Requirements of Twitter’s Design

Understand the requirements and estimation for Twitter's design.

We'll cover the following

Requirements

Let’s understand the functional and non-functional requirements below:

Functional requirements

The following are the functional requirements of Twitter:

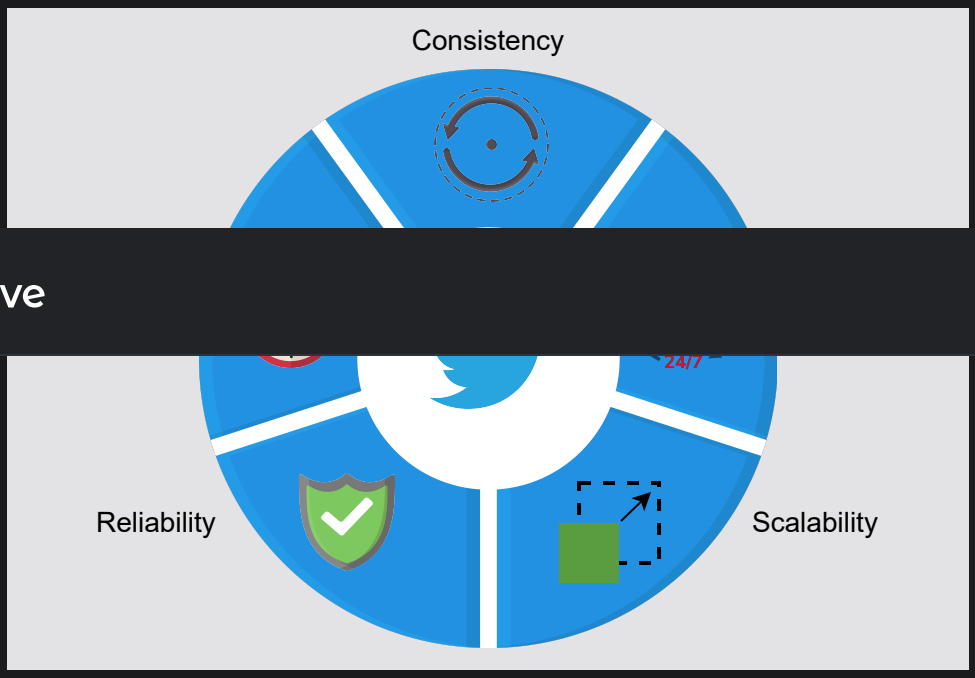
- **Post Tweets**: Registered users can post one or more Tweets on Twitter.
- **Delete Tweets**: Registered users can delete one or more of their Tweets on Twitter.
- **Like or dislike Tweets**: Registered users can like and dislike public and their own Tweets on Twitter.
- **Reply to Tweets**: Registered users can reply to the public Tweets on Twitter.
- **Search Tweets**: Registered users can search Tweets by typing keywords, hashtags, or usernames in the search bar on Twitter.

- **View user or home timeline**: Registered users can view the user's timeline, which contains their own Tweets. They can also view the home's timeline, which contains followers' Tweets on Twitter.
- **Follow or unfollow the account**: Registered users can follow or unfollow other users on Twitter.
- **Retweet a Tweet**: Registered users can Retweet public Tweets of other users on Twitter.

Non-functional requirements

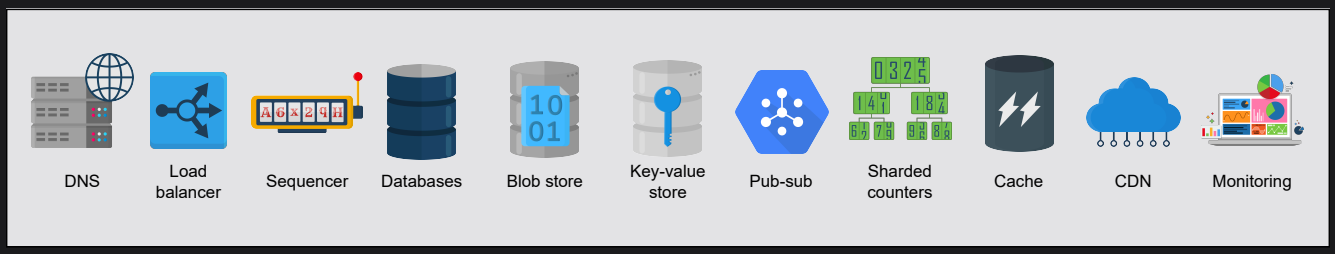
- **Availability**: Many people and organizations use Twitter to communicate time-sensitive information (service outage messages). Therefore, our service must be highly available and have a good uptime percentage.
- **Latency**: The latency to show the most recent top Tweets in the home timeline could be a little high. However, near real-time services, such as Tweet distribution to followers, must have low latency.
- **Scalability**: The workload on Twitter is read-heavy, where we have many readers and relatively few writers, which eventually requires the scalability of computational resources. Some estimates suggest a 1:1000 write-to-read ratio for Twitter. Although the Tweet is limited to 280 characters, and the video clip is limited to 140s by default, we need high storage capacity to store and deliver Tweets posted by public figures to their millions of followers.
- **Reliability**: All Tweets remain on Twitter. This means that Twitter never deletes its content. So there should be a promising strategy to prevent the loss or damage of the uploaded content.
- **Consistency**: There's a possibility that a user on the east coast of the US does not get an immediate status (like, reply, and so on.) update on the Tweet, which is liked or Retweeted by a user on the west coast of the US. However, the user on the west coast of the US needs an immediate status update on his like or reply. An effective technique is needed to offer rapid feedback to the user (who liked someone's post), then to other specified

users in the same region, and finally to all worldwide users linked to the Tweet.



Non-functional requirements of Twitter

Furthermore, we must identify which essential resources must be estimated in the Twitter design. We used Twitter as an example in our Back-of-the-Envelope chapter, so we won't repeat the resource estimation exercise here.



Building blocks used in Twitter design

- **DNS** is the service that maps human-friendly Twitter domain names to machine-readable IP addresses.
- **Load balancers** distribute the read/write requests among the respective services.
- **Sequencers** generate the unique IDs for the Tweets.
- **Databases** store the metadata of Tweets and users.

- **Blob stores** store the images and video clips attached with the Tweets.
- **Key-value stores** are used for multiple purposes such as indexing, identifying the specified counter to update its value, identifying the Tweets of a particular user and many more.
- **Pub-sub** is used for real-time processing such as elimination of redundant data, organizing data, and much more.
- **Sharded counters** help to handle the count of multiple features such as viewing, liking, Retweeting, and so on,. of the accounts with millions of followers.
- A **cache** is used to store the most requested and recent data in RAM to give users a quick response.
- **CDN** helps end users to access the data with low latency.
- **Monitoring** analyses all outgoing and incoming traffic, identifies the redundancy in the storage system, figures out the failed node, and so on.

In the next lesson, we'll discuss the design of the Twitter system.

Back

Completed

Next

System Design: Twitter

High-level Design of Twitter