Yeh **System Design Interview** ka sabse crucial phase hai: **Step 1 - Clarify Requirements.**

Interview mein sabse badi galti log yeh karte hain ki sawal sunte hi system design karna shuru kar dete hain. **Ruko!** Pehle interviewer se poocho ki exactly chahiye kya.

Yahan candidate wahi kar raha hai—woh boundaries set kar raha hai. Har sawal ka main detailed reason samjhata hoon, beginner perspective se.

---

### Phase 1: The Interview Conversation (Sawal Jawab)

Niche har point ka matlab aur uska "Why" (Kyu pucha gaya) samjhaya gaya hai:

#### 1. Message Size & Format

> ***Candidate:*** *Message ka format kya hai? Text ya Multimedia? Size kya hogi?* ***Interviewer:*** *Sirf Text. Size KBs (Kilobytes) mein hogi.*

- **Beginner Explanation:**
  - **Kyu pucha?** Agar message video/image hota (MBs/GBs), toh hume **Blob Storage** (jaise AWS S3) chahiye hota. Lekin kyunki yeh chota text (KB) hai, hum ise seedha disk ya memory mein database ki tarah store kar sakte hain.
  - **Impact:** System fast hoga kyunki data chota hai.

#### 2. Repeated Consumption (Bar-bar padhna)

> ***Candidate:*** *Kya ek message ko baar-baar padha ja sakta hai?* ***Interviewer:*** *Haan. (Note: Yeh traditional Queue nahi,* ***Event Streaming*** *feature hai).*

- **Beginner Explanation:**
  - **Traditional Queue (RabbitMQ):** Message padha -> Delete ho gaya. (Jaise Snapchat).
  - **Streaming (Kafka):** Message padha -> Wahan hi rahega taaki dusra banda bhi padh sake. (Jaise WhatsApp Group chat).
  - **Reason:** Candidate yeh check kar raha hai ki mujhe data delete karna hai ya store karke rakhna hai. Interviewer ne kaha "Store rakhna hai", matlab design complex hoga.

#### 3. Message Ordering (Sequence)

> ***Candidate:*** *Kya jis order mein message aaye, usi order mein process hone chahiye?* ***Interviewer:*** *Haan, order maintain karna zaroori hai.*

- **Beginner Explanation:**
  - **Problem:** Distributed system mein 10 servers hote hain. Agar Server A aur Server B dono message bhej rahe hain, toh kaun pehle pahuncha yeh decide karna mushkil hota hai.
  - **Impact:** "Strict Ordering" mangna matlab system slow ho sakta hai, kyunki hum parallel processing kam kar payenge. (Example: Bank transaction - pehle 'Credit' hona chahiye fir 'Debit', ulta nahi ho sakta).

#### 4. Data Retention (Kab tak save rakhna hai)

> ***Candidate:*** *Data kab tak save rakhna hai?* ***Interviewer:*** *2 weeks tak.*

- **Beginner Explanation:**
  - **RAM vs Disk:** Agar data 2 hafte rakhna hai, toh RAM (Memory) mein nahi rakh sakte (mehnga aur volatile hai). Hume ise **Hard Disk (Persistent Storage)** pe likhna padega.
  - **Reason:** Isse confirm hua ki hume heavy storage management design karna padega.

#### 5. Delivery Semantics (Guarantee)

> ***Candidate:*** *Delivery guarantee kya honi chahiye?* ***Interviewer:*** *"At-least-once" zaroori hai. Ideally sab support karo.*

- **Beginner Explanation:** Yeh bahut important concept hai:

- **At-most-once:** Message bheja, pahuncha toh theek, nahi toh jaane do. (Like Live Video streaming frame - ek choot gaya toh koi baat nahi).
- **At-least-once:** Message pahunchna hi chahiye, chahe do baar chala jaye. (Like Payment notification - paise kate hain toh user ko pata chalna hi chahiye, chahe 2 SMS aa jayein).
- **Exactly-once:** Message ek hi baar process ho. (Like E-commerce order - do baar order place nahi hona chahiye).
- **Impact:** "Exactly-once" achieve karna sabse mushkil aur slow hota hai.

---

### Phase 2: Requirements Breakdown

Jab conversation khatam hui, toh humne requirements ko do hisso mein baata:

#### 1. Functional Requirements (System *kya* karega)

Yeh user features hain jo dikhte hain:

- Producer message bhejega.
- Consumer message padhega.
- Puraana data delete kar sakte hain (Truncate).
- Message order mein aayenge.
- User choose karega ki use "Guarantee" chahiye ya "Speed".

#### 2. Non-Functional Requirements (System *kaisa* hoga)

Yeh performance aur quality attributes hain:

- **High Throughput vs Low Latency:**
  - *Throughput:* Ek second mein kitne message handle kar sakte hain (e.g., 1 Million logs/sec).
  - *Latency:* Ek message ko pahunchne mein kitna time lagta hai (e.g., 2ms for payment).
  - Interviewer keh raha hai ki hume dono cases support karne hain.
- **Scalable:** Agar achanak se traffic 10x ho jaye (e.g., IPL match start hote hi), toh system crash nahi hona chahiye. Naye servers add karke sambhal lena chahiye.
- **Persistent & Durable:** "Durable" ka matlab hai mazboot. Agar server mein aag lag jaye ya power cut ho jaye, toh data loose nahi hona chahiye. Isliye hum data ki copy (Replication) alag-alag nodes pe rakhenge.

---

### Phase 3: Traditional vs Modern (Difference Samjho)

Text ke last mein ek critical difference samjhaya hai jo interview ke liye "Gold" hai.

**Traditional Queue (RabbitMQ):**

- Designed to empty quickly.
- Data RAM mein rehta hai.
- Consumer ne khaya, data gaya.
- Order aage-peeche ho sakta hai.
- **Design:** Simple hota hai.

**Modern Streaming (Kafka/Pulsar - Jo hum design kar rahe hain):**

- Designed to hold data (Log storage).
- Data Disk par rehta hai.
- Consumer ne khaya, data phir bhi wahan hai (Retention).
- Strict ordering hoti hai partition ke andar.
- **Design:** Complex hota hai kyunki hume storage aur replication manage karna padta hai.

### Summary for Interview

Is step ka maqsad yeh dikhana tha ki aap **RabbitMQ** nahi, balki **Apache Kafka** jaisa system design karne ja rahe ho. Interviewer ne jo "Added Features" (Retention, Replay, Ordering) mange hain, woh saare Kafka ke core features hain.