1. Good Evening
2. Lecture begins at 9:05 pm
3. Topic → Design BookMyShow

Agenda

1. Design BookMyShow

Pen → Class Diag

TTT → Interactive App.

PL → MVC

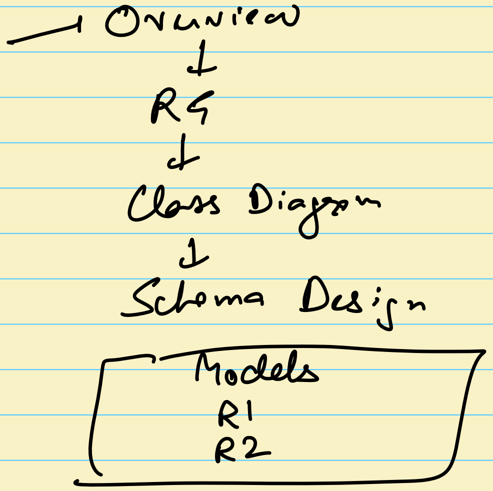
BMS → MVC
SB (DD)
Lombock (9S)
JPA + Hibunat
(DBA)

Splitwise → Api

Overview

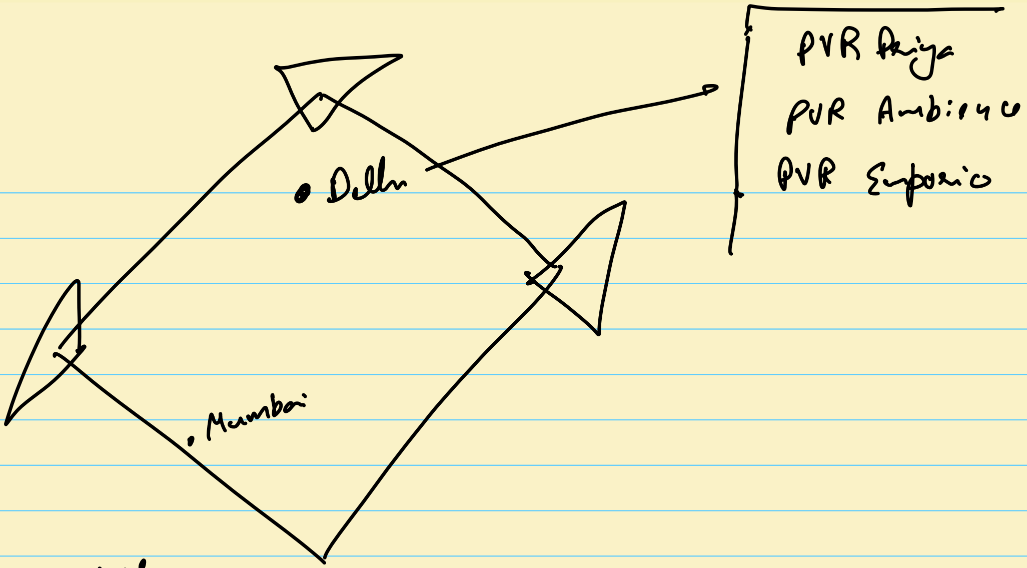
1. Align with interview
2. kind of system → Class Diagram
   → Interactive Apps
   → Backend API

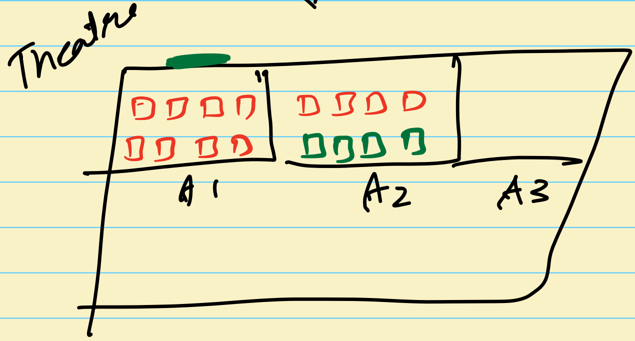
Advise
1. Models
2. Req by Req.



Requirement Gathering

1. Draw a sketch of system
2. User journey ?
3. Behaviours ? →





✓ 1. There are multiple cities. ✓

✓ 2. There are multiple theatres ✓ in a city.

✓ 3. There are multiple auditoriums ✓ in theatre.

✓ 4. There are seats in Audi. ✓

✓ [5. Each seat will have a type → fixed [Silver, Gold, Platinum] → dynamic

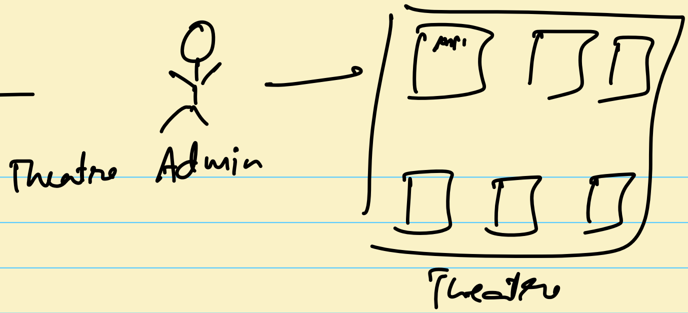
✓ 6. Audi will have multiple shows.

✓ 7. Audi will have features → 2D, 3D, Dolby Atmos

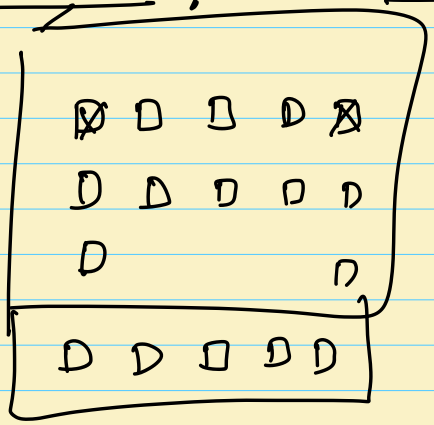
✓ 8. Show will also have features list.

✓ 9. Show will have a start time & end time.

User Journeys



1. Theatre Admin will add shows to Audis.
2. BMS Admin can add cities & theatres.
3. User will search city, select a movie, select a theatre select a show & select seats.



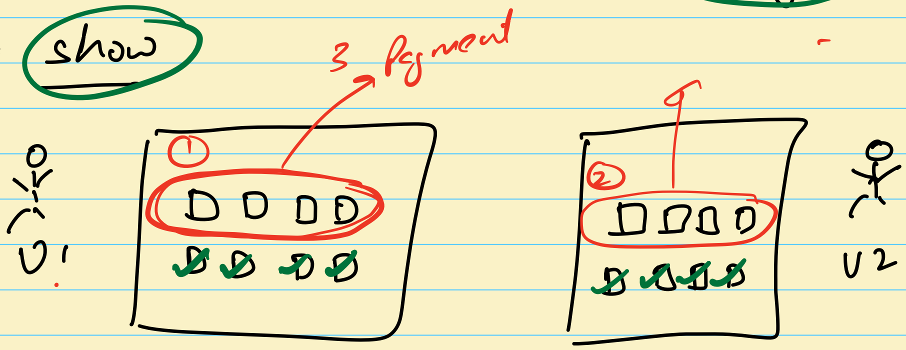
make payment.

gets a ticket.

Behaviours.

1. Price will depend on seat type, & show

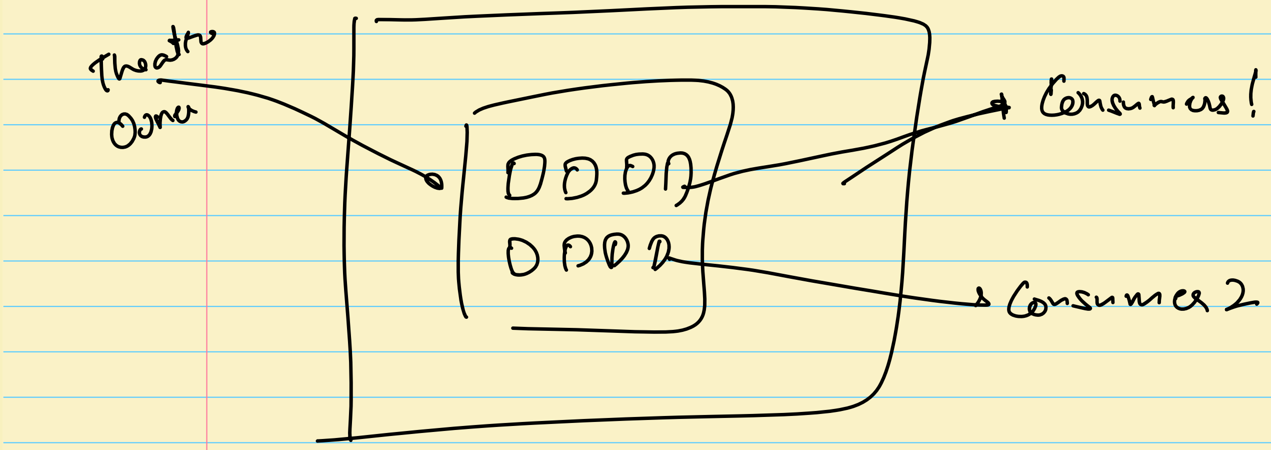
2.



Seats can be locked w/o payment
[Concurrency Handling]

3. Seat status depends on Show + Seat

4. A Ticket can be used to book multiple seats in a single show.



1. Class Diagram

- Entities
- Enums
- Attributes

Break → 10:03 – 10:14

Classes

1. City ✓
2. Theatre ✓
3. Audi ✓
4. Seat ✓
5. Show ✓
6. User ✓
7. Ticket
8. Payment.

Enums

1. Seat Type ★ ✓
2. Seat Status ✓
3. Feature ✓
4. Role ✓
5. PaymentProvider

City ✓

id
name
List<Theatre>

Theatre ✓

id
name
address
List<Audi>

Audi ✓

id
name
List<Seat>
List<Show>
List<Feature>

Seat ✓

id
number
Seat Type
row
col

Show ✓

id
startTime ✓
endTime ✓
movie
List<Feature>

Movie ✓

id
name
descr

e Seat Type ✓

- SILVER
- GOLD
- PLATINUM

e Feature ✓

- 2d
- 3d
- 4d
- Dolby

User ✓

- id
- name
- password
- List<Role> ✓

e Role ✓

- Customer
- TheatreAdmin
- Admin

Ticket ✓

- id
- booking Time
- ☆ List<SeatShow>
- amount
- Auditorium
- Payment

Payment ✓

- id
- transactionId
- PaymentProvider

e PaymentProvider ✓

- RazorPay ✓
- Stripe ✓

Seat Show ✓

- id
- Seat ✓
- Show ✓
- SeatStatus ✓

e SeatStatus ✓

- Available,
- Booked,
- Locked

Show SeatType ✓

- id
- show
- seatType
- price

Schema Design.

1. Make tables for classes & enums
2. Make columns for primitive + string + datetime
✓ 3. Make columns for relationships.

City : id, name

Theatre : id, name, address, cityid

Audi : id, name, theatre id

Seat : id, number, row, col, audiId

Show : id, startTime, endTime, movieId, audiId

Movie : id, name, descr, lang

User : id, name, pwd

Ticket : id, bookingTime, amount, paymentId

Payment : id, transactionId, PaymentProvider

seatShow : id, seatId, showId, seatStatus, ticketId

ShowSeattype : id, price, showId, seatType

showFeature : showId, featureId

audiFeature : audiId, featureId

UserRole userid, roleId

SeatType : id, value

Feature : id, value

Role. : id, value

PaymentProvider : id, value

SeatStatus : id, value

T T T [Undo O(1)]

Coding.