System Design Interview Questions: Caching

What is caching, and why is it important in system design?

Caching is storing frequently accessed data in a temporary storage layer to speed up future data retrievals. It reduces latency, load on databases, and improves user experience.

What are the different types of caching in large-scale systems?

Types include in-memory caching (Redis, Memcached), CDN caching, application-level caching, browser caching, and database query caching.

How does Redis help with caching, and what are its advantages?

Redis is an in-memory key-value store known for low latency and high throughput. It supports TTLs, eviction policies, persistence, and complex data types.

Compare Redis and Memcached.

Redis supports persistence, complex data structures, and replication, while Memcached is simpler and faster for basic key-value use cases.

What is cache invalidation, and why is it hard?

Cache invalidation ensures stale data is removed from the cache. It's hard because timing, data consistency, and dependencies must be carefully managed.

How does TTL (Time To Live) work in caching?

TTL defines how long a cache item remains valid. Once expired, the cache is removed or refreshed from the source.

What is the difference between cache-aside and write-through caching?

In cache-aside, the app reads from cache and loads from DB on a miss. In write-through, writes go to both cache and DB simultaneously.

What's the difference between write-through and write-back cache?

Write-through writes data to both cache and DB. Write-back writes to cache first and syncs to DB later, improving speed but risking data loss.

How do CDNs perform caching at the edge?

CDNs cache static content on servers close to users. This reduces latency and server load by serving content from nearby locations.

How can caching introduce data consistency issues?

Stale data in cache may be returned if the cache isn't updated after a DB change. Synchronization is key.

What are LRU and LFU eviction strategies?

LRU (Least Recently Used) evicts the least recently accessed items. LFU (Least Frequently Used) removes items accessed the fewest times.

How do you implement a cache eviction policy in Redis?

Redis supports policies like noeviction, allkeys-lru, volatile-lru, allkeys-random, etc., which can be configured via the maxmemory-policy setting.

How do you cache database queries?

Store the result of expensive queries in a cache store (like Redis) and serve future requests from there. Invalidate or refresh the cache as needed.

How would you handle cache warming in a distributed system?

Preload popular or critical data into the cache after deployment or restart to prevent cold start delays.

How can you secure sensitive data stored in cache?

Encrypt sensitive data before storing in cache, set appropriate TTLs, and avoid caching personal or confidential information when not needed.

When would you *not* use caching?

Avoid caching for rapidly changing or highly sensitive data, or when data freshness is critical.

What are some common mistakes engineers make with caching?

Mistakes include over-caching, forgetting invalidation, storing large blobs, or relying solely on cache for critical data.

How does browser caching work?

Browsers cache static resources based on HTTP headers like Cache-Control, ETag, and Expires, improving client-side performance.

How do you ensure cache reliability in case of Redis failure?

Use Redis replication, clustering, or fall back to database on cache failure to ensure continuity.

How do you test caching logic during application development?

Use mocks for the cache layer, test cache hit/miss scenarios, TTL expiry, and eviction policy behaviors.