Design a URL Shortener

- Step 1 - Understand the problem and establish design scope
  - Here are the basic use cases:
    - URL shortening: given a long URL => return a much smaller URL.
    - URL redirecting: given a shorter URL => redirect to the original URL.
    - High availability, scability and fault tolerance considerations.

  - Back of the envelope estimation
    - Write operation: 100 million URLs are generated per day.
    - Write operations per second: 100 million /24/3600 = 1160
    - Read operation: Assuming ratio of read operation to write operation is 10:1, read operation per second: 1160 * 10 = 11600.
    - Assuming the URL shortner service will run for 10 years, this means we must support 100 million * 365 * 10 = 365 billion records.
    - Assume average URL length is 100.
    - Storage requirement over 10 years: 365 billion * 100 bytes = 36.5 TB.

- Step 2 - Propose high-level design and get buy-in

  - API Endpoints
    - A URL shortner primary needs two API endpoints:

    1. URL shortening: To create a new short URL, a client sends a POST request, which contains one parameter: the original long URL. The API looks like this:

    POST api/v1/data/shorten
      - request parameter: [longURL : string]

- return shortURL

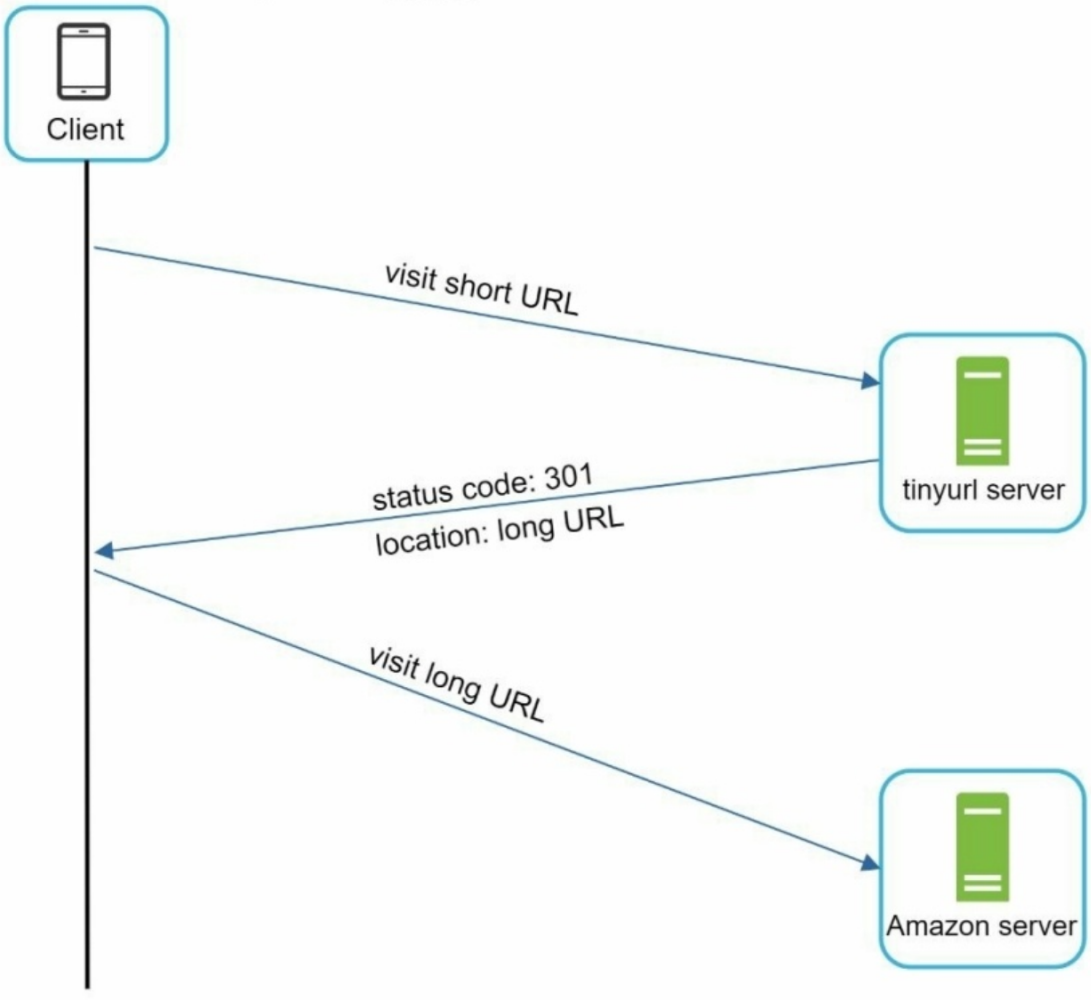
2. URL redirecting: To redirect a short URL to the corresponding long URL, a client sends a GET request. The API looks like this:

GET api/v1/shortURL
- Return longURL for HTTP redirection.

- URL Redirecting
- Once the server receives a tinyurl request, it changes the short URL to the long URL with 301 redirect.

short URL: https://tinyurl.com/qtj5opu

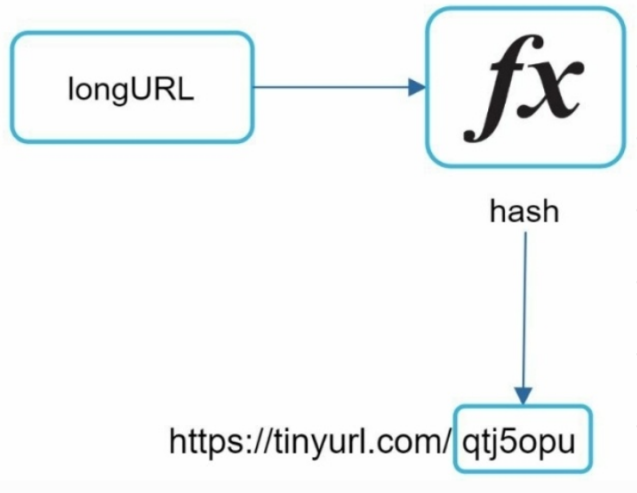
long URL: https://www.amazon.com/dp/B017V4NTFA?pLink=63eaef76-979c-4d&ref=adblp13nvvxx_0_2_im



- 301 redirect vs 302 redirect:
  - 301 redirect: A 301 redirect shows that the requested URL is "permanently" moved to the long URL. Since it is permanently redirected, the browser caches the response, and subsequent requests for the same URL will not be sent to the URL shortening service. Instead, requests are redirected to the long URL server directly.
  - 302 redirect: means that the URL is "temporarily" moved to the long URL, meaning that subsequent requests for the same URL will be sent to the URL shortening service first. Then, they are redirected to the long URL server.

- Each redirection method has its pros and cons. If the priority is to reduce server loads, using 301 redirect makes sense as only the first request of the same URL is sent to URL shortening servers. However, if analytics is important, 302 redirect is a better choice as it can track click rate and source of the click more easily.
- Most intuitive way to implement URL redirecting is to use hash tables. Assuming the hash table stores <shortURL, longURL> pairs, URL redirecting can be implemented by the following:
  - Get longURL: longURL = hashTable.get(shortURL)
  - Once you get the longURL, perform the URL redirect.

- URL Shortening
  - To support the URL shortening use case, we must find a hash function fx that maps a longURL, to the hashValue -

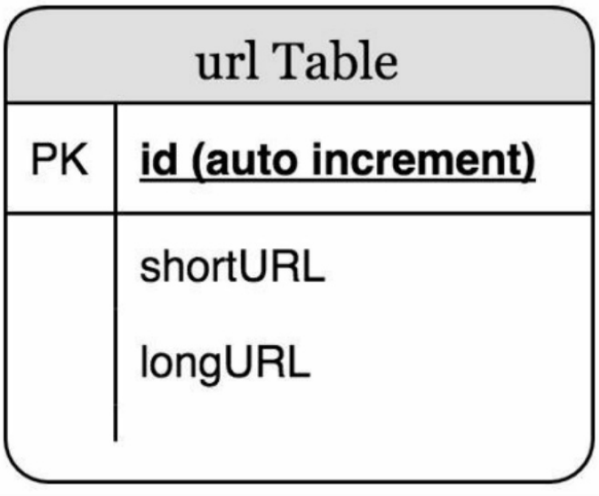


  - The hash function must satisfy the following requirement:
    - Each longURL must be hashed to one hashValue.
    - Each hashValue can be mapped back to the longURL.

- Step 3 - Design Deep Dive

- Data Model

- In the high level design, everything is stored in a hash table, however, this approach is not feasible for real-world systems as memory resources are limited and expensive. A better option is to store <shortURL, longURL> mapping in a relational database.



- Hash Function

- Hash function is used to hash a long URL to a short URL, also known as hashValue.
- Hash Value Length
  - The hash value consists of characters from [0-9, a-z, A-Z] = 62 possible characters. To figure out the length of hashValue, find smallest n such that $62^n \geq 365$ billion.

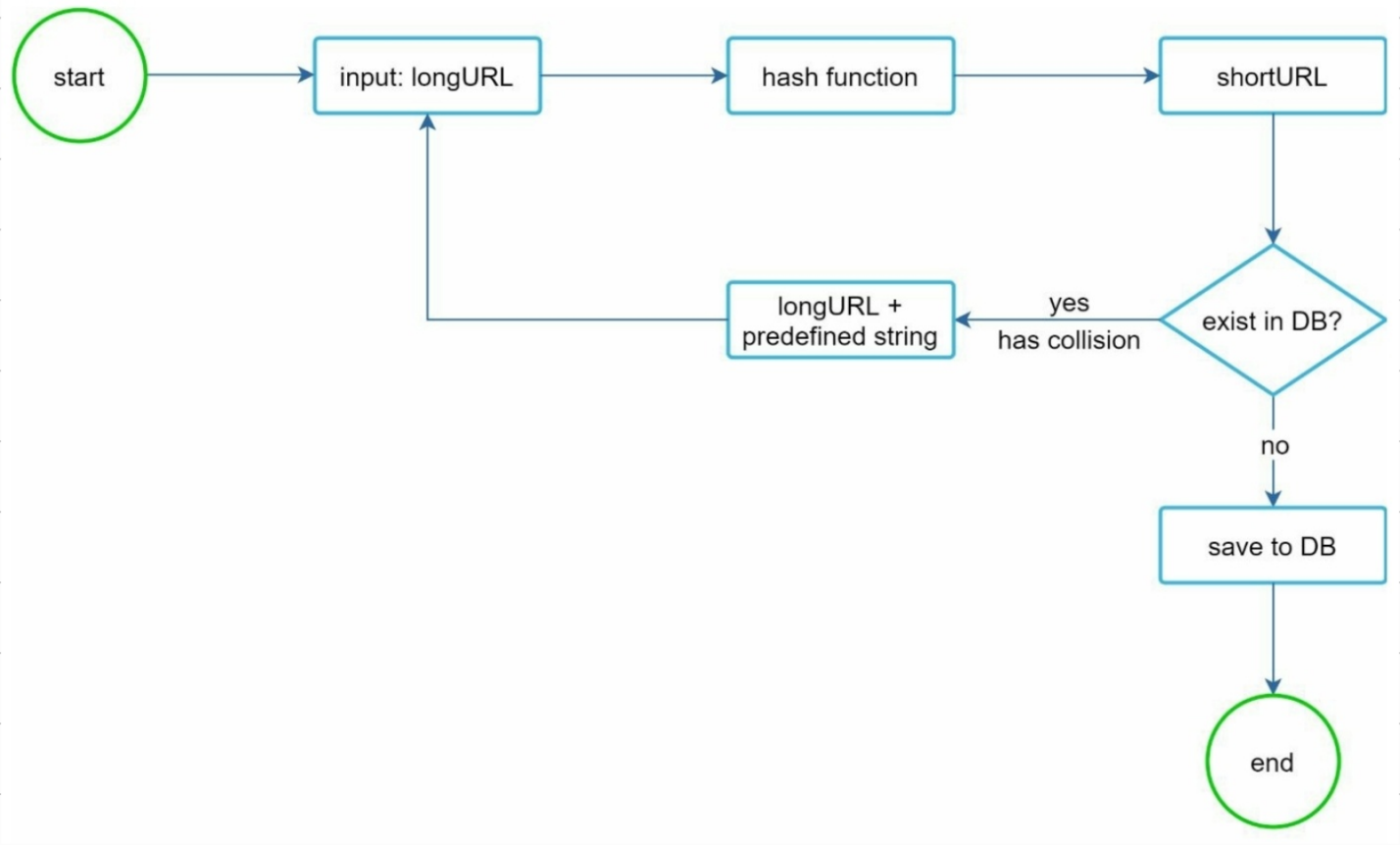
N	Maximal number of URLs
1	62^1 = 62
2	62^2 = 3,844
3	62^3 = 238,328
4	62^ 4 = 14,776,336
5	62^5 = 916,132,832
6	62^6 = 56,800,235,584
7	62^7 = 3,521,614,606,208 = ~3.5 trillion

- When $n = 7$, $62^7 = 3.5$ trillion.

- Hash + Collision Resolution
  - To shorten a long URL, we should implement a hash function that hashes a long URL to a 7 character string. A straightforward solution is to use well-known hash functions like CRC32, MD5 or SHA-1.

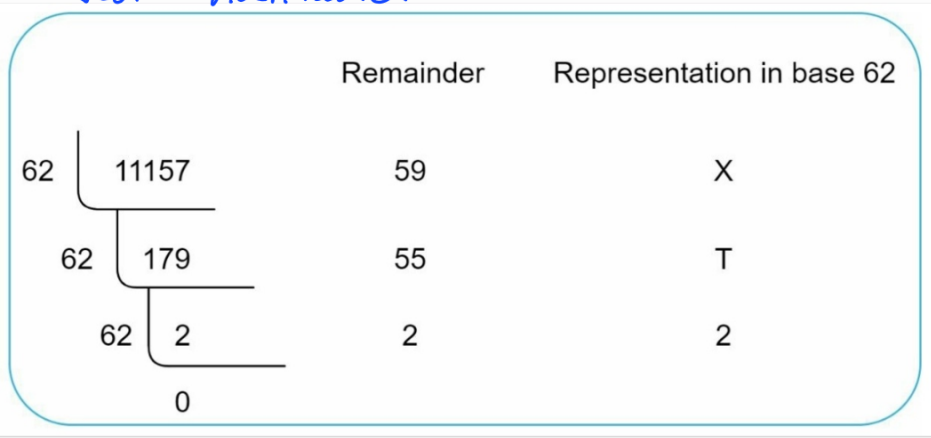
Hash function	Hash value (Hexadecimal)
CRC32	5cb54054
MD5	5a62509a84df9ee03fe1230b9df8b84e
SHA-1	0eeae7916c06853901d9ccbefbfcaf4de57ed85b

  - First approach to shorten the hash values is to collect the first 7 characters of a hash value; however this can lead to hash collisions. To resolve hash collisions, we can recursively append a new predefined string until no more collisions is discovered.



- This method can eliminate collisions; however it is expensive to query the database to check if a shortURL exists for every requests. A technique called bloom filters can improve performance.

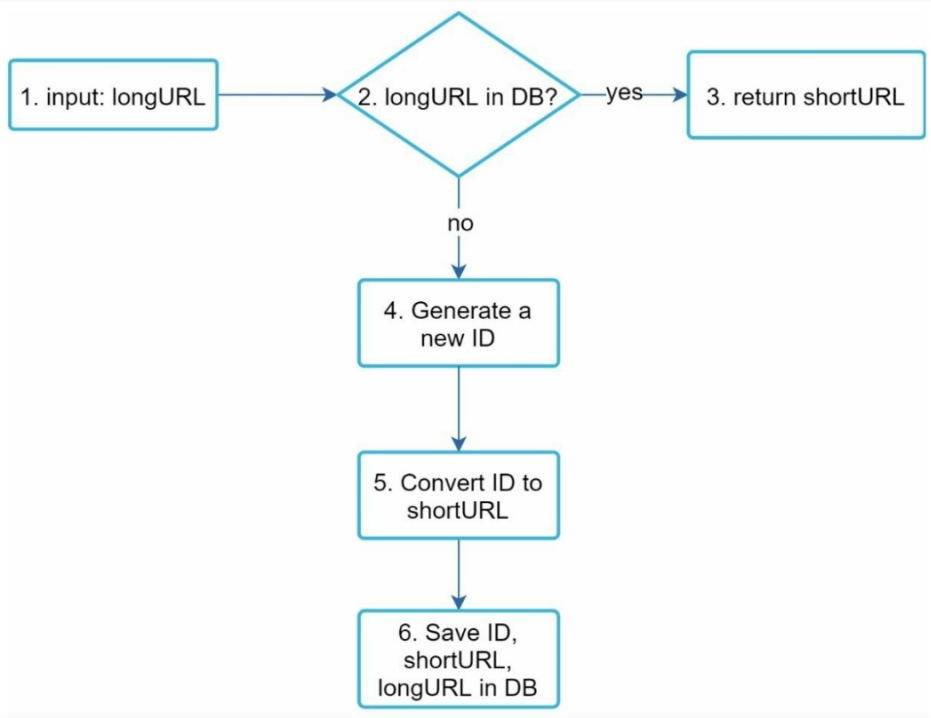
- Base 62 conversion
  - Base conversion helps to convert the same number between its different number representation systems.
  - Base 62 conversion is used as there are 62 possible characters for hashValue.



- Comparison of the 2 approaches

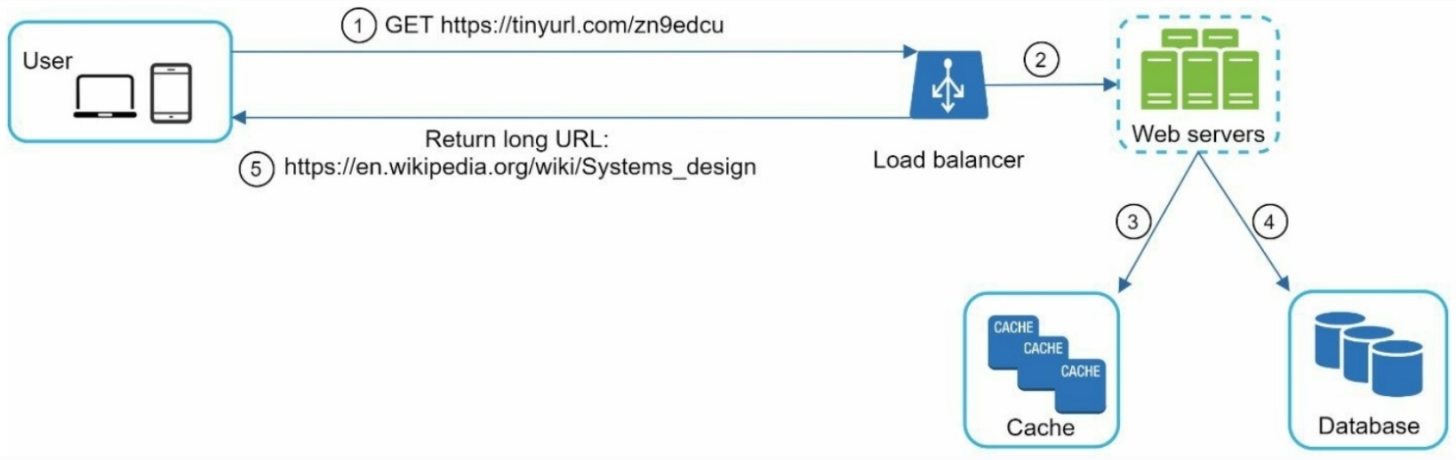
Hash + collision resolution	**Base 62 conversion**
Fixed short URL length.	The short URL length is not fixed. It goes up with the ID.
It does not need a unique ID generator.	This option depends on a unique ID generator.
Collision is possible and must be resolved.	Collision is impossible because ID is unique.
It is impossible to figure out the next available short URL because it does not depend on ID.	It is easy to figure out the next available short URL if ID increments by 1 for a new entry. This can be a security concern.

- URL Shortening Deep Dive



1. longURL is the input.
2. The system checks if the longURL is in the database.
3. If it is, it means the longURL was converted to shortURL before, fetch the shortURL from the database and return it to the client.
4. If not, the longURL is new. A new unique ID is generated.
5. Convert the ID to shortURL with base62 conversion.
6. Create a new database row with the ID, shortURL and longURL.

- URL Redirecting Deep Dive



- The flow of URL redirecting is summarized as follows:
  1. User clicks a shortURL link.
  2. The load balancer forwards the request to web servers.
  3. If a shortURL is already in the cache, return the longURL directly.
  4. If a shortURL is not in the cache, fetch the longURL from the database. If not in database, it is likely a user entered an invalid shortURL.
  5. The longURL is returned to the user.

- Step 4 - Wrap Up
  - Rate Limiter: A potential security problem we could face is that malicious users send an overwhelmingly large number of URL shortening requests. Rate Limiter helps to filter out requests based on IP addresses or other rules.
  - Web Server Scaling: Since the web tier is stateless, it is easy to scale the web tier by adding or removing web servers.
  - Database Scaling: Database replication and sharding are common techniques.
  - Analytics: Data is increasingly important for business success.
  - Availability, Consistency and Reliability.