

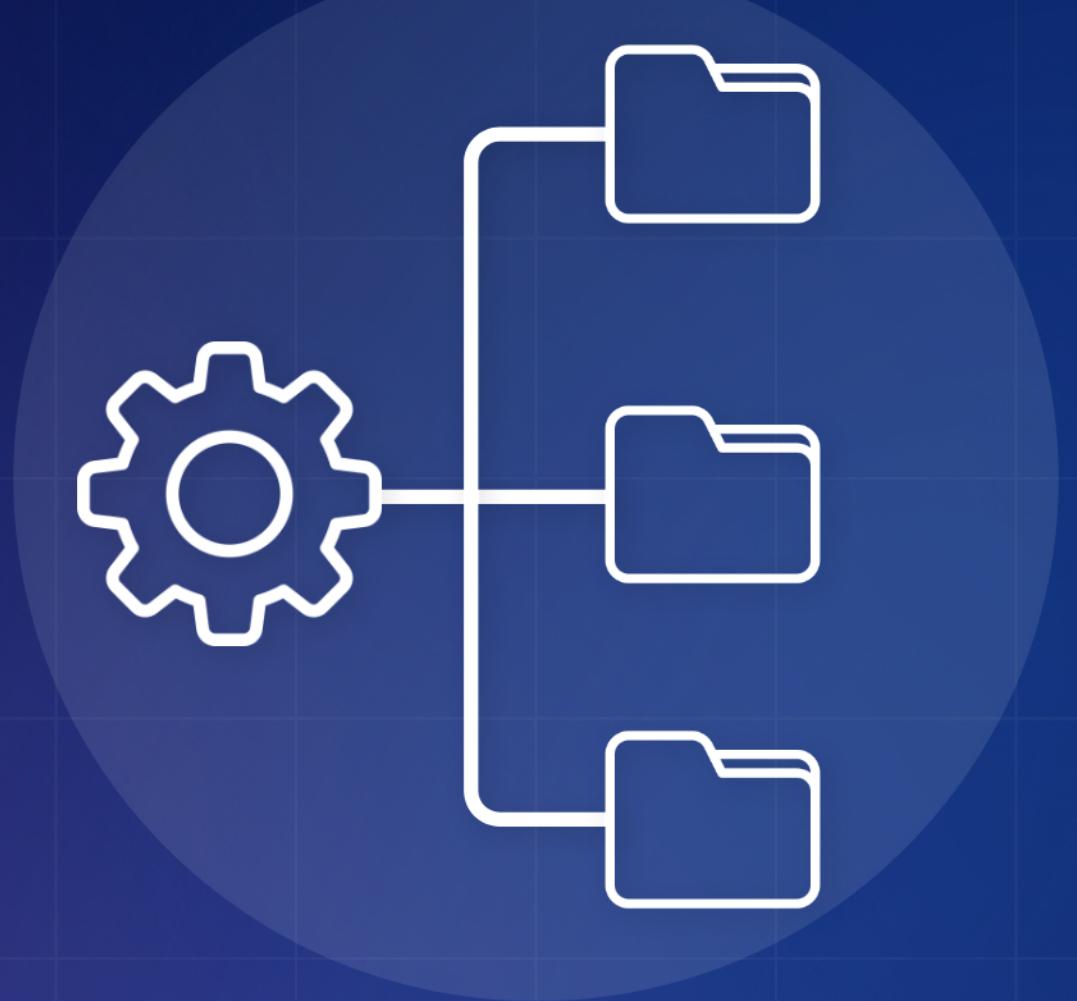
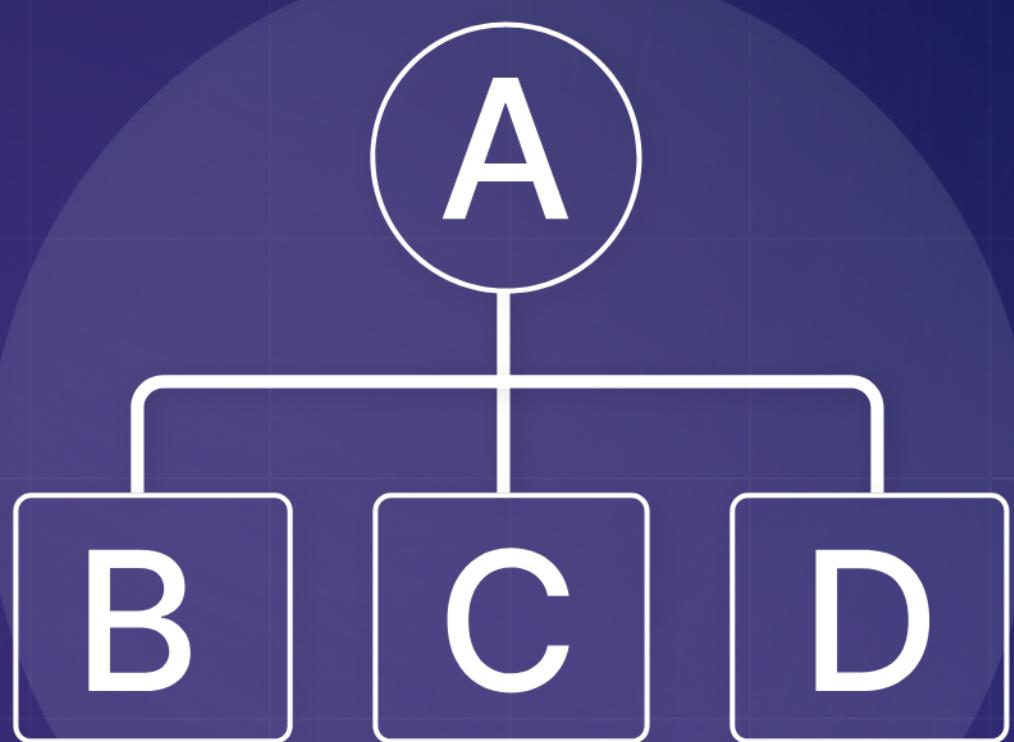
COMPLETE

# HLD & LLD

---

## Guide For System Design Interviews

---



Target Top Product-Based Companies

## \*Disclaimer\*

System Design is the most asked topic in tech interviews. So, make sure you prepare it thoroughly.

**Take the help of this doc and ace your System Design Interviews**

# Low-Level Design (LLD) Guide

## Basics of Low-Level Design

### Question 1:

**What is Low-Level Design (LLD)?**

### Answer 1:

Low-Level Design (LLD) is the process of designing detailed components of a system, focusing on class structures, object relationships, methods, and interactions.

[Reference](#)

### Question 2:

**What is Object-Oriented Design (OOD)?**

### Answer 2:

Object-Oriented Design (OOD) is a design approach based on objects and classes. It focuses on using objects that interact with one another to design software.

[Reference](#)

### **Question 3:**

**What are the Four Pillars of Object-Oriented Programming?**

### **Answer 3:**

Encapsulation, Abstraction, Inheritance, Polymorphism.

[Reference](#)

### **Question 4:**

**What is a Class Diagram?**

### **Answer 4:**

A class diagram is a static structure diagram in UML that describes the structure of a system by showing its classes, attributes, operations, and relationships between objects.

[Reference](#)

### **Question 5:**

**What is the Law of Demeter?**

### **Answer 5:**

The Law of Demeter is a design guideline that promotes loose coupling between components. It suggests that a class should only communicate with its close dependencies.

[Reference](#)

## **Question 6:**

**What is the Dependency Inversion Principle?**

## **Answer 6:**

This principle suggests that high-level modules should not depend on low-level modules; instead, both should depend on abstractions.

[Reference](#)

## **Question 7:**

**What are Interfaces in OOP?**

## **Answer 7:**

Interfaces define abstract methods that a class must implement, enabling polymorphism and abstraction.

[Reference](#)

## **Question 8:**

**What is the Observer Design Pattern?**

## **Answer 8:**

The Observer Pattern defines a one-to-many dependency, where one object (the subject) notifies all its dependents (observers) of state changes.

[Reference](#)

## Question 9:

**What is the Adapter Design Pattern?**

## Answer 9:

The Adapter Pattern allows objects with incompatible interfaces to work together by wrapping them with a new interface.

[Reference](#)

## Question 10:

**What is the Strategy Design Pattern?**

## Answer 10:

The Strategy Pattern defines a family of algorithms and makes them interchangeable by selecting one at runtime.

[Reference](#)

## Question 11:

**What is the difference between an Abstract Class and an Interface?**

## Answer 11:

Abstract Class: Can have both abstract and concrete methods and is used for shared behavior. Interface: Only contains abstract methods (before Java 8), used for defining a contract.

[Reference](#)

## Question 12:

What is Aggregation in OOP?

## Answer 12:

Aggregation represents a 'whole-part' relationship where the contained objects can exist independently of the container.

[Reference](#)

## Question 13:

What is the Strategy Design Pattern?

## Answer 13:

The Strategy Pattern defines a family of algorithms and makes them interchangeable by selecting one at runtime.

[Reference](#)

## Question 14:

What is Composition in OOP?

## Answer 14:

Composition is a 'whole-part' relationship where the lifetime of the contained objects depends on the lifetime of the container.

[Reference](#)

## **Question 15:**

**What is Polymorphism in OOP?**

## **Answer 15:**

Polymorphism allows objects of different classes to be treated as objects of a common superclass, achieved via method overriding or method overloading.

[Reference](#)

## **Question 16:**

**What is Cohesion and Coupling in Design?**

## **Answer 16:**

Cohesion measures how closely related the responsibilities of a single module or class are. Coupling measures how dependent different modules or classes are on one another.

[Reference](#)

# Practical Questions for LLD

 **Question 1:**

**Design a Parking Lot System**

**Practice**

 **Question 2:**

**Design an Online Book Rental System**

**Practice**

 **Question 3:**

**Design a Movie Ticket Booking System**

**Practice**

 **Question 4:**

**Design an ATM System**

**Practice**



## Question 5:

Design a Hotel Management System

Practice



## Question 6:

Design a Food Delivery System

Practice



## Question 7:

Design a Ride-Sharing System (Like Uber/Lyft)

Practice

# High-Level Design (HLD) Guide

## Basics of High-Level Design

### Q Question 1:

What is High-Level Design (HLD)?

### A Answer 1:

High-Level Design (HLD) focuses on the overall architecture of a system, describing its major components, their relationships, and the flow of data.

Reference

### Q Question 2:

What are the key components of High-Level Design?

### A Answer 2:

Architectural Patterns: Client-server, microservices, and layered architecture. Data Flow: Describes the movement of data within and between system components. Databases: Choosing between SQL and NoSQL databases, data partitioning, and replication. Communication: HTTP, RPC, or message queues.

Reference

## Question 3:

**What is a Load Balancer?**

## Answer 3:

A load balancer distributes network or application traffic across multiple servers to avoid overwhelming a single server, improving availability and reliability

[Reference](#)

## Question 4:

**What is Horizontal and Vertical Scaling?**

## Answer 4:

Horizontal Scaling: Adding more machines to handle the load (scale-out). Vertical Scaling: Increasing the capacity of a single machine (scale-up).

[Reference](#)

## Question 5:

**What is Caching, and why is it used?**

## Answer 5:

Caching stores frequently accessed data in memory to improve performance by reducing access time for that data.

[Reference](#)

## Question 6:

What are Databases in HLD?

## Answer 6:

SQL (Relational): Structured databases like MySQL, PostgreSQL.

NoSQL (Non-relational): Schema-less, unstructured databases like MongoDB, Cassandra.

[Reference](#)

## Question 7:

What is the CAP Theorem?

## Answer 7:

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

[Reference](#)

## Question 8:

What are Message Queues?

## Answer 8:

Message Queues decouple different components of a system by sending messages asynchronously. Examples: RabbitMQ, Kafka etc.

[Reference](#)

## **Question 9:**

### **What is Sharding in Databases?**

#### **Answer 9:**

Sharding splits a database into smaller, more manageable pieces, called shards, each containing a subset of the data.

[Reference](#)

## **Question 10:**

### **What is Data Partitioning?**

#### **Answer 10:**

Data Partitioning splits large datasets into smaller segments, improving performance and enabling distributed storage.

[Reference](#)

## **Question 11:**

### **What are Microservices?**

#### **Answer 11:**

Microservices architecture breaks a system into small, loosely coupled services, each responsible for specific functionality.

[Reference](#)

## **Question 12:**

**What is a CDN (Content Delivery Network)?**

## **Answer 12:**

A CDN is a network of distributed servers that deliver content based on the geographic location of the user, origin of the webpage, and the delivery server.

[Reference](#)

## **Question 13:**

**What is Eventual Consistency?**

## **Answer 13:**

Eventual Consistency means that a system will become consistent over time, given that no new updates are made.

[Reference](#)

## **Question 14:**

**What is the Circuit Breaker Pattern?**

## **Answer 14:**

The Circuit Breaker Pattern is used to detect failure and prevent an application from trying to execute operations that are likely to fail.

[Reference](#)

## Question 15:

What is Rate Limiting?

## Answer 15:

Rate Limiting controls the number of requests a user or system can make to a service over a set period of time.

Reference

# Practical Questions for HLD

 **Question 1:**

Design a URL Shortening Service (Like Bitly)

[Practice](#)

 **Question 2:**

Design a Content Delivery Network (CDN)

[Practice](#)

 **Question 3:**

Design a Ride-Sharing System (Like Uber/Lyft)

[Practice](#)

 **Question 4:**

Design a Chat System (Like WhatsApp)

[Practice](#)

## Question 5:

Design a Distributed File Storage System (Like Google Drive)

[Practice](#)

## Question 6:

Design a Video Streaming Service (Like Netflix)

[Practice](#)

## Question 7:

Design an E-Commerce System (Like Amazon)

[Practice](#)

# Projects for LLD and HLD Implementation

## LLD Projects

**Project 1:**  
Ride-Sharing App (Like Uber/Lyft)

Reference

**Project 2:**  
Library Management System

Reference

**Project 3:**  
Library Management System

Reference

## **Project 5:**

### **Hotel Management System**

**Reference**

## **Project 6:**

### **ATM Machine System**

**Reference**

# HLD Projects

## Project 1: Scalable Video Streaming Platform

Reference

## Project 2: Distributed File Storage System

Reference

## Project 3: Social Media Platform

Reference

## Project 4: Real-Time Chat System

Reference

## **Project 5:** **Online Marketplace System**

**Reference**

## **Project 6:** **ATM Machine System**

**Reference**



## WHY BOSSCODER?

 **750+** Alumni placed at Top Product-based companies.

 More than **136% hike** for every **2 out of 3** working professional.

 Average package of **24LPA**.

The syllabus is most up-to-date and the list of problems provided covers all important topics.

Lavanya  
 Meta



Course is very well structured and streamlined to crack any MAANG company

Rahul .  
 Google



[EXPLORE MORE](#)