30 Most Important System Design Concepts

Curated by: Priya Bagde

## 1. Client-Server Architecture

A model where the client requests resources and the server provides them.

🔄 Example: Like ordering food at a restaurant—customer (client) orders, waiter (server) serves the meal. The client doesn’t cook the food, just requests and receives it.

## 2. IP Address

A unique identifier for a device on a network.

🔄 Example: Think of it as your home address used to receive letters or packages. Each device on the internet needs an address to send and receive data.

## 3. DNS

Converts domain names into IP addresses.

🔄 Example: Like a phonebook that translates names (like "Mom") into phone numbers. You type "google.com", DNS finds its actual IP to connect you.

## 4. Proxy / Reverse Proxy

Intermediaries that forward requests and responses between client and server.

🔄 Example: A receptionist that forwards your call to the right department. Reverse proxy is like a security guard checking calls before letting them through.

## 5. Latency

The time delay between request and response.

🔄 Example: Like the time between ordering a pizza and it being delivered. Lower latency means faster response, just like faster pizza delivery.

## 6. HTTP/HTTPS

Protocols used for communication over the web, HTTPS adds encryption.

🔄 Example: HTTP is like sending a postcard (anyone can read it), HTTPS is like a sealed envelope. HTTPS ensures secure and private communication.

## 7. APIs

Interfaces that allow software to communicate.

🔄 Example: Like a waiter taking your order and bringing it to the kitchen. You don’t need to know how the kitchen works—just use the menu (API).

## 8. REST API

An architectural style for stateless, cacheable client-server communication.

🔄 Example: Like ordering food where each request is complete and independent (e.g., “1 pizza to this table”). No need to remember previous orders.

## 9. GraphQL

A query language for APIs that allows clients to request specific data.

🔄 Example: Like building your custom burger instead of ordering a fixed menu. You get exactly what you want—nothing more, nothing less.

## 10. Databases

Systems used to store and manage data.

🔄 Example: Like a library storing books where each book is indexed and searchable. You can read, write, and organize information efficiently.

## 11. SQL vs NoSQL

SQL uses structured schema; NoSQL supports flexible and scalable data storage.

🔄 Example: SQL is like a well-organized school with fixed subjects and schedules; NoSQL is like a flexible workshop. Use SQL for structured data, NoSQL for dynamic or large-scale unstructured data.

## 12. Vertical Scaling

Increasing a single server’s capacity.

🔄 Example: Like upgrading your computer with more RAM and CPU. It improves performance but has a limit.

## 13. Horizontal Scaling

Adding more machines to distribute the load.

🔄 Example: Like hiring more chefs in a kitchen to serve more customers. Scales better for large systems.

## 14. Load Balancers

Distribute incoming traffic across multiple servers.

🔄 Example: Like a traffic police officer directing vehicles to less busy roads. Ensures no server gets overwhelmed.

## 15. Database Indexing

Speeds up data retrieval by creating pointers to data.

🔄 Example: Like a book index that helps you jump directly to the topic. Without it, you'd have to read every page.

## 16. Replication

Copying data across multiple servers for availability.

🔄 Example: Like having multiple copies of your keys—if you lose one, others still work. Helps with reliability and backups.

## 17. Sharding

Splitting a database into smaller parts for scalability.

🔄 Example: Like splitting a huge phonebook by city or surname to manage better. Each shard handles a subset of the data.

## 18. Vertical Partitioning

Splitting database columns into separate tables.

🔄 Example: Like separating personal info and order history into different folders. Improves performance and maintainability.

## 19. Caching

Storing frequently accessed data in-memory to reduce latency.

🔄 Example: Like saving your favorite contacts on speed dial. It avoids looking them up each time.

## 20. Denormalization

Combining tables to reduce joins and improve read performance.

🔄 Example: Like copying your address on multiple forms to avoid checking a separate record. Speeds up reads, but increases data redundancy.

## 21. CAP Theorem

Only two of Consistency, Availability, and Partition Tolerance can be guaranteed at once.

🔄 Example: Like choosing between being fast, accurate, or resilient—you can only pick two. No system can perfectly achieve all three at once.

## 22. Blob Storage

Stores unstructured data like images, videos, or files.

🔄 Example: Like a digital locker for dumping anything—PDFs, videos, backups. No strict format needed.

## 23. CDN

Distributes content geographically to reduce load time and latency.

🔄 Example: Like storing popular products in local stores instead of shipping them from HQ. Delivers faster by being closer to users.

## 24. WebSockets

Enables real-time, two-way communication between client and server.

🔄 Example: Like a phone call where both people can talk anytime. Great for live chat, games, and notifications.

## 25. Webhooks

Allows services to send real-time data to other systems via HTTP.

🔄 Example: Like receiving a text alert when your food order is ready. No need to keep checking manually.

## 26. Microservices

A software architecture where each service runs independently.

🔄 Example: Like a company with different departments (HR, sales, tech) working independently. Each service handles a specific function.

## 27. Message Queues

Allows asynchronous communication between services.

🔄 Example: Like a queue of notes passed between departments—each reads at its own pace. Ensures smooth processing without overloading.

## 28. Rate Limiting

Restricts the number of requests a client can make to prevent abuse.

🔄 Example: Like allowing only 5 free samples per customer to avoid misuse. Protects system from spam or attacks.

## 29. API Gateways

A single entry point for all API requests, handling routing, security, and more.

🔄 Example: Like a receptionist who handles all your requests and forwards them to the right team. Simplifies and secures access to backend services.

## 30. Idempotency

An operation that can be repeated without changing the result.

🔄 Example: Like pressing the elevator button multiple times—it still comes once. Essential for safe retries in APIs.