

📄

✍️

ChatGPT ▾

📄 ChatGPT

🔍 Explore GPTs

Today

Design Mastery Roadmap ⋮

Matrix Sum & Product

Yesterday

Keyboard Status Display ALP

Perfect Liked That!

Previous 7 Days

Assembly Language Differences

Exponents Division Simplification

MAC vs. MAC

Time Six Hours Ago

Dynamic Window With React

at my current level im comfortable with OOps, so tellme should i learn LLD and HLD before system design and whats the roadmap if system design

🌀

Learning Low-Level Design (LLD) and High-Level Design (HLD) is crucial for mastering system design. Here's a structured roadmap to help you proceed:

Step 1: Strengthen Core Concepts

- Object-Oriented Programming (OOP):**
 - Ensure a deep understanding of OOP principles: Encapsulation, Abstraction, Inheritance, and Polymorphism.
 - Practice designing classes and interfaces.

Step 2: Low-Level Design (LLD)

LLD involves designing the components of a system and their interactions in detail.

- Design Patterns:**

↓

 - Study and implement common design patterns: Singleton, Factory, Observer, Strategy, Adapter, etc.
 - Understand when and how to use each pattern.
- SOLID Principles:**
 - Single Responsibility Principle (SRP)
 - Open/Closed Principle (OCP)
 - Liskov Substitution Principle (LSP)
 - Interface Segregation Principle (ISP)
 - Dependency Inversion Principle (DIP)
- Code Review and Refactoring:**
 - Practice reading and reviewing code.
 - Learn to refactor code for better design and maintainability.

Step 3: High-Level Design (HLD)

HLD involves designing the architecture of the system at a higher level of abstraction.

- System Components:**
 - Understand the main components of a system: client, server, database, API, etc.
- Architectural Patterns:**
 - Study common architectural patterns: Layered (n-tier), Client-Server, Microservices, Event-Driven, etc.
- Scalability and Reliability:**
 - Learn about load balancing, caching, database sharding, replication, etc.
 - Understand concepts of fault tolerance and high availability.

Step 4: System Design

This is where you apply LLD and HLD to build complex systems.

- Design Principles:**
 - Learn about CAP Theorem, ACID vs BASE, and eventual consistency.
- Practice Design Problems:**
 - Design systems like URL shortener, messaging queue, social media feed, etc.
 - Break down the problem, identify components, and outline interactions.
- Real-World Systems:**
 - Study real-world systems and architectures (e.g., Google's Bigtable, Amazon's DynamoDB).

Recommended Roadmap:

- OOP Mastery:**

1. OOP Mastery:

- Read "Design Patterns: Elements of Reusable Object-Oriented Software" by the Gang of Four.
- Implement various design patterns in a language of your choice.

2. LLD Focus:

- "Clean Code" by Robert C. Martin. ⬇
- "Refactoring: Improving the Design of Existing Code" by Martin Fowler.

3. HLD and System Design:

- "Designing Data-Intensive Applications" by Martin Kleppmann.
- "Site Reliability Engineering" by Google.
- "The Art of Scalability" by Martin L. Abbott and Michael T. Fisher.

4. Practical Experience:

- Participate in system design interviews or mock interviews.
- Contribute to open-source projects or build your own projects.

Practice Platforms:

1. LeetCode, HackerRank, and GeeksforGeeks:

- Practice system design problems.

2. Design Gurus and Educative.io:

- Courses focused on system design interviews.

By following this roadmap, you will build a strong foundation in both low-level and high-level design, enabling you to excel in system design.

🔊 📄 ↺ 🔒 ⚙️



Upgrade plan
Get GPT-4, DALL-E, and more



Message ChatGPT



ChatGPT can make mistakes. Check important info.

