#8weeksLLdchallenge

Day 1

- How to make scalable applications?
- LLD must for HLD, opensource, FAANG, etc
- understanding backend of swiggy, zomato, ola, etc

Prerequisites :- C++/Java

Searching in an array → Linear (if unsorted) / → Binary (if sorted)

Sorting : Quick, merge

} isolated problem

⇓

solution in DSA

⇓

make an app around it

⇓

LLD

- LLD involves building an application around DSA integrating multiple concepts.

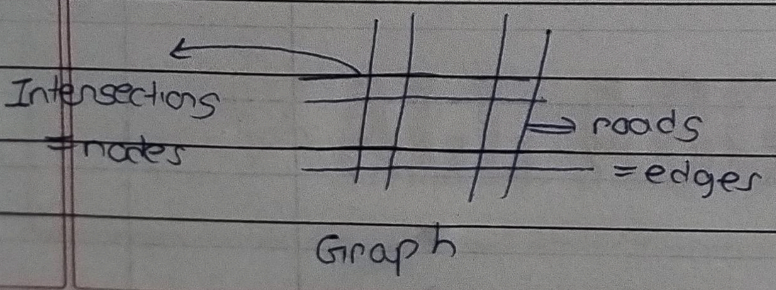
Quick ride application

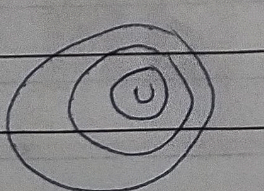
Anurag	Maurya
↳ DSA ✓	↳ DSA ✓
LLD ×	↳ LLD ✓

Anurag

Source → Dest

- Dijkstra's algo
- How to assign order?





P.queue for each user

⇓

min-heap

- pop.

- No object/entities
- No relationship
- No data security
- No Notif/Payment
- No scaling

Maurya
- ↳ Objects/Entities
  - → Users
  - → Rider
  - → Location
  - → Notif
  - ↳ Payment
- ↳ Relationship
- ↳ Data security
- ↳ Scaling

After this, DSA comes into picture.

- LLD's focus
  - ↳ Scalability (millions of users) → manage increased loads
  - → Maintainability → (new features integrated smoothly)
    - ↳ code debugging
  - → Re-usability
    - ↳ code highly reusable

- Plug and play model:-
  - allows seamless integration of code across different apps without issues [notification, payment, etc]
  - Tightly coupled code is problematic as it restricts flexi

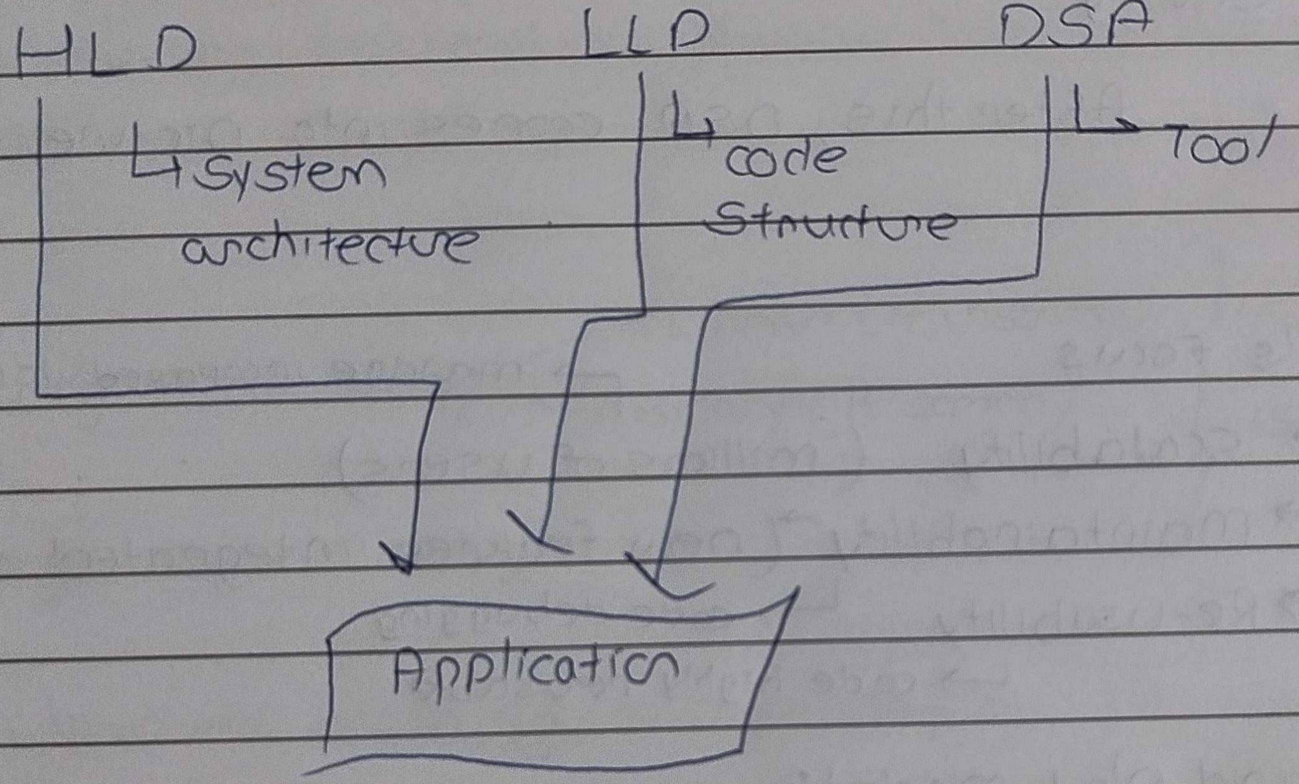
- What is not LLD?

- HLD (High Level Design)
  - ↳ Tech stack of app (Java/Spring Boot)
  - → DB using (SQL/NoSQL)
  - → server scale
  - → Cost optimization

HLD Interviews:- 0 code
- ↳ architecture
- ↳ high level diagrams vs LLD (class diagrams)

Date
Page

- LLD focuses on detailed implementation whereas HLD emphasizes the overall architecture & integratic
- HLD provides a broad overview of the system, such as module interactions and application structure



- If DSA is the brain of app, LLD is skeleton.