Case Study-37 System Design
Handling High Traffic & Scaling Bottlenecks

🚨 Problem

Applications fail under heavy traffic because:

- **Monolithic architecture becomes slow**
- **Single database becomes a choke point**
- **Lack of caching leads to high latency**
- **No load balancing across servers**

This results in outages, slow response time, and poor user experience.

🛠️ Solution

Use a scalable system design approach:

- **Break monolith into microservices for independent scaling**
- **Add CDN + caching layers (Redis/Memcached)**
- **Use Load Balancers to distribute traffic**
- **Introduce Horizontal Scalability with auto-scaling groups**
- **Opt for sharded or replicated databases**

This builds a system that can gracefully handle millions of users without performance drops. Scalable system design improves performance by introducing microservices, caching, load balancing, and database sharding to handle high traffic efficiently.

#SystemDesign #Architecture #Scalability #LoadBalancing #Caching #Microservices #HighTraffic #DistributedSystems