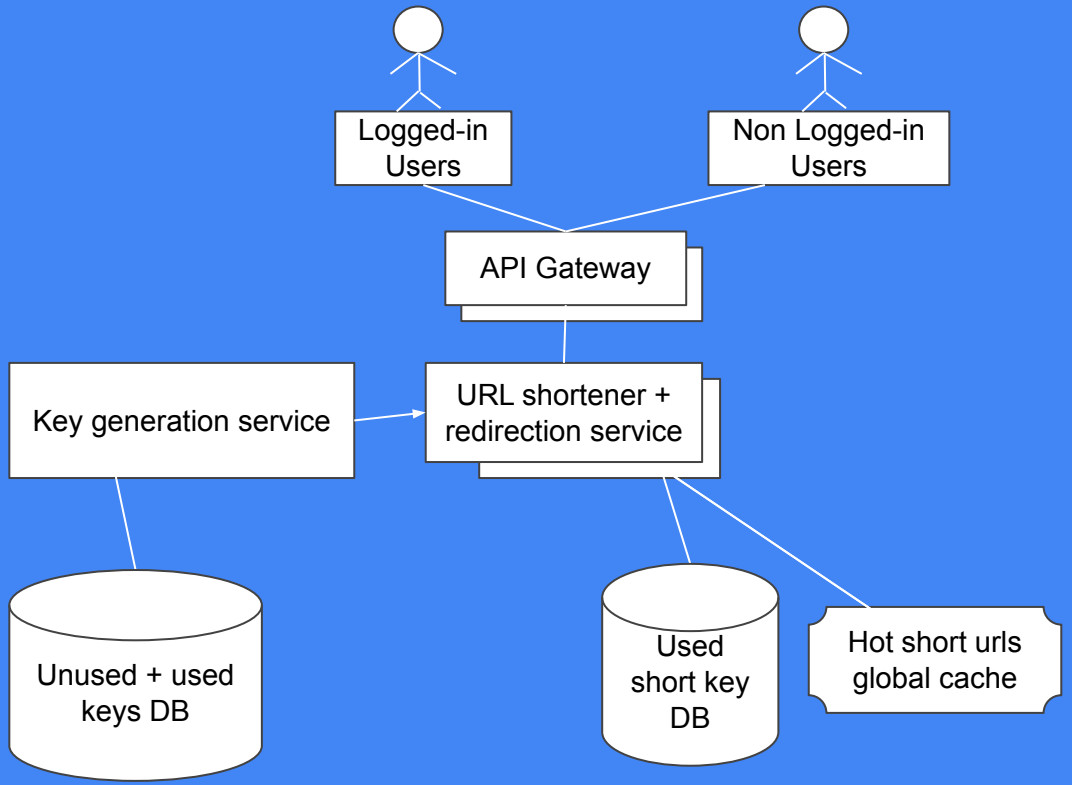
https://www.youtube.com/@15minutesystemdesign



https://www.youtube.com/@15minutesystemdesign

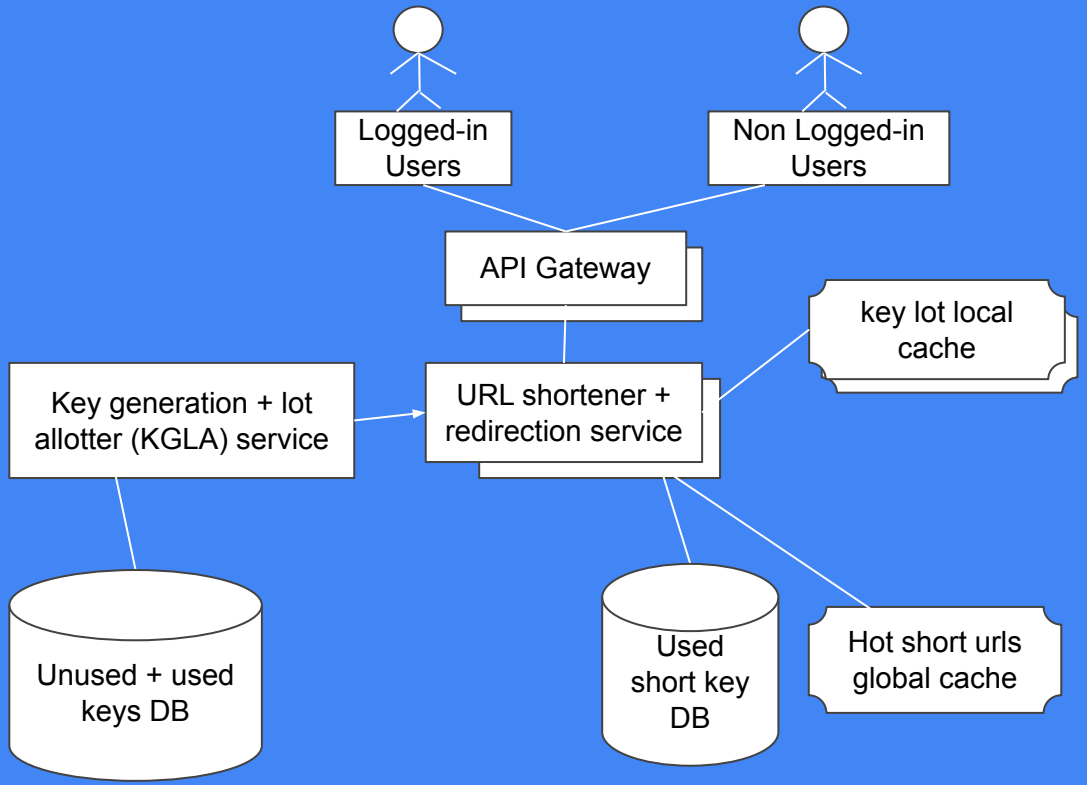


https://www.youtube.com/@15minutesystemdesign



Can we load bulk short keys in lot on each server periodically ?

https://www.youtube.com/@15minutesystemdesign



Short Keys Allotment Algorithm -

(a) Each URL shortener server checks periodically if key lot local cache need to be refilled

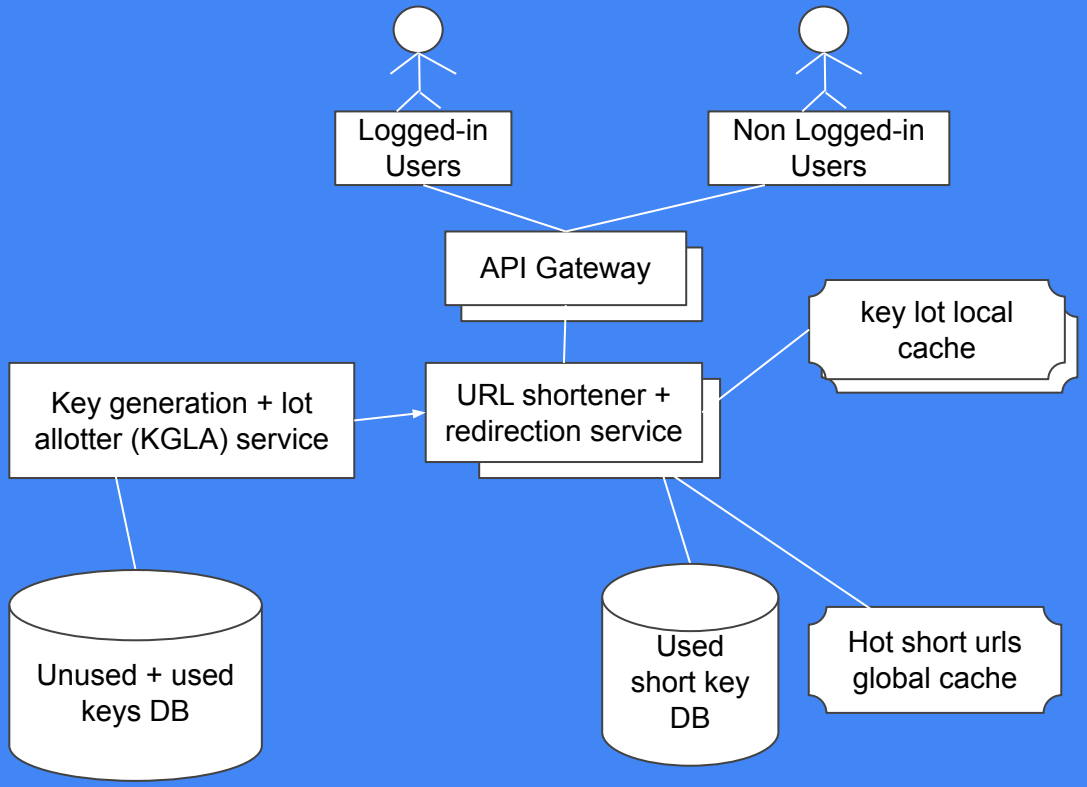
(b) If yes, that server (server_id =2) calls KGLA service to provide next lot of keys (1 lot = 1,00,000 keys)

(c) KGLA service moves 1 lot of keys from unused keys table to used keys table and mark them allotted to server_id =2

(d) KGLA service sends 1 lot of unused keys to requesting server

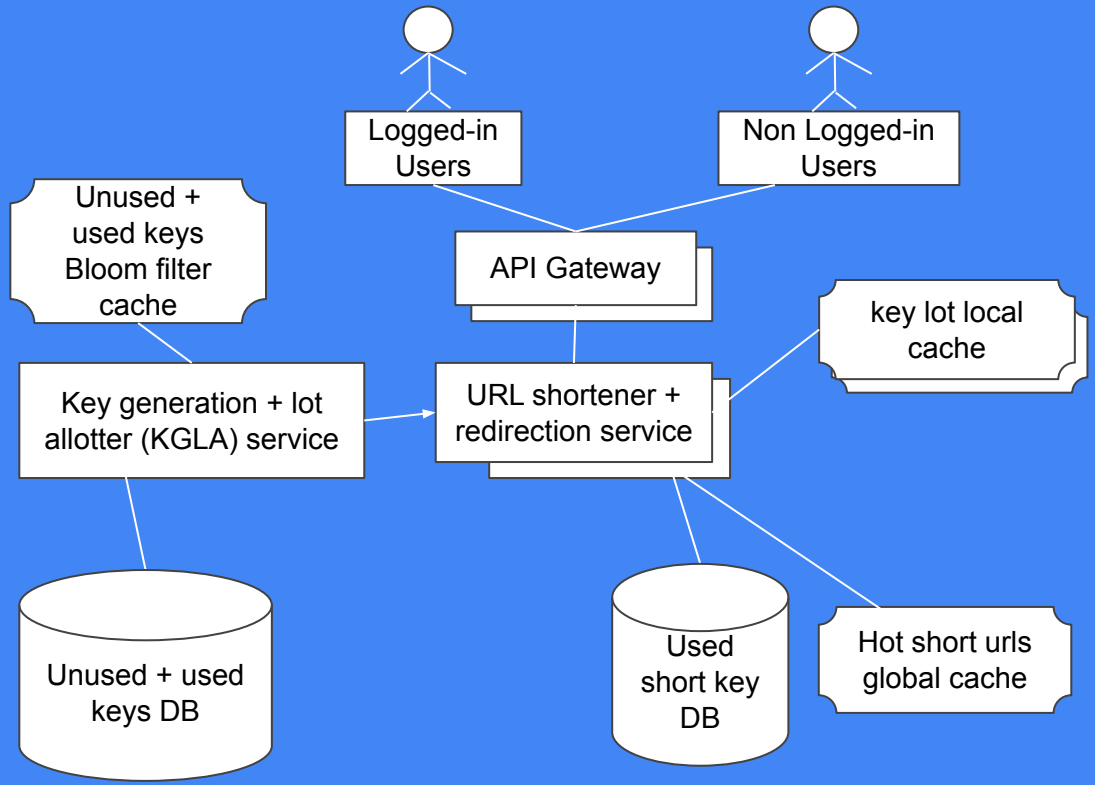
(e) server loads allotted keys in key lot local cache

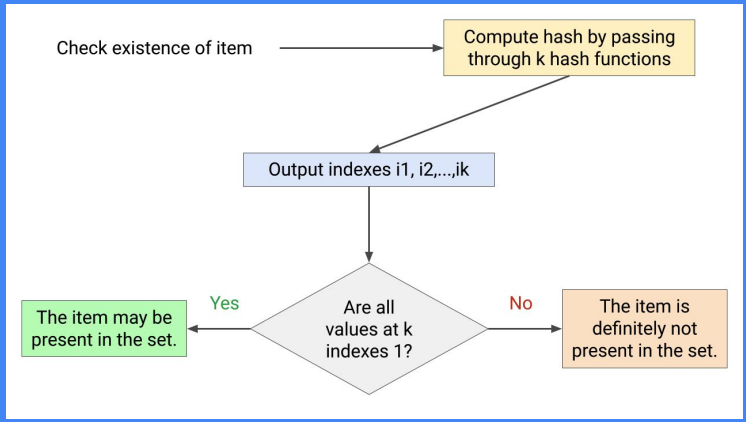
https://www.youtube.com/@15minutesystemdesign



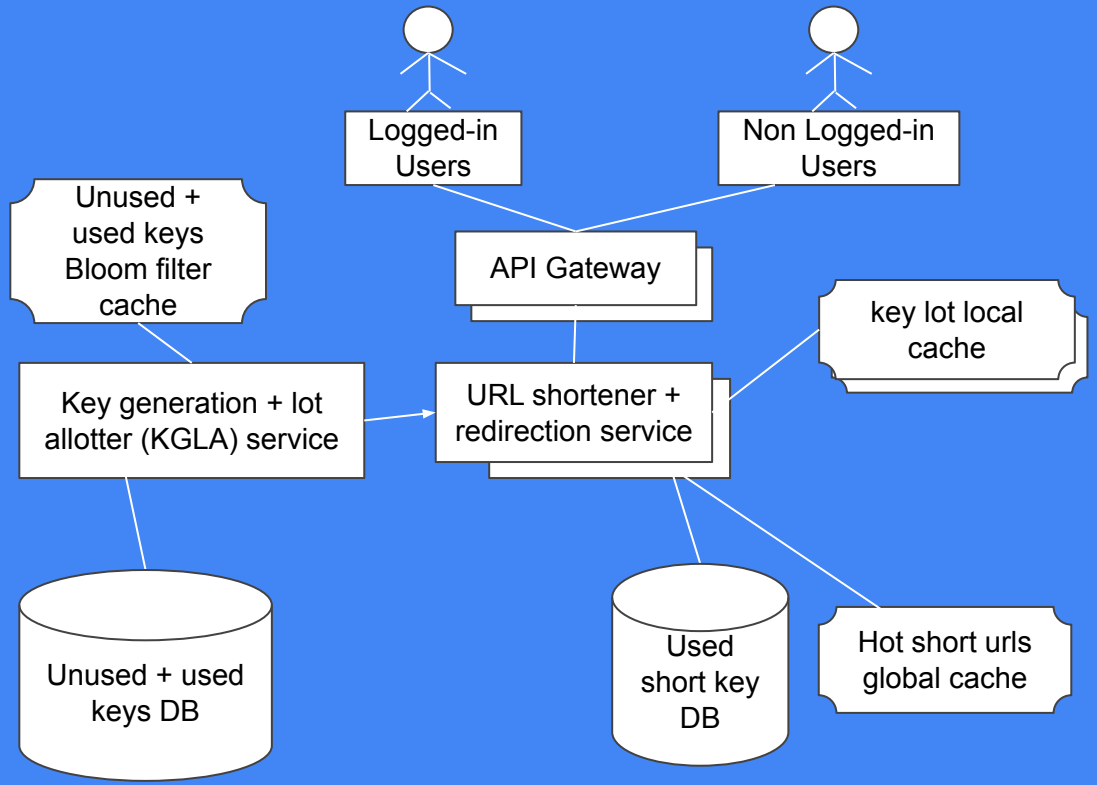
How can KGLA service check faster, if generated key definitely does not exist in Unused + used keys db ?

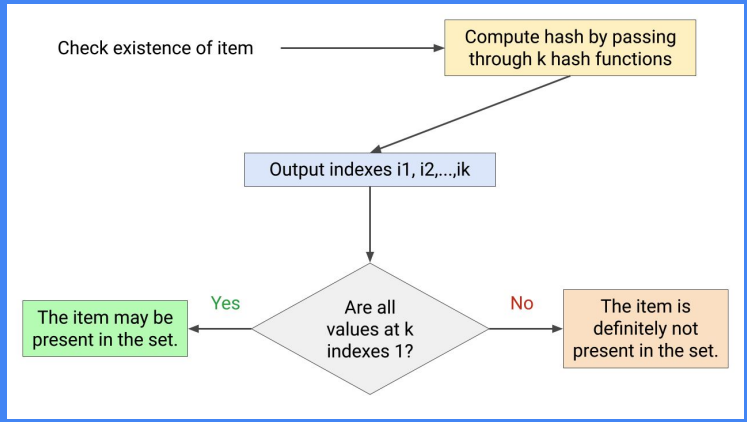
https://www.youtube.com/@15minutesystemdesign





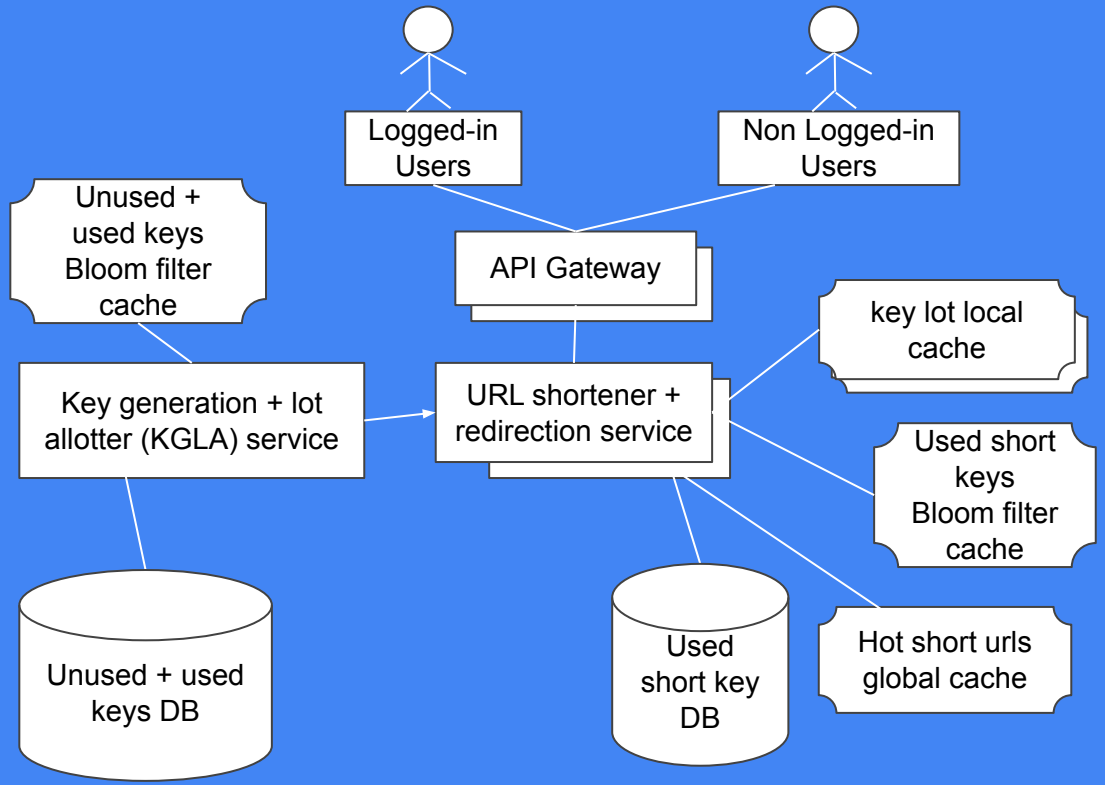
https://www.youtube.com/@15minutesystemdesign

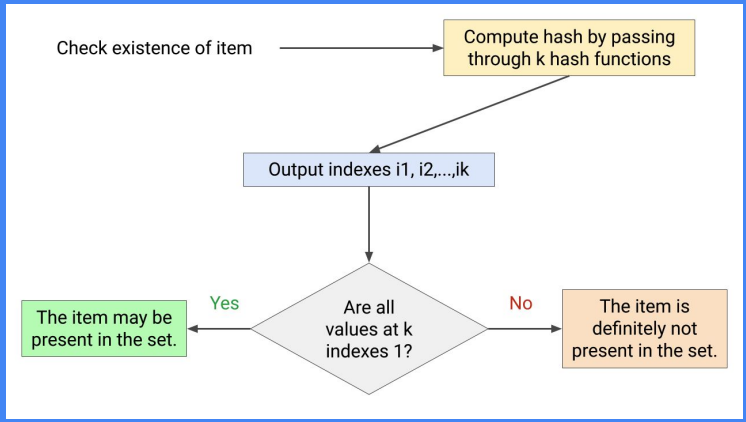




How can url shortener service check faster if custom short key definitely does not exist ?

https://www.youtube.com/@15minutesystemdesign



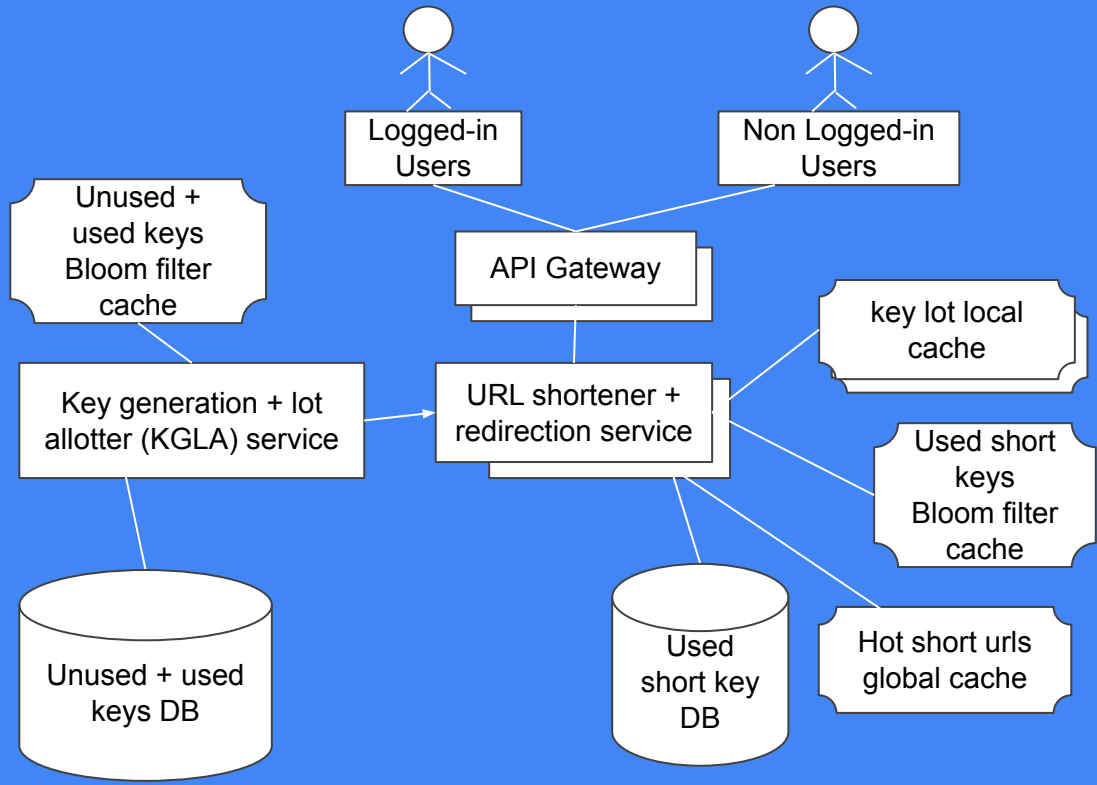


A bloom Filter with 0.0001% false positive and ZERO false negative would require just 24MB space to do "key exist check" on 10 Million keys using 10 hash functions.

Membership test with Bloom filter would have O(m) space complexity and O(k) time complexity

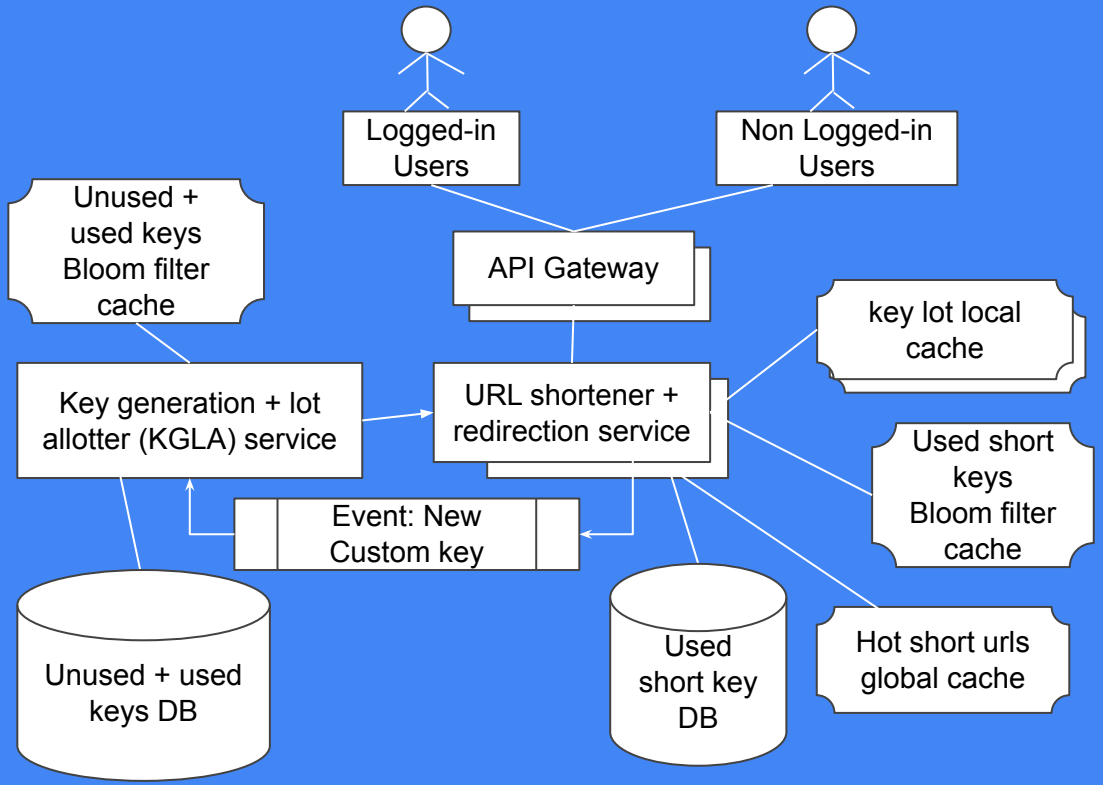
m = no. of bits in bloom filter
k = no. of hash functions
Mostly, $k < \log_2 n$

https://www.youtube.com/@15minutesystemdesign

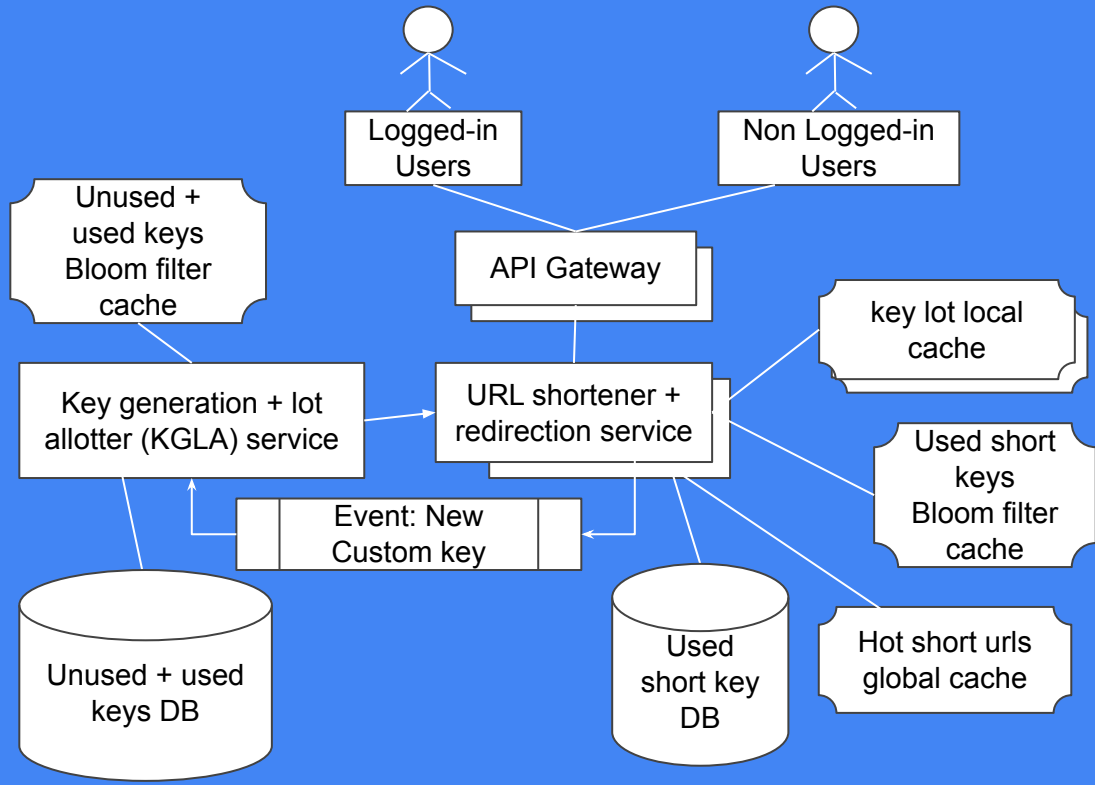


If a custom key is being used, how to inform KGLA service that this key should not be allotted to any other server ?

https://www.youtube.com/@15minutesystemdesign

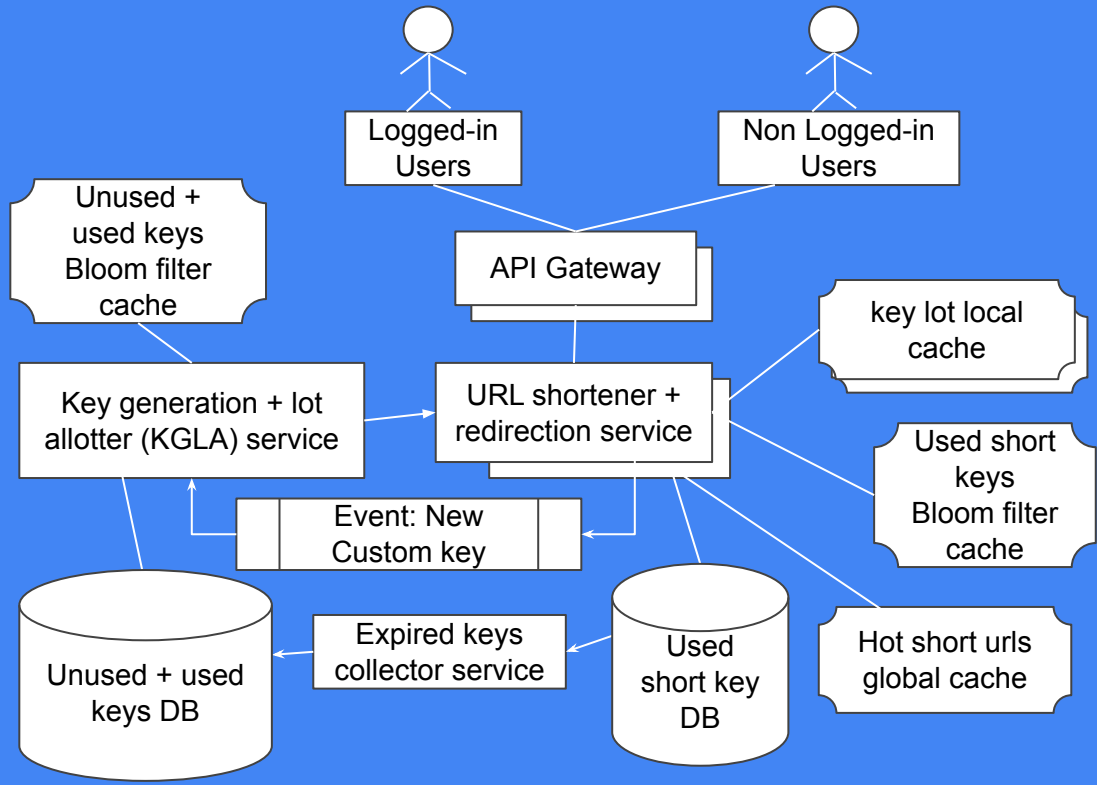


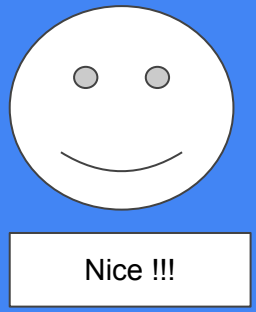
https://www.youtube.com/@15minutesystemdesign



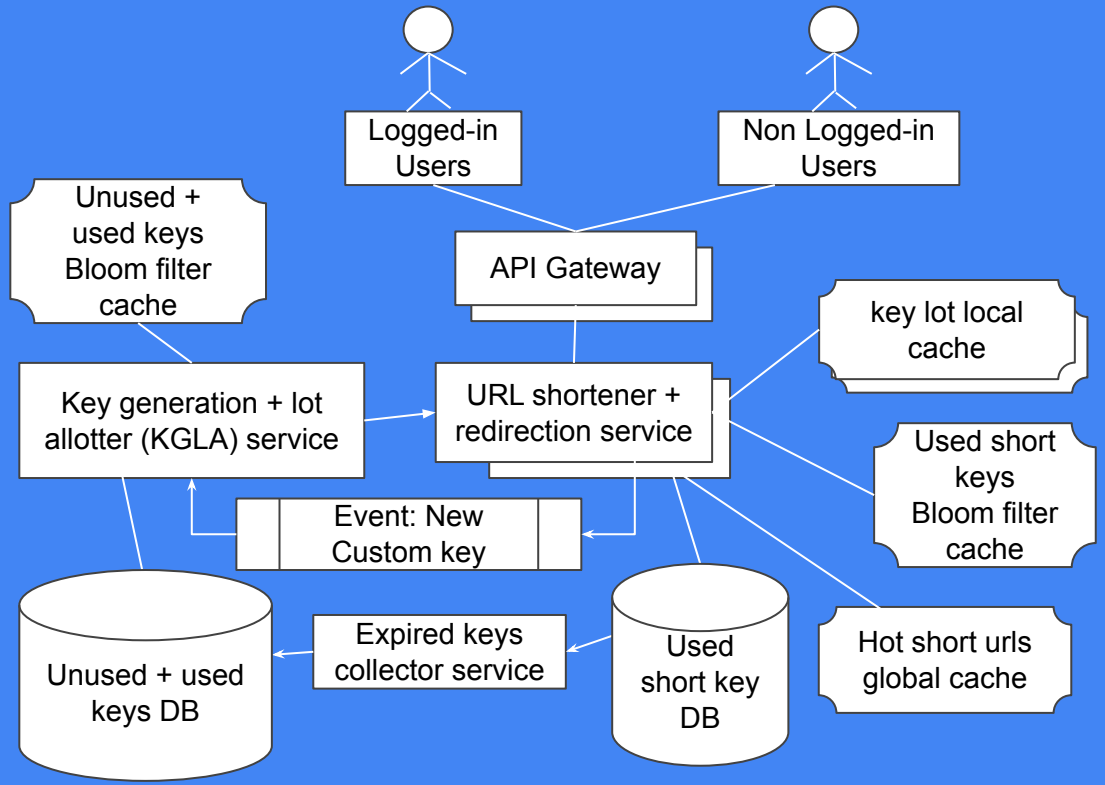
Can we collect expired short url keys for reuse?

https://www.youtube.com/@15minutesystemdesign





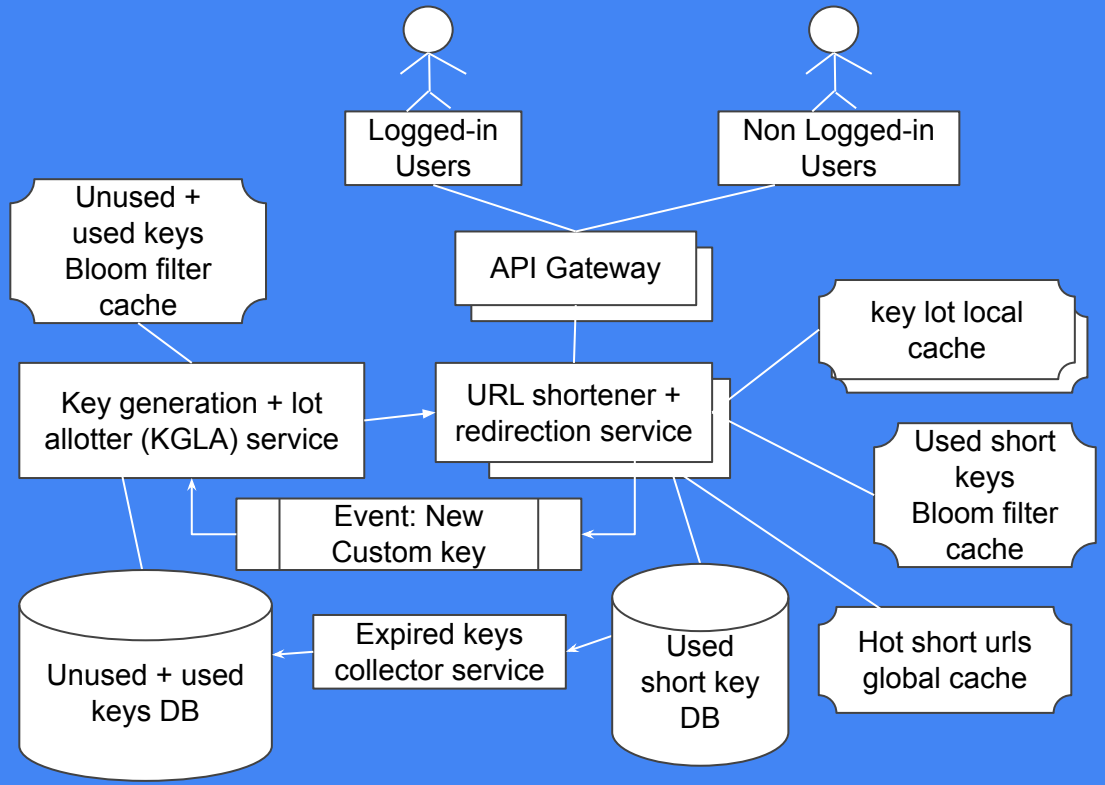
https://www.youtube.com/@15minutesystemdesign



Offline key generation Algorithm -

Check periodically if unused + used keys db need to be refilled
If yes repeat the following until refill count is achieved
(a) Generate random 7 character long alphanumeric key
(b) Check in unused + used keys bloom filter cache if "key definitely not present"
(c) If true then save in Unused + used keys db and update unused + used keys bloom filter cache
(d) If false repeat from step (a)

https://www.youtube.com/@15minutesystemdesign



Offline key generation Algorithm -

Check periodically if unused + used keys db need to be refilled
If yes repeat the following until refill count is achieved
(a) Generate random 7 character long alphanumeric key
(b) Check in unused + used keys bloom filter cache if "key definitely not present"
(c) If true then save in Unused + used keys db and update unused + used keys bloom filter cache
(d) If false repeat from step (a)

URL Shortening Algorithm -

(a) If no custom short key is provided, remove one key from keys lot local cache. This key would be definitely not present in used short keys. Goto step (c) directly
(b) Else if custom key is provided, check in used short key bloom filter cache if "key definitely not present"
(c) If true, update short key mapping to given long url in used short key DB, update used short keys bloom filter cache. Publish event for kgla service to mark custom key being used in unused+used keys db and associated bloom filter cache
(d) If false throw exception "custom key already exist"

https://www.youtube.com/@15minutesystemdesign

Like, share, Subscribe

https://www.youtube.com/@15minutesystemdesign