Whenever we are learning something, we have to ask about what, why and when. So now I am going through the system-design concepts. So what ever concepts I write, everywhere I will be using What, Why, and When.

Now let's start, what is system design and why are we using that and when are we supposed to think of this.

1. System design is the process of architecting a software system—just like how civil engineers design buildings. In the same way they decide on materials and structural strength, we design software systems to define how they will handle traffic, scale with users, interact with other components, and meet performance or reliability goals. It's about figuring out how a system should work, from high-level architecture down to component interaction.

2. Why to use? Because we have to architect the system flow in such a way we should not mess it up on the later stage, its better to have the entire design before start. We can avoid future bottlenecks and scalability issues. Without a proper design, systems can become hard to scale or debug later, especially as they grow.

3. When are we using? Simple while we are building any new software product/ service or when scaling an existing system.

Let's jump to types of system design.

We have mainly two types, one is High Level System Design often called as HLD and Low Level Design often called as LLD. But why is that called as HLD and LLD? Because HLD gives an aerial overview like what component has to be placed where? It includes tech stack decisions, components/architecture, communication patterns. Whereas LLD is of designing the classes / functions or database design etc. Completely into application level kinda.

HIGH LEVEL SYSTEM DESIGN

So, yeah to design this HLD System, I should be knowing few concepts to speak on. I need few fundamentals, right?

Here are those major foundational concepts on top of this we will be building our systems.

There are many concepts about 30+ to mention, but we will touch base the most important and foundational one now.

1. Scalability
2. Load Balancing
3. Caching
4. Database Sharding
5. CDN
6. Database Indexing

7. Queues/Workers
8. CAP Theorem

We can also discuss about Service Discovery, Api gateway, Rate Limiting, Microservices, Auth, and Event driven architecture.

For now, we will learn about the above basics/fundamentals.

1. Scalability:
    a. What?: Its is the systems ability or potential to scale or handle increased load without affecting the performance.
    Can be classified into two types: Vertical Scaling or Horizontal scaling.
    Vertical Scaling: Add more RAM to a single machine. Like increasing the capacity of single server to handle the load.
    Horizontal Scaling: Adding of more servers/nodes to the system like adding multiple servers.
    b. Why to use? To avoid performance issues during high traffic.
    c. When to use? When user base is expected to grow, or performance is degrading under load.

2. Load Balancing:
    a. What? It distributes incoming traffic across multiple servers.
    b. Why to use? Mainly to improve the availability of the system, and performance.
    c. When to use? When you need high availability and better user experience.

3. Caching:
    a. What is this? Its to store a frequently accessed data in memory so it could be retrieved faster.
    b. Why are we using? Because to reduce the latency and server load
    c. When to use? When we have a data that is frequently accessed but doesn't change a lot. Also to have a faster response.

4. Database Sharding:
    a. What is this? This is splitting of large DB into smaller, faster and manageable pieces called shards.
    b. Why its needed? Because when the DB is split it ensures faster performance and scalability.
    c. When to use? When the DB is too large to handle it efficiently or may be when we are distributing data across multiple servers.

5. Content Delivery Network (CDN):
    a. What is this? It's a network of distributed servers that deliver static content to users based on their location.
    b. Why to use this? To reduce the latency and load times by serving the content from the nearest server to the user.
    c. When to use? When we are serving global customers, also when files don't change frequently.

6. Database Indexing:
    a. What? This is creating a data that improves the speed of data retrieval operations on a database.

   b. Why? To make searches faster, especially on huge data sets.
   c. When? When we are frequently querying on certain columns.

7. Queues/Workers
   a. What? These are used when we call async functions which are independent and run in the background like sending mails, processing messages using queues.
   b. Why? To avoid users to wait for long running tasks, so that the processes can be added to queue and keep the system running.
   c. When? Like email notifications, sending message data, sometime to the servicebus we put the data so it can finish it in its time.

8. CAP Theorem.
   a. CAP stands for Consistency, Availability and partition tolerance. Remember in distributed systems, we can only achieve two of the three.
   b. Why? Designing the trade-offs in distributed systems.
   c. When? When we are building distributed services.