Day 1 :

System Design

Introduction To System Design

* What is How-Level Design (LLD) ?

HLD → detailed Design process in Software Development

→ Main focus : 'implement individual' components described in the High Level Design.

→ Analogy: Construct actual house from the plan.

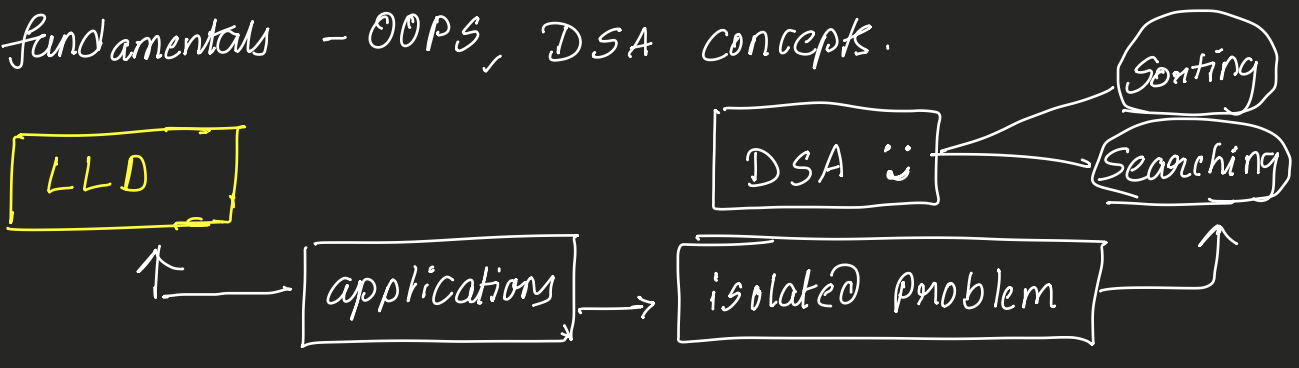
→ Includes UML Diagrams, Data structures & Algorithms

— Source [GFG]

In other words,

we have an isolated Problem. for instance Searching an element in the array / Sorting an array. we have algorithms to solve it like Linear Search, Binary Search, Merge Sort, Quick Sort.

we actually build an application using the core computer fundamentals - OOPS, DSA concepts.



⭐ What is the edge of being strong with LLD along with DSA ?

Illustrative Story :

Scenario: Build a ride-booking app like Uber/Lyft.

Student 1 : Good at DSA but Not LLD

Student 2 : Good at both DSA, LLD ✓

Student 1 approach :

- Decomposition of problem :
  - Identifies the city map to graph Nodes, roads to edges. (Basically good correlation).
  - Use Dijkstra's algorithm to find shortest path.
  - Use a min-Heap PQ to match riders closest to Passengers.
- **Gaps :**
  1) No detailed analysis of functional and non-functional requirements.
  2) No identification of classes/entities (User, Rider, Location, Notification, Payment).

3) Data security Omitted. (Handling of sensitive information)

4) Missing integration points (notifications, Payment Gatways)

5) Missing the consideration for scaling to millions of users.

Student 2 approach :

- Decomposition of problem :
  - Entity identification

    Objects : User, rider, Location, Notification service, Payment...
  - Define relationships & interactions

    i) How user and riders are matched.

    ii) How can the Notification Service and Payment Gatway can be integrated?
  - Non- functional Requirements:

    i) Data security is addressed.

    ii) Scalability : Application should handle millions of users without compromising on performance.

Then use DSA and computer fundamental concepts to implement the application components.

eg: Embed shortest-path Algorithm & driver matching heap using the OOPs design framework.

⭐ Core LLD Principles and Focus Area:

1) Scalability

- Handle large user volumes easily.
- Code structure should allow rapid, low-effort expansion (adding servers, features).

2) Maintainability

- New features should not break the existing ones.
- Code should be properly structured & easily debuggable.

3) Reusability

- Write loosely-coupled, "Plug and Play" modules (eg. Generic Notification or matching algorithms and use across apps like Doordash, Amazon Delivery, Swiggy).

⭐ How is LLD different from HLD?

- High-level Design (HLD) focus on System architecture not the code structure / implementation.
- Some key focus areas: Techstack used, Database: SQL Vs NoSQL
- Server scaling and deployment: Autoscaling, load balancers, cost optimization on AWS / GCP / Azure.
- Minimizing cloud / server expenses per load.

ⓐ Summary & Key Takeaways :

- DSA : Brain of the application : Using algorithms to solve the specific tasks.
- Low-Level Design: Skeleton, Object Models, Class Diagrams, code organization, where algorithms fit in.
- High-Level Design: Architecture system wide infrastructure, tech stack, databases, servers.

"If DSA is the brain, LLD is the skeleton of your applications".

These concepts are crucial to understand and not just memorize them. These functional and non-functional requirements should be gathered and the discussion is open-ended. You need to understand these for making trade offs among them.

Youtube link: http://youtube.com/watch?v=AK0hu0Zxua4