0) Why HLD is important?

i) Advantages & Disadvantages of monolithy.

ii) What is a microservice

iii) Communication between microservices

iv) Distributed transactions

v) Advantages & Disadvantages of microservices

vi) when to use what?

Sandeep

=> 2019 - CSE => Techno India

→ Capgemini => FS => MorganStanley

Scaler → Ajio.com => BE

→ Visa => SSE

→ Atlassian => FS.

- Java 17
- Kotlin
- GraphQL
- AWS

Monolith

Sachin :- Shipkart.com => { domain }

↓

Ecommerce

=> modules :

i) Catalog module
ii) Search
iii) Cart
iv) Order
v) Payment
vi) Notifications
vii) Shipment | Tracking | Returns | LMS

last mile service

logistics

viii) User's module
ix) Merchant's module

First phase :- Explore | PMF | MVP

↓

Product market fit

=> Why learning HLD is important ?

→ HLD => High level design

i) Perform well

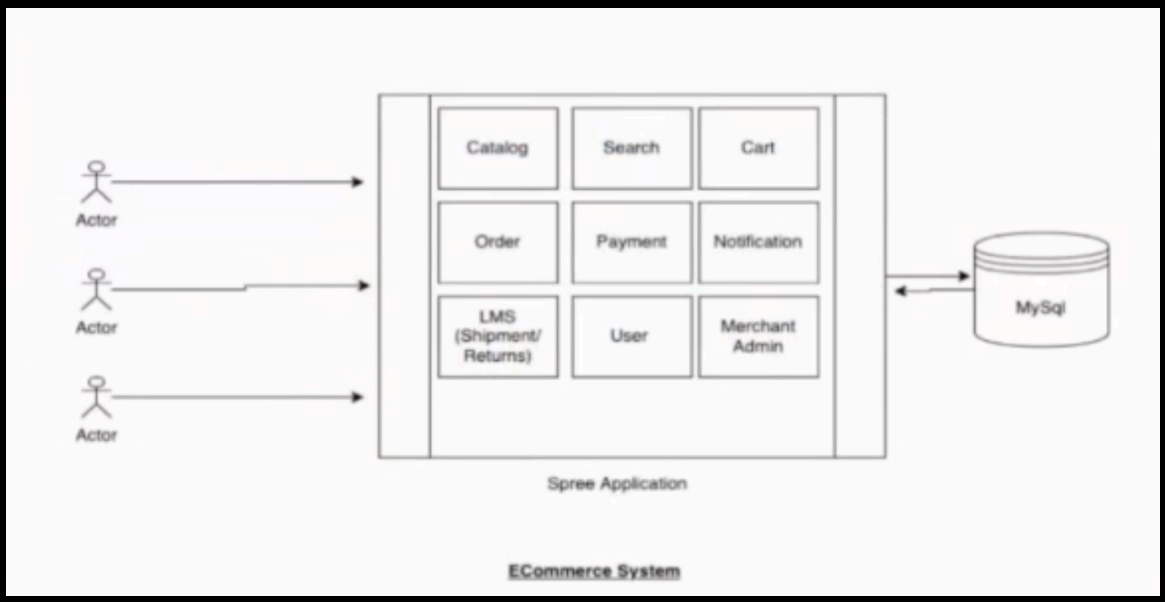
ii) Cracking interviews

Popular ecommerce

i) Magento → PHP

ii) ATh → Java

iii) Spree → ROR



2000 ⟶ 2 orders/day

2005 ⟶ 2000 orders/day

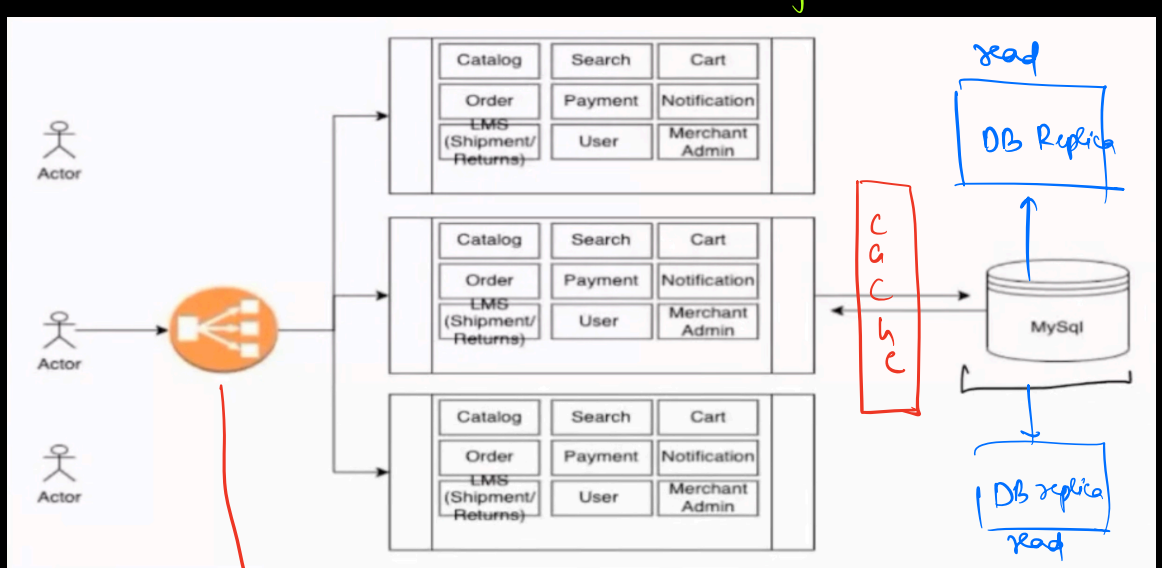
Vertical Scaling

- 1 DB - machine
- 100 TB
- 1000 TB

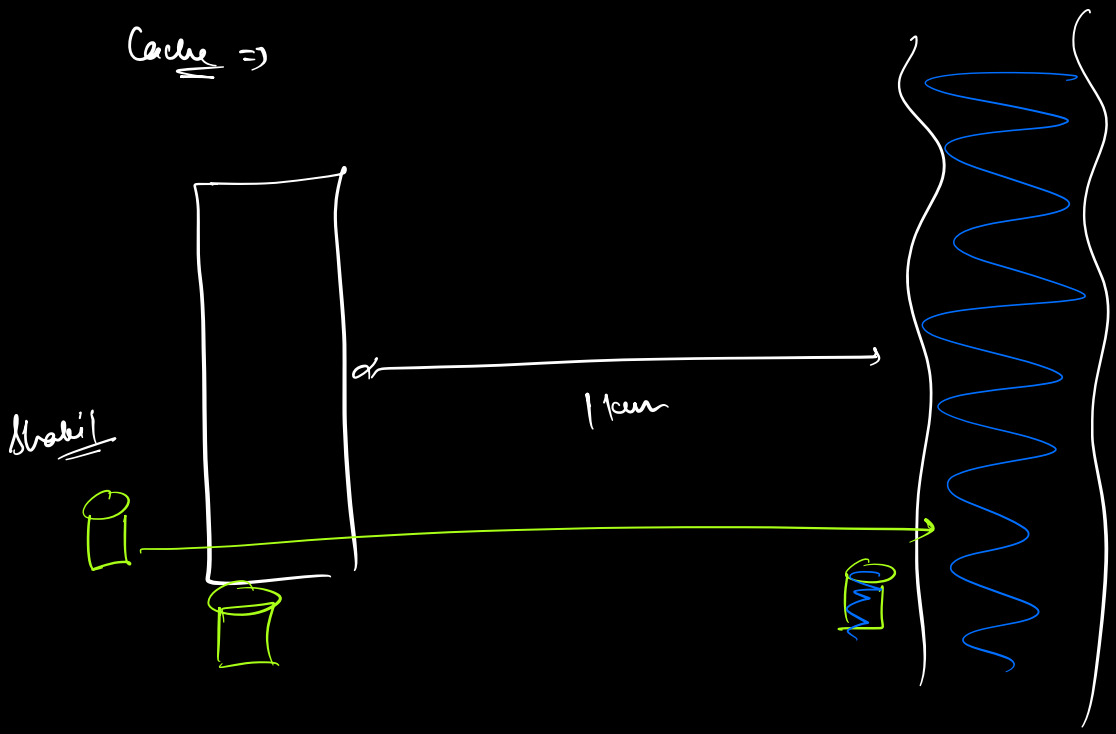
Horizontal Scaling

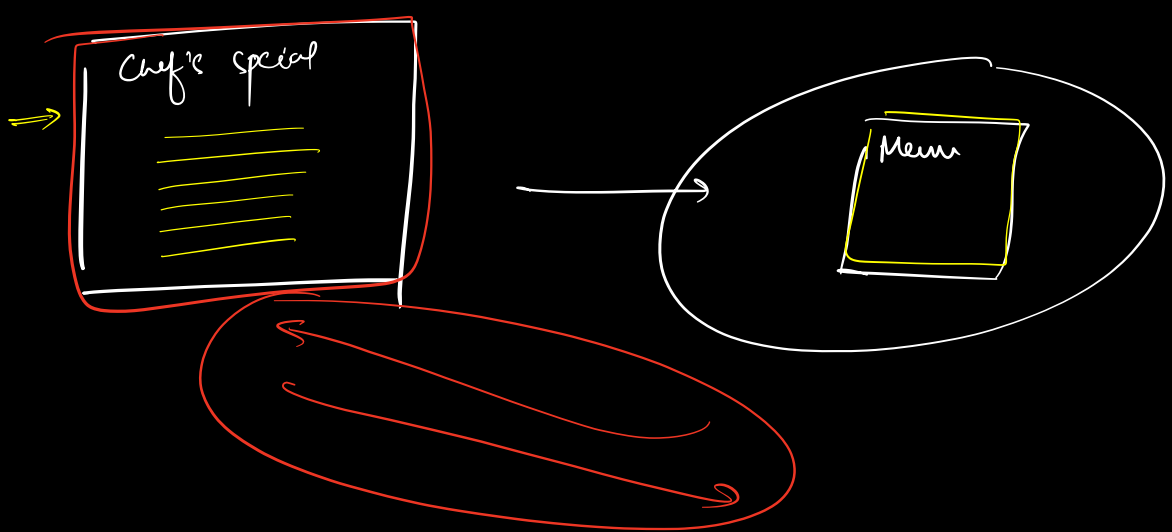
- adding more machine
- 1 ⟶ 100 TB
- 10 ⟶ 100 TB each

Horizontal Scaling



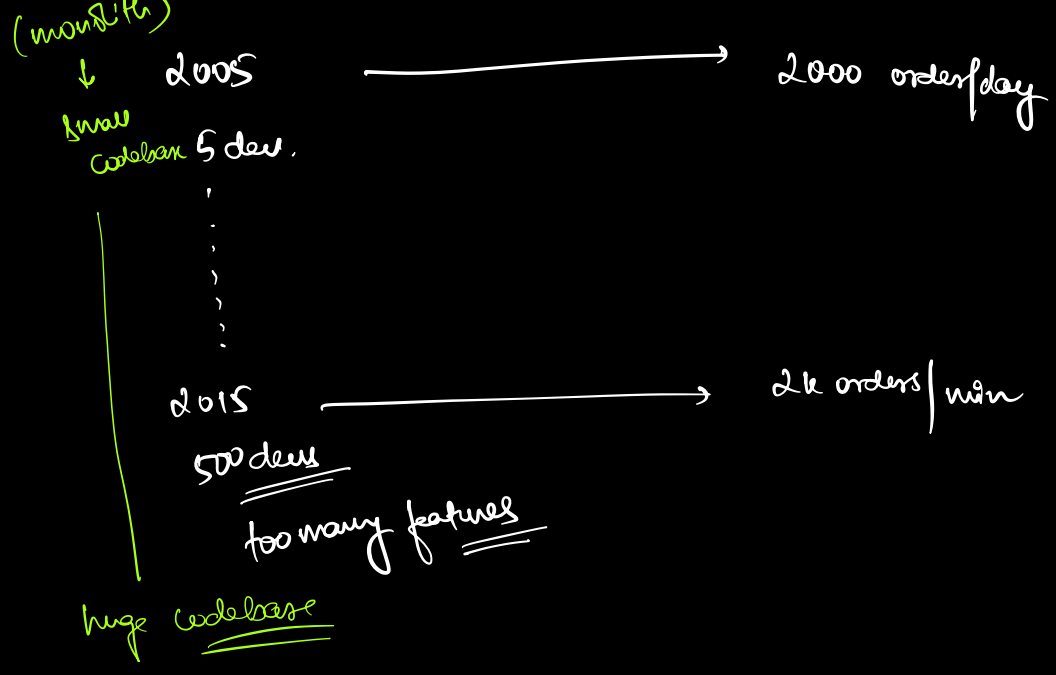
Load Balancer


Cache =>
Shabil
1 km


Chef's special
Menu

Advantages of monolith

- Easy to understand and implement
- End to end testing is very easy
- Able to maintain with a very small team
- Easy to develop for quick launch.



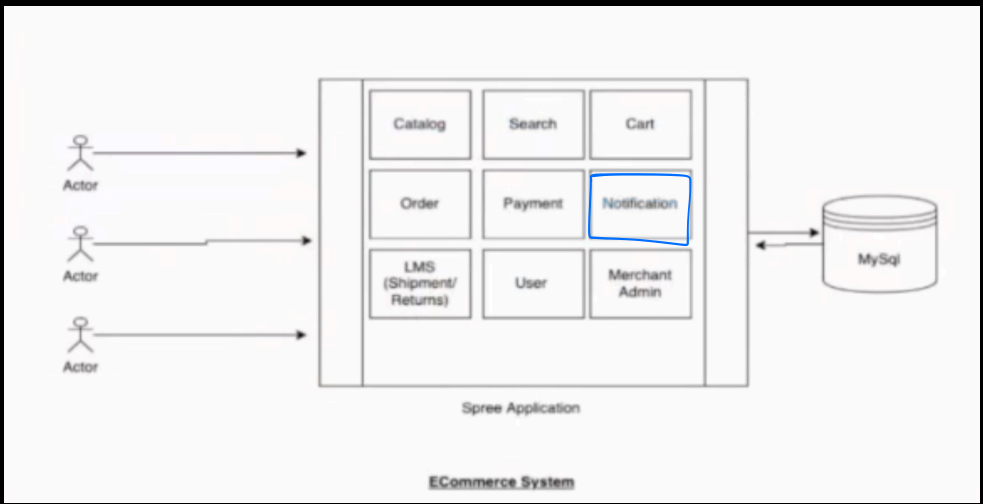
Drawbacks of monolith:

- Developer onboarding → complex code
- Making changes are difficult: tightly coupled, and minor changes can lead to huge issues.
- Selective scaling is not possible
- Insufficient use of resources
- Barrier to new tech stack
- Difficult to debug
- Extremely slow dev loop
- [Big ball of mud] ← no one knows the app entirely
- Availability

Monolith:- Apps that consists of one system where all the codebase, business logic, DB are interconnected and dependent on each other

⇒ move from monolith to microservice

2) Decomposition of micro-service:-



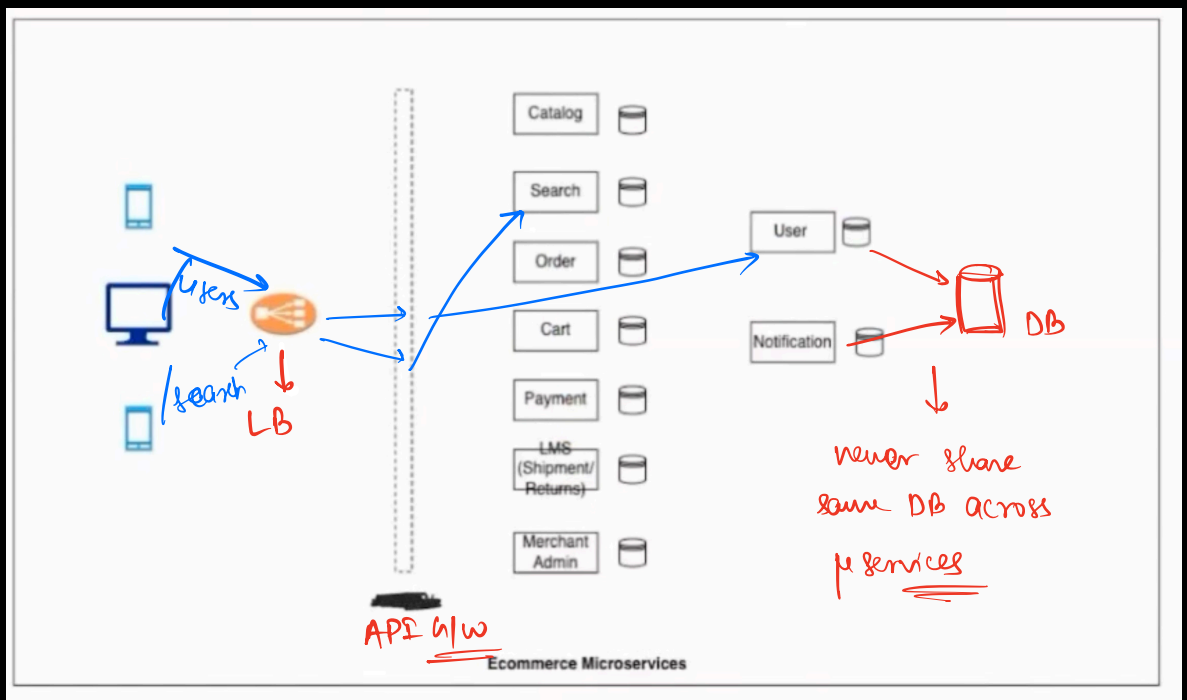
2) Decompose:-

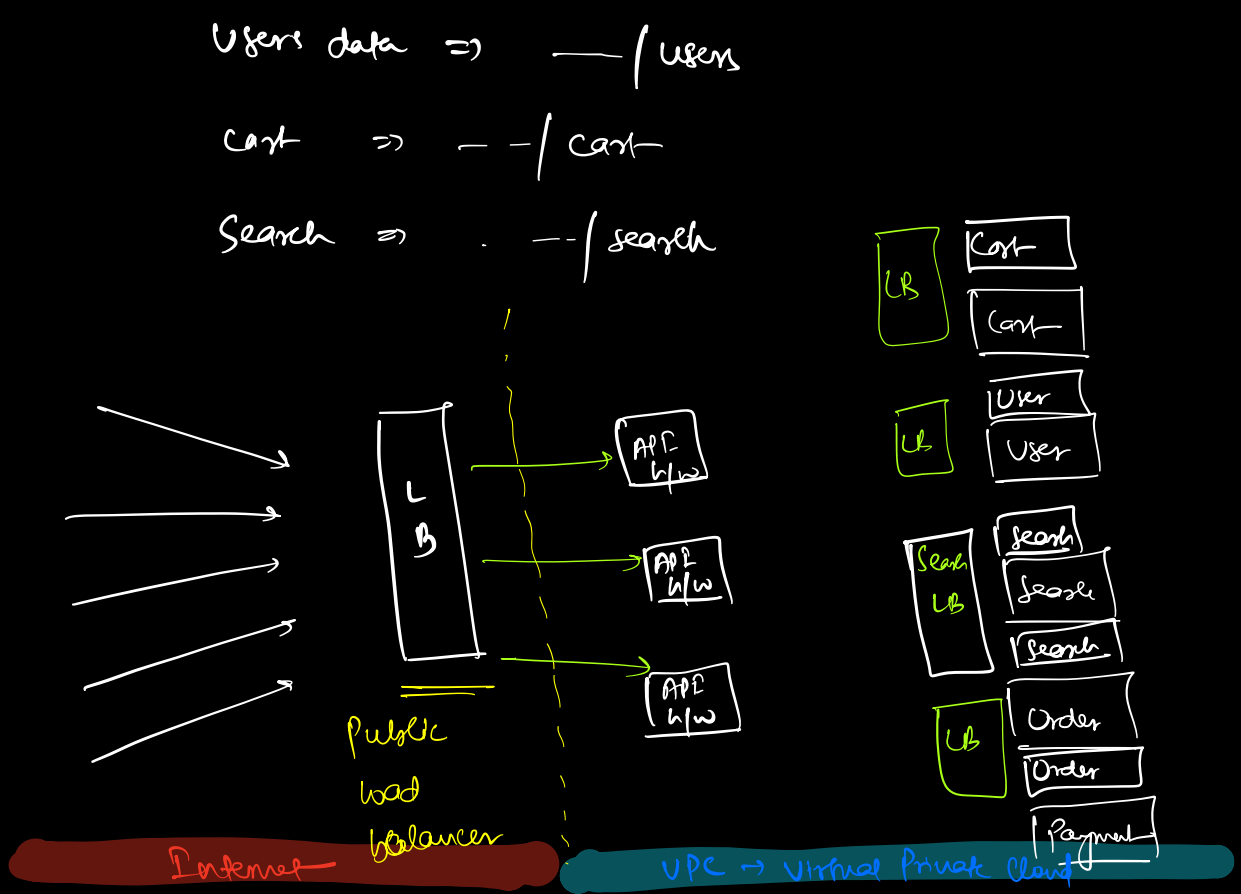
Choose a module

1) Based on scale

11) Easy to decompose

111) BU (business units)


Catalog
Search
Order
Cart
Payment
LMS (Shipment/ Returns)
Merchant Admin
User
Notification
DB
Users
/search
LB
never share same DB across μ services
API G/W
Ecommerce Microservices


Users data => /users
cart => /cart
Search => /search
LB
API G/W
API G/W
API G/W
Public load balancer
Internet
Cart
Cart
User
User
Search
Search
Search
Order
Order
Payment
VPC → Virtual Private Cloud

Advantages of μservices:-

i) Independent components ⇒ fault isolation.
no cascading failures

ii) Easy to add features & deploy

iii) Selective scaling

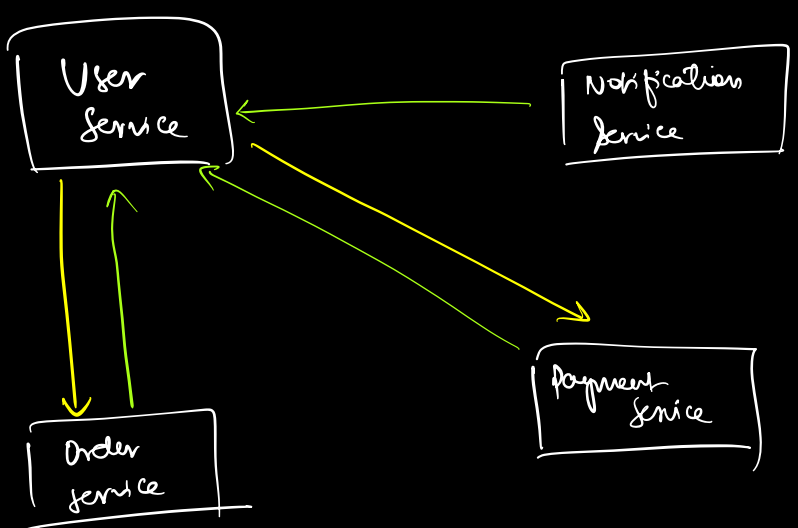
iv) Tech stack flexibility

v) Onboarding developers is easy

vi) Small teams, clear ownership

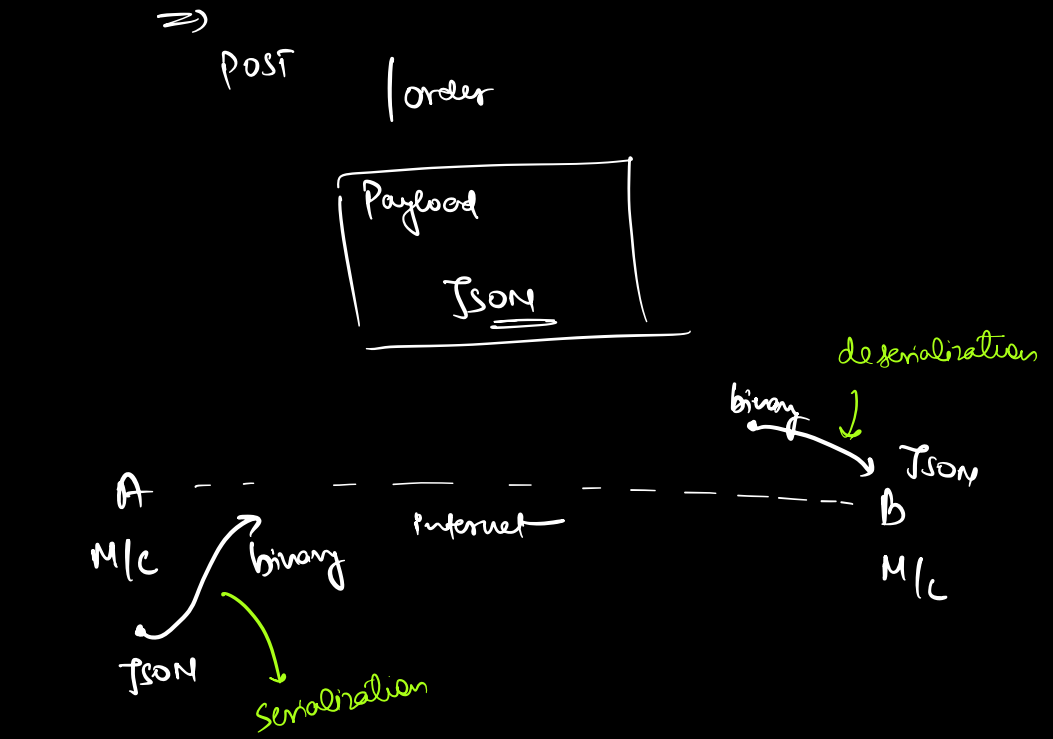
vii) easy to detect bugs

viii) faster dev loop

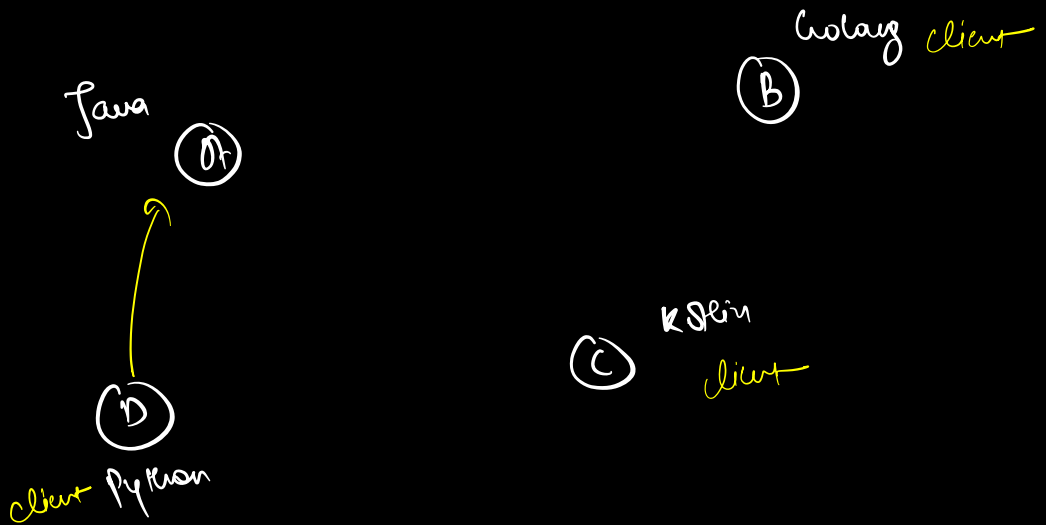


Communication b/w μservices:

1) REST APIs



⇒ Slow due to multiple steps.



⇒ lot of code required

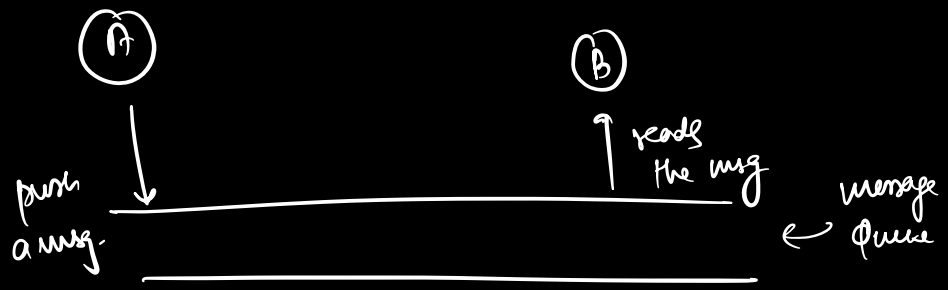
ii) RPC ⇒ remote procedural calls.

↓

Protobuf ⇒ binary

gRPC ⇒ Google RPC

iii) Events ⇒ Event driven system



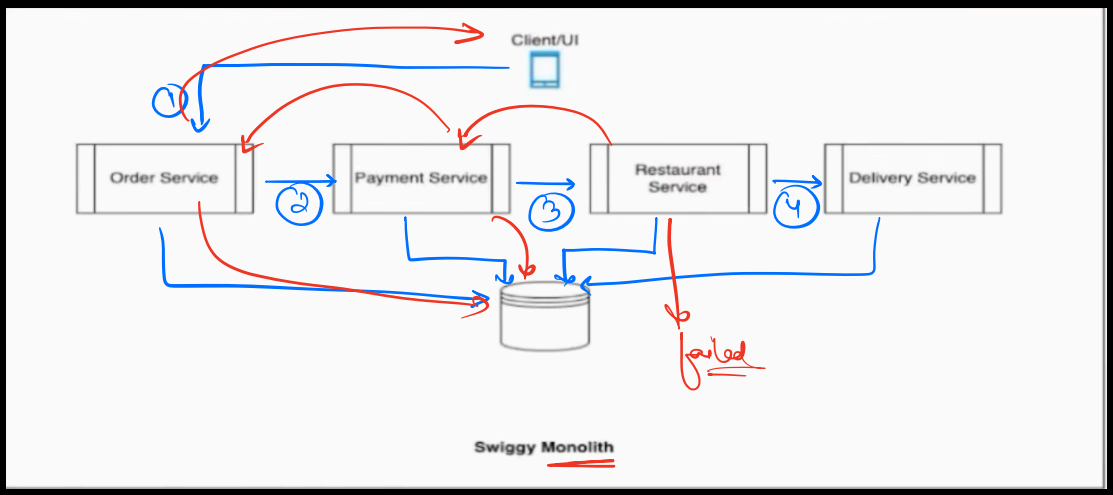
eg ⇒ Kafka, RabbitMQ, Tibco, AWS SQS

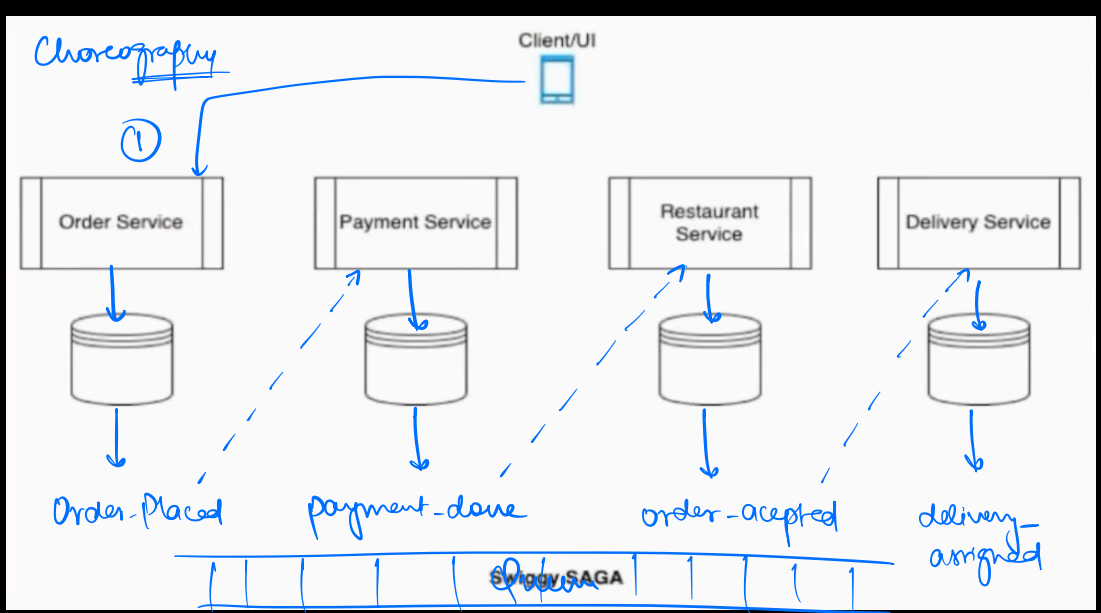
⇒ Observer design pattern ⇐

Pub/Sub ⇒ model.

⇒ Distributed transactions :

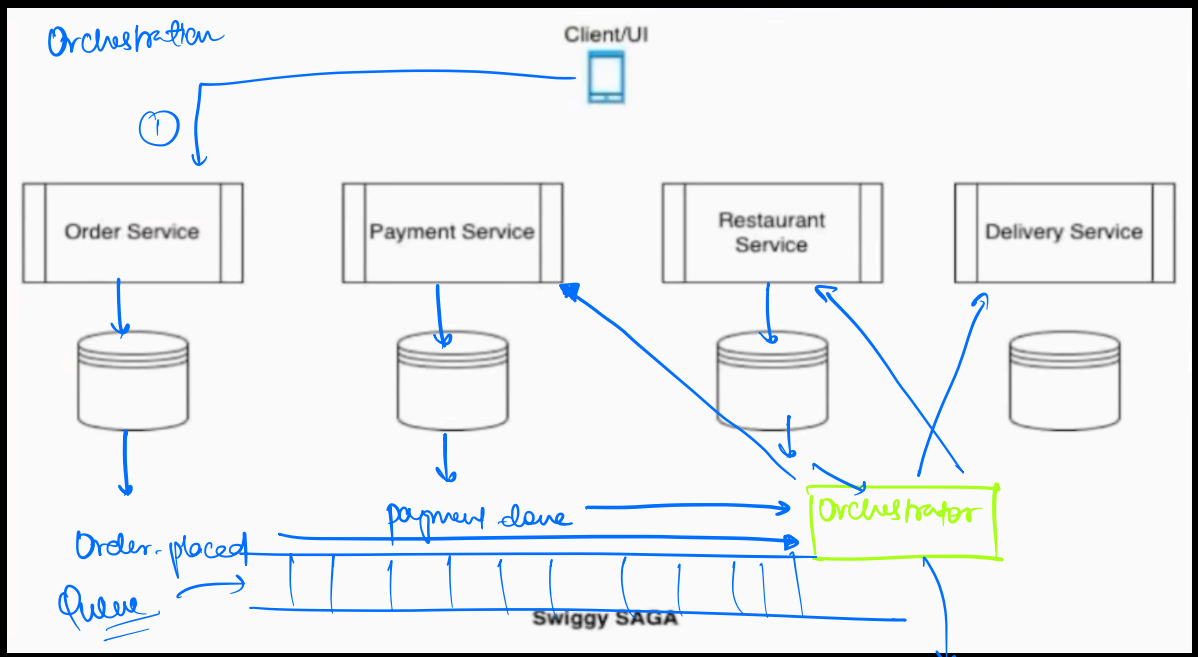
eg ⇒ Swiggy [food delivery app]





Saga ⇒ distributed transaction

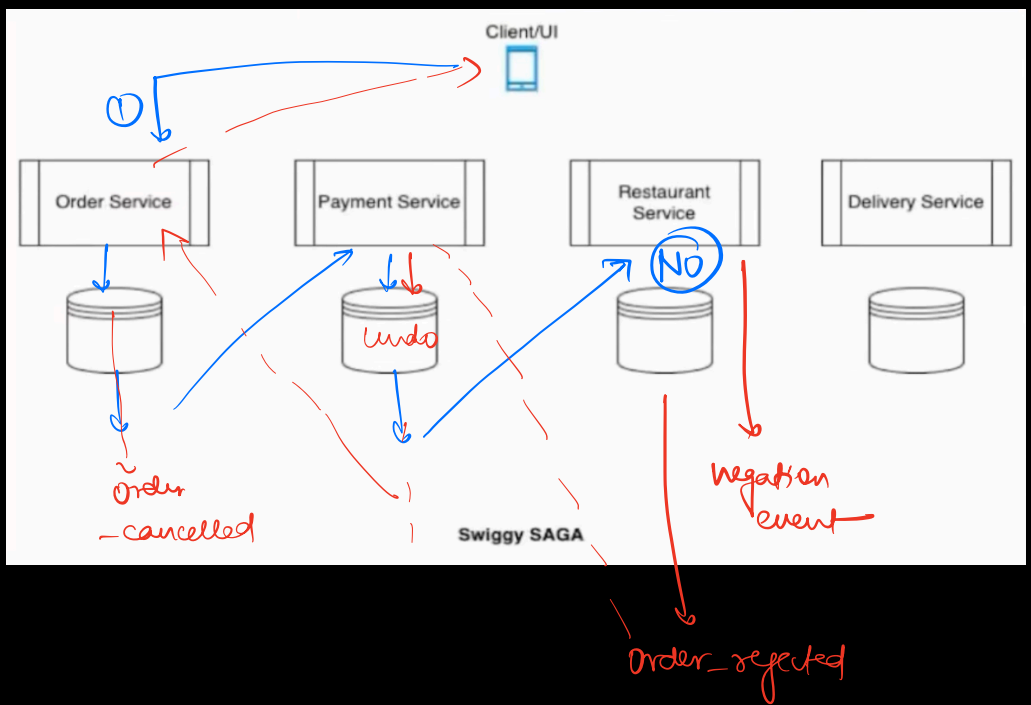
- Orchestration ⇒ someone running it
- Choreography ⇒ happens on their own



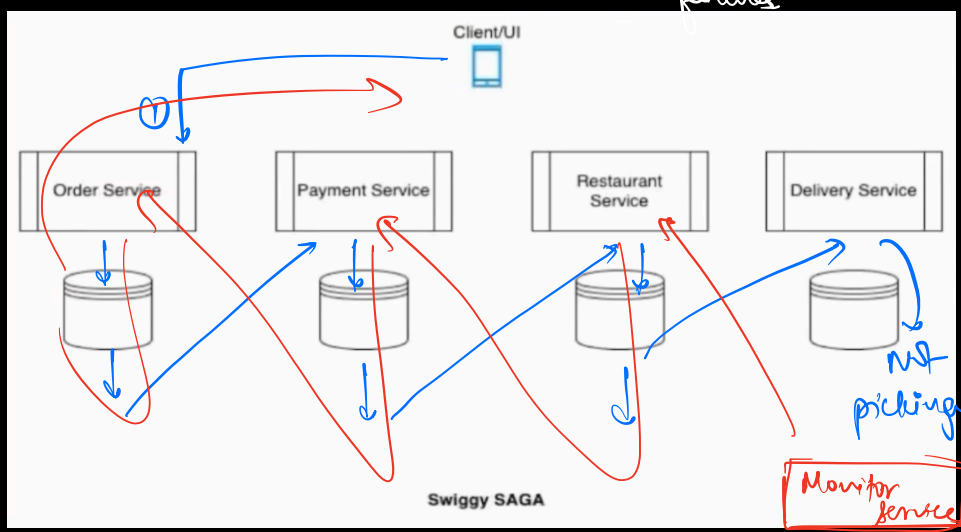
replication can be done

Handling failures:

⇒ **Compensating transactions:** taking care expected failures



⇒ **Self Compensating transactions:** taking care unexpected failures



disadvantages of microservices:

- Difficult to build, need highly skilled S/w engg.
- Heavy dependency of devops.
- Lot of monitoring is required
- N/w hops ⇒ increase latency

⇒ When to use monolith:-

- Small team
- Simple appn
- Low scale
- Quick launch
- Low expertise

⇒ When to use micro-services:-

- Scale
- complex appn
- big and talented engg. team
- availability & faster dev loop.

⇒ Start with monolith but with intention to move to μservices

Designing Data Intensive appn