CAP Theorem

CAP theorem(Brewer's theorem), is a fundamental concept in distributed systems that states that a distributed database can only provide two of the following three guarantees at any given time:
1. Consistency (C): All nodes in the system see the same data at the same time. After a write operation any subsequent read return the latest data.
2. Availability (A). Every request(read or write) receives a response, whether successful or not. The system remains operational 100% of the time.
3. Partition Tolerance (P). The system continues to function even if network partitions occurs.

In a distributed system, network partitions are inevitable, so CAP theorem essentailly forces system designers to choose between consistency and Availability.

Scenarios in the context of CAP

1. CP (Consistency + Partition Tolerance):
The system prioritizes consistency over availability during a network partition.
If nodes can't communicate, some nodes may become unavailable to maintain a consistent state.

Example:
Banking systems: If a user tries to withdraw money from two ATMs at the same time, the system must ensure consistency to prevent overdrawing. During a partition, it may block one of the ATM to maintain data integrity.

2. AP(Availability + Partition Tolerance)
The system prioritizes availability over consistency during a partition.
It ensures the system is always operational but allows temporary inconsistency.

Example:
DNS : Even during network partitions, DNS must resolve domain names to IP addresses to ensure availability. Some responses might not reflect the most recent updates.

3. CA (Consistency + Availability)
Technically impossible in distributed systems during a partition, as the system must sacrifice one to tolerate partitions.

Example:
Single node systems: These can provide both consistency and availability but can't tolerate partitions as they are not distributed.

How to Use CAP Theorem in an interview

1. Understand Requirement: Analyze the trade-offs based on the system priorities.
2. Explain Trade-offs: Clearly articulate why you are prioritizing one over the other.
3. Present Solutions: Suggest techniques(eg. caching, replication, quorum-based writes) to mitigate the downside of the chosen approach.