⇒ How to approach schema design?

⇒ Design TicTacToe

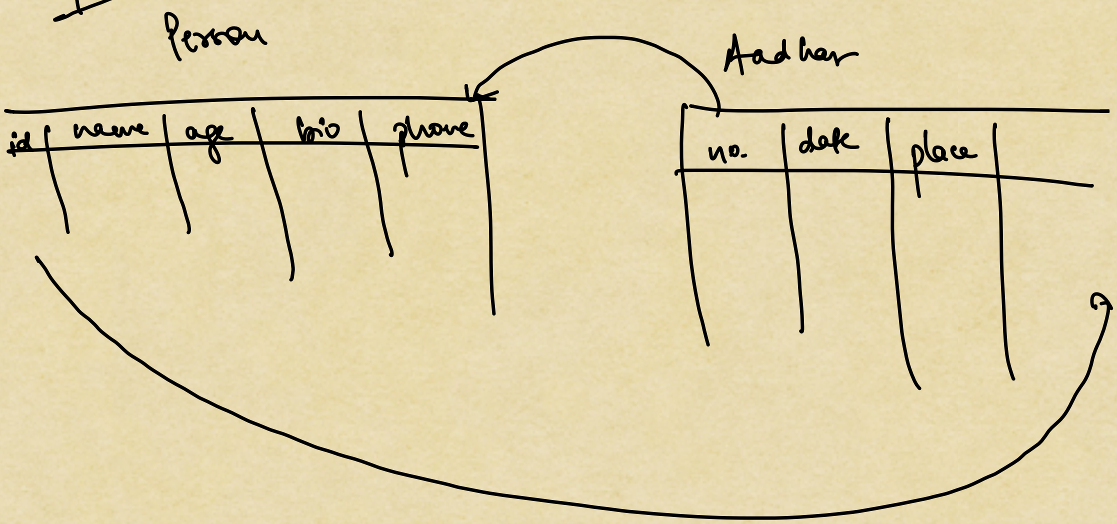
⇒ How to approach schema design:-

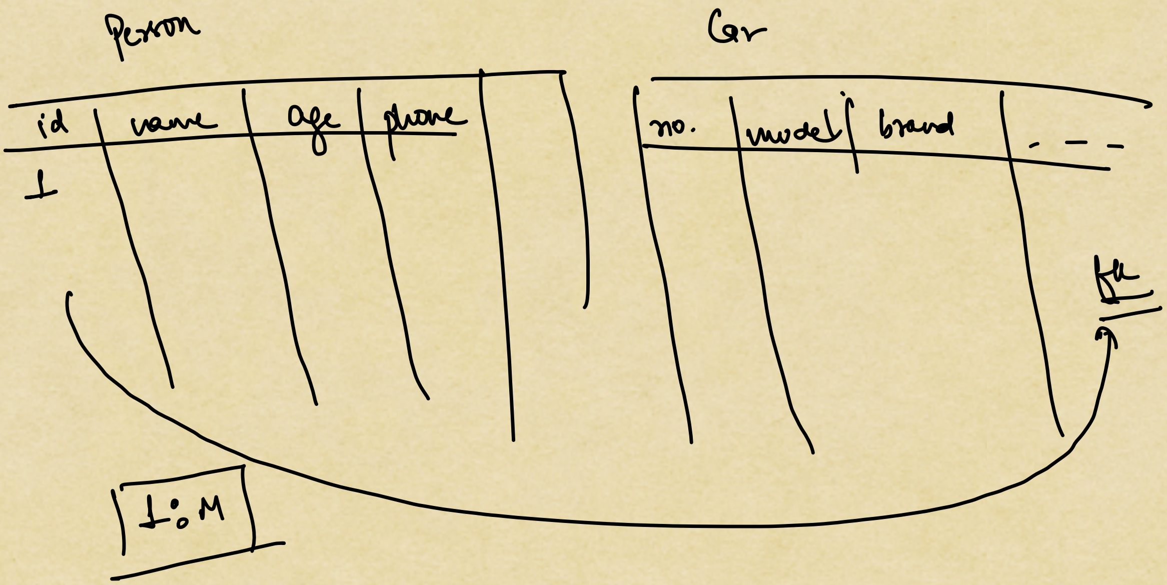
$\boxed{1:1}$ ⇒ id of any side can go as fk on the other side

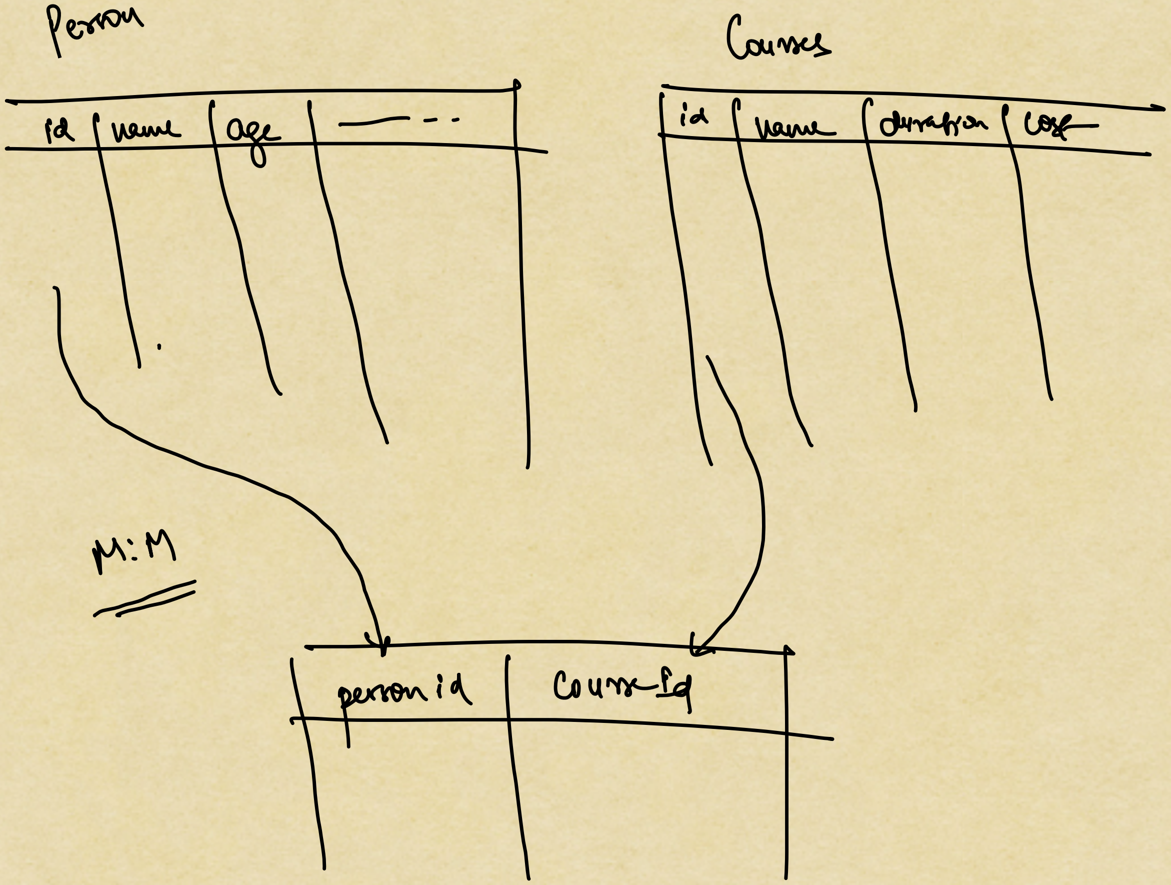
$\boxed{1:M \text{ or, } M:1}$ ⇒ id of 1 side goes as fk on the other side

$\boxed{M:M}$ ⇒ mapping table

1/1




Person
Car
id
name
age
phone
no.
model
brand
- - -
1
fk
1:M


Person
Courses
id
name
age
— - -
id
name
duration
cost
M:M
person id
course_id

→ Schema Design

→ Classes → [tables]

```
class Movie {
        id
        title
        releaseDate;
        desc;
        List<Actor> actors;
}
```

Movie

id	title	releaseDate	desc

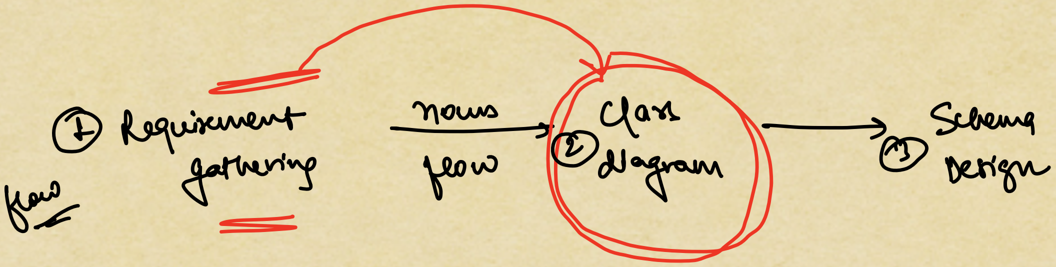
Actor

id	name	age

Movie-Actor

movieId	actorId

```
Actor {
        id
        name
        age
        List<Movie> movies
}
```



⇒ How to code

⇒ layered architecture

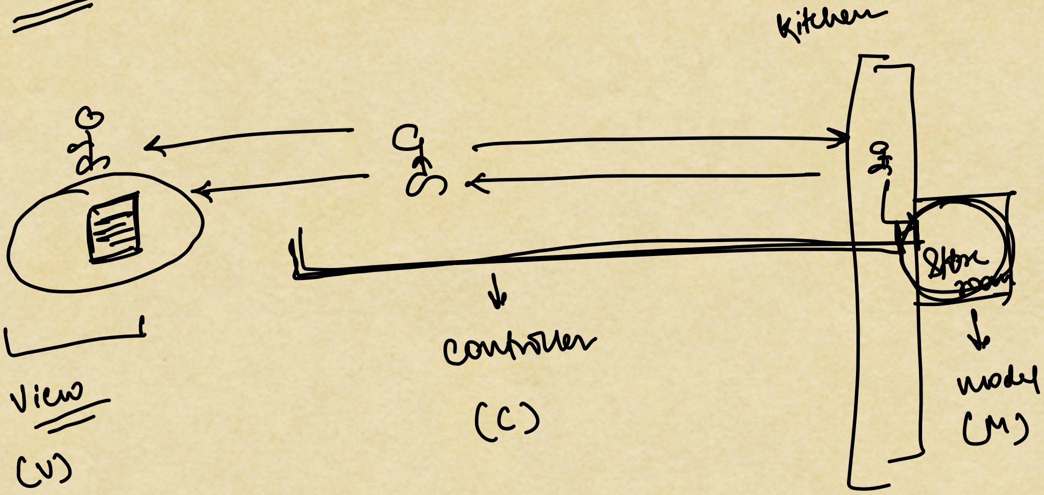
vs

⇒ domain driven architecture

⇒ MVC design pattern :

M → model
V → view
C → controller

Restaurant



↓ layered architecture (BE)

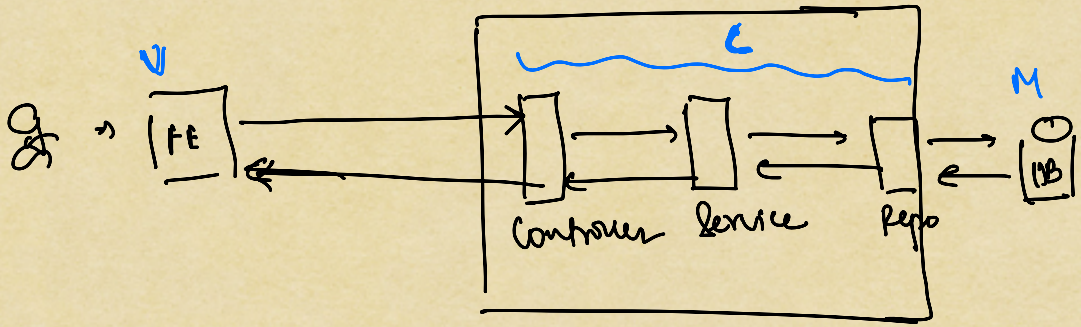
1st → / controller ⇒ code ⇒ incoming reqs & outgoing responses
last ←

/ service ⇒ business logic

/ repository or, DAO (data Access Object)
⇒ talks to database

$\boxed{\text{DB}}$

/models or, /entities ⇒ all our entities will go here



↓
Server box
↓
Java|SpringBoot code

* Layered architecture

⇒ controller → ProductController
→ CartController
→ PaymentController
.
.

⇒ service → ProductService
→ CartService
→ PaymentService - -

Domain

product user payment coupon cart

product
↓
ProductController
" Service
" Repo

payment
↓
PaymentController
" Service
" Repo

cart
CartController
" Service
" Repo

folder ⇒ domain ⇒ product → ProductController
→ ProductService
→ ProductRepo
.
.
.

Trick

BMS

4
- User ⇒ register & login
- Admin ⇒ add movies, theatres
- user ⇒ book tickets
- user ⇒ see booking history

User ⇒ customer & admin

Movie

Theatre

Ticket

↓

models

⇒ layered architecture :)

⇒ controller ⇒ all controller

⇒ service ⇒ all service

⇒ repo ⇒ all repo.

↓

functioning

tip ⇒ code feature by feature

⇒ controller	UserController MovieController
service	UserService MovieService
repo	UserRepo MovieRepo.
model	User Movie

↓

entity

DESIGN TICTACTOE

X	x	X
	O	X
O	O	O

⇒ Persist ⇒ NO

DB ⇒ NO

⇒ <u>Requirement gathering</u>

i) Size of the board ⇒ User input ⇒ N ⇒ N X N

ii) No. of player ⇒ N-1

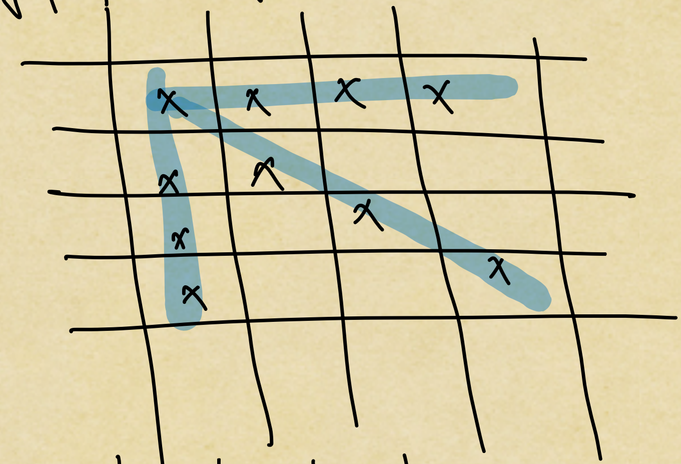
iii) Every player should have a unique symbol

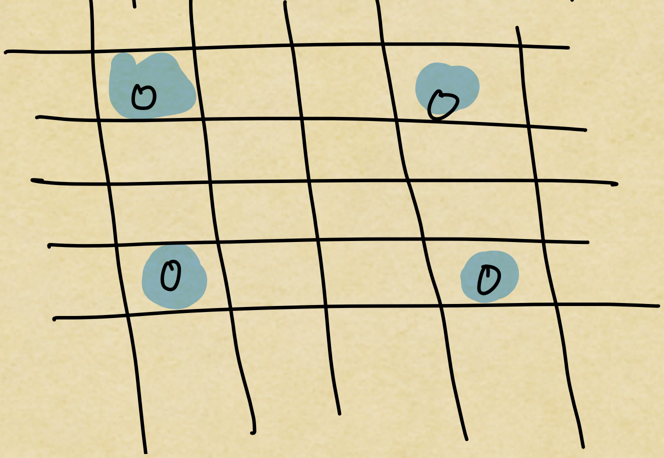
iv) Will there be bots ⇒ Yes

v) maxm no. of bots ⇒ 1

vi) Winning strategy ⇒

- Same symbols in a row or column or diagonal
- any player can get his/her symbol in all 4 corner





viii) who will start, and playing order :)

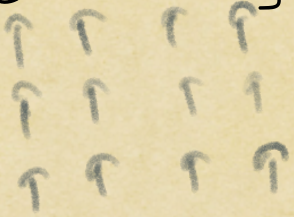
↓ random

↓ randomise the list and follow it

[A B C D] ⇒ randomise

↓

[C A D B]



⇒ when game ends ?

→ If someone wins

→ no empty cells [entire board is full]

* we can watch replay of the game !

(once the game finishes)

* we cant pause the game and nobody can leave the game

* Players can undo their move

* implement the check for winner

=> TC => O(1)

SC => anything

X	X	X

10 mins

[undo]

TicTacToe → command line