Today's content

1. Cardinality revision
2. How to approach schema design
3. MVC
4. How to code
5. Design tic-tac-toe

Cardinalities revision

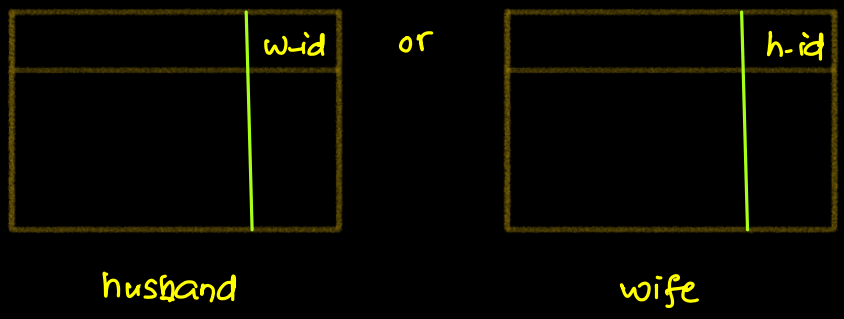
1 : 1

1 : m

m : 1

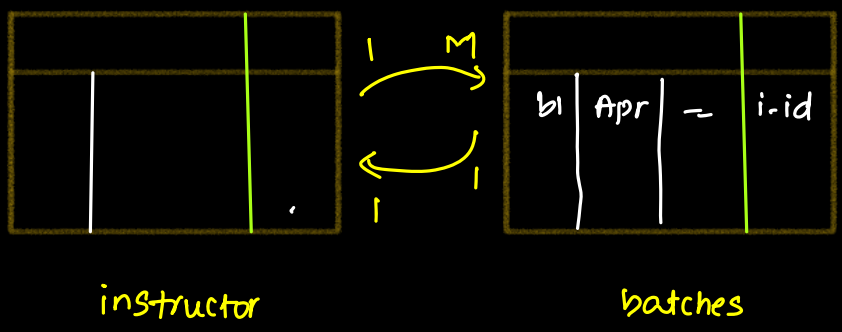
m : m

1 : 1 : How to represent?

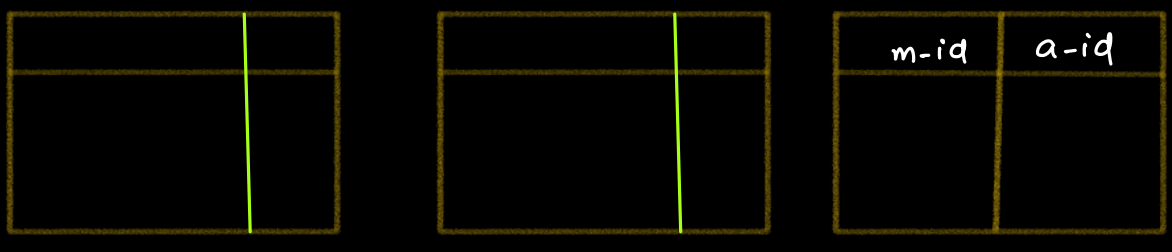


1 : m . How to represent?

m : 1



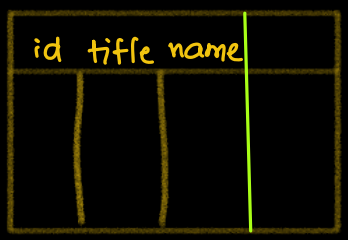
m : m How to represent?



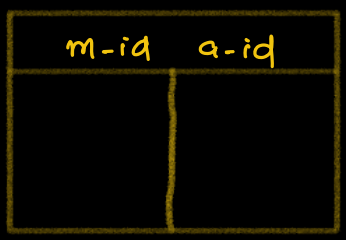
2. How to approach schema design

1. For each class representing an entity in the class diagram, you create table.
2. For every primitive type, becomes column in the table
3. Non-primitive types
    1. Find the cardinality rel^n.
    2. Represent that cardinality.

```
class Movie
    id
    title
    name
    List<Actor>
```



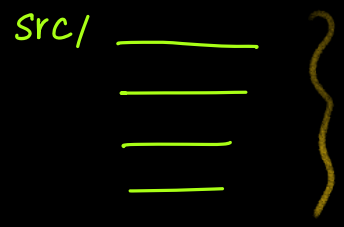
Movie



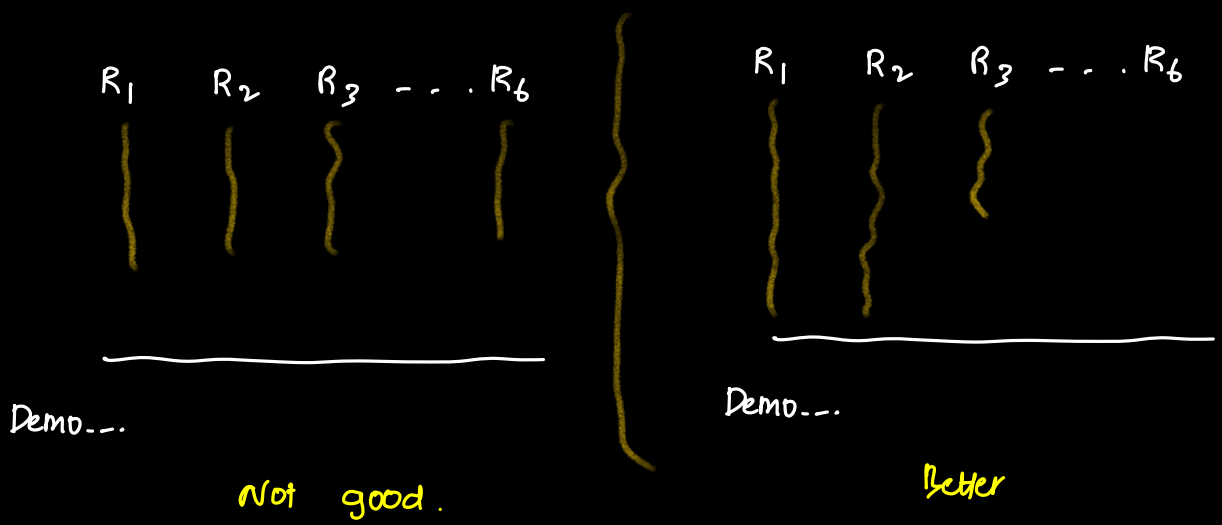
movie-Actor

How to code

1. Project structure : Your codebase should be structured as per industry standards.



2. Go feature by feature.



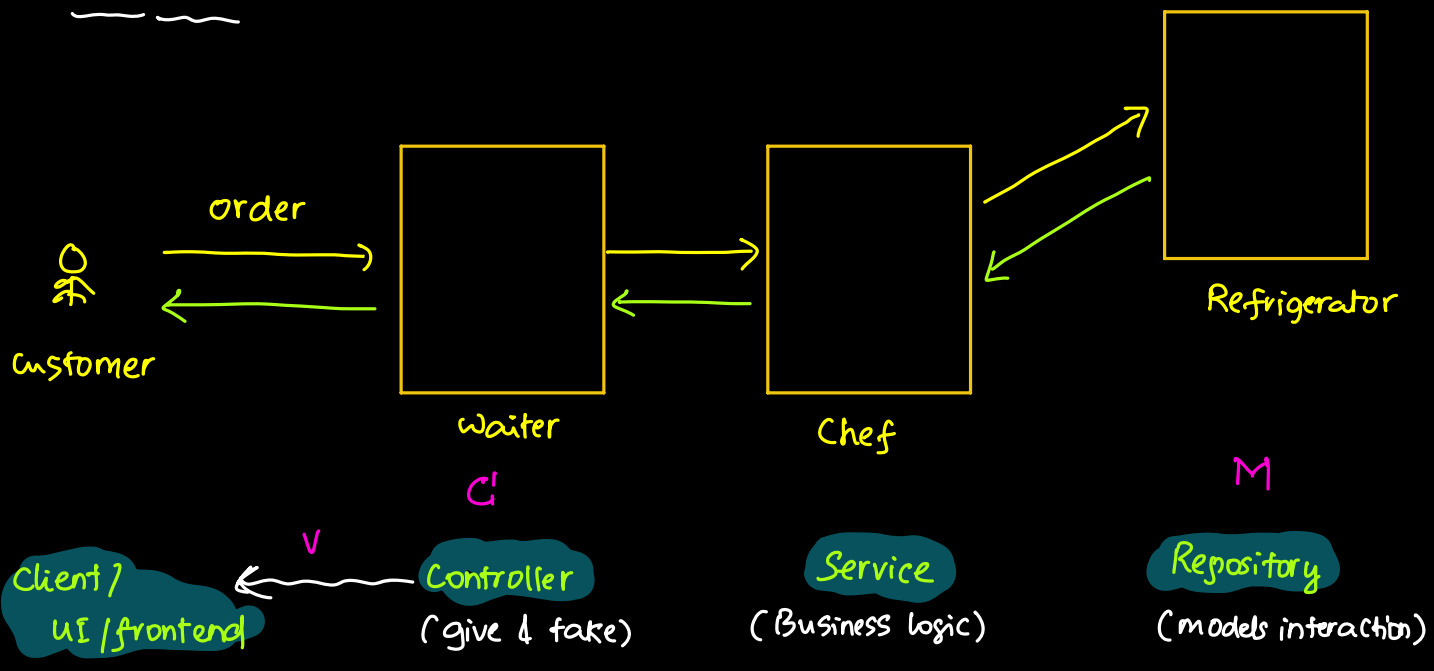
Why? So that interviewer goes back with an impression that given enough time, you're able to code all the features.

MVC?

M - Model
V - View
C - Controller

} way to segregate the responsibilities across different things classes.

Restaurant



How data is returned to the client.
↓
presented

→ View (html) {JSON/ XML}

These days view is not coded in backend

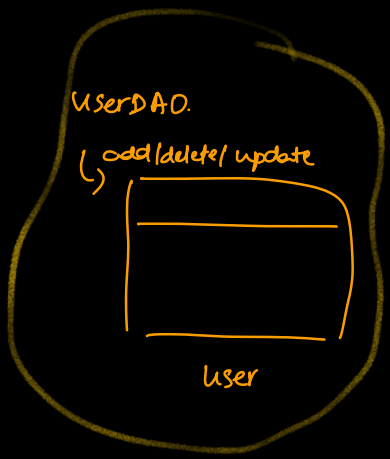
There's no "service" in MVC?

earlier Controller → Repository

With time it made sense to have another layer "Service".

Code structure

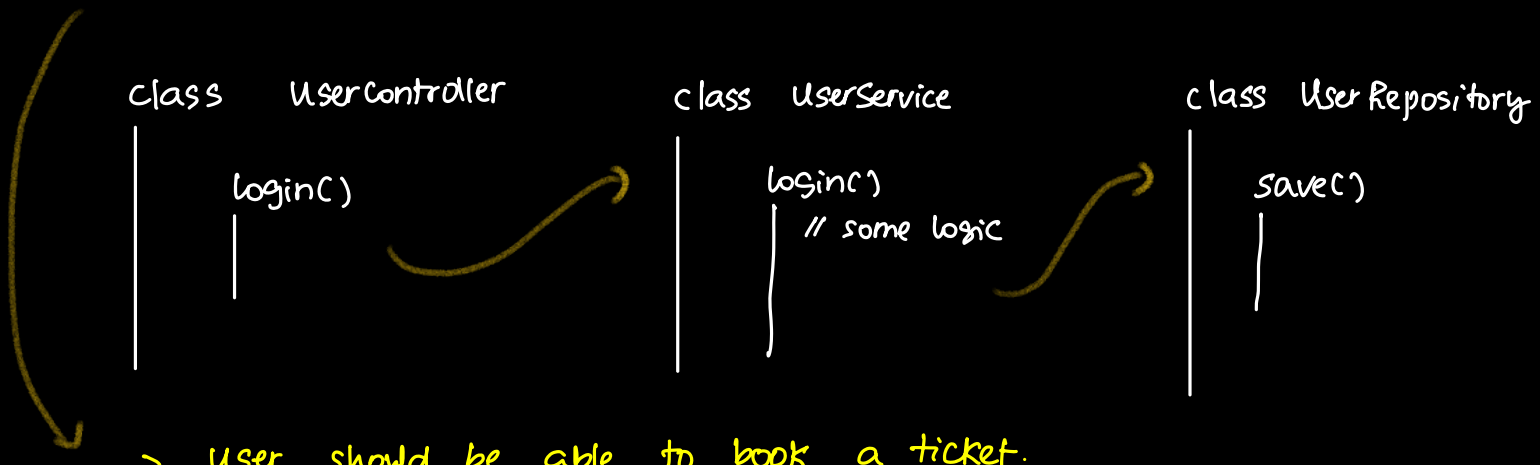
```
src/
    models/
        User
        Show
        Movie
    Controller/
        UserController
        ShowController
    Services/
        UserService
        ShowService
    repositories/
        UserRepository
        ShowRepository
```



Take on the requirements one by one - -

1. User should be able to login.

→ Create all the models in your class diagram. => login()

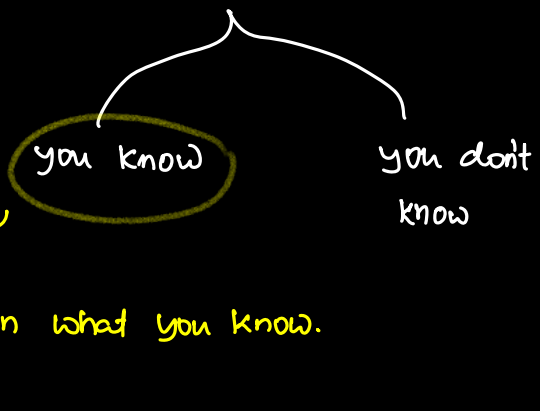


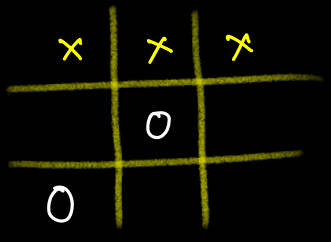
-> User should be able to book a ticket.

// Even if there's no DB connection, it's good to create this structure.

Design Tic Tac Toe

Overview





Yes -- Yes - -

1. Just entities or entire working app?

   Yes, entire working app.

   Data → No persist. [No DB].

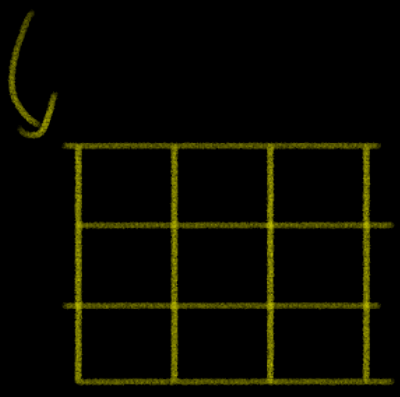
   Interaction → Via command line.

2. Requirement gathering.

   Suggest features / ideas / ask about corner cases (clarifications)

1. Is the board size fixed?

NO__ NXN.



2. No. of players?

NO ___ (N-1) players to play.

3. Symbols for each player?

Can we allow any character to be the symbol? Yes. } Builder (validations)

ASK: Validate symbols.

4. Do we play with humans? bots are there?

Yes!__ There can be bots__

difficulty level.

5. Yes!__

Easy, Medium, Tough } Strategy Pattern

6. How many bots can be there?

Just 1 bot, others are all human.

7. Timers?__

NO!___

8. first move?__

Random order for list of players.

[2 4 1 3]

How game starts? Prerequisites before starting.

How game ends? < first player reaches dest. ✓

all player reaching dest.

9. How is winner decided

list the rules, allow user to select at the start.

ex: only via row
only via col
only via dig
} strategy

10. Game ends when we've 1 winner.

11. exit in between? => No.

Common feature for game design.

1. leader board
2. Tournaments
3. Rewards
4. Pause/Resume/Timer
5. Exit in blw → What to do?
6. UNDO.
7. Replay of the game.

UNDO

↳ Timer? -- No -- for now.

1. Any player can press undo, any no. of times.
2. It'll rollback to previous state.

Global undo

No matter who's pressing, we'll revert the last movement.

Time to spend - - -

3 minutes on overview.

max 10 minutes on requirements.
[5-8 min].

HW.

1. Draw class diagram of Tic-Tac-Toe
2. Check if someone has won or not in O(1) time

for victory ? _ _

BF : $O(n^2)$

SOS : $O(n)$, SC : $O(n)$.

OS : $O(1)$,

Sub. optimal soln → optimal soln.

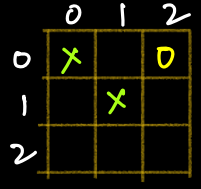
UNDO feature [valid for all games].

3 ways

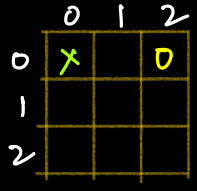
1. Kuch Kuch hota hai → [some things happen]
2. Om Shanthi Om → [Takes re-birth to marry the actress]
3. Doraemon → [Go back in her life using time machine].

① List<Move> l = {[0,0,X], [0,2,O], [1,1,X]}

undo

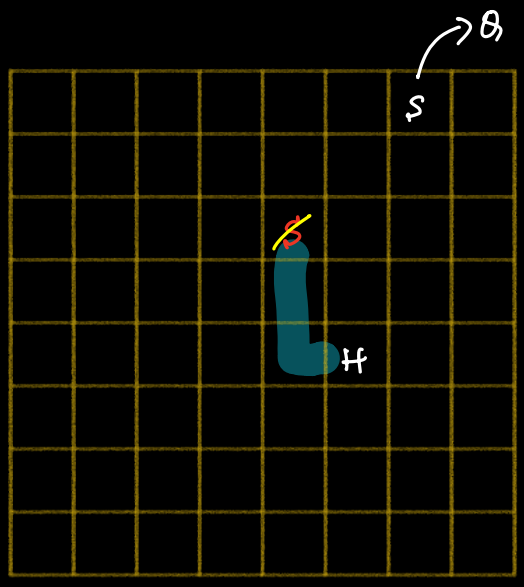


List<Move> l = {[0,0,X], [0,2,O], [ ]}



Steps:
1. Stores moves in a list
2. Remove last move from list.
3. Empty the cell in that move from Board.

Can we do this in Check?

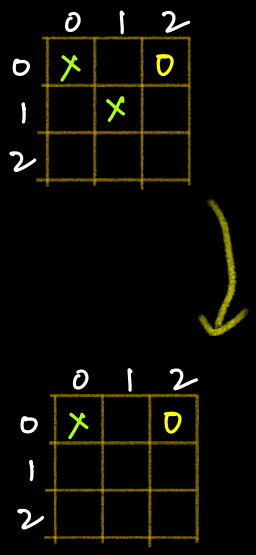


H kills S -> takes its position.
Soldier reaches opponents base and becomes queen.

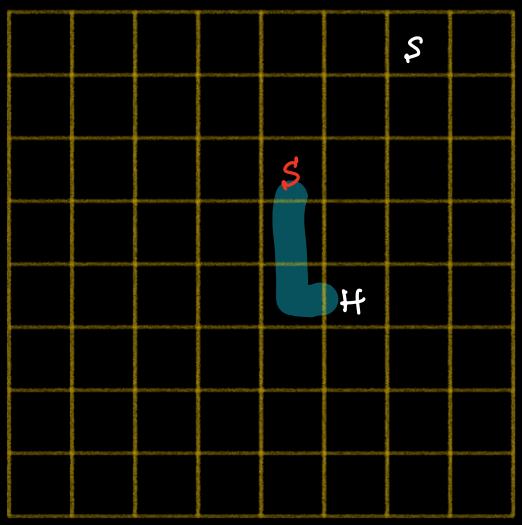
Undo
=> Not just pushing H back to its position

(2)

list<Move> l = {[0,0,x], [0,2,0], [1,1,x]}



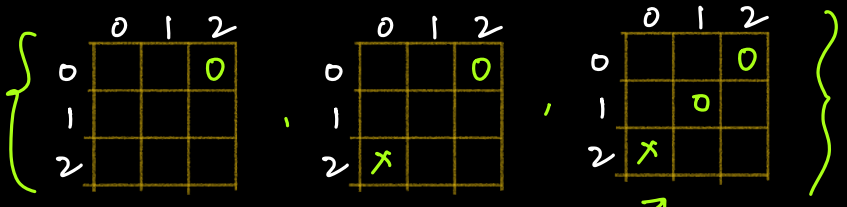
Om Shanti Om => Clearing it and re doing all the moves from start.



m1 , m2 , m3 , m4 , m5

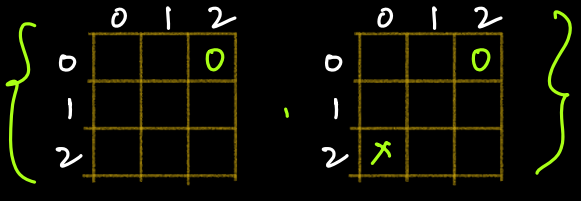
1. To undo m5.
2. Clear the board
3. Re-do all the steps from [m1 → m4].

③. Store entire board.

List<Board> l = {  }

Board state

undo.

List<Board> l = {  }

Board state

1. Easy to impl.
2. less TC, SC
3. May not work all games

Kuch Kuch hota hai

1. More TC
2. Works for all games

Om Shanthi Om

1. More SC
2. Works for all games

Doraemon

interface