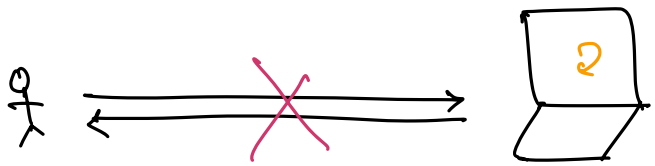
Agenda.

1. What is LLD ?
2. Why LLD is important ?.
3. How to approach any LLD problem ?.
   - → Requirements Gathering
   - → Class Diagram
   - → Schema Design
4. Doubts.

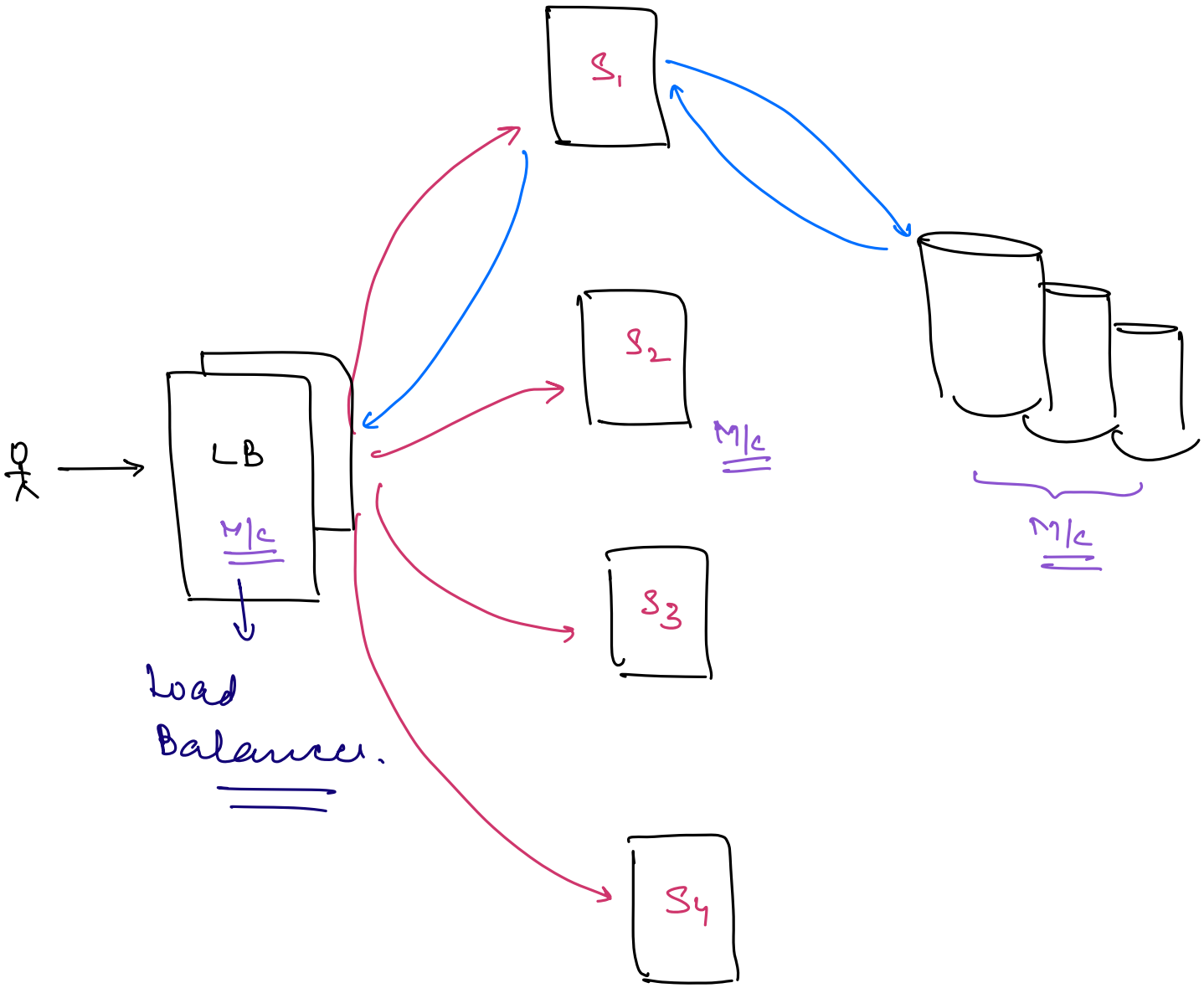
Low Level Design ⟷ High level Design.

Amazon | Google | FB | ....

→ Single Point of failure



→ huge traffic.



⇒ Distributed System.

⇒ HLD.

↳ Overview of the System.

⇒ LLD.

↳ Software / App^n

⇒ Code.

⇒ Study of writing good code

⇒ Properties of Good Code.

1) Understandable.

2) * Maintainable

3) * Extensible.

⇒ Maintainable

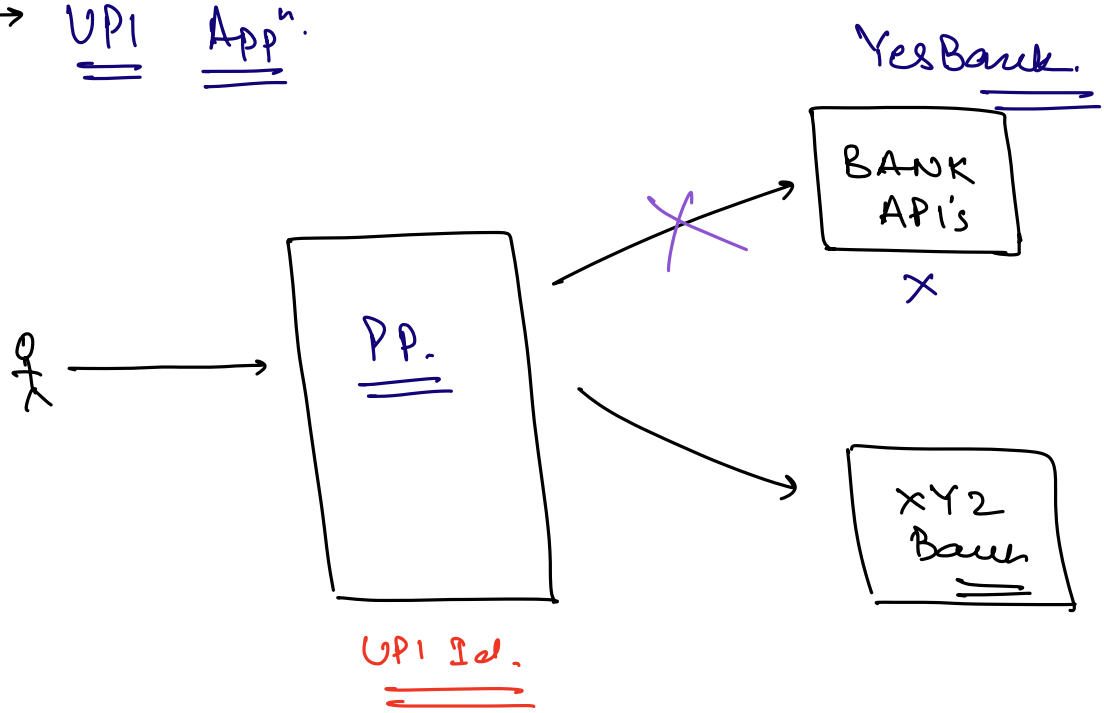
↳ Easy to find bugs in your code.

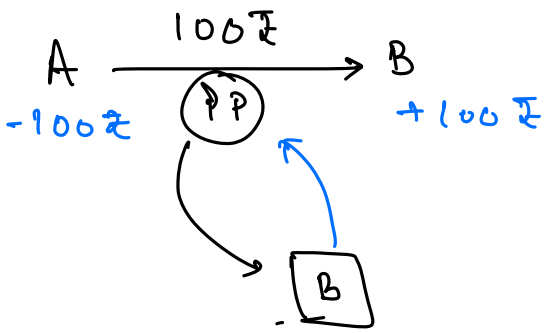
⇒ Extensible.

↳ Easy to Add new features.

⇒ PHONEPE. → fintech.

↳ UPI App^{n}.





2020

↳ Ban on YB.

⇒ PP. X

⇒ LLD. → Design Principles

↘ Design Patterns.

#

Coding → ~12% time

Debugging

Testing

Design

Documentation

Code Review.

Deployment.

88%.

LLD makes 88% of time for a SE more productive.

How to approach any LLD problem ?.

Client |

PM |

EM.

1) Requirement Gathering.

→ Understand the requirements.

→ Clarify the requirements.

→ Edge cases | Corner cases.

2) Class Diagram.

- Classes | Interfaces
- Define all the entities in our system.
- Design Patterns.

Way to writing code in different scenarios.

- Singleton
- Adapter
- Strategy
- Factory
- Builder
- Observer

3) Database | Schema Design.

- Which all the tables are going to be there in our DB.
- Relationship b/w the tables.
- Cardinality.
  - 1 : 1
  - 1 : M | M : 1
  - M : M.

Normalization

4) Code.

Flipkart
Phone
Swiggy
Zomato
Cred
Zepto
—

1st round as Machine Coding round.

(2-3 hrs).

LLD round with working code

=> Amazon
MS
—
—
—

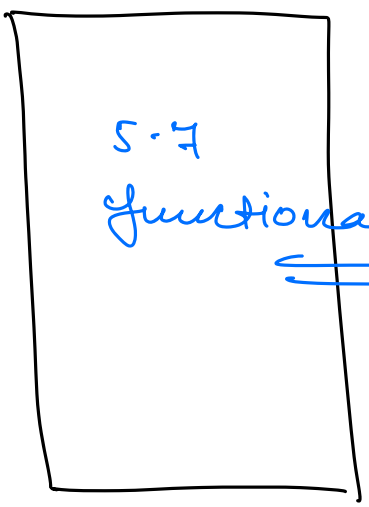
LLD without code.

Design round.

LLD of Payment App

=> Machine Coding round.

Doc



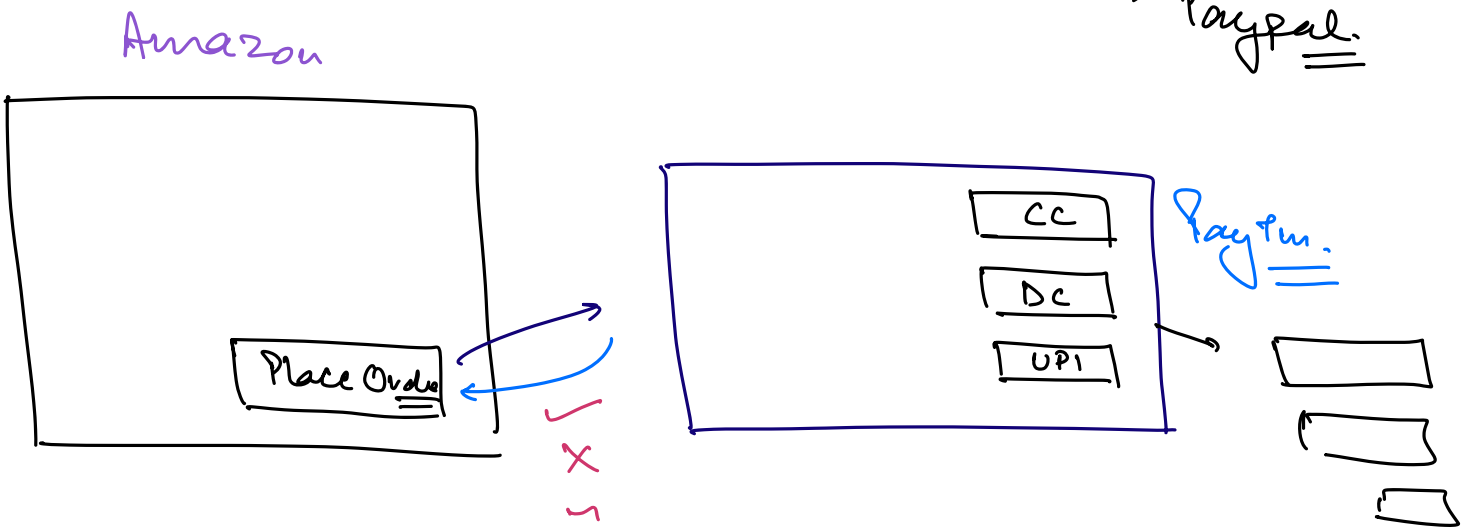
→ Requirement Gathering

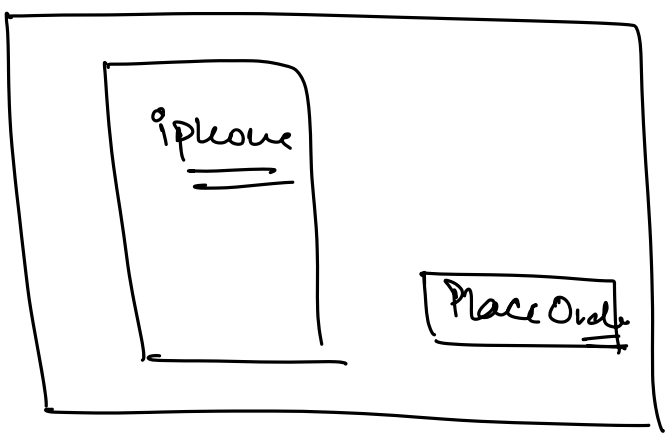
→ Class Diagram

→ Schema Design.

PayTm ⇒ Payment GW.

Razorpay
Phonepe
PayTm
Paypal.

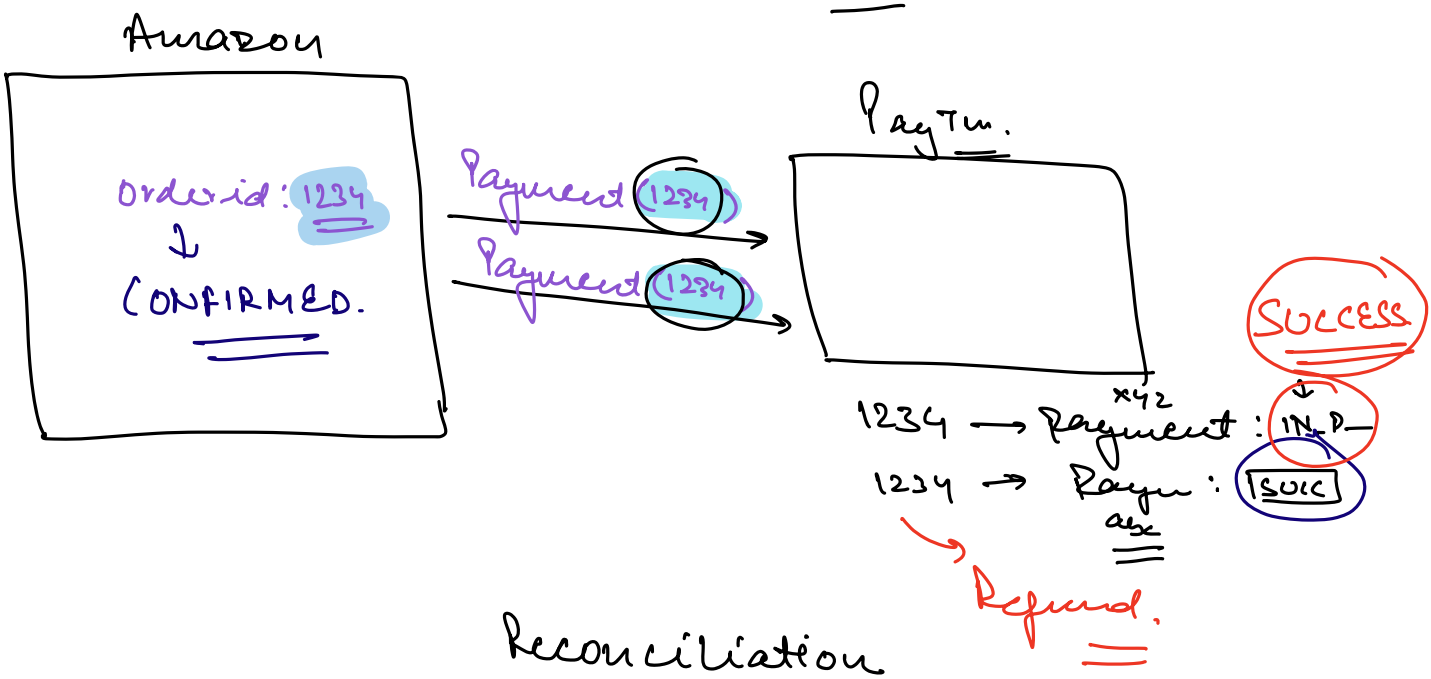




① Payment happens first

Order gets placed first. ②

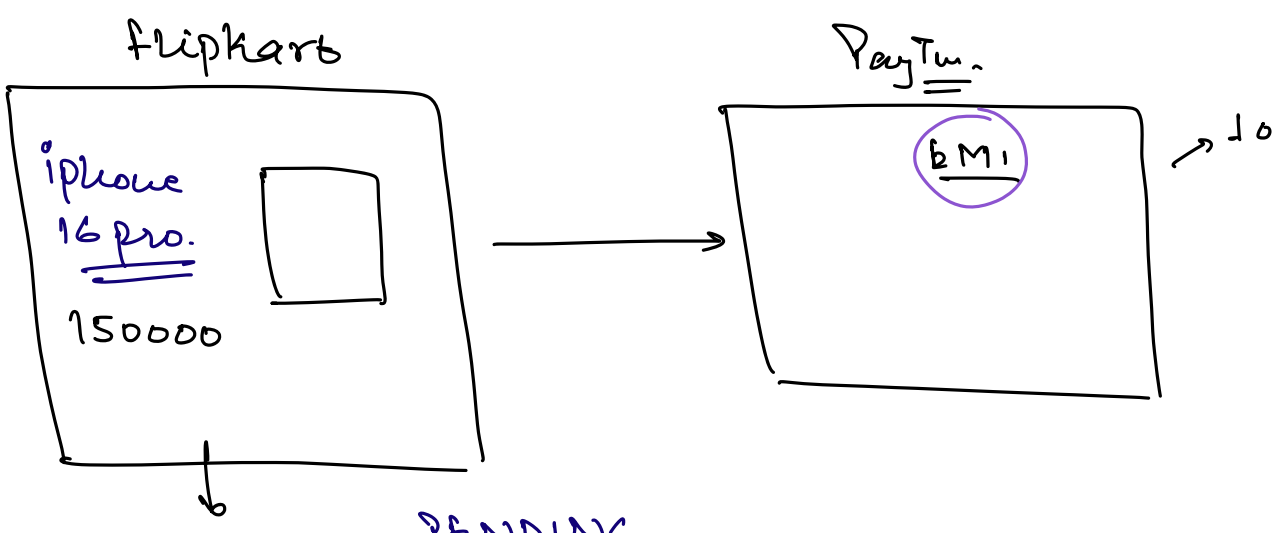
⇒ Amazon first creates an order in the Pending state and then whenever the payment happens Amazon will mark the status as CONFIRMED. if the payment doesn't get completed then Amazon will cancel the order.



Reconciliation

Transaction | Payment

PayTm : PG.

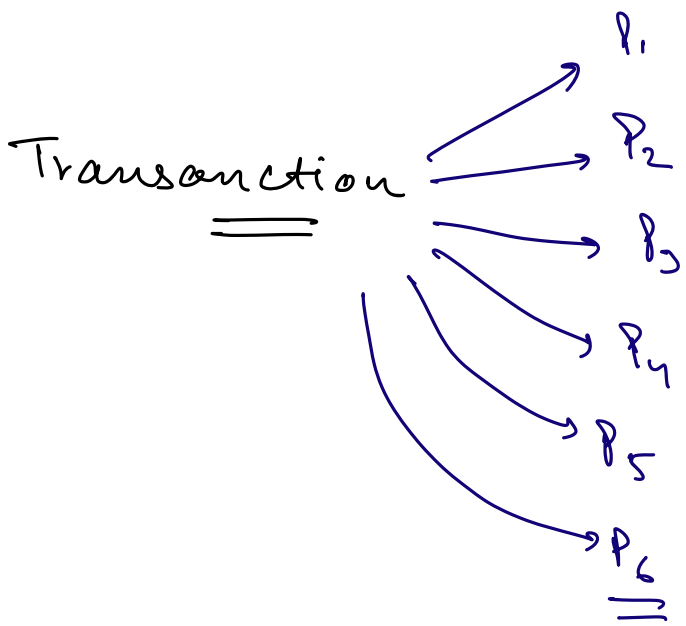


Order | Transaction → PENDING.

of 1,50,000

Transaction ⟶ 1 Payment ✓

6 months EMI



⇒ Refund.

↳ Original Source.

#

Amazon

1000₹ → 200₹ wallet

1000₹ → 800₹ UPI | CC | DC | NB.

↓

Cancel.

Transaction of 1000₹

P_1 : 200₹ wallet.

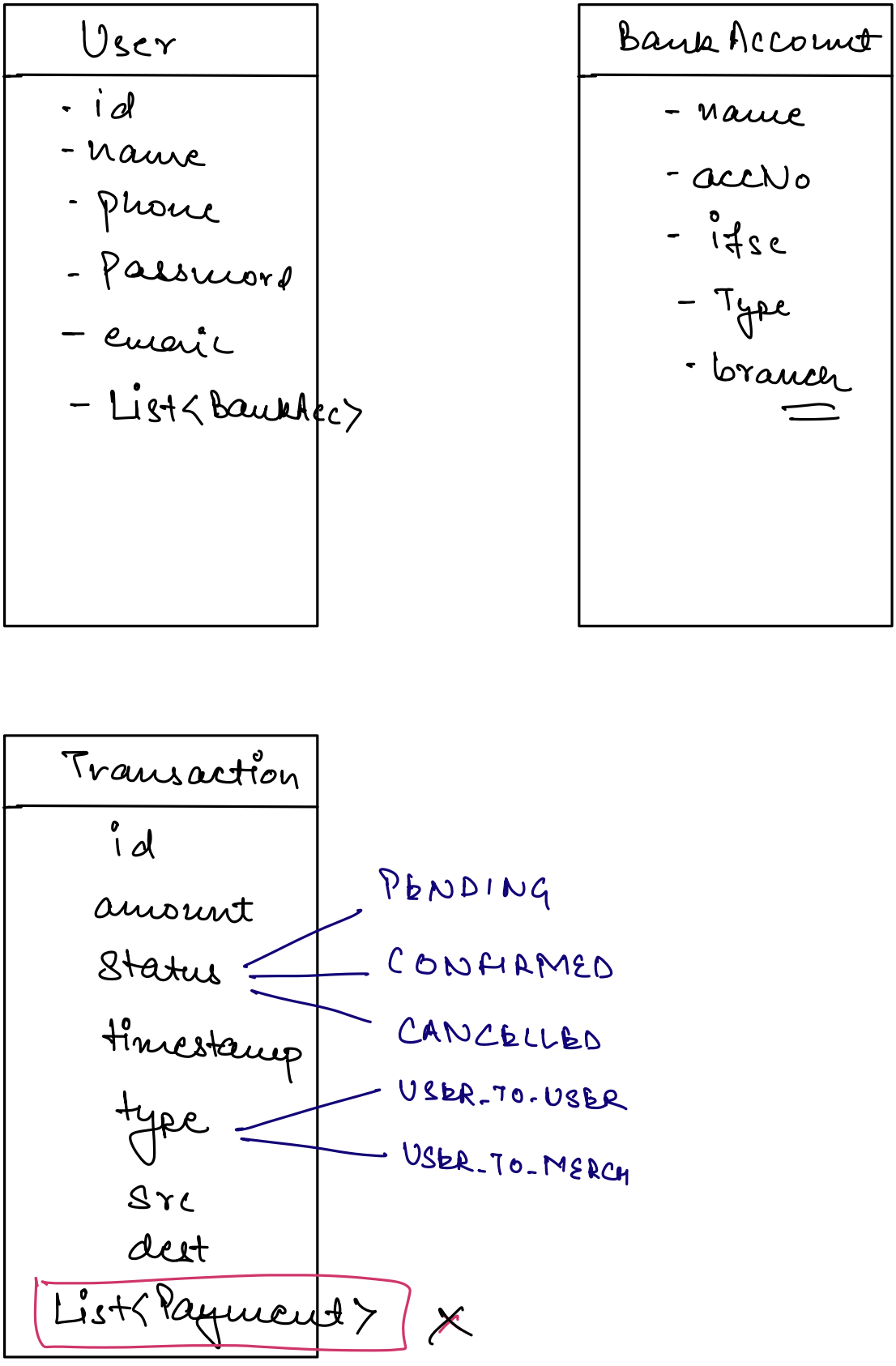
P_2 : 800₹ : CC

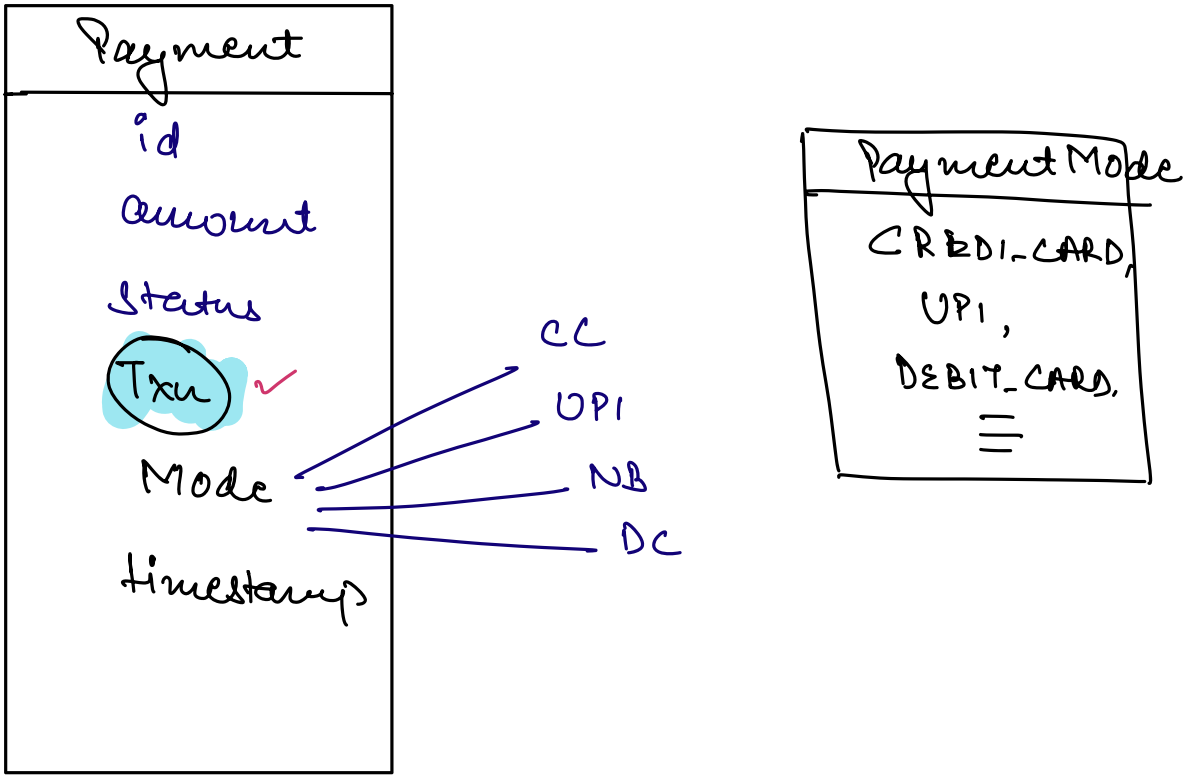
Split Payment.

Class Diagram.

⇒ Go through all the requirements & figure out all the Nouns for which we need to store data in the DB. for every such Noun create a Class.

(Entity)


User
- id
- name
- phone
- password
- email
- List<BankAcc>
BankAccount
- name
- accNo
- ifsc
- Type
- branch
Transaction
id
amount
Status
timestamp
type
src
dest
List<Payment>
PENDING
CONFIRMED
CANCELLED
USER_TO_USER
USER_TO_MERCH



Schema Design. [Relational DB].

↳ DB design.

→ for every entity, create a table in the Database.

→ figure out the attributes for every entity

↳ Primitive : Simple attrs (int | bool | String | ...)

↳ Non Primitive : Object of another class.

Create a Column

Non Primitive. : find the relationship b/w the entities. Cardinality.

- 1 : 1
- 1 : M | M : 1
- M : M.

users

id	name	phone	password	. . . . .

bank_accounts.

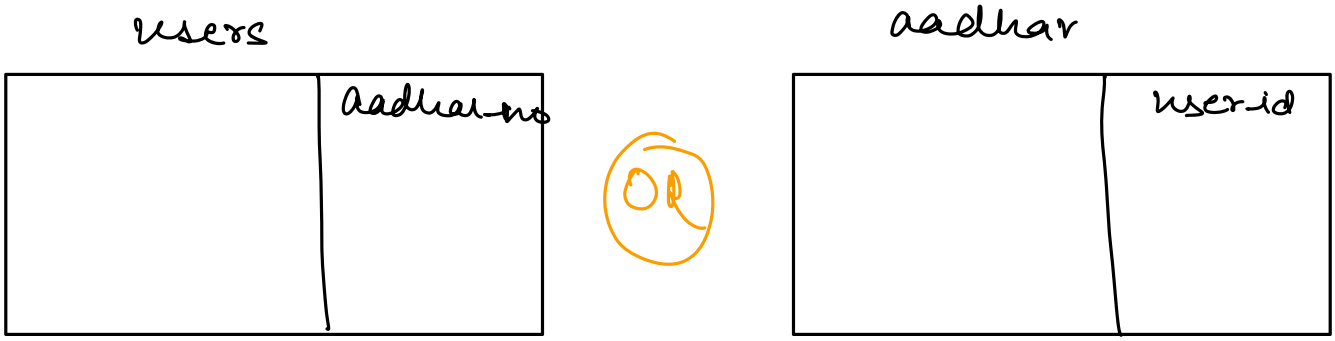
id	accNo	ifsc	- - -

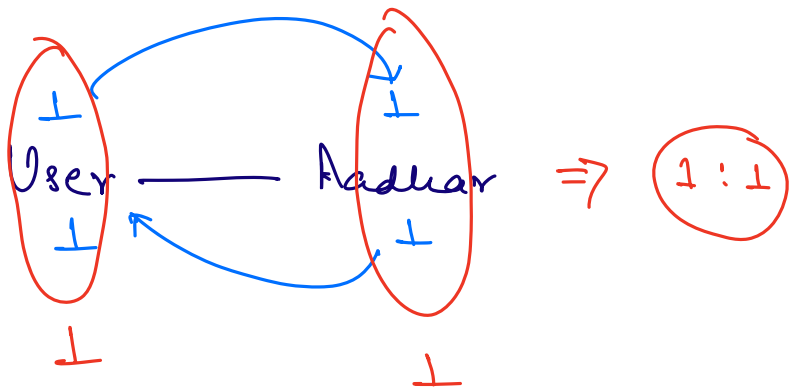
transactions

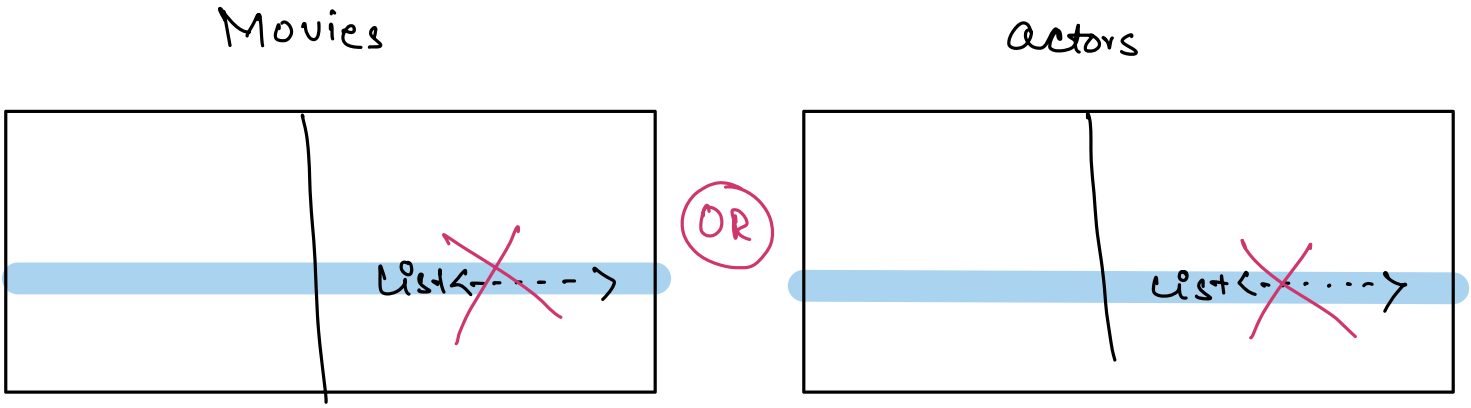
Payments.

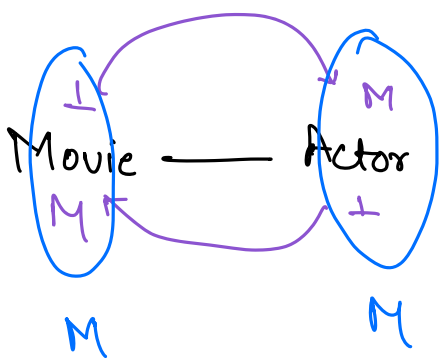
Examples.

1 : 1









⇒ M : M.

movies_actors

movie_id	actor_id.

mapping table.

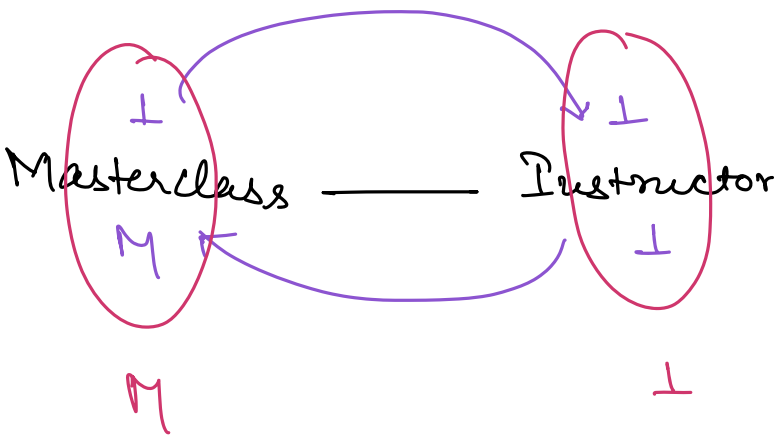
⇒

masterclasses

ins_id	

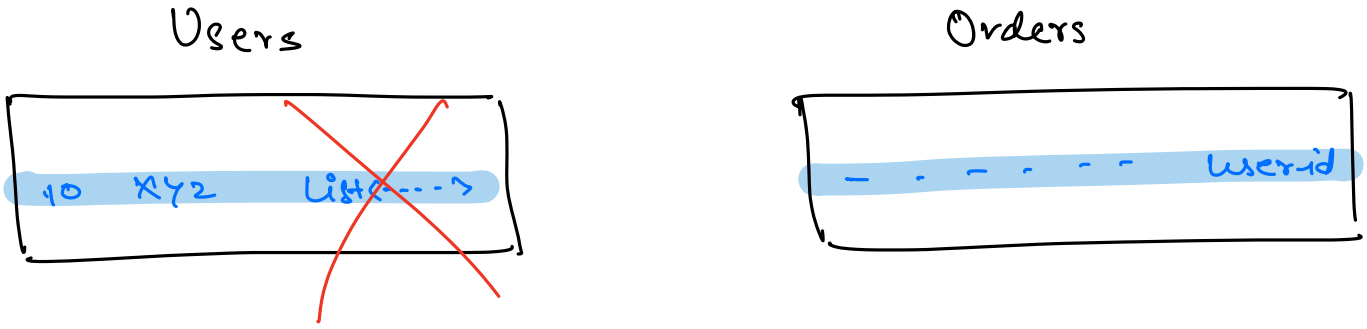
instructors

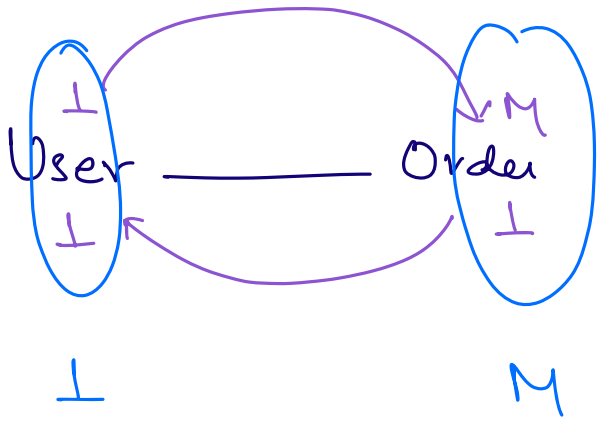
teacher	List<---7



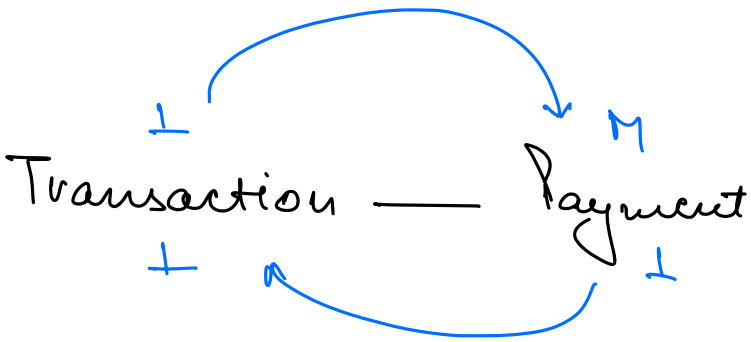
⇒ M:1

⇒ Id of (1) side on (M) side.





$\Rightarrow$ 1 : M



$\Rightarrow$ 1 : M

$\Rightarrow$ txn-id in the Payments table.
