■ System Design Cheat Sheet

■ Scalability Concepts

- Vertical Scaling: Scale UP by adding CPU/RAM to one machine (simple but limited).
- Horizontal Scaling: Scale OUT by adding more machines (complex but powerful).
- Remember: 'Bigger box vs more boxes.'

■ Load Balancers, Caching, Queues

- Load Balancers: Distribute requests across servers (Nginx, HAProxy).
- Caching: Redis/Memcached for DB queries; CDN for static assets. (Shortcut to memory).
- Queues: Kafka/RabbitMQ for async processing. (Buffer for tasks).

■ RESTful API Design

- Stateless, resource-based endpoints.
- Use HTTP methods: GET, POST, PUT, PATCH, DELETE.
- Version APIs (/v1/users) & paginate results.
- Keep it predictable and consistent.

■ CAP Theorem

- Consistency: All nodes same data.
- Availability: System always responds.
- Partition Tolerance: System works despite network splits.
- Pick 2: CP (banking), AP (social media). Always P in distributed systems.

■ Database Sharding & Consistency Models

- Sharding: Split DB by key (A–M on DB1, N–Z on DB2). Watch out for joins & hotspots.
- Consistency Models:
- - Strong: Read = latest write (banking).
- - Eventual: May read stale, but catches up (social feeds).
- - Causal: Preserves cause-effect order (likes after posts).