Lecture 1 Notes :

1. What is Low-Level Design (LLD)?

Definition: Designing the internal structure ("skeleton") of an application by identifying classes/objects, their relationships, data flows, and how DSA solutions plug into this structure.

- **DSA**: Solves isolated problems (e.g. "find shortest path in an array/graph") using algorithms like binary search, quicksort, Dijkstra's, heaps, etc.
- **LLD**: Determines which objects exist in the system and how they interact, then applies DSA inside that structure.

2. Illustrative Story: Two Approaches to Building "**QuickRide**"

- Scenario: Build a ride-booking app ("QuickRide") like Uber/Ola.

Anurag's DSA-First Approach:

1. **Problem decomposition:**

- Map city intersections to graph nodes, roads to edges.
- Use Dijkstra's algorithm to compute shortest route.
- Use a min-heap (priority queue) to match riders to closest drivers.

2. **Gaps:**

- No identification of classes/entities (User, Rider, Location, Notification, Payment).
- Omits data security (masking phone numbers).
- Missing integration points (notifications, payment gateways).
- No consideration for scaling to millions of users.

Maurya's LLD-First Approach:

1. **Entity identification:**

- Objects: User, Rider, Location, NotificationService, PaymentGateway, etc.

2. **Define relationships & interactions:**

- How User and Rider connect via Location.
- How NotificationService and PaymentGateway integrate.

3. **Non-functional concerns:**

- Data security: Protect personal info.
- Scalability: Architect code to handle millions of users without performance collapse.

4. **Then apply DSA:**

- Embed shortest-path algorithm and driver-matching heap inside this object-oriented framework.

3. Core LLD Principles & Focus Areas

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

2. **Maintainability**
   - New features shouldn't break existing ones.
   - Code should be easy to debug and locate bugs.

3. **Reusability**
   - Write loosely coupled, "plug-and-play" modules (e.g. generic notification or matching algorithms usable across apps like Zomato, Swiggy, Amazon delivery).

4. What LLD Is Not (vs. HLD)

- High-Level Design (HLD) focuses on system architecture, not code structure:
- Tech stack: Choice of languages/frameworks (e.g. Java Spring Boot).
- Database: SQL vs. NoSQL vs. hybrid.
- Server scaling & deployment: Autoscaling, load balancers, cost optimization on AWS/GCP.
- Cost considerations: Minimizing cloud/server expenses per load.

5. Summary & Takeaways

- DSA = Brain of an application: algorithms solve specific tasks.
- LLD = Skeleton: object models, class diagrams, code organization, and where algorithms plug in.
- HLD = Architecture: system-wide infrastructure, tech stack, databases, servers.

6. Key line to remember

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