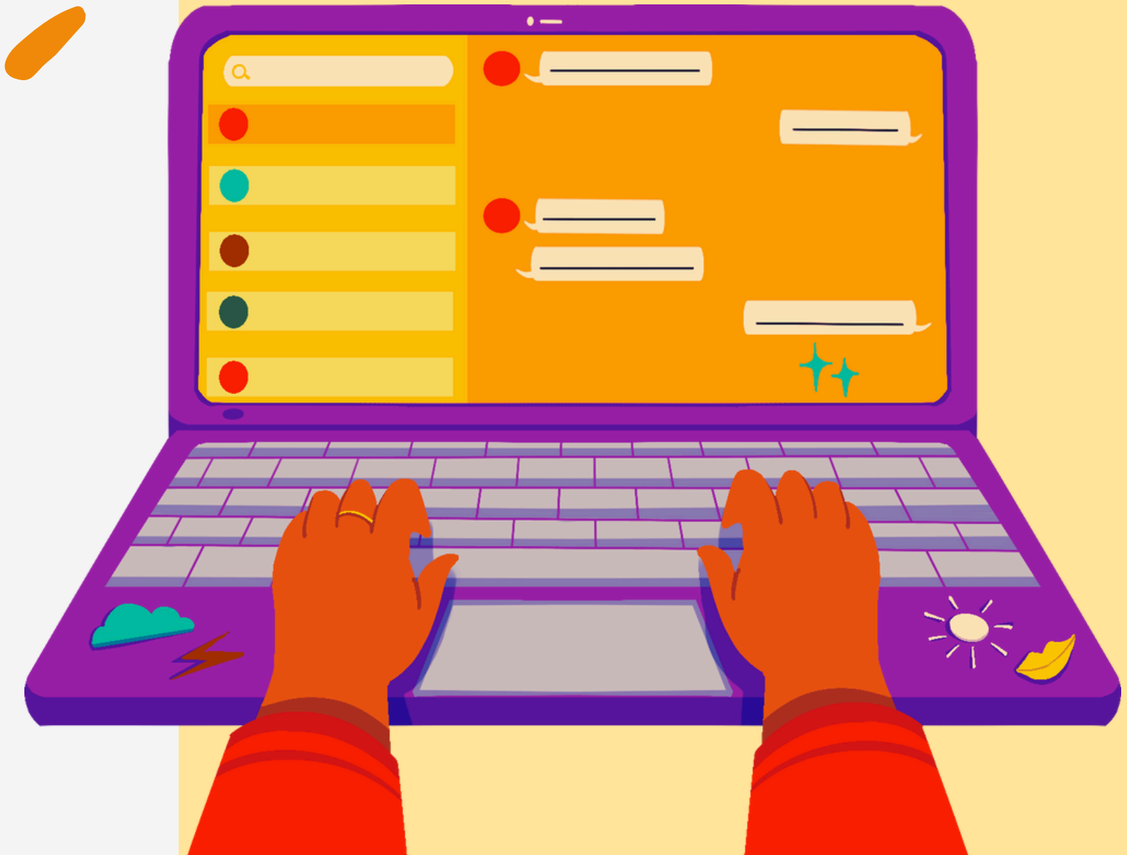
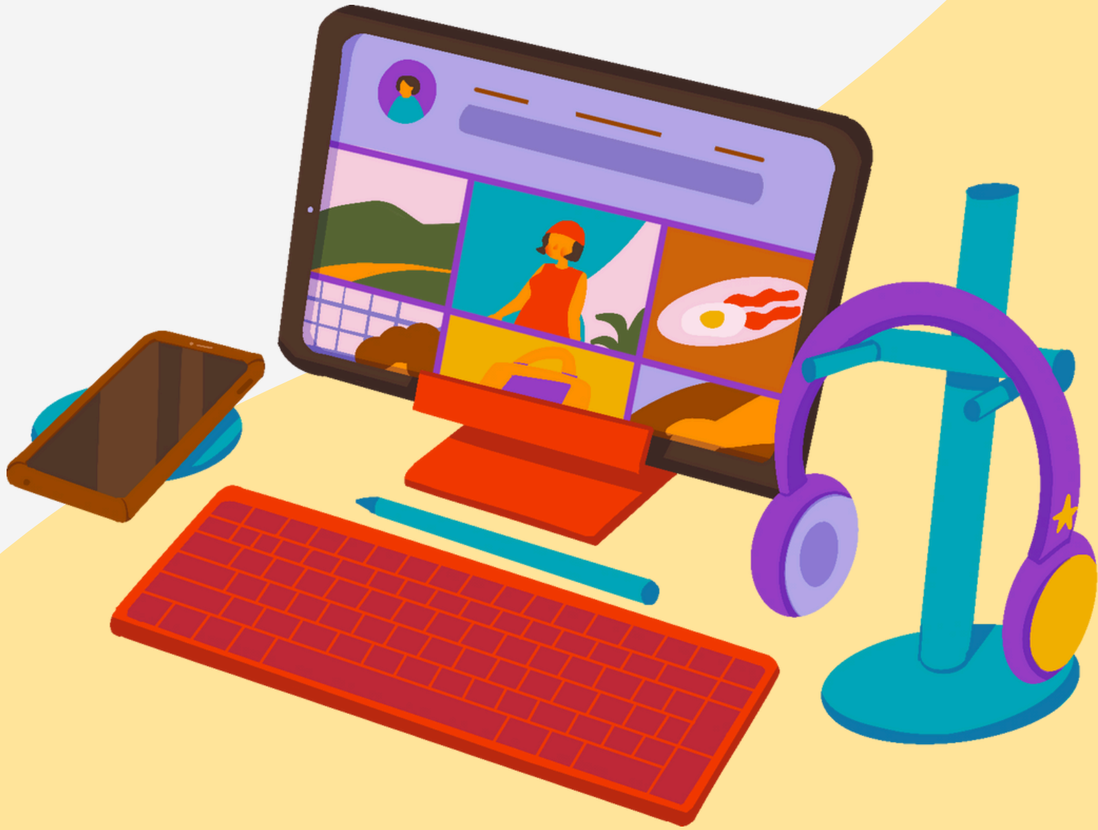
DesignGurus.io

DISTRIBUTED SYSTEMS



DesignGurus.io

WHY DISTRIBUTED SYSTEMS



Distributed systems provide scalability, fault tolerance, and resource sharing.

They allow us to handle large-scale applications and data efficiently.

DesignGurus.io

KEY CONCEPTS

1. Nodes
2. Scalability
3. Fault Tolerance
4. Consistency
5. Availability
6. Partition Tolerance
7. CAP Theorem
8. Replication
9. Sharding
10. Load Balancing
11. Distributed Consensus
12. Distributed Transactions
13. Middleware

DesignGurus.io

1. NODES

Individual computers in a distributed system.

Each node can perform tasks and communicate with other nodes.

DesignGurus.io

2. SCALABILITY

The ability of the system to handle increased load by adding more nodes.

Distributed systems can scale horizontally by adding more machines.

DesignGurus.io

3. FAULT TOLERANCE

The system's ability to continue operating properly in the event of a failure of some of its components.

Distributed systems achieve this by having redundancy and replication.

DesignGurus.io

4. CONSISTENCY

Ensuring all nodes see the same data at the same time. This can be challenging due to network delays and failures.

DesignGurus.io

5. AVAILABILITY

The system's ability to be operational and accessible when needed.

High availability is achieved through redundancy and failover mechanisms.

DesignGurus.io

6. PARTITION TOLERANCE

The system continues to operate despite network partitions that cause some nodes to be unable to communicate with others.

DesignGurus.io

7. CAP THEOREM

The theorem states that a distributed system can only provide two out of three guarantees: Consistency, Availability, and Partition Tolerance.

DesignGurus.io

8. REPLICATION

Copying data across multiple nodes to ensure reliability and fault tolerance.

DesignGurus.io

9. SHARDING

Dividing a database into smaller, more manageable pieces (shards) that can be distributed across multiple nodes.

DesignGurus.io

10. LOAD BALANCING

Distributing the workload evenly across all nodes to ensure no single node is overwhelmed.

DesignGurus.io

11. DISTRIBUTED CONSENSUS

Achieving agreement among distributed nodes on a certain value or state.

Common algorithms include Paxos and Raft.

DesignGurus.io

12. DISTRIBUTED TRANSACTIONS

Ensuring a sequence of operations across multiple nodes completes successfully and maintains system integrity.

DesignGurus.io

13. MIDDLEWARE

Software that provides common services and capabilities to applications outside of what's offered by the operating system.

It helps in communication and data management among distributed nodes.

DesignGurus.io

FOLLOW FOR MORE UPDATES ON SYSTEM DESIGN AND CODING

Mock Interviews | Courses | Interview Bootcamps