Payment apps (LLD) low level design × high level design

→ how code base for particular s/w is structured

→ methodology of writing good code

↓ extensible (easy to add new code)
↓ maintainable
↓ Readable (understandable)
↓ Durable

high level design
↓
– an overview of something –
↳ working and interactions of various infrastructure layers to serve s/w system at desired scale & efficiency

DB sharding

S O L I D
single responsibility
open close

Approach (LLD) → class diagram (no need to mention all classes)
→ only need to mention models so classes related to design patterns/design principles
→ ...

Adapter

DB Schema design
→ tables
→ attributes
→ relationships b/w tables
→ cardinality
→ any tables for relations

LLD of payment App → might be UPI Apps
→ payment gateway
→ wallet Apps

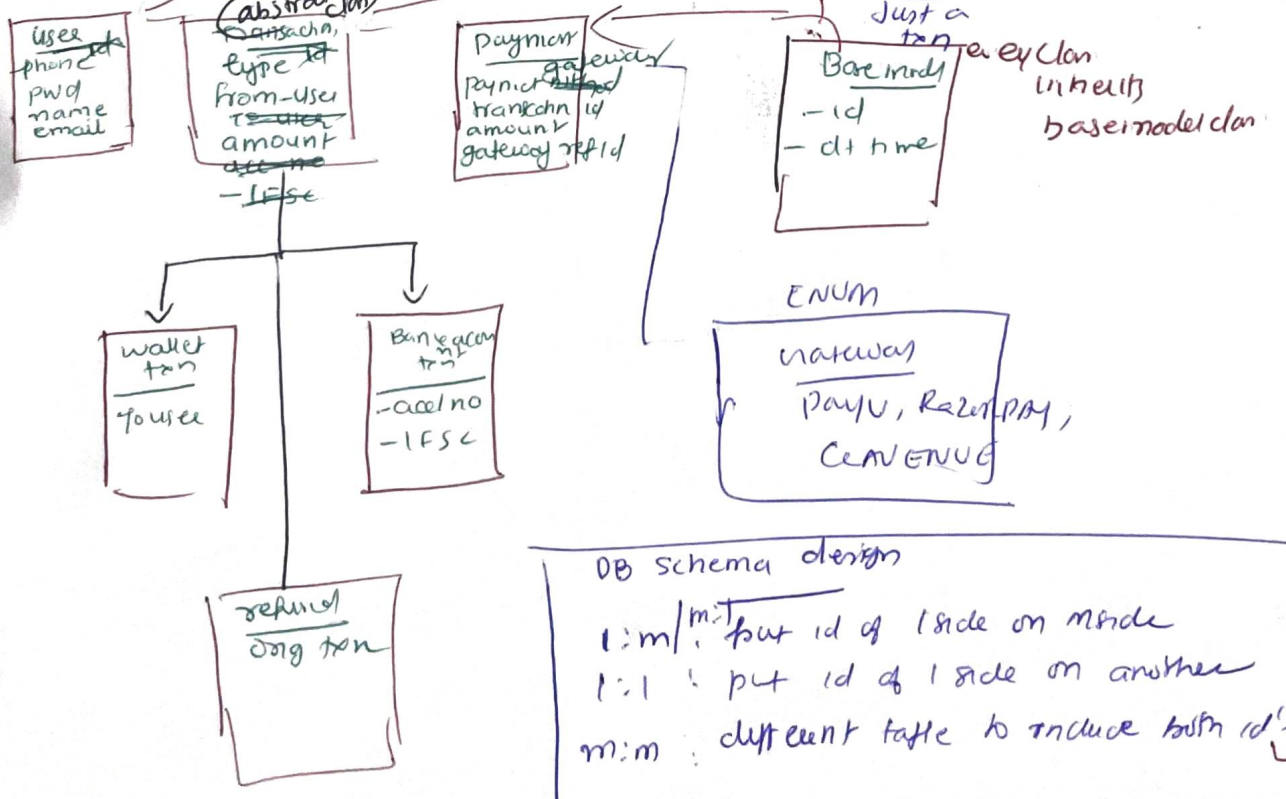
overview

↳ Wallet Apps ?

Class diagrams
go through all req. if that req has nouns (all entity about which storing data is DB)
↳ create a class for it

User / transaction / BankAccount / Payment / Refund / payment method

Just a txn entity class inherits baseModel class



DB schema design

1:m : put id of 1 side on m side
1:1 : put id of 1 side on another
m:m : different table to include both id's