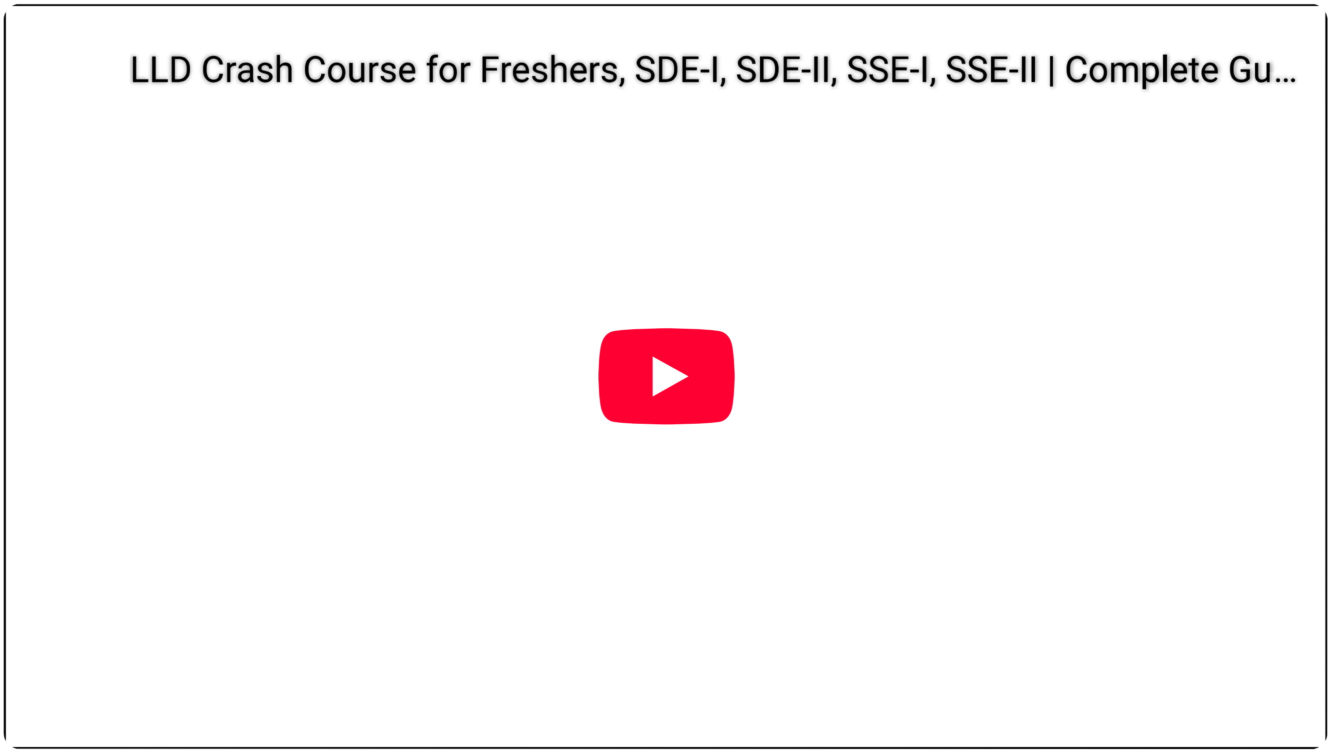
Search articles...

LLD Crash Course


LLD Crash Course for Freshers, SDE-I, SDE-II, SSE-I, SSE-II | Complete Gu...

Topic Tags:

System Design LLD Crash Course

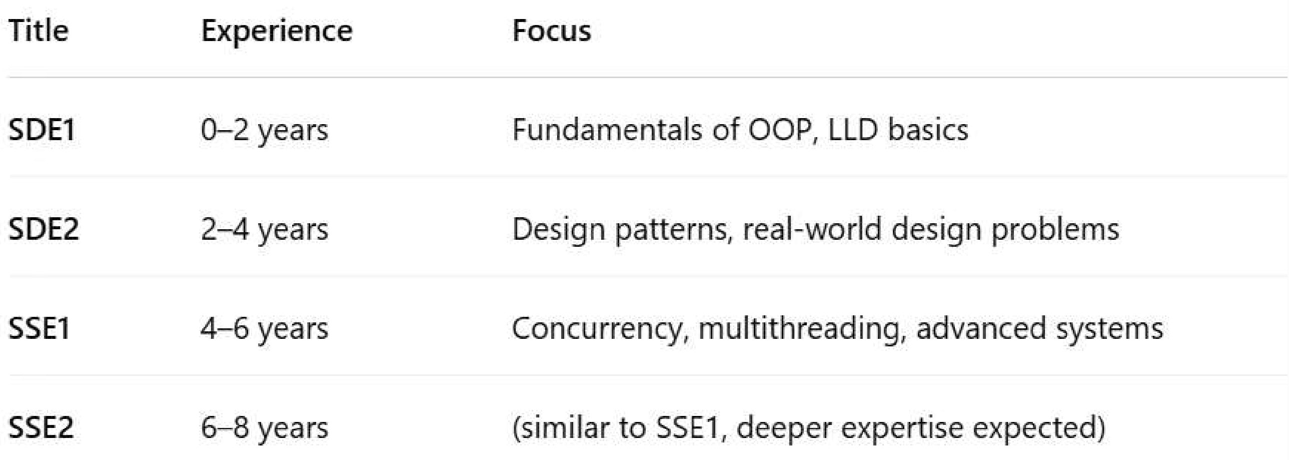
🚀 Low-Level Design (LLD) Crash Course for Software Engineers (0–6 Years)

In the realm of software engineering, the ability to design robust and efficient systems is paramount. Low-Level Design (LLD) plays a critical role in this process by focusing on the internal working of individual components and how they interact within a system. As companies increasingly prioritize practical design and problem-solving skills, mastering LLD is essential for aspiring and growing software engineers.

This crash course is tailored for engineers across different experience levels—particularly those with 0 to 6 years of experience—and aims to equip you with the foundational knowledge and practical skills to tackle real-world design challenges confidently. You'll explore key concepts, essential design patterns, and frequently asked LLD problems, along with tips and tricks for interviews.

📈 Experience Levels

- **SDE1** (Software Developement Engineer 1) : 0-2 years of experience
- **SDE2** ((Software Developement Engineer 2) : 2-4 years of experience
- **SSE1** (Senior Software Engineer 1) : 4-6 years of experience
- **SSE2** (Senior Software Engineer 2) : 6-8 years of experience



Title	Experience	Focus
SDE1	0–2 years	Fundamentals of OOP, LLD basics
SDE2	2–4 years	Design patterns, real-world design problems
SSE1	4–6 years	Concurrency, multithreading, advanced systems
SSE2	6–8 years	(similar to SSE1, deeper expertise expected)

> *Note: These titles and levels can vary by company, but this framework provides a general guideline.*

🧱 For SDE1 (0–2 Years): The Foundation

For the very freshers and someone with 0-2 yrs of experience should focus on the design principles and some introductory knowledge of -

- What is Low level System Design ?
- How to Approach LLD Problems ?
- LLD Interview Tips

These are essential for all levels, but especially critical when starting out.

🧠 Object-Oriented Programming (OOP) Basics

Ensure you're confident with:

- Introduction to Classes and Objects

- Class Relationships : A Deep Dive
- Constructor and It's types
- This Keyword in OOPS
- Polymorphism and It's Types
- Inheritance and It's types
- Encapsulation
- Abstraction
- Access Modifiers
- Generics and Wildcards

📐 Core Design Principles

- SOLID Principles (most important—revise frequently)
- DRY Principle
- KISS Principle
- YAGNI Priniciple

🔧 SDE2 ((Software Developement Engineer 2) : 2-4 years of experience

If you have some experience companies go on in asking some design patterns like creational, behavioral, structural etc. Let's see how we should prepare for them firstly if you have less time to prepare just go through the important concepts mentioned below and if you have much time then you can also prepare the other topics mentioned in good to know list.

🎨 Creational Design Pattern -

To get to know about the concept and basic intro of this pattern go through this article and the video attached to it:

Creational Design Pattern Introduction

Important topics :

- Factory Design Pattern
- Builder Design Pattern
- Singleton Design Pattern

Good to know :

- Abstract Factory Pattern
- Prototype Design Pattern

🎭 **Behavioral Design Pattern :**

To get to know about the concept and basic intro of this pattern go through this article and the video attached to it:

Behavioral Design Pattern Introduction

Important Topics:

- Strategy Design Pattern
- Observer Design Pattern
- Iterator Design Pattern
- Command Design Pattern
- State Design Pattern

Good to Know:

- Mediator Design Pattern
- Template Design Pattern
- Chain of Responsibility
- Visitor Design Pattern
- Memento Design Pattern

🏗️ **Structural Design Pattern :**

To get to know about the concept and basic intro of this pattern go through this article and the video attached to it:

Structural Design Pattern Introduction

Important Topics:

- Adapter Design Pattern
- Composite Design Pattern

Good to know:

- Facade Design Pattern

- Decorator Design Pattern
- Bridge Design Pattern
- Proxy Design Pattern
- Flyweight design Pattern

🎮 LLD Interview problems:

Must Prepare:

- Design Tic Tac Toe Game
- Design Snake & Food Game
- Design Parking Lot
- Design Car Rental System
- Design File System

you should focus on preparing for the above problems as they primariliy covers most of the concepts.

Good to Know (if time permits):

- Design Chess Game
- Design Elevator System
- Design Inventory Management System
- Design Vending Machine
- Design Logging System
- Design Splitwise
- Design ATM Machine

These questions are also frequently asked but if you are short on time you can prepare the important one mentioned in the previous list.

⚙️ For SSE1 (4–6 Years): Advanced Topics

This level demands strong understanding of concurrency, multithreading, and system design.

🧵 Concurrancy in java:

Important topics:

- Threads - Thread class and Runnable Interface

- Thread Executors
- Thread Synchronization
- Locks and Types of Locks
- Semaphore
- Java Concurrent Collections

Good to know:

- Thread pool and Thread lifecycle
- Thread Communication
- Future and CompletableFuture

🧪 Concurrancy interview problems:

Important problems:

- Design Bounded Blocking Queue
- Multithreaded Web Crawler

Good to know:

- Print Zero Even Odd
- Fizz Buzz Multithreaded
- The Dining Philosophers

⚡ LLD+ Concurrency interview problems:

Important Problems:

- Design Movie Ticket Booking System
- Design cache
- Design Pub-Sub Model like Kafka
- Design Rate Limiter

⏰ Last-Minute Preparation (1–2 Days)

If you're running short on time and have just 1 or 2 days to prepare, focus on high-impact topics that are frequently asked in interviews and can give you the most return on your time investment.

Here's a smart way to prioritize:

🎯 Must-Cover Topics

- LLD Interview Tips (Understand how to approach and communicate design problems during interviews.)

- SOLID Principles (These are fundamental and often directly referenced during interviews.)

Key Design Patterns

- Factory Design Pattern
- Builder Design Pattern
- Singleton Design Pattern
- Strategy Design Pattern
- Observer Design Pattern

System Design Problem:

- Design Movie Ticket Booking System – A classic LLD problem that touches multiple concepts in one design.

🚫 What You Can Skip (or Just Skim)

- **OOPs Basics:**

You've likely covered this in college or during earlier prep. A quick skim or video revision should be enough.

- **Advanced Topics (Concurrency, Threading, Distributed Systems):**

These are crucial but time-consuming. If you're crunched for time, it's okay to skip deep dives and just **review concepts at a high level**.

✅ Bonus Tip

Even if you're skipping other detailed topics, skim through them quickly to at least be familiar with terms and patterns. Sometimes, just knowing what exists is enough to start a meaningful conversation in interviews.