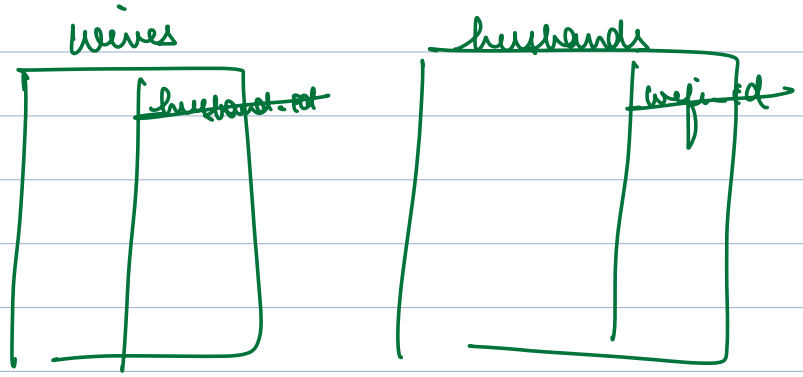
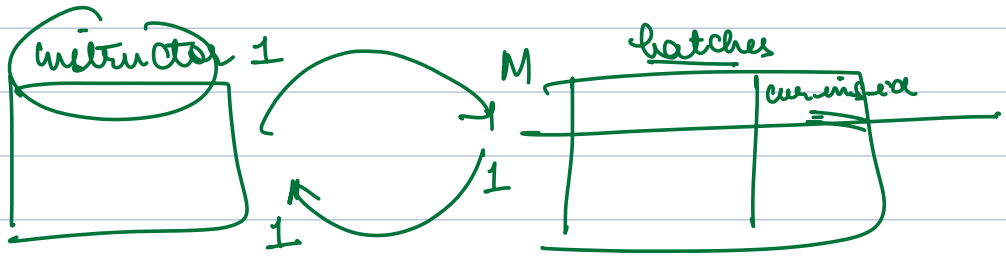
Agenda

1. How to approach Schema Design
2. How to code
3. Design Tic Tac Toe
   - Overview
   - Req
   - class Diagram
   - Surprise? → A interesting topic

HOW TO APPROACH SCHEMA DESIGN

- 1:1 ⇒ id of one side on other side.
- 1:m or m:1 ⇒ id of 1 side on m side
- m:m ⇒ mapping tables





1 : M

Steps to approach schema design

(1) For each class in class diagram create a table.

Rep an entity

class Movie
id
title
name
⇒ list <Actors>

actors
id | name | dob

movies
id | title | name

Schema Design

movie-actors
movie_id | actor_id

1 M
M A
M 1

(2) For each primitive attribute in each class, put that as a column in the corresponding table. (non object) (non association)

int / boolean / string

(3) For every non primitive attr (rel^n with another class)

a-) find the cardinality of rel^n

b) Rep treat rel^n as per rules to

up that cardinality.

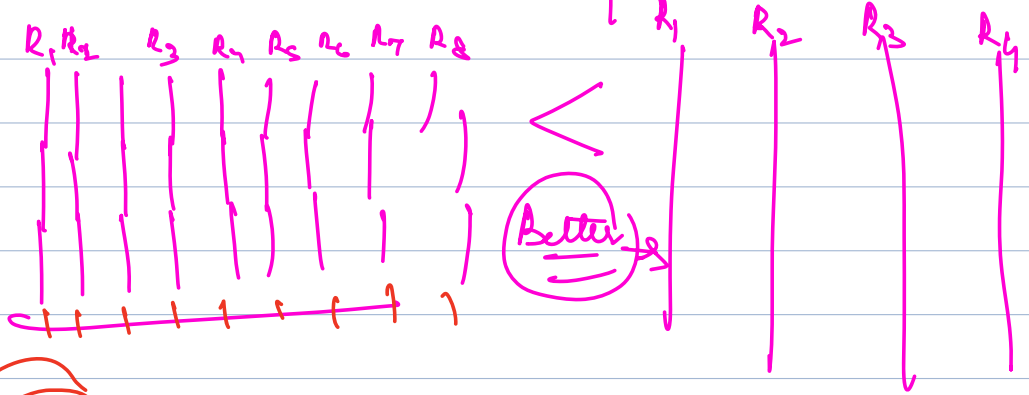
HOW TO CODE ?

→ (1) Project Structure (your codebase should be structured as per best practices in the industry)

MVC Architecture →

src/

(2) Atleast some of the req must be running end to end.



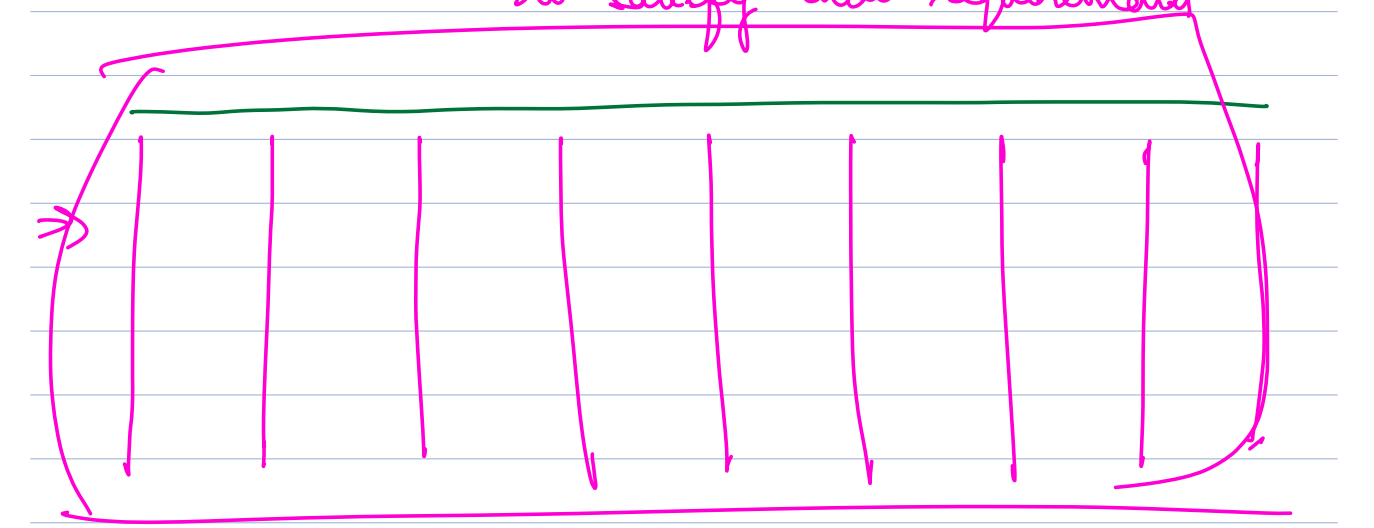
How?

→ (1) Code all the models / Classes in the

Class diagram

② Go req by req (in any order)

→ imp each class / interface needed to satisfy that requirement



① Class Movie { Class Show { Class Actor {

}

② User should be able to login.

Class User Controller { Class User Service {

login() { login() {

}

}

class UserApp {

saveC() {

}

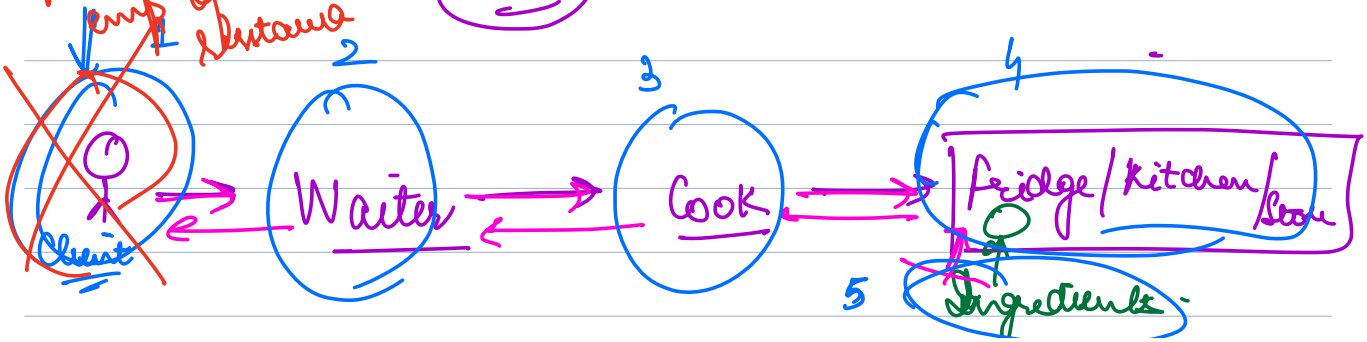
② User should be able to book ticket.

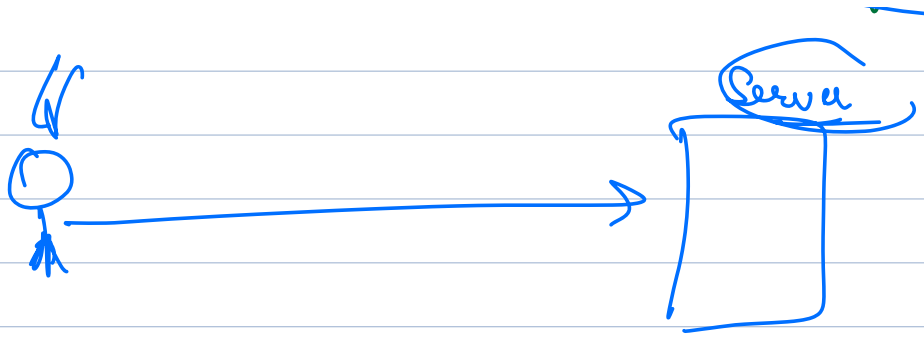
Time is up

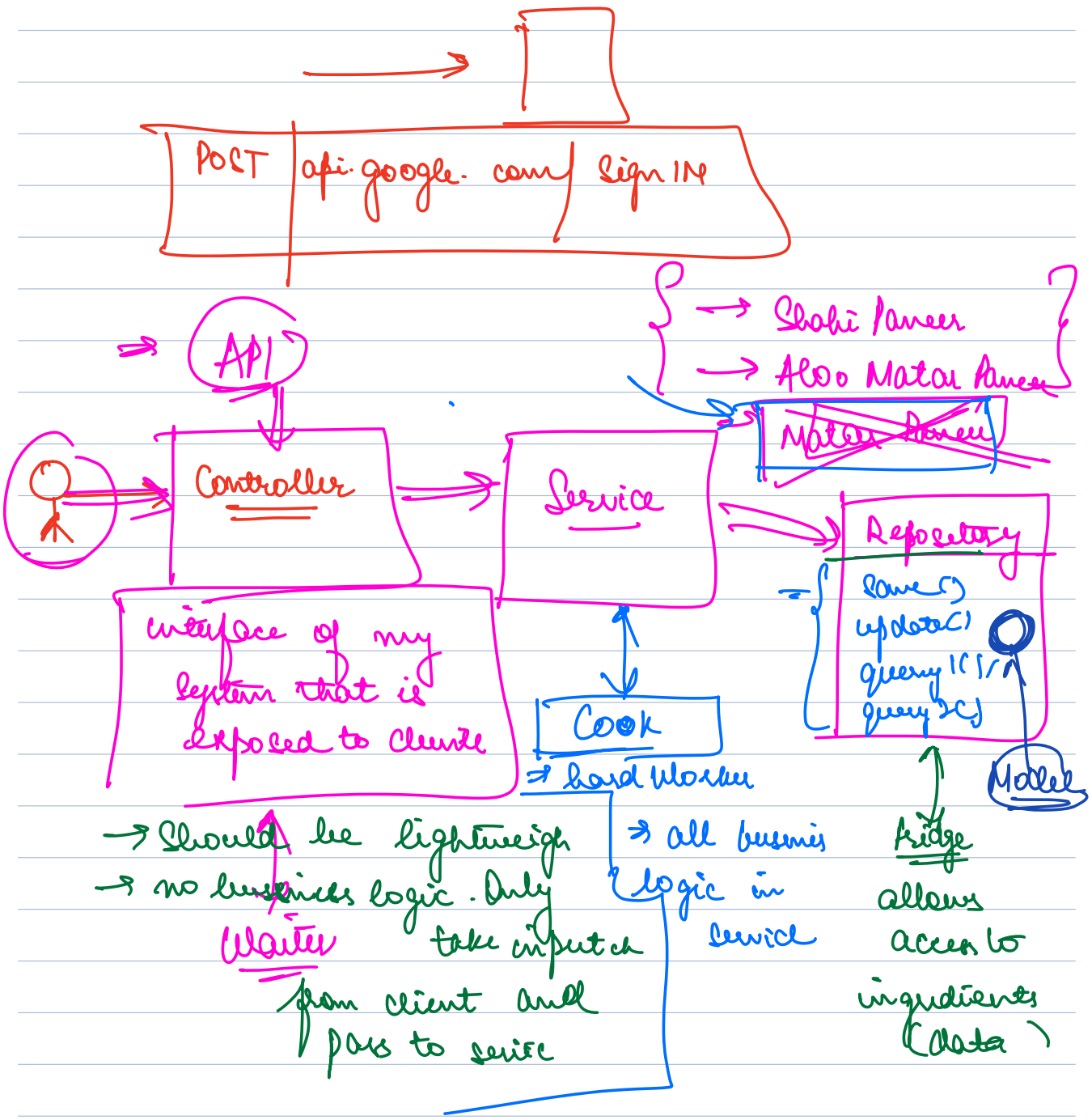
Why? So that the interviewer goes back with the impression that given more time interviewee will be able to code.

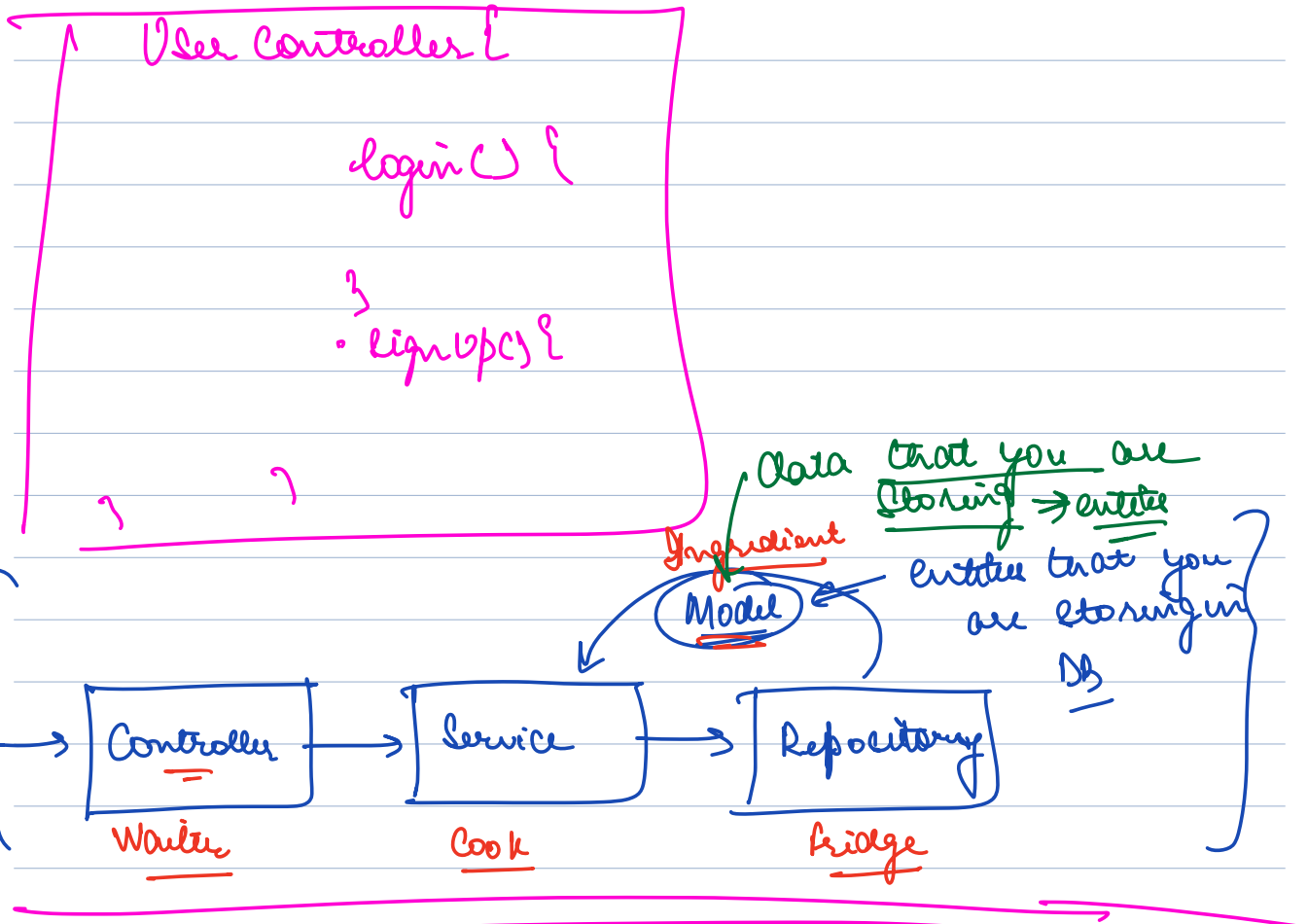
What are models?
How to structure codebase? → MVC

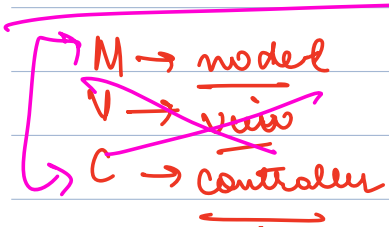
→ Segregate responsibilities into multiple classes → SRP

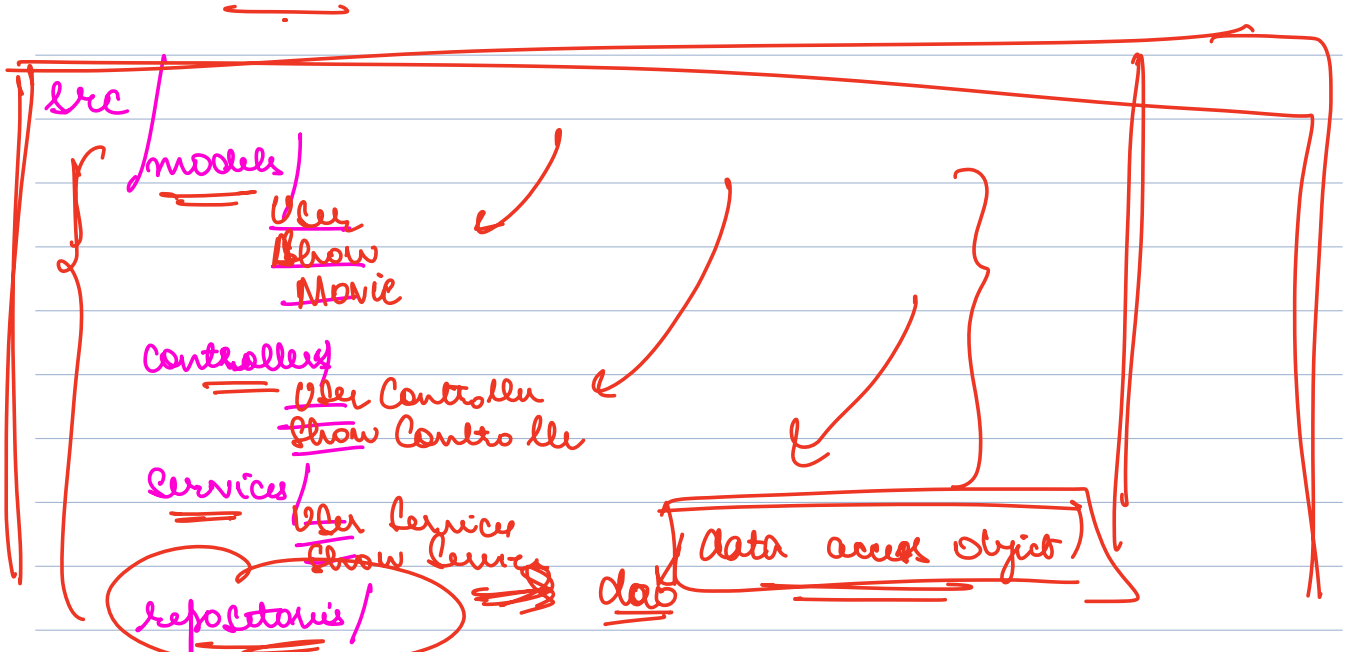


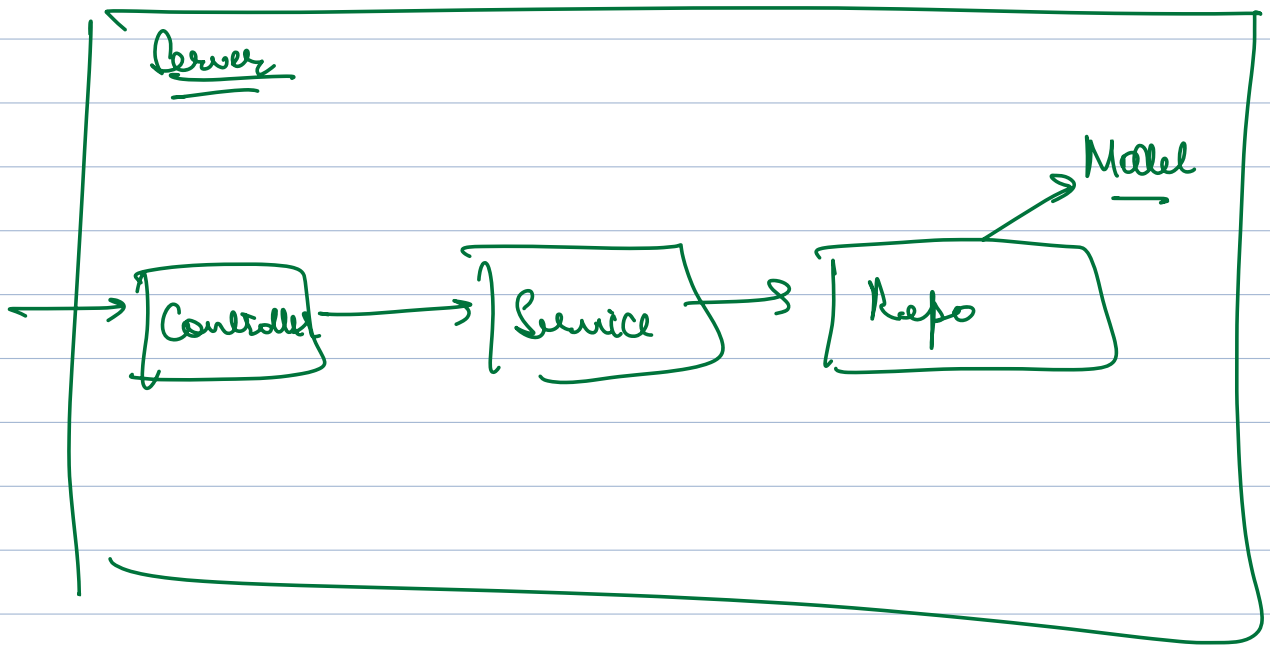

Server

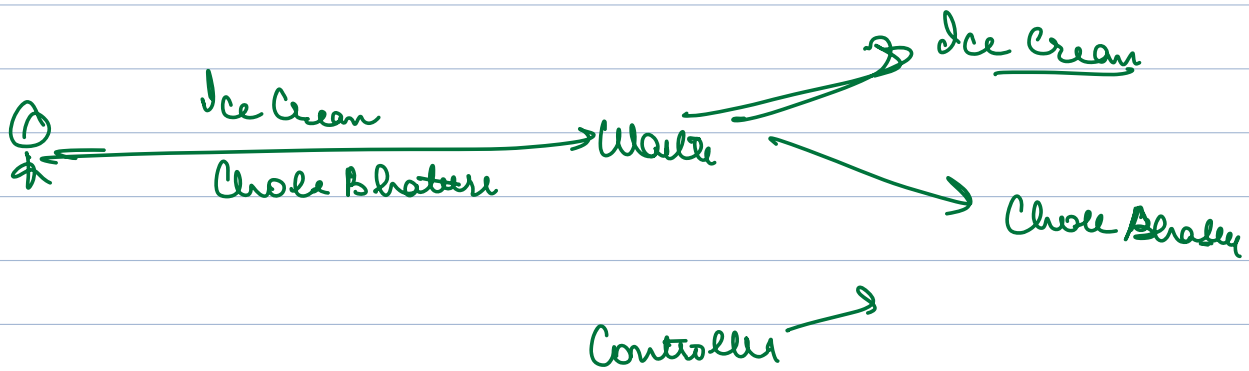

POST
api.google.com/signIN
API
Controller
Service
Repository
Shahi Paneer
Aloo Matar Paneer
Matar Paneer
interface of my system that is exposed to client
Cook
Model
→ Should be lightweight
→ no business logic. Only take input from client and pass to service
Waiter
→ all business logic in service
Bridge
allows access to ingredients (data)


User Controller {
login() {
}
signup() {
}
}
Data that you are storing → entities
Ingredient
Model
entities that you are storing in DB
Controller
Service
Repository
Waiter
Cook
Fridge


M → model
V → view
C → controller


src
models/
User
Show
Movie
controllers/
User Controller
Show Controller
services/
User Service
Show Service
repositories/
db
data access object





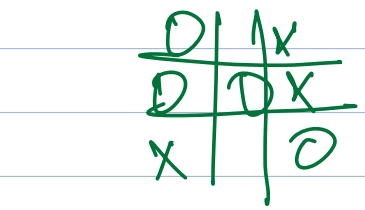
In MC round, do we need to connect to a real DB or can I store data in memory?

Ans → Depends

- → Ask about this to your interviewer while getting overview of problem statement.
- → even if you can store in mem, you have to follow code structure → Repository: Save in a list

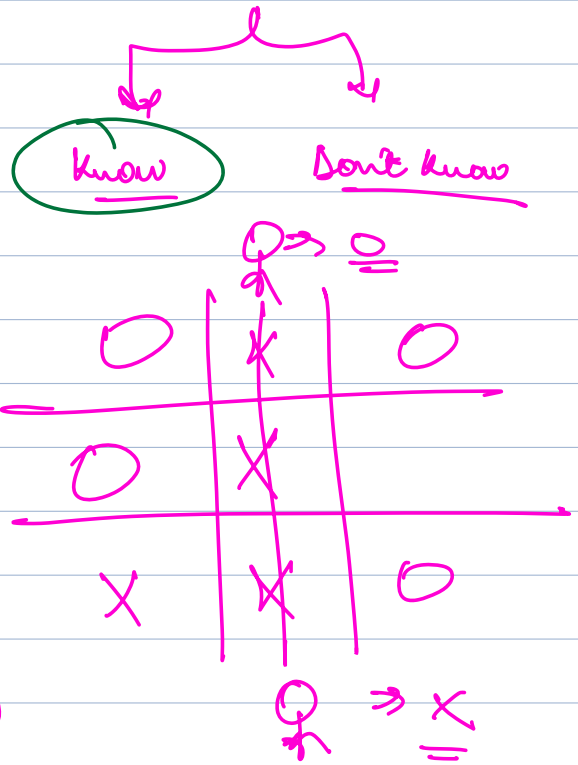
Design TicTacToe

Overview



Real World ↝
Person — X
Command Line ↝

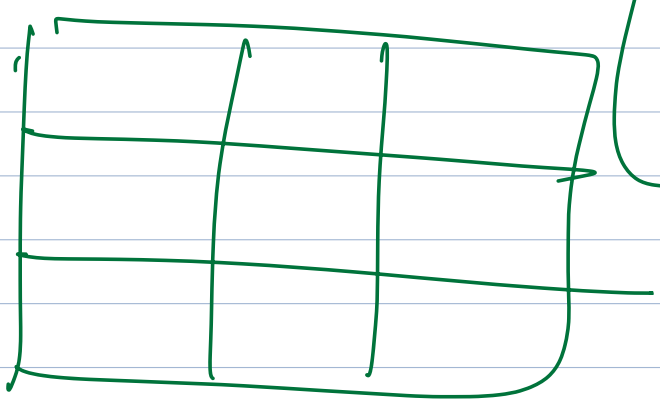
5-8 mins
(10 mins)



Requirement Gathering

→ Suggest features / ideas
or Corner cases (Clarification)

features that we will come up with today, most of them are also valid for other games
→ Chess, Snake & Ladders, Card game



(1) Size of board can be any [n x n]

(2) # of players = (n-1)

(3) Every player can choose their symbol at start of game { We shouldn't allow 2 players with same symbol. } Validation is Builder

(4) Will there be bots ⇒ (Yes) (Computer player)

(5) Bots can have diff difficulty

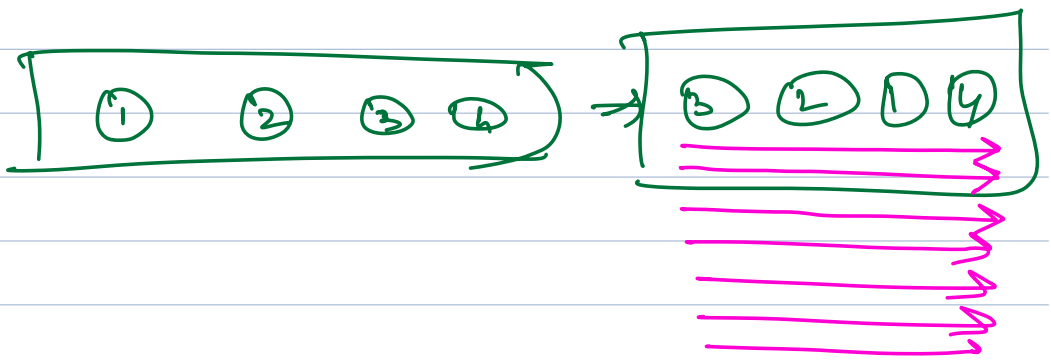
(6) Only 1 Bot per game.?

(7) Who will make first move ?

→ How game starts

→ How game ends.

→ at the start of game we randomize the list of players. And they play in the order in list



(8) How is winner decided ?

→ We want to keep our game extensible. User at start of game can list what all rules to apply to decide a

winner.

1. → All colⁿ of 1 row having same symbol
2. → All corners have symbol
3. → All cells of 1 col having same symbol

(1) (2)

↓ Strategy

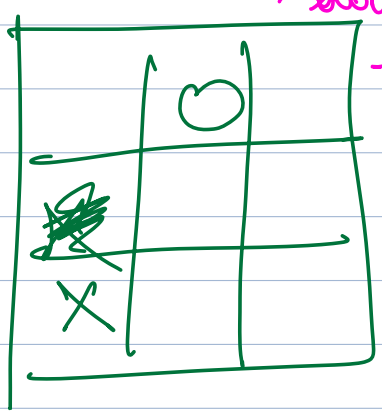
9. Game ends as soon one winner or a draw.

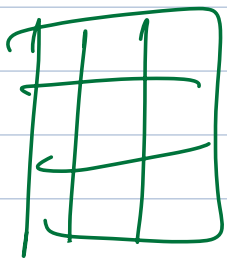
10. No one can exit in bfw

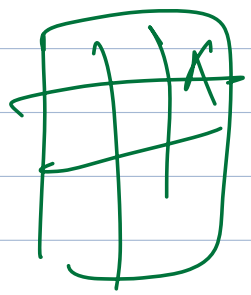
11. Undo a move
   - ↳ Global undo button.
   - ↳ Anyone can press anytt of times
   - ↳ Undoes whatever the last move was

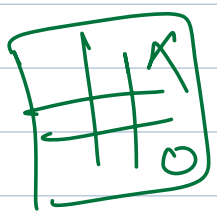
12. Show replay of the game.
   - ↓ doesn't include undone moves

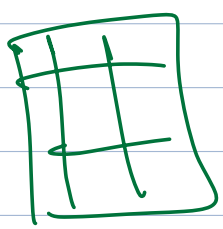
- → Board resize
- → Leaderboard
- → Tournaments
- → Pause / Resume / Timer
- → Exit a game in bfw.
- → Cells are blocked.
- → Undo
- → Replay
- → Show will game start
- → Show will game end -

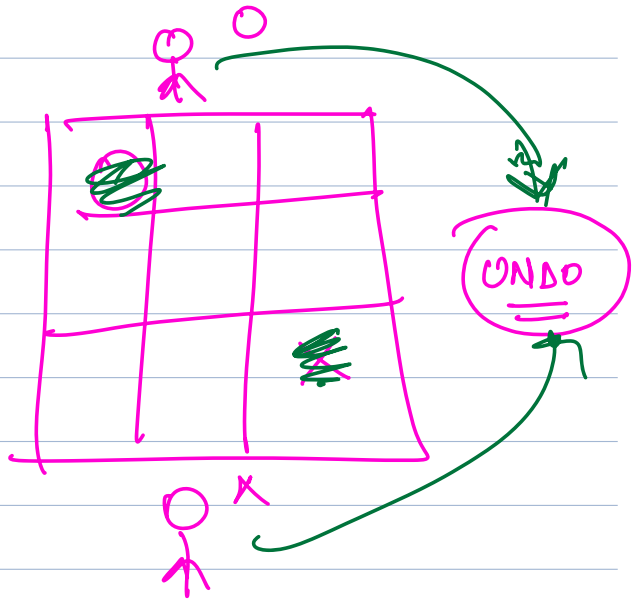












3 Mins: Overview
8-9 mins: Req Gathering

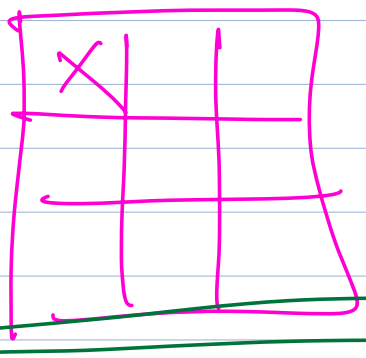
Follow up: Can you implement a check for victory in O(1)

Grid : Tri

	TC	SC
BF	$O(n^2)$	
DFS	$O(n)$	$O(n)$
BS	$O(1)$	

HW

1. Draw class diagram of tic tac toe
2. After a particular move, how could know if someone has won in OC1)

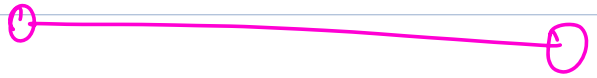


HOW TO IMPLEMENT UNDO FEATURE

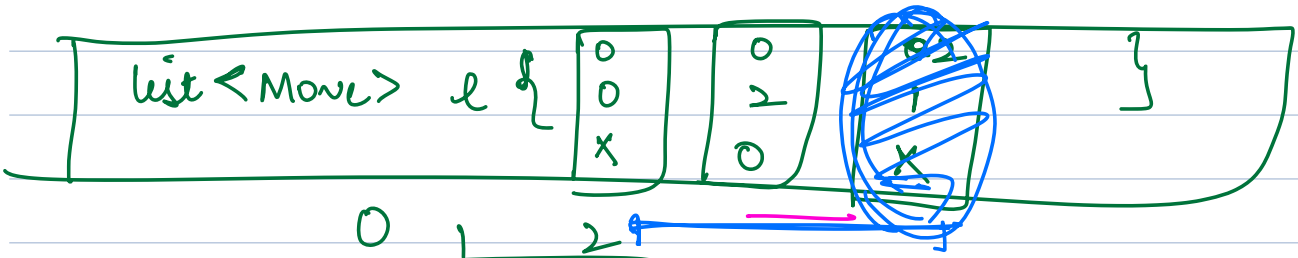
3 ways

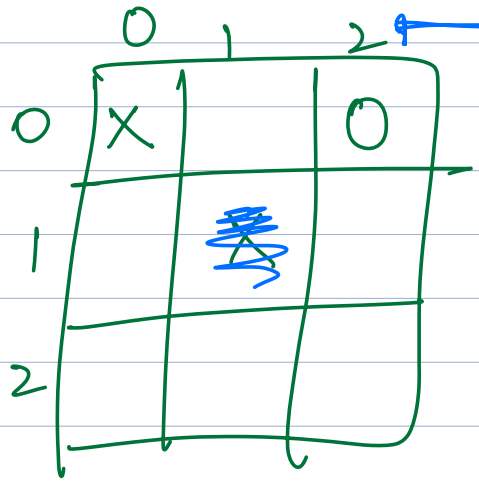
1.) Kuch Kuch Hota Hai
2.) Om Shanti Om
3.) Doraemon

→ You loved someone and break up happened. & you want to marry them.



① you were married you undo your marriage you marry the other person

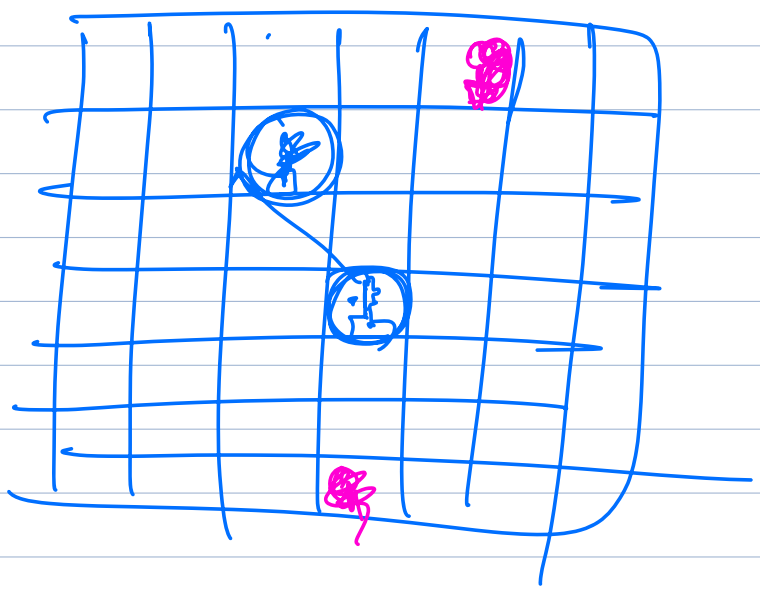




① Store moves in a list

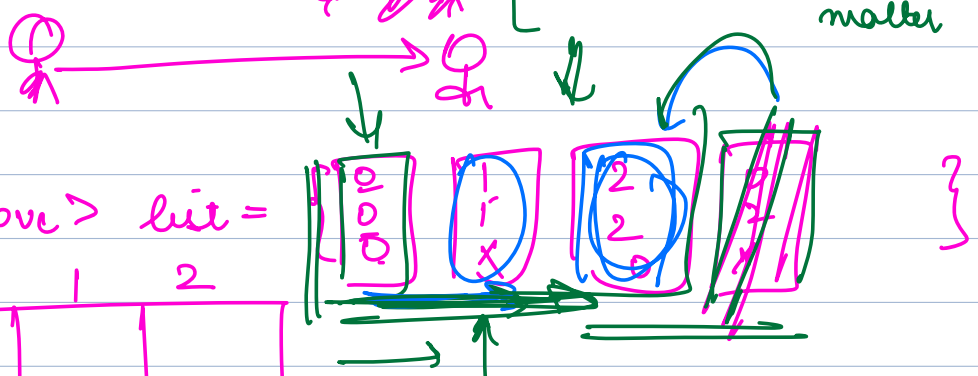
② Remove last move from list

③ Empty the cell in that move → DONE

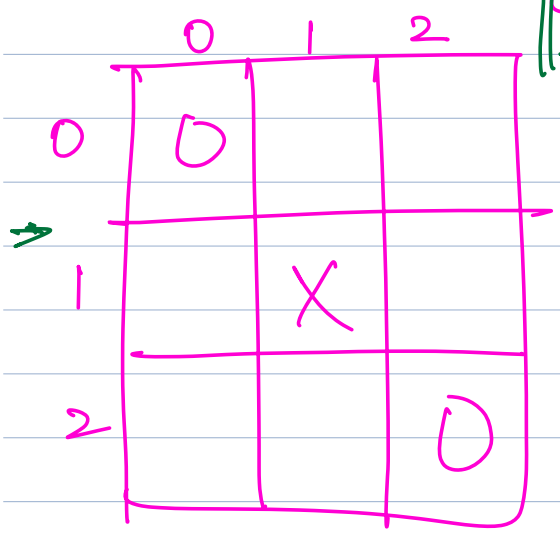


Om Shanti Om

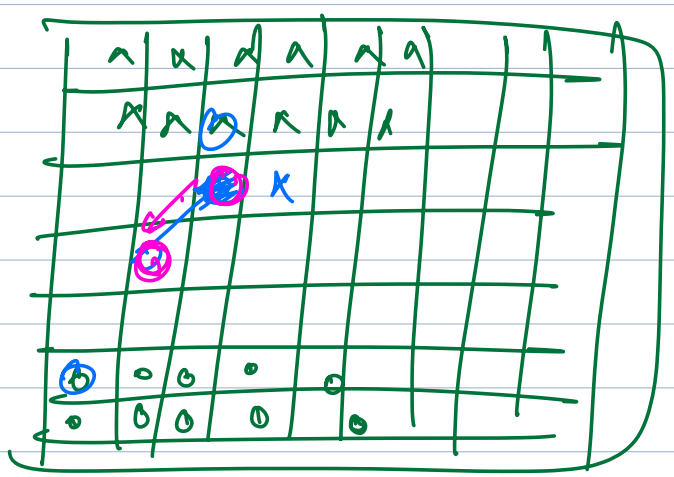
→ Time complexity is higher
→ It doesn't matter

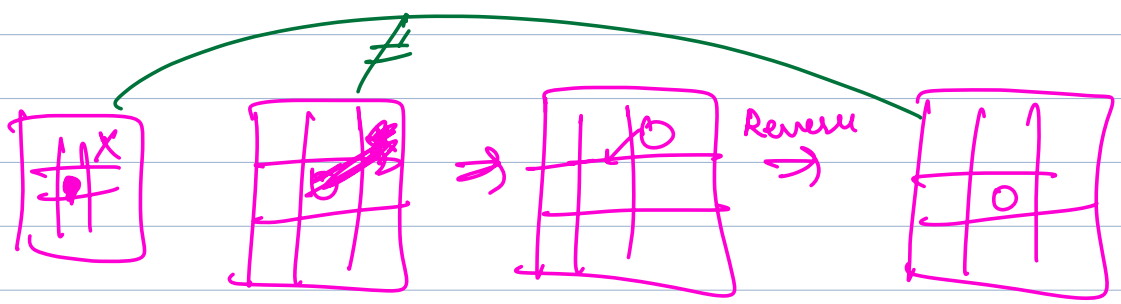


List <Move> list =



① You remove last move from list → Undo
② Clear the board → Redraw
③ Apply all moves to it



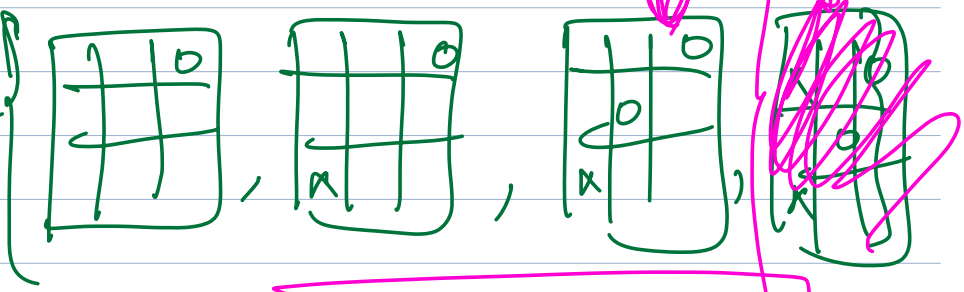


Doraemon

Time Machine

Store the list of state of board after every move

list<board *> =



⇒ good TC

⇒ easy to implement

⇒ bad SC

Summary

Undo Strategy

① If reversing effects of a move is easy
⇒ store list of moves
⇒ remove last move
⇒ reverse its effect

② If reversing not easy

(I) TC doesn't matter ⇒ DSM
⇒ store list of moves
⇒ clear board
⇒ apply all but one move

(II) SC doesn't matter ⇒ Doraemon
⇒ store list of board state
⇒ revert to prev state

12th ⇒ 11th

New UI Ok ——— 11 Ok

GameBuilder {

}