22 December 2025 07:42 PM

What is System Designing ?

- System design is the process of designing the elements of a system such as the architecture, modules and components and the data that goes through that system.

Types of System Design

i) LLD {Low level Design}
ii) HLD {High level Design}

HLD	LLD
☆ Describe the main components that would be developed for the resulting product.	Describe the design of each element mentioned in the High-Level Design of the system
☆ The system architecture details, database design, services and processes, the relationship between various modules, and features.	Classes, interfaces, relationships between different classes, and actual logic of the various components.

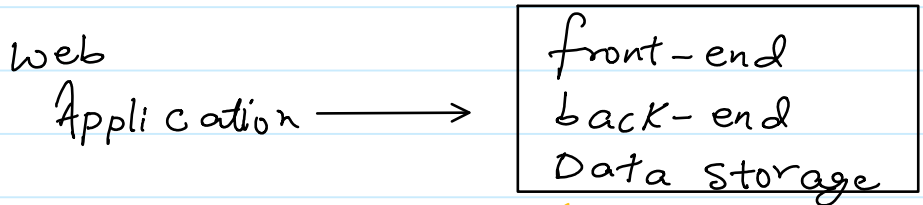
Architecture ⇒ Internal design details for building the applications.

22 December 2025 07:55 PM

Monolithic Architecture

- If all the components and functionalities of a project are entangled and combined in a single Codebase, then that is a monolithic application.

- Monolithic Architecture has less complexity ⇒ Easier to understand ⇒ Higher productivity.



↳ written & Deployed together

- Monolithic system is also known as centralized system.

Advantages / when it is good

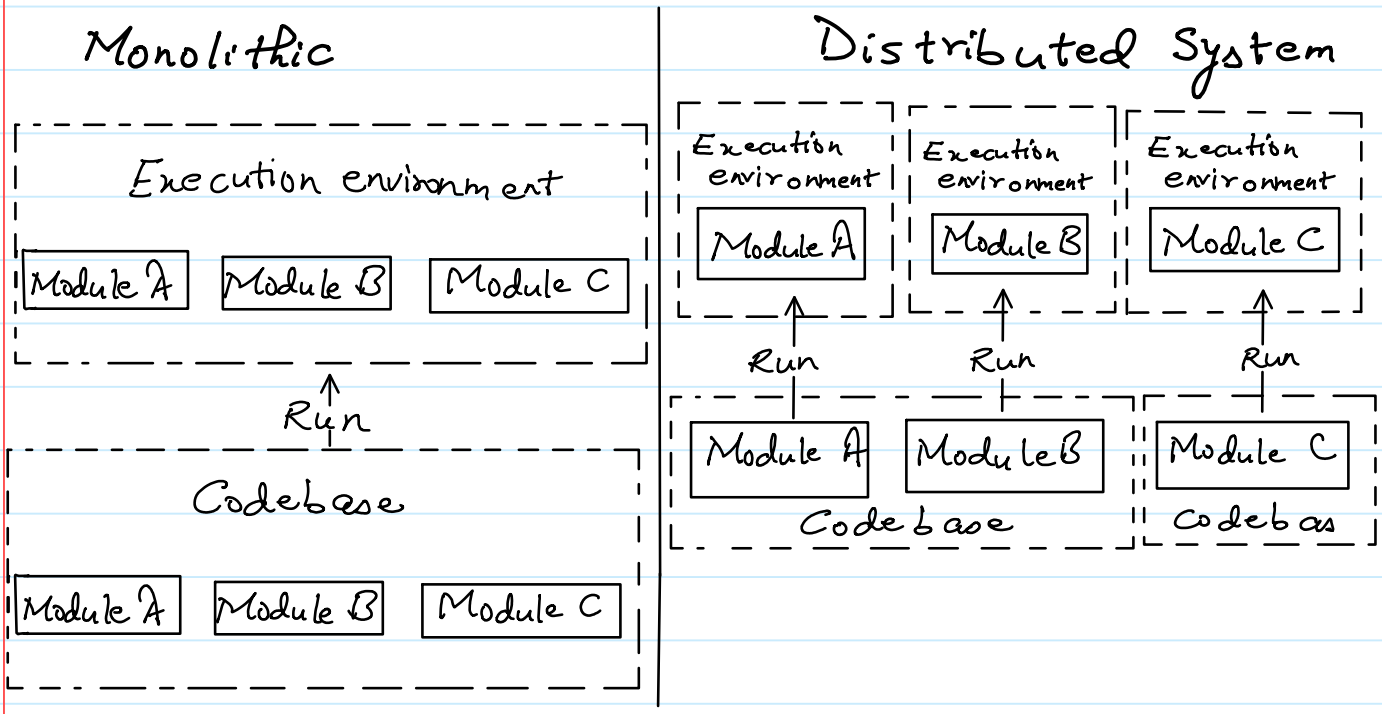
1. When we are just starting.
2. In monolithic architecture, all the modules are present in the single system, so they require fewer network calls as compared to other architectures.
3. It is Comparitively easier to secure monolithic system
4. Integration testing is easier.
5. Less confusion.

Disadvantages

1. In monolithic architecture, every module is combined in a single system, so if there is an error or bug in a single module, it can destroy the Complete system.

2. In monolithic architecture, whenever a single module is updated, the whole system needs to be updated to reflect the changes to the users. All modules are present in a single system and are connected to one another, so the whole system needs to be updated.

3. In monolithic architecture, if there is any change in a single module's programming language or framework, it will affect the entire system. The entire system need to be changed because every module is interlinked and tightly coupled.



☐ **Distributed system:** A distributed system is a collection of multiple individual systems connected through a network that share resources, communicate and coordinate to achive common goals.

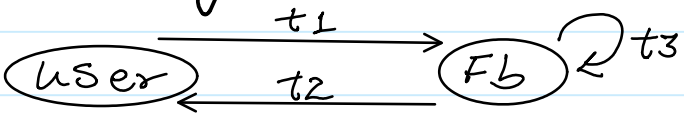
Advantages

1. Scalable → Horizontal scale
2. No single point of failure. for eg → if user service is not working it will not affect authentical service, product service, order service etc.
3. Low latency

Disadvantages

1. Complex, because many services communicate among them.
2. Management required like load balancing
3. Difficult to secure
4. Message may be lost in between nodes.

Latency



Latency = Network delay + Computational delay

$\Delta t = t1 + t2 + t3$

Monolithic

Δt = Computational Delay

<

Distributed

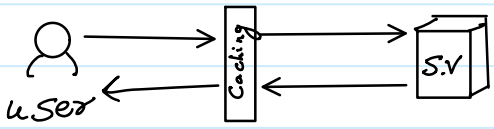
Δt = Comp. delay + Network delay

Reducing latency

23 December 2025 08:32 PM

Reducing latency

1) Caching 2) CDN 3) Upgrading system



Caching VS CDN

- CDNs are geographically distributed networks of proxy servers and their objective is to serve content to users more quickly.
- Caching is the process of storing information for a set period of time on a computer.

Throughput

The volume of work or information flowing through a system. Throughput is the amount of data transmitted per unit of time. It is the process flow rate. Throughput is measured in bits per second i.e bps

- Distributed system will have more throughput than monolithic application because:
  - i) No limit on resources
  - ii) Load balancing

Causes of Low throughput

i) latency ii) protocol overhead iii) Congestion

Improving throughput

i) CDN ii) Caching iii) Distributed system
iv) load balancing v) upgrading system

Availability, Replication & Redundancy

1. What is Availability?

- Definition: Availability is the percentage of time a system is operational and accessible to perform its intended function.
- Concept: It essentially measures "uptime". A highly available system is one that is accessible almost all the time, even during components failure
- Formula:

$$\text{Availability} = \frac{\text{Uptime}}{\text{Total time}} \times 100\%$$

{total time = uptime + downtime}

2. What is Redundancy?

- Definition: Redundancy is the practice of duplicating critical components or functions of a system so if one fails, another can take place
- Goal: To remove Single point of failure (SPOF)
- Types of redundancy:
  - i) Hardware Redundancy: {Extra Hard drive, servers}
  - ii) Software Redundancy: {Running multiple instances}

Replication

23 December 2025 09:13 PM

Q. What is Replication ?

- Definition: Replication refers to creating and maintaining exact copies of data across multiple servers or databases.

- Goal:

  Fault Tolerance: If one database node goes down, data is still safe and accessible from another.

  Performance: Read requests can be distributed (load balanced) across multiple replicas to reduce latency.

  Disaster Recovery: Keep data in different geographical regions.

Fault Tolerance $\propto$ Availability

How to increase Availability

i) Replication
ii) Distributed system
iii) Redundancy

Consistency:

Q. What is consistency ?

Definition: Consistency ensures that every read request returns the most recent write. Regardless of which node or server a client connects to, they should see the exact same data

Consistency

26 December 2025 08:23 PM

Dirty Read: When a client read data that is outdated or has not yet been fully committed across all systems, it is called "Dirty Read".

Real-world example:

- Bank Balance: If you deposit ₹100, the ATM should immediately show the updated balance.
- Movie Ticket: If user A books the last seat on Delhi server, User B on Pune server should immediately see it as "Booked" to prevent doule booking.

2. Monolithic vs Distributed systems

- Monolithic: Generally **natively consistent** because there is often only one database instance. All requests go to the same source, so no synchronization is needed.

- Distributed: Inherently prone to **inconsistency** because data is replicated across different servers.
- sync time (t): If server A updates at t1 and server B takes time to sync t2, the system is inconsistent for the duration max (t1, t2).

3. Factors Improving Consistency

a) Increase network speed: Faster synchronization between nodes reduces the window of inconsistency.

b) Stop Read operations: Block client from reading data until all nodes are successfully updated.

c) Data Locality: Reduce the physical distance between replicas to speed up data transfer.

4. Types of Consistency Models

Type	Definition	Behaviour	Use Case
strong Consistency	The system does not allow reads until all node/replicas are updated	-Zero Dirty Reads. -Higher data integrity -Higher latency. (slower) because the system "locks" during updates	Payments, Banking, Ticket Booking
Eventual Consistency	The system allows read even if some nodes are not yet updated	-Possible Dirty -High availability and speed. -"Eventually" all nodes will become Consistent.	Social Media (Instagram/Facebook like or posts)
Weak Consistency	There is no gurantee that all nodes will update.	-updates may or may not propagate to all nodes -Heavily dependent on specific business logic.	Real-time analytics, Voip/Videa streaming

CAP Theorem

26 December 2025 10:05 PM

CAP Theorem

C — Consistency
A - Availability
P - Partition Tolerance

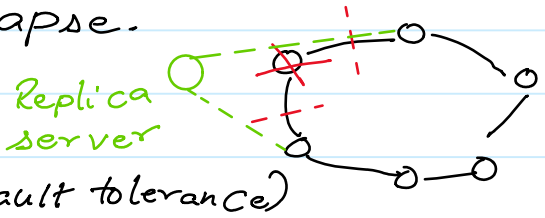
- For a distributed system, the CAP Theorem states that it is possible to attain only two properties and the third would be always compromised.

- The system requirement should define which two properties should be chosen over the rest.

☆ Consistency(C): All clients see the same data at the same time.
eg→ If user A book a ticket user B must immediately see it as booked.

☆ Availability(A): Every requests gets a response (sucess/failure), even if data is slightly old.
eg→ Google search or Youtube always loads, even if search index is 5 min old.

☆ Partition Tolerance: The system continues to work even if communication between servers (node) breaks.
eg→ If a wire is cut or server crashes, the system must not collapse.



Crucial note: In any distributed systems (like cloud/internet), network failures (partitions) are inevitable. Therefore, Partition Tolerance (P) is mandatory. You cannot choose CA in real distributed system.

Application	Classification	Reasoning
Blog Website	AP (Availability)	If a blog post doesn't load instantly for everyone, it's fine. It's more important that the site stays online .
Multiplayer Games (PUBG)	AP (Availability)	Speed/Uptime is critical. A slight lag in player position is acceptable compared to the game crashing .
Stock Trading	CP (Consistency)	Prices must be exact. Showing ₹5000 to one user and ₹10000 to another is disastrous. It's better to show "System Down" than wrong prices .
Video Streaming	AP (Availability)	If the site goes down, users leave. High availability is key to retention .
Ticket Booking (IRCTC)	CP (Consistency)	You cannot sell the same seat to two people (Double Booking). The system must lock seats strictly .
Video Chat	AP (Availability)	If WhatsApp goes down for 1 second, it's bad. It must always be up. Slight lag is okay .
Banking	CP (Consistency)	Accurate balance is non-negotiable. Banks often schedule downtime rather than risk inconsistent transactions .

Lamport Logical clock (Distributed systems)

Q. Why do we need it ?

- The problem : In a distributed system (eg. server in USA, India, Australia), physical clocks are unreliable because
  - △ Different time zones
  - △ Network latency makes synchronization difficult
  - △ Physical clocks drift (run faster/slower) over time.

- The Goal : We don't need to know the exact time (eg. 10 PM) We only needs to sequence of events (i.e Event A happened before B)

Lamport logical Clock

27 December 2025 09:52 PM

2. What is a Lamport logical clock?

⇒ It is a simple algorithm used to order events in a distributed system without using physical time.

⇒ Every process server maintains a simple counter initialized to 0 or 1. This counter act as its clock.

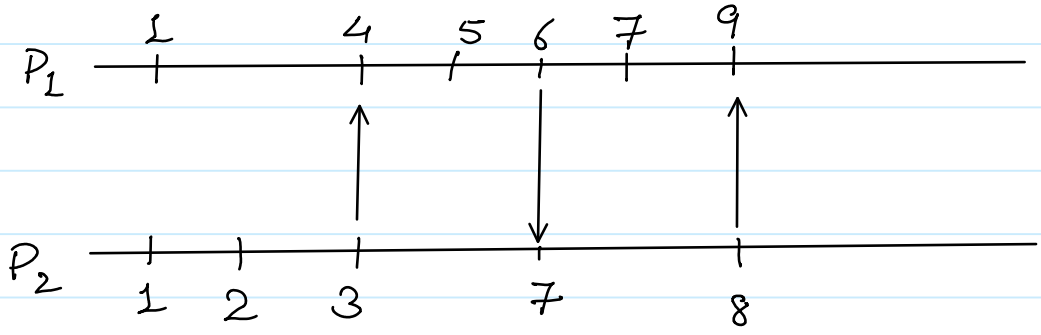
Rules :

i> internal / Send Event :

- Before any local event or sending a message, increment the counter
- Formula Counter = Counter + 1

ii> Receive Event :

- When receiving a message with a sender's timestamp (T_{sender}), update local counter to be higher than both
- Formula: Counter = max(local_counter, T_{sender}) + 1



Vertical vs Horizontal Scaling

28 December 2025 01:55 PM

⇒ Vertical scaling: Increasing the power of a single machine or server.

⇒ How: Adding more RAM, upgrading the CPU, increasing hard disk space on the same server.

Pros	Cons
i> Easy Implementation ii> Less Management: You have only one machine to manage. iii> No Network Latency: Everything is in one box	i> Single poin of failure. ii> Hardware limit: There is limit how much ram/cpu we can put in a machine iii> Cost: High-end "super-computer or mainframes are expensive.

⇒ Horizontal Scaling: Adding more machines to the pool to share the workload.

How: Instead of upgrading 1 server, you buy 4 new standard servers and distribute the traffic/data across them.

Pros	Cons
i> No single point of failure: If one crashes another take its place. ii> Unlimited growth: Theoretic-ally, you can keep adding more servers iii> Cost effective: You can use cheeper hardware.	i> Complex management: Managing 100 server is harder than managing one. ii> Data Consistency: Keeping data across multiple server is difficult iii> Higher power/space

Redundancy and Replication

28 December 2025 02:15 PM

Redundancy: Redundancy is simply the duplication of nodes or components so that when a node or components fails, the duplicate node is available to service customers.

Types of Redundancy:

i) Active Redundancy: Active Redundancy is considered when each unit is operating/active and responding to the action. Multiple nodes are connected to a load balancer, and each unit receives an equal load.

ii) Passive Redundancy: Passive Redundancy is considered when one node is Active or operational and the other is not operating. During the breakdown of the active node, the passive node maintains availability by becoming the active node.

Replication: Redundancy + Synchronization

⇒ Context: Usually refers to Database (stateful servers)

⇒ Why sync: Unlike code servers, database change constantly (Insert/update/delete).

Types of Replication:

i) Active Replication (Multi-master)

- All database nodes are active.
- Any code can accept Read and Write requests.

Load Balancers

28 December 2025 02:32 PM

Challenge: Complex to keep data synced in real-time.

2 Passive Replication (Master – Slave)

- Master Node: Handles Writes (Insert/update)
- Slave Node: Handle Reads (select). They get data updates from master.
- If master fail: One slave is promoted to become master.

⇒ Master-slave replication can either be synchronous or Asynchronous. The difference is simply the timing of propagation of changes. If changes are made to the master and slave at the same time, it is synchronous, If changes are queued up and written later, it is asynchronous