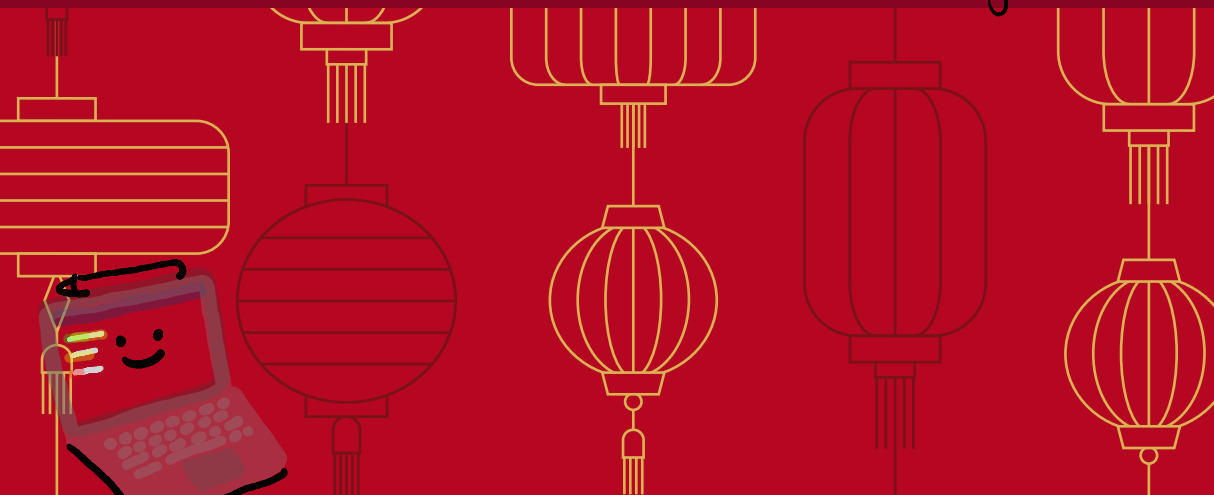
Low level Design case studies

– Rajeev



Case Study : Book My Show

5 key steps of Object Oriented Analysis & Design

1. Gather All Requirements
2. Create Use-case Diagrams
3. Create class Diagrams
4. Create Skeleton Code
5. Complete code implementation

STEP1 : REQUIREMENTS

① App will be listing current movies & shows

② App allows user to search for movies

③ App will be listing different shows for a movie

④ App allows registered user to book ticket for a show

⑤ App allows theatres to add/edit a show

Some questions we may ask interviewer

1. Multi screens support ??
2. Payement methods ??
3. Seat allocation methods ??
4. Multi-lingual movies ??

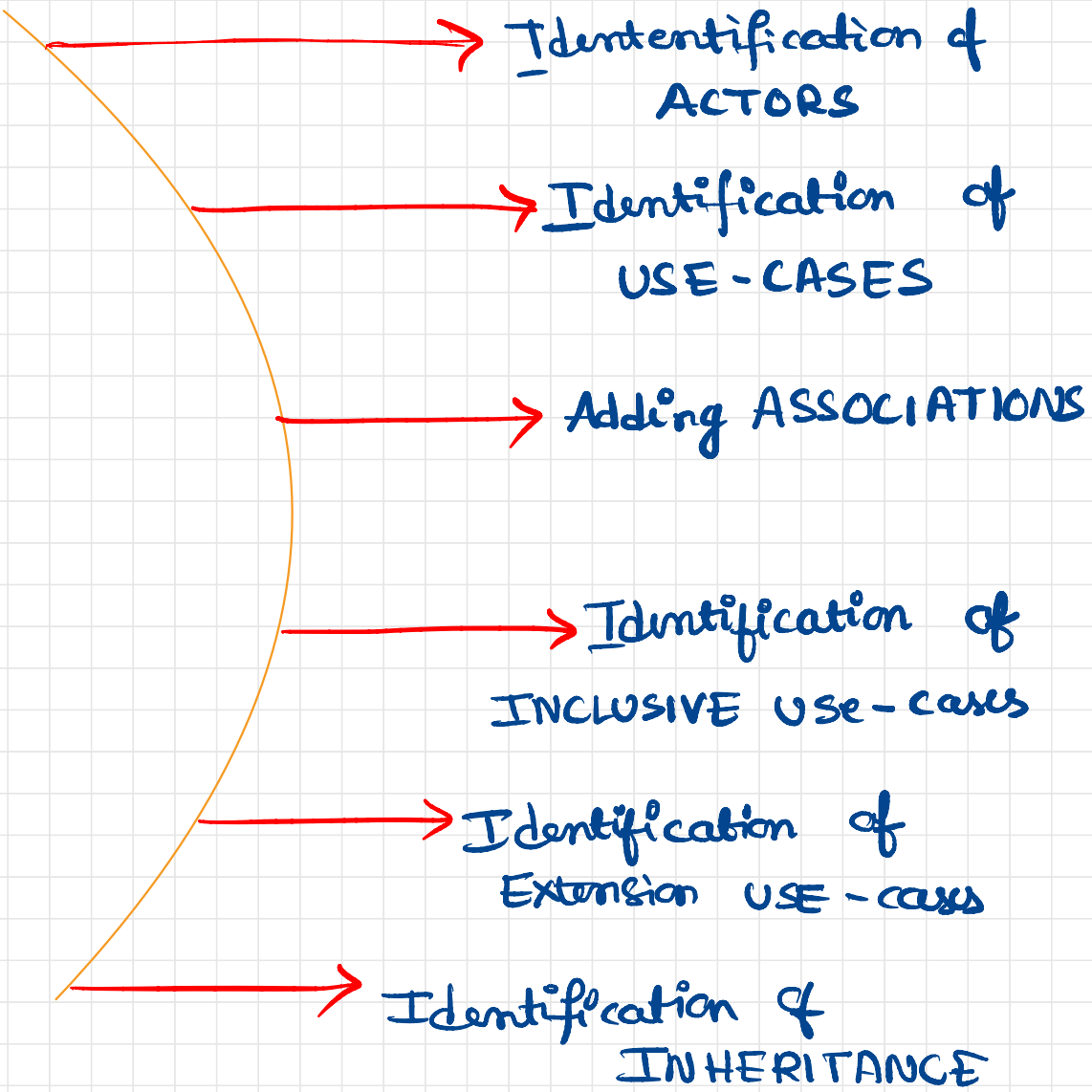
Note: In our case we are making few assumptions to develop our M.V.P. And later this M.V.P would easily be extended

Few Assumptions that can be made by us

- Every theatre has single screen.
- Every theatre has some capacity
- Registered users can book tickets but guests can only search movies
- Registered users will have history of bookings.
- Not considering payement module.

- Movies can be in 2 langs.
- Any 4 genres.

Steps to draw use-case diagrams



① In this use case the actors are "User" and "theater"

② whatever the actions that the actor takes becomes use-cases. Here use cases are

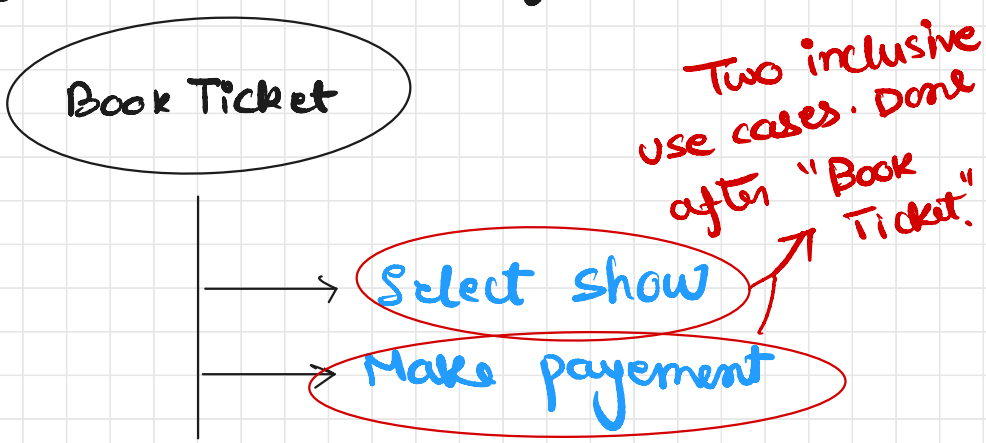
User - search Movie, Book Ticket, Login, Register

Theater - Add Show, Update Show

* Represent use cases in oval-shape

③ Connect user actions to actors with a "bold" line. This defines associations

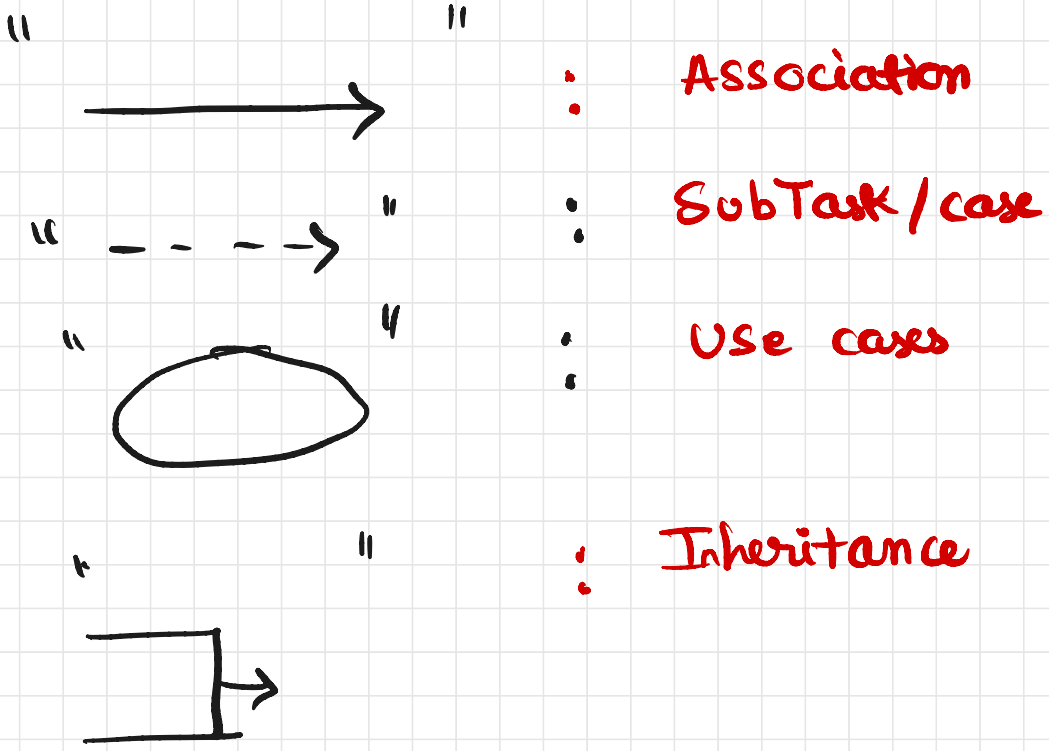
④

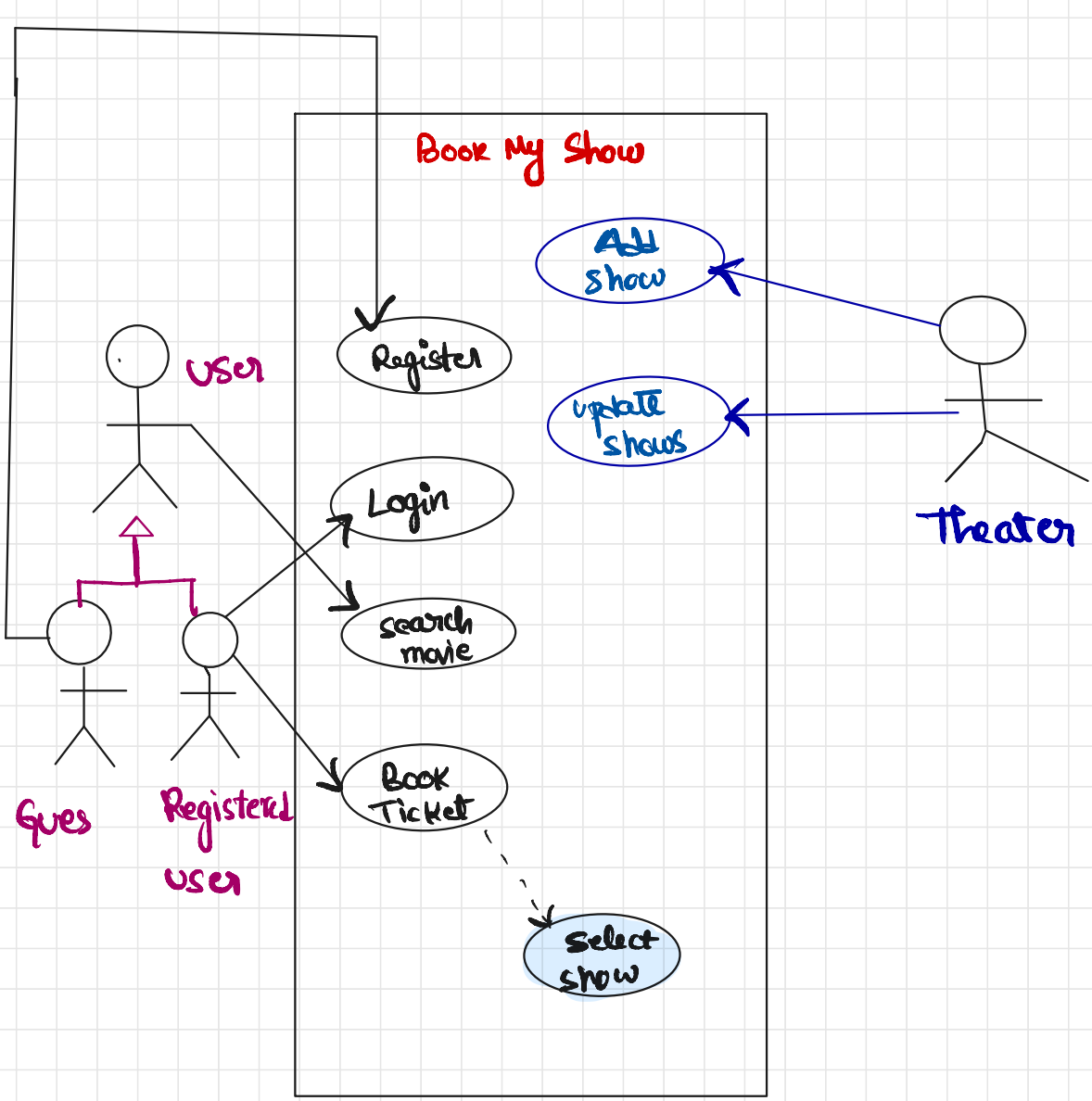


Represent sub tasks/cases with "dotted" lines

⑤ Generalisations. Two types of actors are present

① Guest user
② Registered user → user




Book My Show
Add show
Register
update shows
User
Login
Theater
search movie
Book Ticket
Guest
Registered user
Select show

Create Class Diagrams

- Go to Requirement and tag all the nouns. Ily for Assumptions as well

Nouns from Requirements

- Theater
- Ticket
- Registered User
- User
- Show
- Movie
- App

Nouns from Assumptions

- Genre
- Screen
- Language
- Guest user

. Assume we have a class for all Above nouns

Different Relations between Class that Could exist.

- Association
- Aggregation
- Composition
- Generalization

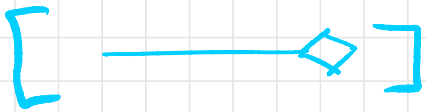
Relations

- App consists of "Users" and "theaters" so its a compositional relationship from user and theater to App. Remember to add multiplicity (*)

- There are two types of user: ① Guest and ② Registered user. But both are of type "User". Hence we can have "generalization

denotion of relation

- Every "Movie" will have a 'genere' and a 'language'. Hence we have "aggregation" relation btwcen "Movie", "lang.", "Genre"

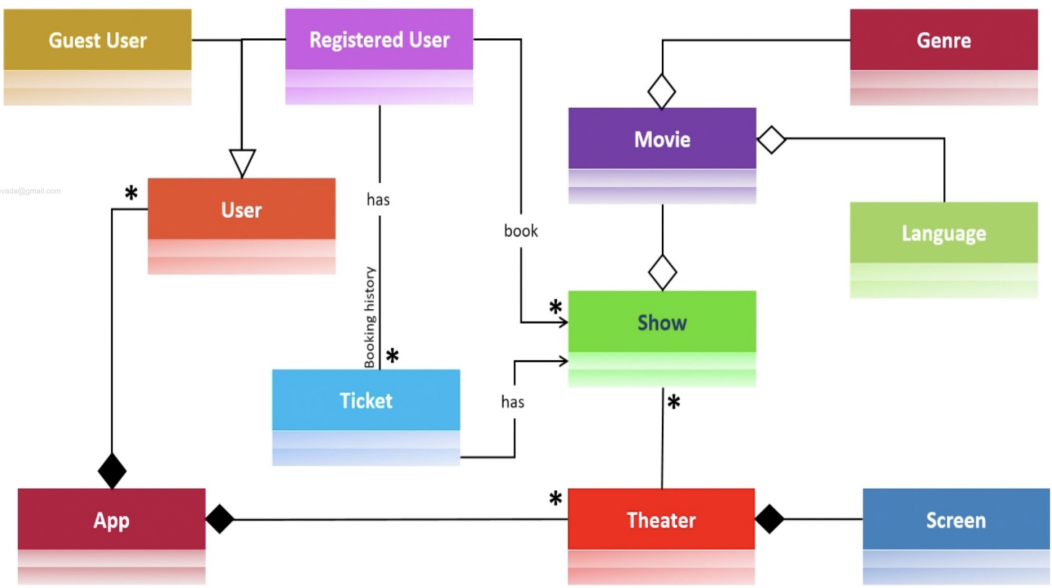


denotion

- Every theatre has a single screen (from assumption). So there is a "compositional" relation.
- A show consists of a movie. we have a aggregation relationship between show and movie. (Note one show only has one move ⇒ no multiplicity)
- A Registered user can book as many shows as he/she wants. So we connect them with one to many association

- A Registered user can see list of booked tickets so we have one to many relation.
- A single show has a ticket. So its a one to one relation.

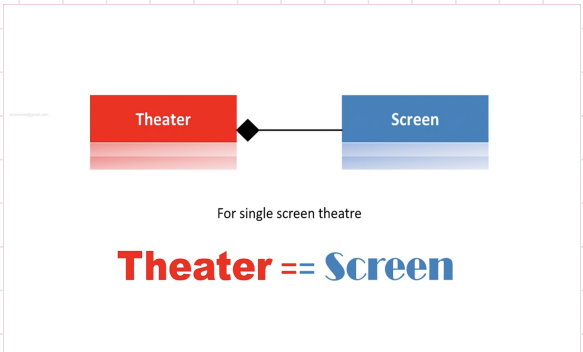
Class Diagram with Relations added to each class.



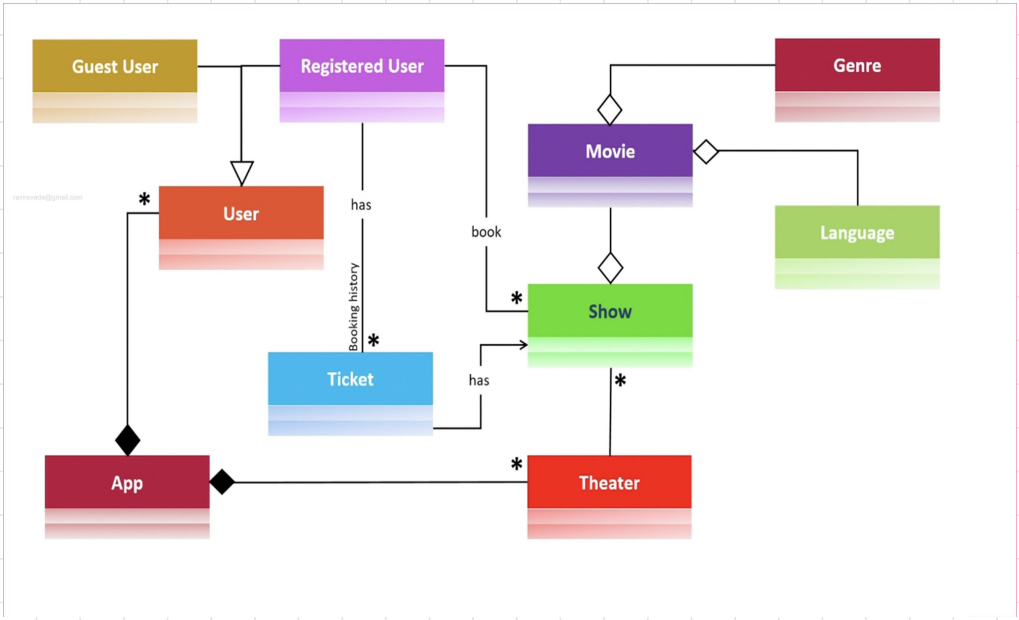
Looking for some Refinements

- Every theatre has a single screen (assumption)

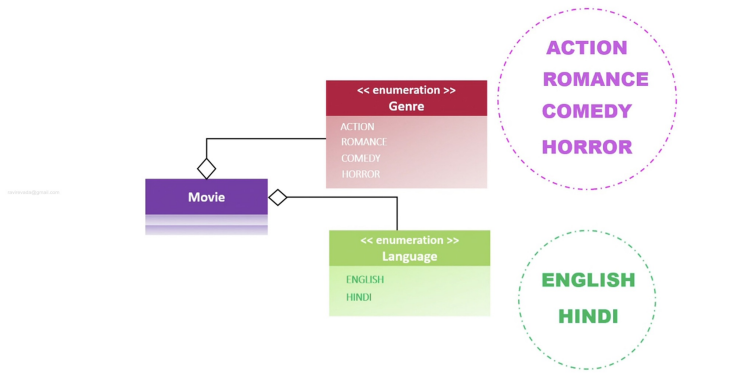
⇒ theatre == screen



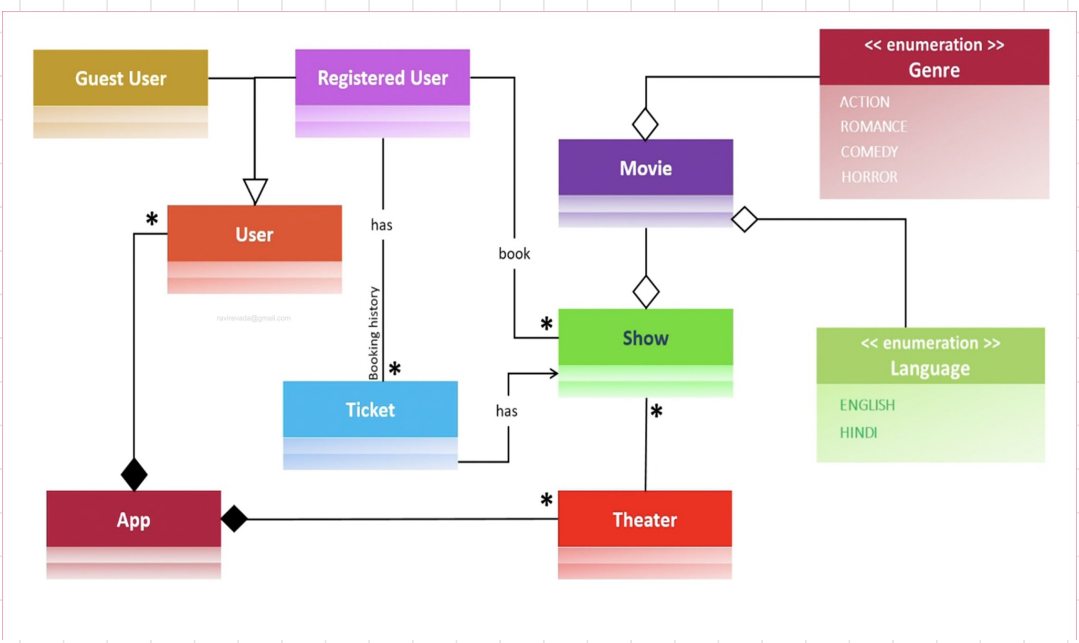
After Refinement (1)



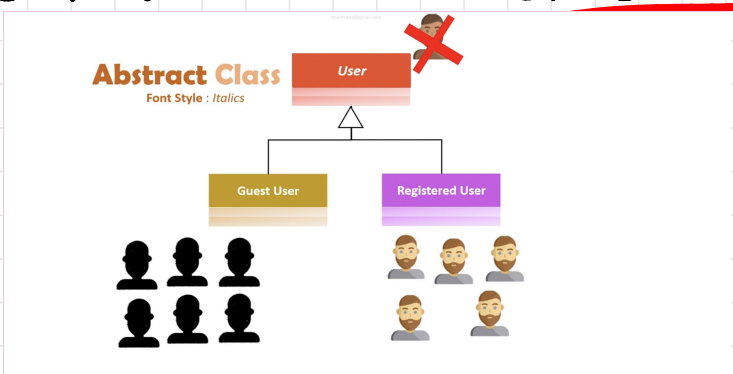
- Every Movie would be one of 4 genres (assumption) So. No need for a class. we can go for enum
- Ily Every Movie would be of 2 langs. (assumption)



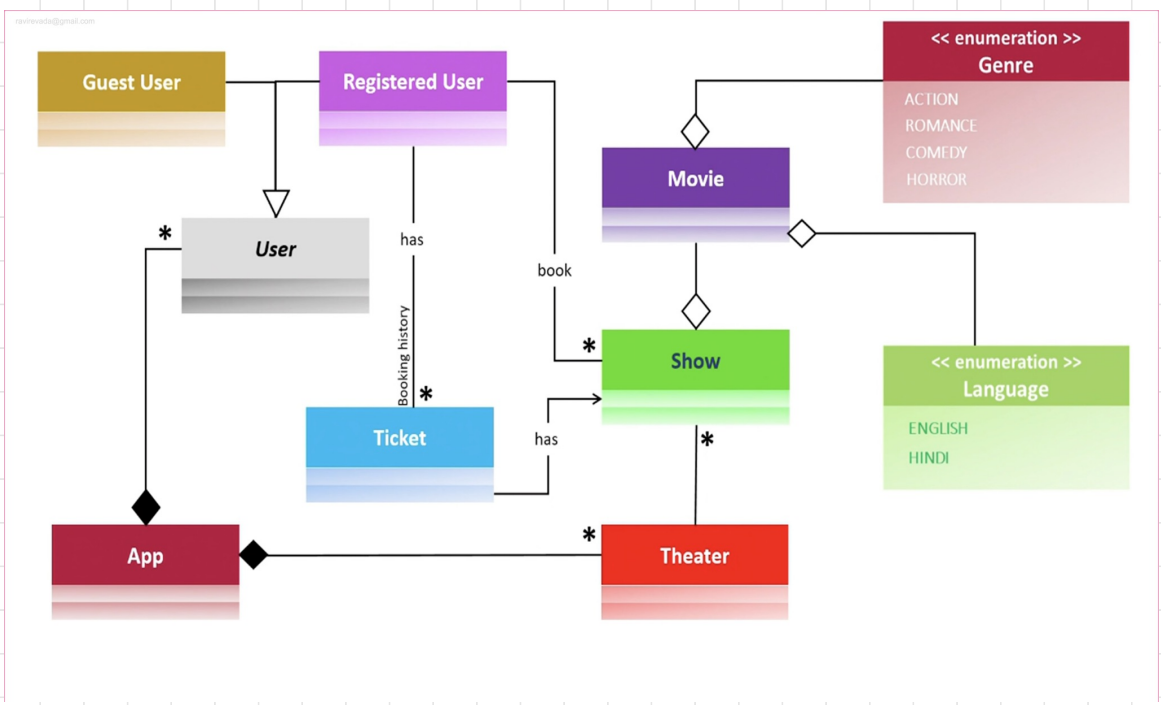
After Refinement (2)



- A user will be a object of a guest user class & a registered user class. So we will never create an object for "User" class. So we make 'User' as abstract



After Refinement (3).



Adding attributes and methods.