1. System Design Fundamentals & Foundations of Low-Level Design

Tuesday, 6 January 2026 1:28 PM

Why System Design

- Flexible
- Reusable
- Easy to maintain
- Easy to test
- Ready for real-world systems & interviews

1. What is System Design?

Definition

System Design is the process of **defining architecture, components, interfaces, and data** for a system to satisfy given requirements.
In simple words:
How do we build software that works well today and scales tomorrow?

Why System Design is Important

- Handles **real-world scale**
- Avoids **rewriting code**
- Improves **performance & reliability**
- Makes teamwork easier
- Critical for **product companies interviews**

Evolution of Systems

Era	Description
Single Machine	One app, one server
Client-Server	UI + Backend
Monolith	Everything in one codebase
Distributed Systems	Multiple services
Microservices	Independent deployable services
Cloud-Native	Auto-scaling, resilience

Real-World Examples

- **Facebook** → billions of users, distributed storage
- **Google** → search, caching, indexing
- **Netflix** → microservices, fault tolerance

2. HLD vs LLD

High-Level Design (HLD)

Focuses on **architecture**
Includes:

- Services
- Databases
- APIs
- Load balancers
- Data flow

Low-Level Design (LLD)

Focuses on **code & classes**

Includes:

- Classes
- Interfaces
- Relationships
- Methods
- Design patterns

Comparison Table

HLD	LLD
Big picture	Code-level
Architecture	Classes & methods
Scalability	Maintainability
Tech decisions	OOP decisions

3. Requirements Engineering

Functional Requirements (FR)

What the system does

Examples:

- User can register
- User can login
- User can place order

Non-Functional Requirements (NFR)

How the system behaves

Examples:

- Scalability
- Performance
- Security
- Availability
- Latency