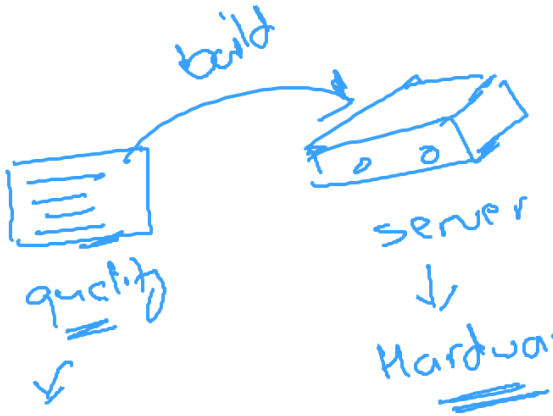
System Design
- Node SD
- SD !limited to language
- OOP = Object Oriented Programming
- LLD, HLD
  - Low Level Design
  - High Level Design
- Significance?
  - Knowledge?
  - Job Perspective?
  - Better Engineer?
    - Design Scalable System
      - Scalable : Traffic
    - More Structured System
- Expectations?
  - Right Now
  - System Design / JDs
  - Basics :
- Overview
  - TS
  - OOPs
  - LLD
  - HLD

→ H → HLD

→ S → LLD

Maintable




L → B

server { • apis

DB { • schema

1. Storage
2. Speed
3. Security

MD

vP / Principle Engine

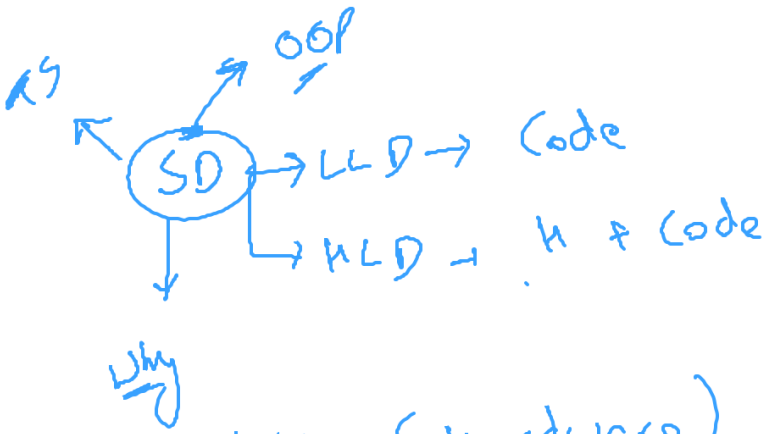
Example
1) Netflix
2) Hotstar

↑
SD

TS
OOPS
LLD = Code = Maintainable Code = Add more Feature
- JS?
- TS?
- AI : Python, TypeScript
HLD = Server, Database server, etc.

Why TS?
- TS vs JS
- type => data type
- declare the type of variable
- string, number, boolean, etc.
- Will adding type create Scalable and Maintainable system? No
  - Object Oriented Programming
- Benefit of TS?
  - Types => Early error detection
    - let num = 23;
      num = “name”;
    - enforce the type
  - OOPS

Typescript
- run?
- js vs ts
- javascript : browsers : client
- node.js : server
- typescript is the superset of javascript
    - anything done by js can be done by ts
- type checking => early error detection (compile time error)
    - ts does not deal with run time errors
    - compile time vs run time
        - compile = syntax : , : type
        - runtime = logic : undefined values, inifinity
- working of js?
- working node.js?
- working of TS?
    - TS -> JS (Transpilation)
    - Compilation vs Transpilation
        - Compilation : High level language -> Low level language
            - High Level Language : C, C++, Java, Javascript, TS
            - Low Level Language : assembly language : 0s,1s
        - Transpilation : High Level language -> High level language
            - conversion of .ts file to .js




1. Scalable (Hardware)
2. maintable (software)