1. Overview.

- entities / complete flow
- input being read [Rest api / cmd line]
- Data persistence [DB].

2. Requirement gathering

- Concurrent bookings should be handled.
- You should support regions, languages
- Each region will have multiple theatres
- only movies can be booked.
- Each theatre will have multiple screens, they'll can play diff movie at the time.
- Each screen will have different shows
- screen / movie / theatre / seat

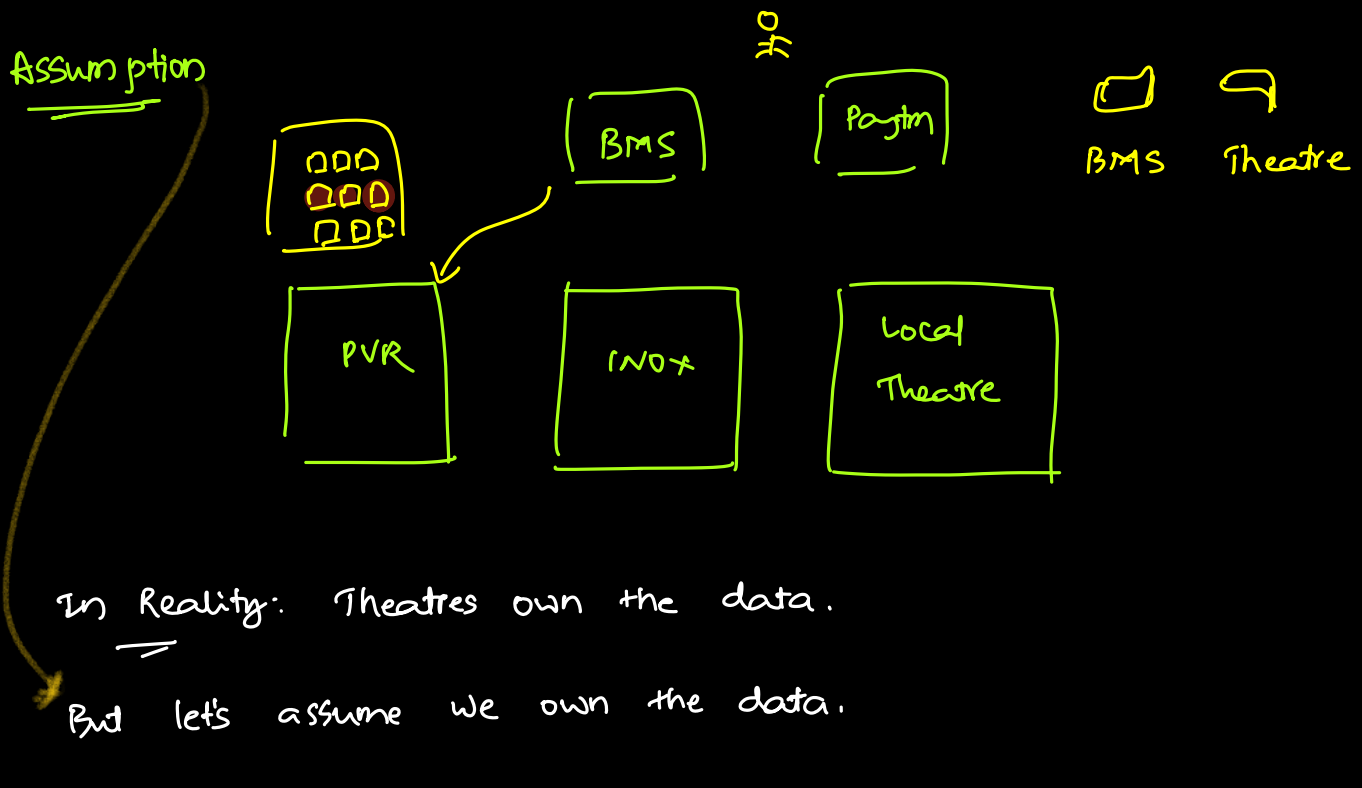
  Show → for a movie + particular audi + particular time
- search fun → X
- In one booking, 10 seats.
- Add ons → X
- Price will be a fun of

  movie, theatre, seat type, time

  show + seat type
- Payments → online

UPI, Netbanking, card etc. [3rd party]

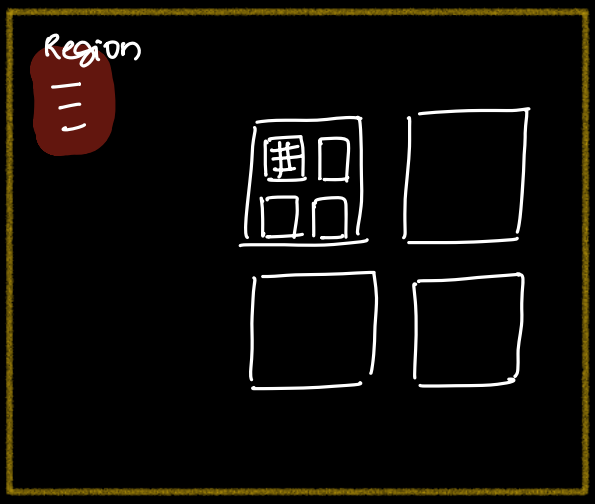
- Partial payment → Yes.
- Wallet → ✗
- Cancellation → ✓
- screen → [2D, 3D, Dolby-audio] features
- movie → [actors, reviews, no of bookings etc].



In Reality: Theatres own the data.

But let's assume we own the data.

class diagram



Region
- id
- name
- List<Theatre>

Theatre
- id
- name
- List<Screen>

Screen
- id
- name
- List<Seat>
- List<Feature>

Feature
3D, 2D,
DOLBY

not an enum ⇒

Seat
- id
- number
- seatType
- row
- col

Show
- id
- startTime
- endTime
- movie
- screen

ShowSeat
- show
- seat
- status
~~- price~~

ShowSeatTypePrice
- show
- seatType
- price

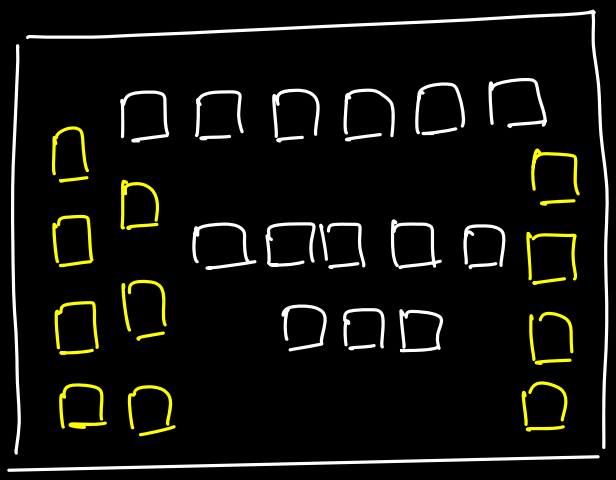
User
- id
- name
- phNo
- List<Booking>

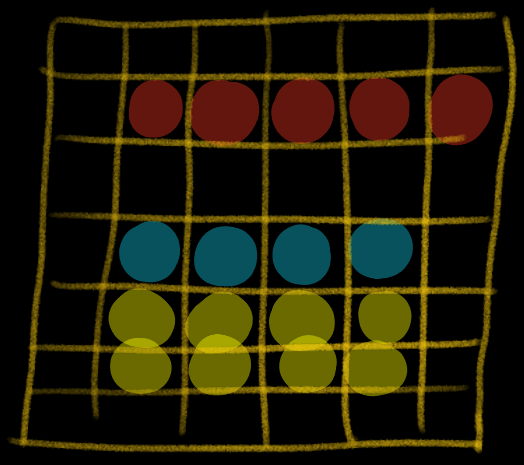
Booking / Ticket
- id
- amount
- List<Payment>
- status
- List<ShowSeat>

Payment
- id
- refNo
- status
- mode
- amount
- pgateway

CANCELLED,
BOOKED.

how do we know the layout of the screen?





rol, col for seat.

Home work

(i) Create schema

(ii) Create models

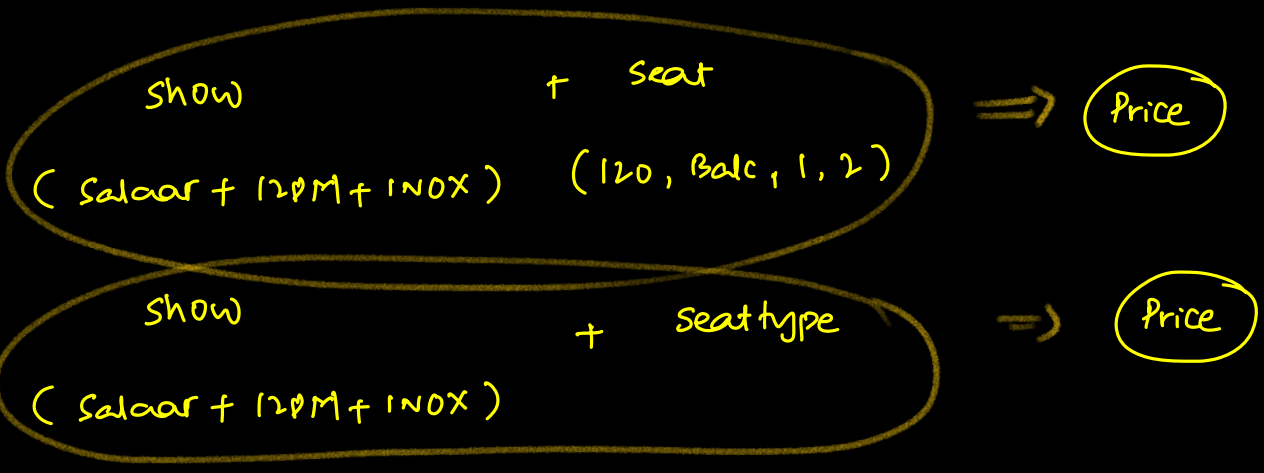
(iii) For one feature => Book a ticket.

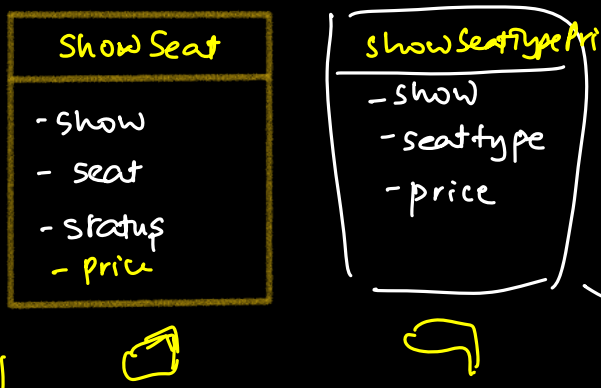
Spring boot + DB interactions

Spring data jpa

=> in Detail in project module

show seat type price discussion.





showId	seatId	
1	1	balcony
1	2	
1	3	
1	4	
1	5	FC
1	6	
1	7	
1	8	

Price

showId	seattype	price
1	balcony	—
1	FC	—