Best Practices to Follow

1. Solidify the requirements – both Functional and Non-Functional
2. Scope the Problem
3. Capacity/ Resource Estimation
4. System API’s
5. Database Design

* Identify and understand data and its characteristics to look for appropriate data storage systems and data processing components for the system design.
* Some important questions to ask ourselves when searching for the right systems and components include the following:
  + What’s the size of the data right now?
  + At what rate is the data expected to grow over time?
  + Is the data read-heavy or write-heavy?
  + How will the data be consumed by other subsystems or end users?
  + Do we need strict consistency of data, or will eventual consistency work?

1. Present the building blocks of the Design
   * Figuring out which components we’ll use, where they’ll be placed, and how they’ll interact with each other.
2. Propose a Design Diagram and get an agreement

* Elaborate on why some component was used instead of another—for example, Firebase versus SQL.

1. Workflow
2. Discuss specific design components
3. Design Evaluation
4. Additional Details

**1. Solidify the requirements – both Functional and Non-Functional**

Functional Requirements:

1. What features are important? What is the desired output?
2. Extra features like custom-URL? Default timespan? User specific expiration date?
3. Any Size limits? Paste size?

Non-Functional Requirements:

1. Highly available
2. Minimum latency
3. Consistency can take a hit (CAP Theorem)
4. Service should be real-time with minimum latency
5. **Scope the Problem**
6. What Clients do we need to support? Mobile apps, Web browsers, smart TV, Tablets.
7. Do we need to support international users?
8. Is encryption required?
9. What are the supported video resolutions?
10. **Capacity Estimation and Constraints**
11. Traffic Estimates
12. How many daily active users (DAU’s)?
13. If service like URL, pastebin, etc., then how many services/requests per month/day?
14. If Read/ Write is performed, then Read/Write Ratio?
15. How many RPS (requests/ second) can a server handle?

From the above information you can make Traffic estimates (QPS) and Total Servers Required

* QPS = Total Requests per day / 86400 secs
* Total Server Required = Number of Daily Active users / RPS of a server

1. Storage Estimates
2. Time duration for which objects are required to be stored in years?
3. Usage per day? E.g., Tweets/ day, Average time spent on the product/day?
4. File size requirement? E.g., Image or Video Average size?
5. Distribution of the requests/day? E.g., Ten percent of the tweets contain images, whereas five percent of the tweets contain a video.
6. Storage required to store a metadata? E.g., The tweet text and its metadata require a total of 250 Bytes of storage in the database.

From the above information you can calculate Total storage required for one day/ year

* Total Storage/ day = (Metadata storage + Photo storage + Video storage) / day
* Total Storage/ year = Storage/ day x Days/ year

1. Bandwidth Estimates

For Bandwidth Estimates use the following steps:

1. Estimate the daily amount of incoming data to the service.
2. Estimate the daily amount of outgoing data from the service.
3. Estimate the bandwidth in Gbps (gigabits per second) by dividing the incoming and outgoing data by the number of seconds in a day.
4. How many reads/ day (if read/write ratio is not provided in q3 of Traffic estimates)

From the above information you can calculate Incoming traffic and Outgoing traffic

* Incoming Traffic (always consider write data. E.g., Tweets/Videos posted) = (Total Storage/ day (calculated in Storage Estimates)/ 86400 secs) x 8 Gbps
* Outgoing Traffic (always consider read data. E.g., Tweets/ Videos viewed) = (Total Outgoing data/ day) x 8 Gbps

1. Memory Estimates
2. Ask if 80-20 rule is acceptable?

From the above information you can calculate Size of the Cache required to store hot storage/ day

* Cache Size = (Total incoming requests per day (calculated from QPS in traffic estimates) x Avg. Size of the request (e.g. tweets including videos and images)) x 0.2

1. **System APIs**

* Briefly Discuss what API’s will be needed and what results will be returned
* Any extra APIs needed?

1. **Database Design**

* Understand nature of the data
* Discuss which database will be used SQL Vs NoSQL?
* Identify and understand data and its characteristics to look for appropriate data storage systems and data processing components for the system design.
* Some important questions to ask ourselves when searching for the right systems and components include the following:
* What’s the size of the data right now?
* At what rate is the data expected to grow over time?
* Is the data read-heavy or write-heavy?
* How will the data be consumed by other subsystems or end users?
* Do we need strict consistency of data, or will eventual consistency work?

1. **Present the building blocks of the Design**

* Figuring out which components we’ll use, where they’ll be placed, and how they’ll interact with each other.

1. **Propose a Design Diagram and get an agreement**

* Elaborate on why some component was used instead of another—for example, Firebase versus SQL.

1. **Workflow**
2. **Discuss specific design components**
3. **Design Evaluation**
4. **Additional Details**