# System Design Interview Q&As

## 🔷 Section 1: Scalability & Basics

**Q1. What is system design?**
**Answer:** System design is about building scalable, reliable, and maintainable systems by balancing architecture, databases, and performance trade-offs.
**Example:** Designing Instagram with millions of users.
💡 Tip: Interviewers care about *your thought process*, not memorization.

**Q2. What is scalability?**
**Answer**: Scalability means handling growing load by adding more resources — vertical = bigger machine, horizontal = more machines.
**Example**: Amazon adding servers during Black Friday.
💡 Tip: Say *cloud-native apps prefer horizontal scaling*.

**Q3. Difference between vertical vs horizontal scaling?**
**Answer**: Vertical = upgrade CPU/RAM of one machine, Horizontal = add multiple servers. Horizontal is more fault-tolerant.
**Example** : Facebook uses horizontal scaling for billions of requests.
💡 Tip: Always mention horizontal scaling for modern apps.

**Q4. What is high availability?**
**Answer**: High availability ensures systems stay online with minimal downtime using replication and failover.
**Example**: Netflix is always available through multi-region failover.
💡 Tip: Say “HA = reliability + business continuity.”

**Q5. What is fault tolerance?**
**Answer**: Fault tolerance means the system continues working despite failures using redundancy and retries.
**Example**: Google Cloud runs across zones with auto-failover.
💡 Tip: “No single point of failure” is the key phrase.

## 🔷 Section 2: Load Balancing

**Q6. What is a load balancer?**

**Answer**: A load balancer distributes traffic across servers to prevent overload and improve availability.

**Example**: AWS ELB routes traffic evenly.

💡 Tip: Mention performance + fault tolerance.

**Q7. Difference between L4 vs L7 load balancers?**

**Answer**: L4 balances traffic based on IP/TCP, while L7 works at HTTP/HTTPS and routes by content.

**Example**: L7 sends /images to image servers.

💡 Tip: Say "L7 = smarter, L4 = faster."

**Q8. What are sticky sessions in load balancers?**

**Answer**: Sticky sessions tie users to the same server for session consistency.

**Example**: Shopping cart tied to one server.

💡 Tip: Mention drawback — "less scalable."

**Q9. How do DNS and load balancers work together?**

**Answer**: DNS resolves domain → LB → routes traffic to servers.

**Example**: google.com → global DNS → LB → server.

💡 Tip: DNS + LB is the backbone of global services.

**Q10. Global vs Local load balancing?**

**Answer**: Local = distributes traffic in one data center, Global = across multiple regions.

**Example**: Cloudflare for global LB.

💡 Tip: "Global LB = geo-based routing."

## 🔷 Section 3: Caching

**Q11. What is caching?**

**Answer**: Caching stores frequently accessed data in fast storage to reduce response time.

**Example**: Redis cache for user sessions.

💡 Tip: Always link caching to *speed + cost savings*.

**Q12. Client-side vs Server-side caching?**

**Answer**: Client-side = browser/CDN, Server-side = DB query cache, Redis.

**Example**: Browser cache images; Redis caches queries.

💡 Tip: Show both for full answer.

**Q13. What are cache eviction policies?**

**Answer**: Methods to remove old data: LRU (Least Recently Used), LFU, FIFO.

**Example**: Redis uses LRU by default.

💡 Tip: Say "LRU = most common."

**Q14. CDN vs Cache?**

**Answer**: CDN = caching at the edge (closer to users), cache = app/database level.

**Example**: Cloudflare CDN vs Redis cache.

💡 Tip: CDN = global reach, cache = local speed.

**Q15. What is cache invalidation?**

**Answer**: Removing stale data from cache using TTL, write-through, or write-back.

**Example**: Expiring old product prices.

💡 Tip: Mention "cache invalidation is hard but critical."

## 🔷 Section 4: Messaging & Queues

**Q16. What is a message queue?**

**Answer**: A queue decouples producers and consumers for async processing.

**Example**: Order service → Payment via RabbitMQ.

💡 Tip: Always connect queues with *scalability*.

**Q17. Pub/Sub vs Message Queue?**

**Answer**: Queue = one-to-one, Pub/Sub = one-to-many subscribers.

**Example**: Kafka for Pub/Sub in microservices.

💡 Tip: Say "Pub/Sub = broadcast, Queue = task handoff."

**Q18. What is backpressure in queues?**

**Answer**: When consumers are slower than producers, messages pile up.

**Example**: Payment service lagging behind orders.

💡 Tip: Answer with "scale consumers or throttle producers."

**Q19. Dead letter queue (DLQ)?**

**Answer**: DLQ stores failed/unprocessed messages for debugging.

**Example**: Invalid payment messages in DLQ.

💡 Tip: Say "DLQ improves reliability."

**Q20. Why use queues in system design?**

**Answer**: Queues smooth spikes, allow retries, and enable async workflows.

**Example**: Email notifications via queue.

💡 Tip: "Queues = scalability + fault tolerance."

## 🔹 Section 5: Consistency & Reliability

**Q21. What is CAP theorem?**
**Answer**: In distributed systems, only 2 of Consistency, Availability, Partition Tolerance can be guaranteed.
**Example**: SQL = CA, DynamoDB = AP.
💡 Tip: Always mention trade-offs.

**Q22. Strong vs Eventual consistency?**
**Answer**: Strong = instant sync, Eventual = sync after some delay.
**Example**: SQL = strong, Amazon S3 = eventual.
💡 Tip: Use “shopping cart” as example.

**Q23. What is replication?**
**Answer**: Copying data across multiple servers for reliability and performance.
**Example**: Read replicas in MySQL.
💡 Tip: “Replication = high availability.”

**Q24. What is sharding?**
**Answer**: Splitting DB into smaller parts (shards) to scale horizontally.
**Example**: Users A–M in Shard1, N–Z in Shard2.
💡 Tip: “Sharding = scale but complex joins.”

**Q25. What is quorum in distributed systems?**
**Answer**: Minimum number of nodes needed to approve an operation.
**Example**: Raft/Paxos consensus.
💡 Tip: Say “quorum ensures consistency.”

## 🔹 Section 6: Best Practices

**Q26. How do you design for high traffic?**
**Answer**: Use load balancers, caching, queues, and CDNs with horizontal scaling.
**Example**: Twitter during trending events.
💡 Tip: Always say “design for peak load.”

**Q27. What is idempotency in APIs?**
**Answer**: Same request multiple times = same result (no duplicate effect).
**Example**: Payment retry won’t double charge.
💡 Tip: Mention *idempotency key*.

**Q28. How do you monitor large-scale systems?**

**Answer**: Use logging, metrics, and tracing tools like ELK, Prometheus, Grafana.

**Example**: Track API latency with Grafana.

💡 Tip: Always add *alerts + dashboards*.

**Q29. What is blue-green deployment?**

**Answer**: Two environments (Blue = live, Green = new). Switch traffic when new is stable.

**Example**: AWS CodeDeploy with Blue/Green.

💡 Tip: Say “safe CI/CD rollout.”

**Q30. What are common bottlenecks in large systems?**

**Answer**: DB queries, I/O latency, network bandwidth, poor architecture.

**Example**: Slow SQL query = app lag.

💡 **Tip**: Say “profiling + monitoring fix bottlenecks.”