Week 1 – Lesson 1

1 What is System Design?

Making decisions under constraints

2 Basically answering questions like

- How many users?
- How fast?
- How reliable?
- How much data?
- What happens when things fail?

Not what tools, but why those tools?

3 Example:

problem: you need to build a system that shows "Number of likes on a post".

Naive solution: User → Server → Database → Server → User

- Every "like" updates DB
- Every "view" reads DB

} Works for 100 users

What breaks at scale?

Let's say:

- 1 million users
- Each post has 1000s of views per second

Hence, Database becomes slow

Server CPU increases

Latency goes up

Users see ~~delay~~ delay

→ This is system design pain

4 Core Idea #1: BOTTLENECKS

A system is only fast as its slowest part

Common bottlenecks:
- CPU
- Memory
- Network
- Disk (OB)

System design = finding and fixing bottlenecks

5 Scale is not Magic #2

Vertical Scaling
- Bigger machine
- More RAM, CPU

- Expensive
- Has limits

Horizontal scaling
- more machines
- split load

- cheap
- Scales better
- more complexity

most real systems = horizontal scaling

6 **Mental model**

Think of system design like a restaurant
- One chef → slow with many consumers
- more chef → faster
- Cash counter becomes bottleneck
- kitchen storage runs out

→ **Systems behave the same way.**

7 **Mini-design task**

Task 1 (Think, don't code yet)

Design a like counter system that supports:
- millions of users
- Reads are much more than writes
- likes must not be lost.

Answer these:
Where is data stored?
What happens when 1 million users refresh?
What is the bottleneck?