(1) FEATURE EXPECTATIONS [5 min]

---

1. Use cases
2. Scenarios that will not be covered
3. Who will use
4. How many will use
5. Usage patterns

(2) ESTIMATIONS [5 min]

---

1. Throughput (QPS for Read and Write queries)
2. Latency expected from the system (for read and write queries)
3. Read/Write ratio
4. Traffic estimates
    - Write (QPS, Volume of data)
    - Read (QPS, Volume of data)
5. Storage estimates
6. Memory estimates
    - If we are using a cache, what is the kind of data we want to store in the cache
    - How much RAM and how many machines do we need for us to achieve this?
    - Amount of data you want to store in disk/SSD

(3) DESIGN GOALS [5 min]

---

1. Latency and Throughput requirements
2. Consistency vs Availability [Weak/strong/eventual => consistency | Failover/replication => availability]

(4) HIGH-LEVEL DESIGN [5-10 min]

---

1. APIs for Read/Write scenarios for crucial components
2. Database schema
3. Basic algorithm
4. High-level design for Read/Write scenario

(5) DEEP DIVE [15-20 min]

---

1. Scaling the algorithm
2. Scaling individual components:
    - Availability, Consistency and Scale story for each component
    - Consistency and availability patterns
3. Think about the following components, how they would fit in and how it would help
    a) DNS
    b) CDN [Push vs Pull]
    c) Load Balancers [Active-Passive, Active-Active, Layer 4, Layer 7]
    d) Reverse Proxy
    e) Application layer scaling [Microservices, Service Discovery]
    f) DB [RDBMS, NoSQL]
        - RDBMS
            - Master-slave, Master-master, Federation, Sharding, Denormalization, SQL Tuning
        - NoSQL
            - Key-Value, Wide-Column, Graph, Document
            - Fast-lookups:
                - RAM [Bounded size] => Redis, Memcached
                - AP [Unbounded size] => Cassandra, RIAK,Voldemort
                - CP [Unbounded size] => HBase, MongoDB,Couchbase, DynamoDB
    g) Caches
        - Client caching, CDN caching, Web server caching, Database caching, Application caching, Cache @Query level, Cache @Object level
        - Eviction policies:
            - Least Recently Used(LRU)
            - Least Frequently Used(LFU)
            - First in First Out (FIFO)
        - Cache Loading Policies
            - Cache aside
            - Write through
            - Write behind
            - Refresh ahead
    h) Asynchronism
        - Message queues
        - Task queues
        - Backpressure
    i) Communication
        - TCP, UDP, REST, RPC, Thrift, GraphQL

(6) JUSTIFY [5 min]

---

1. Throughput of each layer
2. Latency caused between each layer
3. Overall latency justification