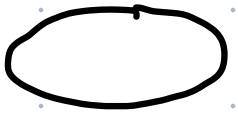
Use Case Diagram

→ Represents set of use cases of the system.



Use Case

Communication line b/w actor & use case

Actor (if system <<external system>>)

generalisation (actor / use case)

<<include>>


Includes an existing use case will be executed always

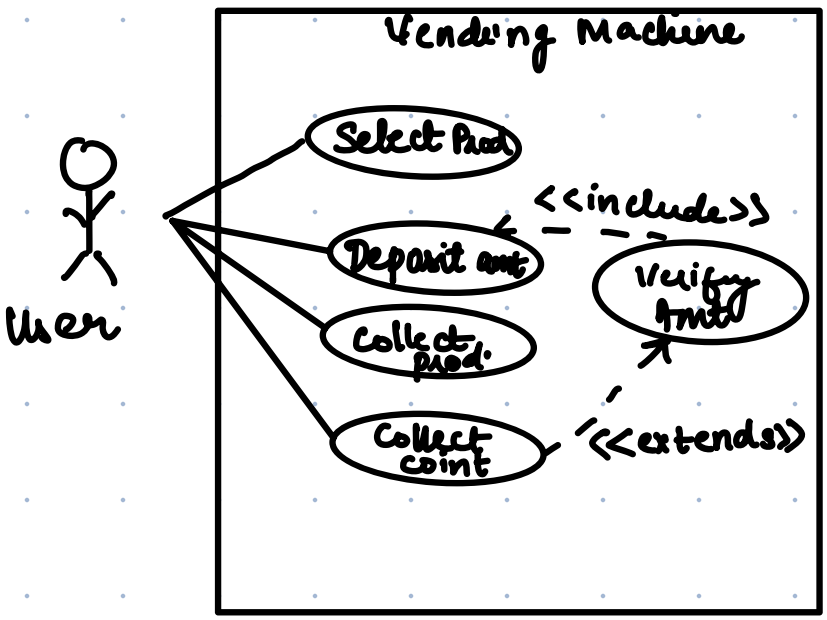
<<extend>>


Extends an additional use case conditionally executed

Q. Vending Machine

Actor : User

Use Cases : Select drink , Deposit coin , Verify Coins ,
Collect Product , Collect Coins



Refine : Actor : User , Technician

Extra Use Case : Maintanence , Make Repairs
Replenish Product

↙
Check Inventory

Bank

Q. Actor : Customer , Technician , Bank ← external system

Use Case : Check Balance , Dep. Funds , Withdraw

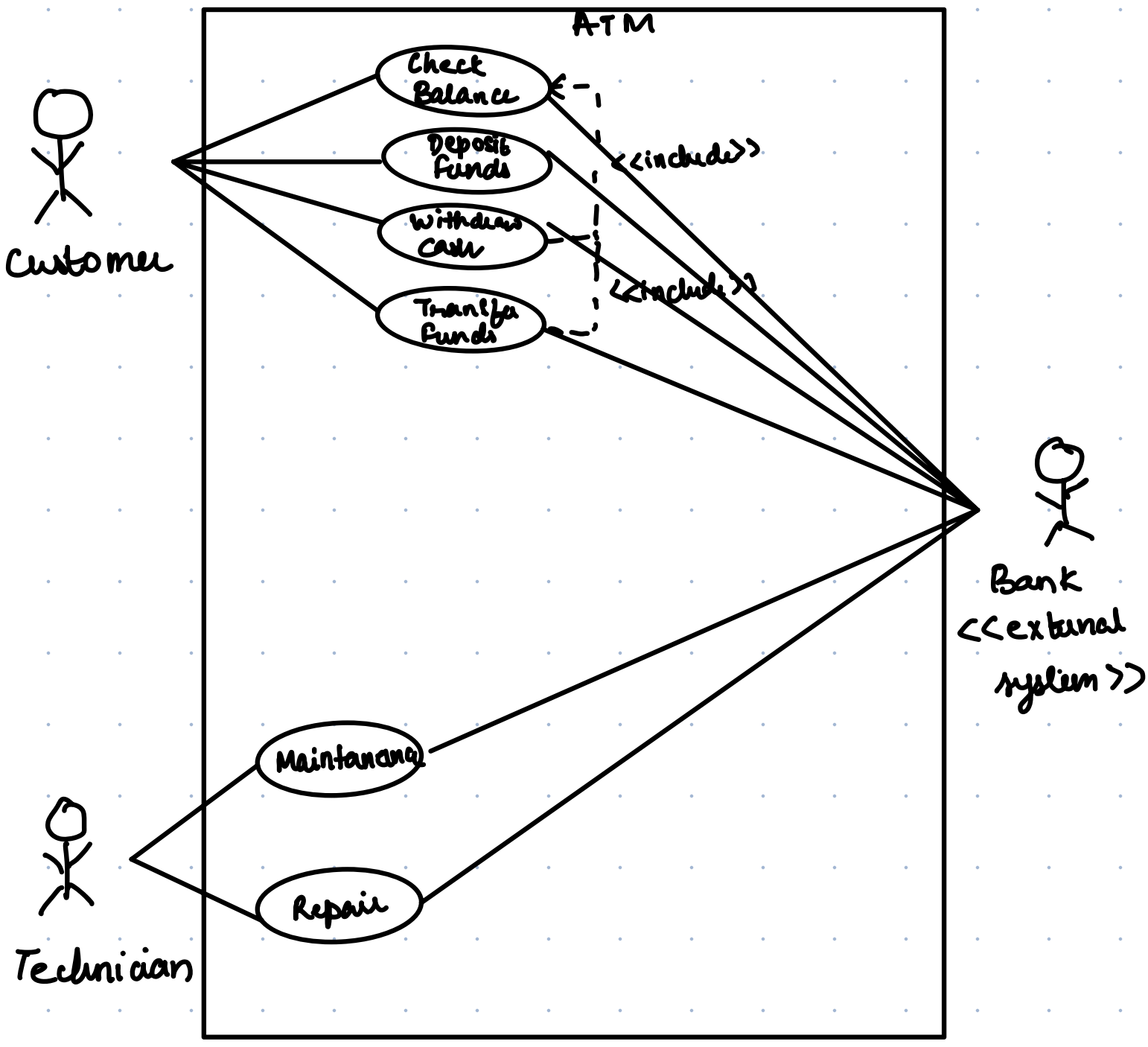
↑

Cash , Transfer Funds

↓

Selecting Acc. Type

Maintanence , Repair

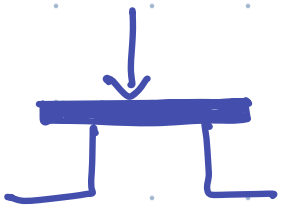


Activity Diagram

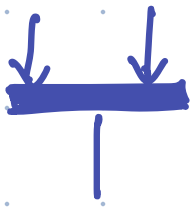
● Start State

◉ Final State

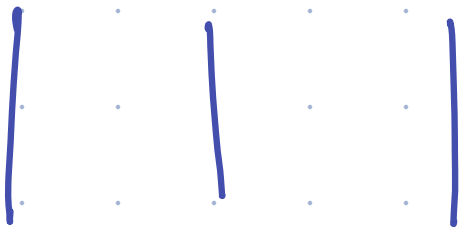
⟶ Flow arrow



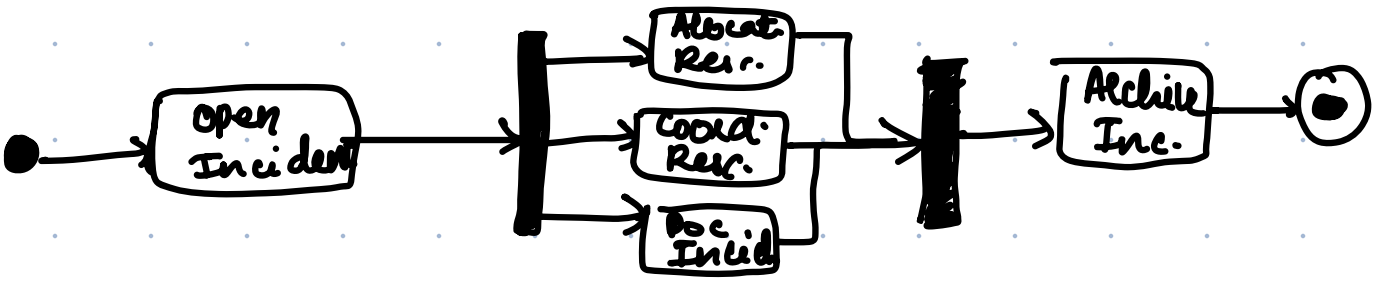
Fork

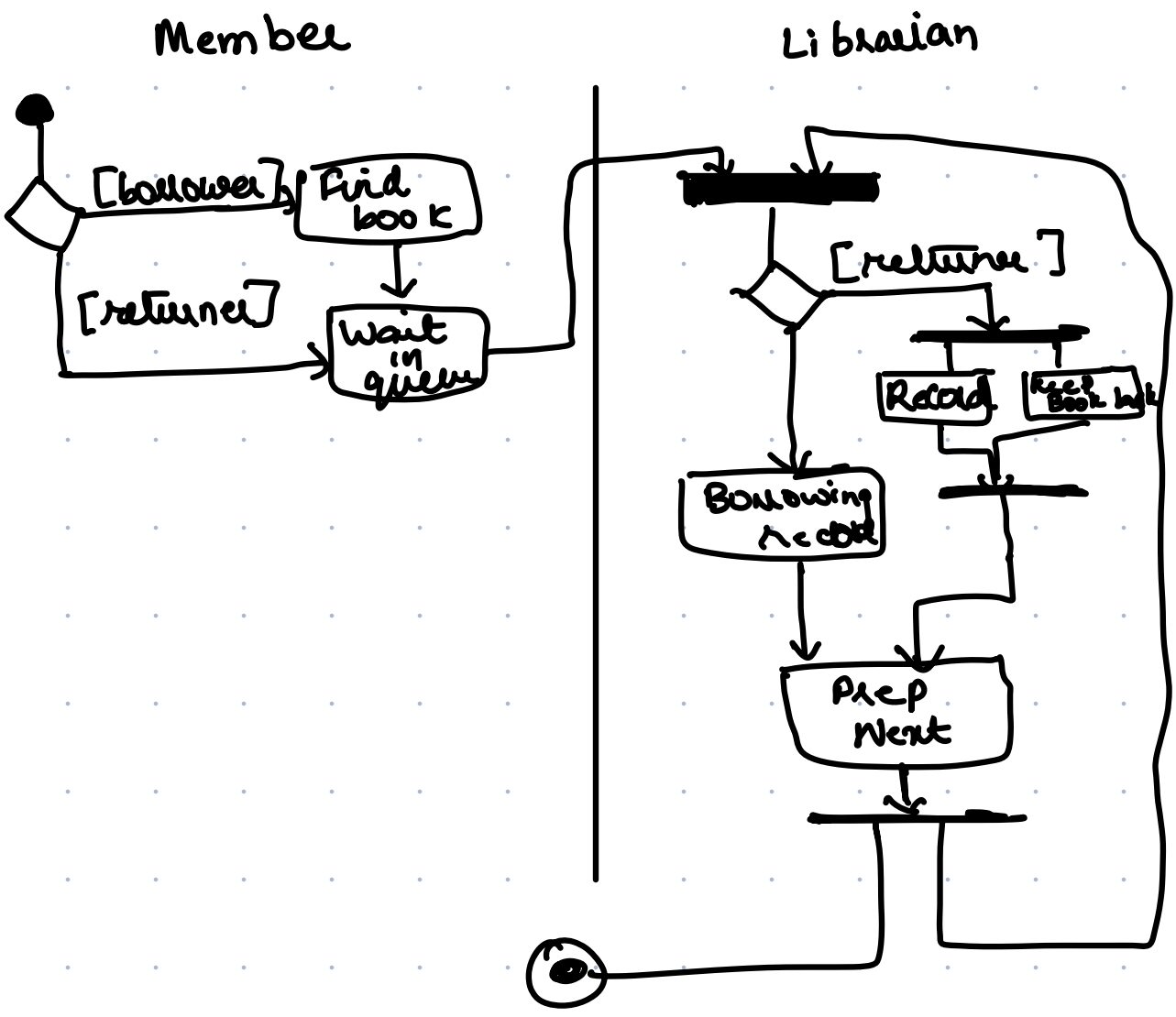


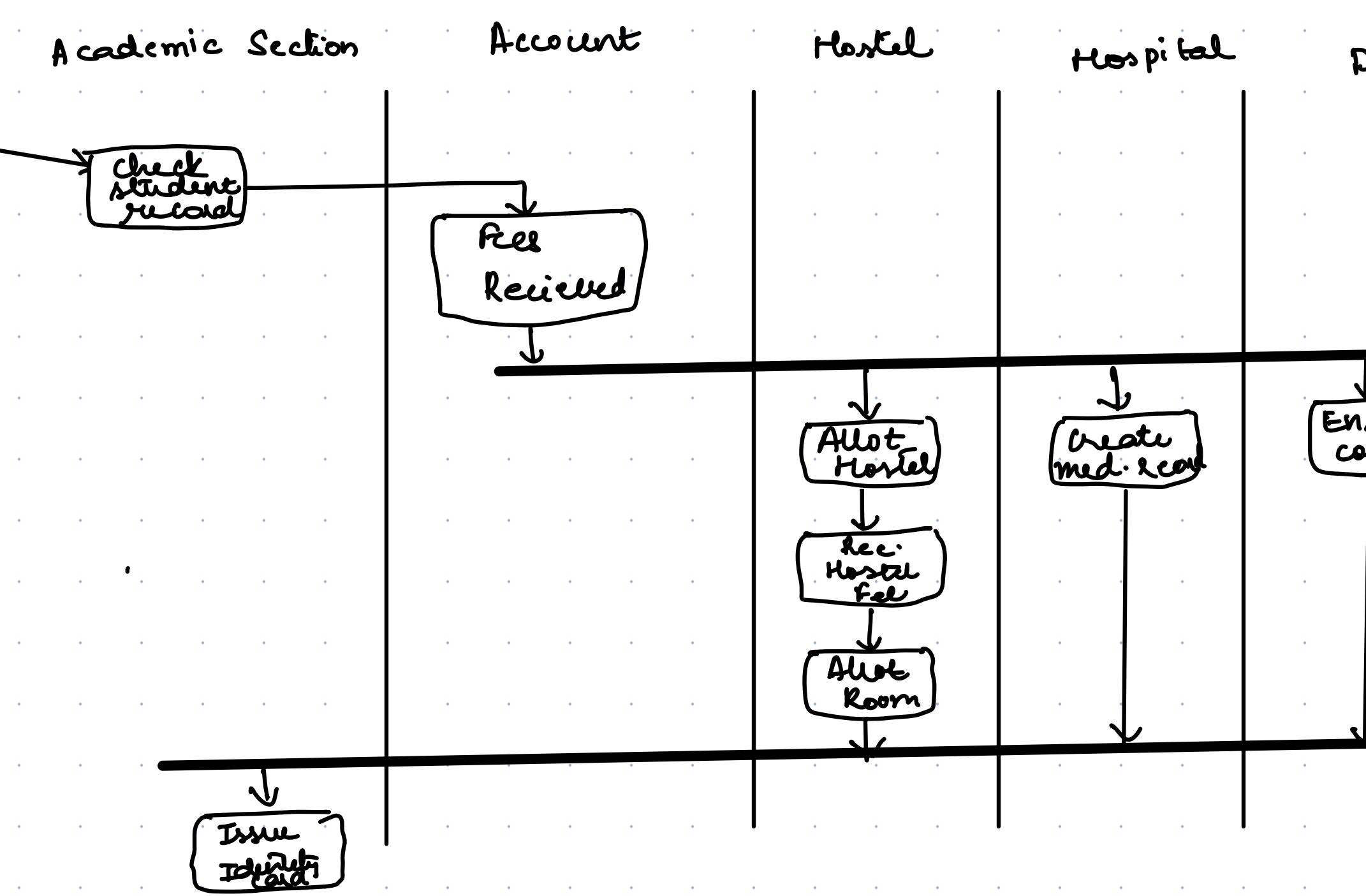
Join



swim lanes — group activities acc to whose performing them




Member
Librarian
[borrower]
Find book
[returner]
Wait in queue
[returner]
Record
Borrowing record
Prep Next


Academic Section
Account
Hostel
Hospital
Check student record
Fee Recieved
Allot Hostel
Rec. Hostel Fee
Allot Room
Create med. record
Issue Identity Card

Hospital Management

Patient

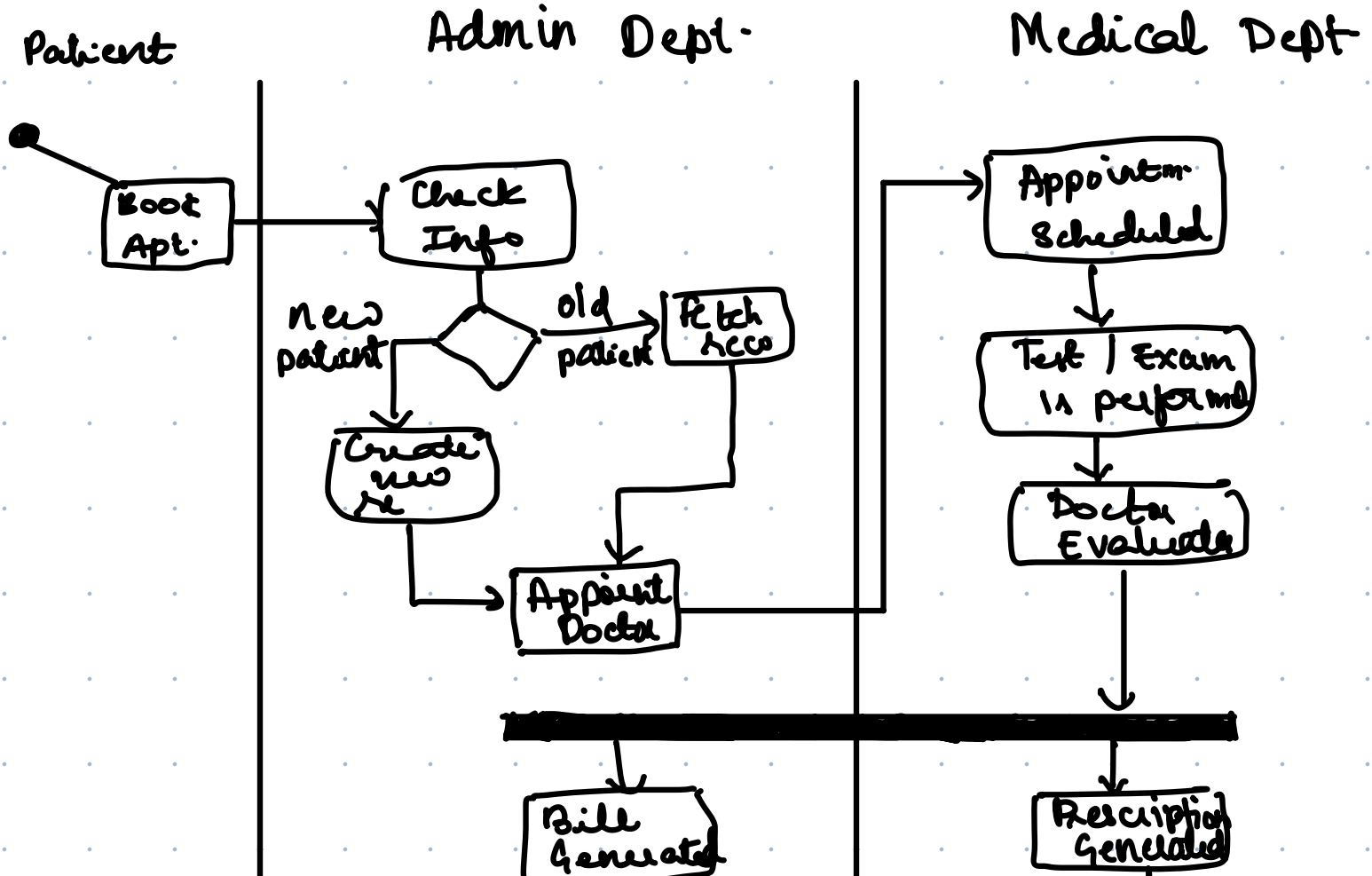
Admin

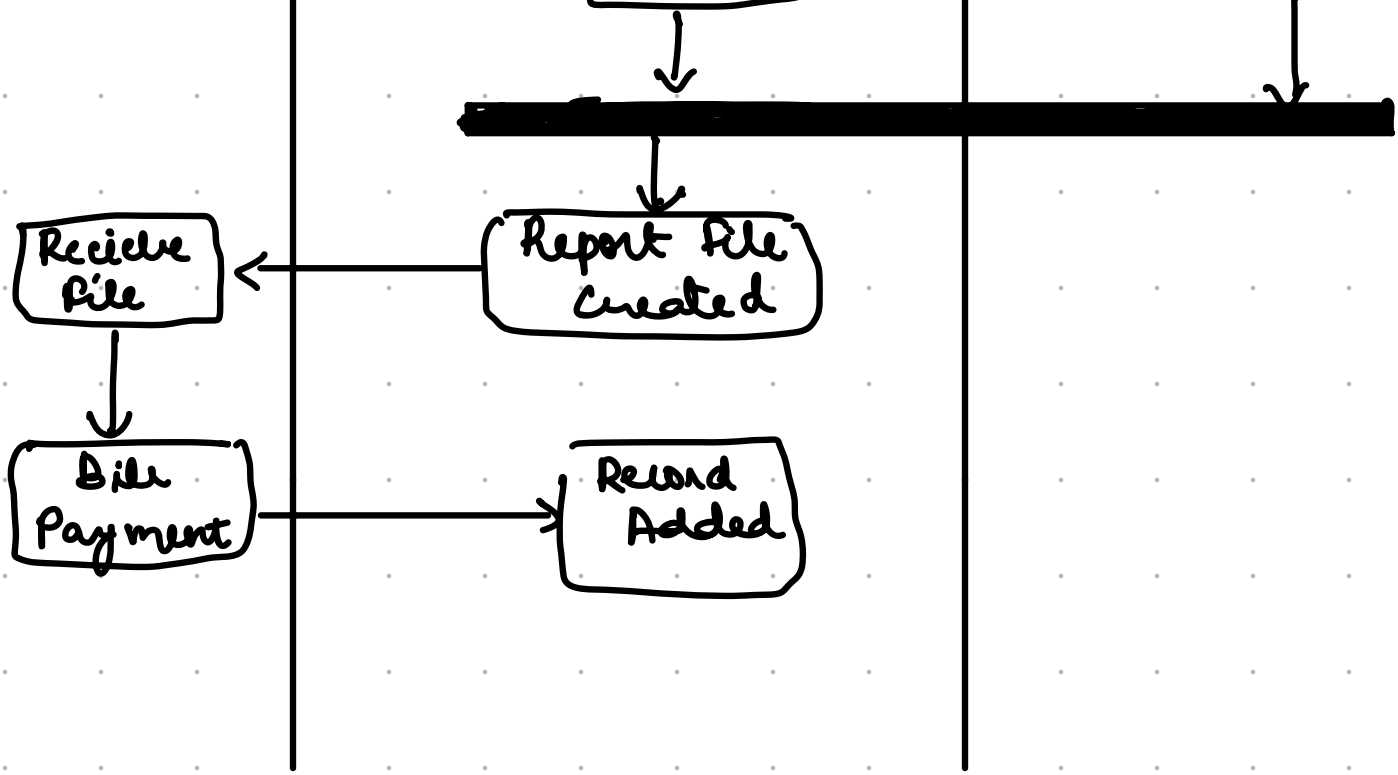
Doctor

>> Join

>> FMK

>> Descision





* Write in Present Working tense

eg: generating Prescription like that

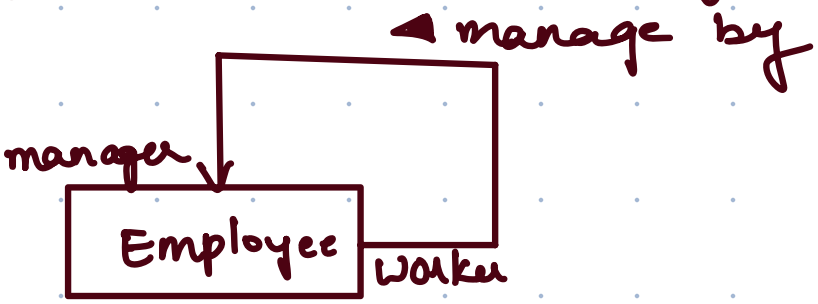
Class Diagram

1) Association : An association b/w two classes indicate that object at one end of an association recognizes the object at the other end, and may send messages

Directed → you label the relationship and provide a small triangle indicating direction to increase readability

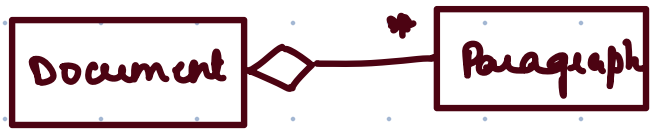
Employee works for Company

2) Reflexive Relationship : is a relationship from one class back to itself.



3) Aggregation: is a type of association used to model "whole to its parts" relationship b/w objects

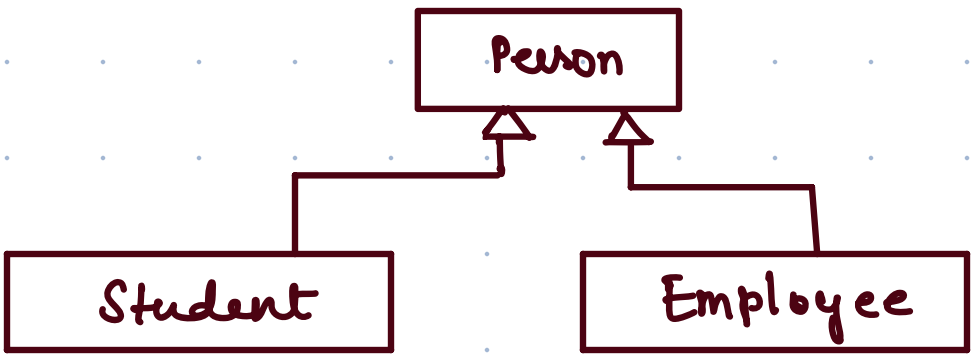
→ Basic Aggregation



→ Composition Aggregation: strong indication of ownership parts are created with the whole & destroyed when the whole is destroyed.

4) Dependency: Dependency relationship is shown as a dotted arrow from dependent class to independent

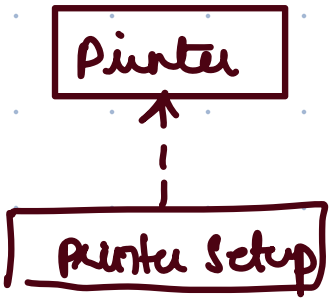
5) Generalization: is a type of association to model "is-a relationship" b/w base/parent class and sub/child class.



Student, Employee are specializations of Person.

↳ derive methods & attributes of base class

6) Realization: a realization relationship indicates that one class implements a behaviour specified by another class.



Sequence Diagrams

→ Emphasis on time ordering
→ Represent dynamic behaviour of object.
→ Which object communicates with which and what message triggers those communications.

Object (a: Object)

↗ name of object

Smith: Patient

↗ Object

: Patient

↖ anonymous

unknown class ↘

Smith

Timeline of an object

Activation box → points at which object is active

* Iteration marker — for all objects

[condition] message sent only if condn is true.

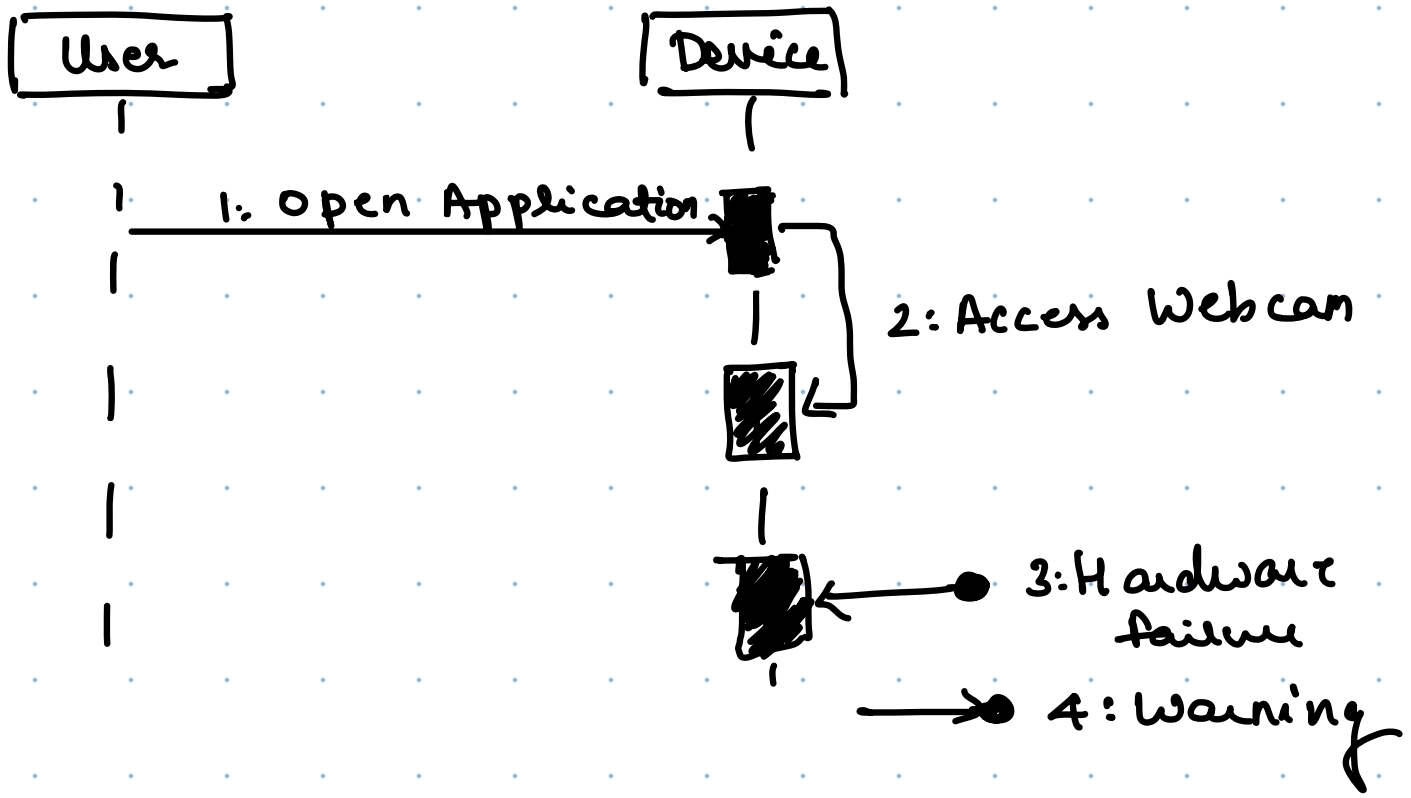
Synchronous message requires a response before interaction can continue.

Asynchronous message dont need reply

Self message

lost : when sender is known but not reciever

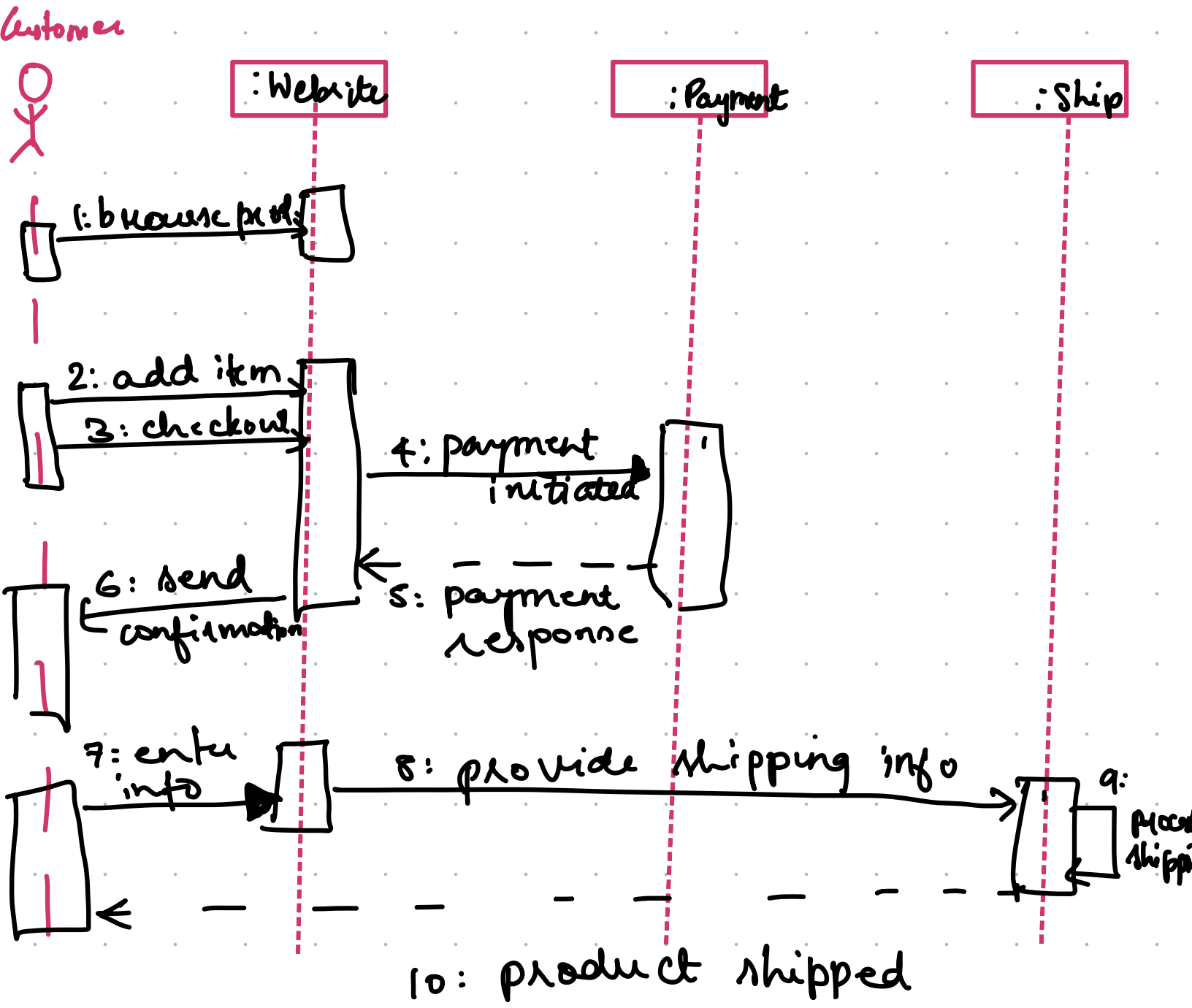
found: sender is unknown
but reciever is



Lifeline v/s Actor ← Depict objects external to the system

↓

potrays an object internal to the system


Customer
:Website
:Payment
:Ship
1: browse prod
2: add item
3: checkout
4: payment initiated
5: payment response
6: send confirmation
7: enter info
8: provide shipping info
9:
10: product shipped

Collaboration Diagram

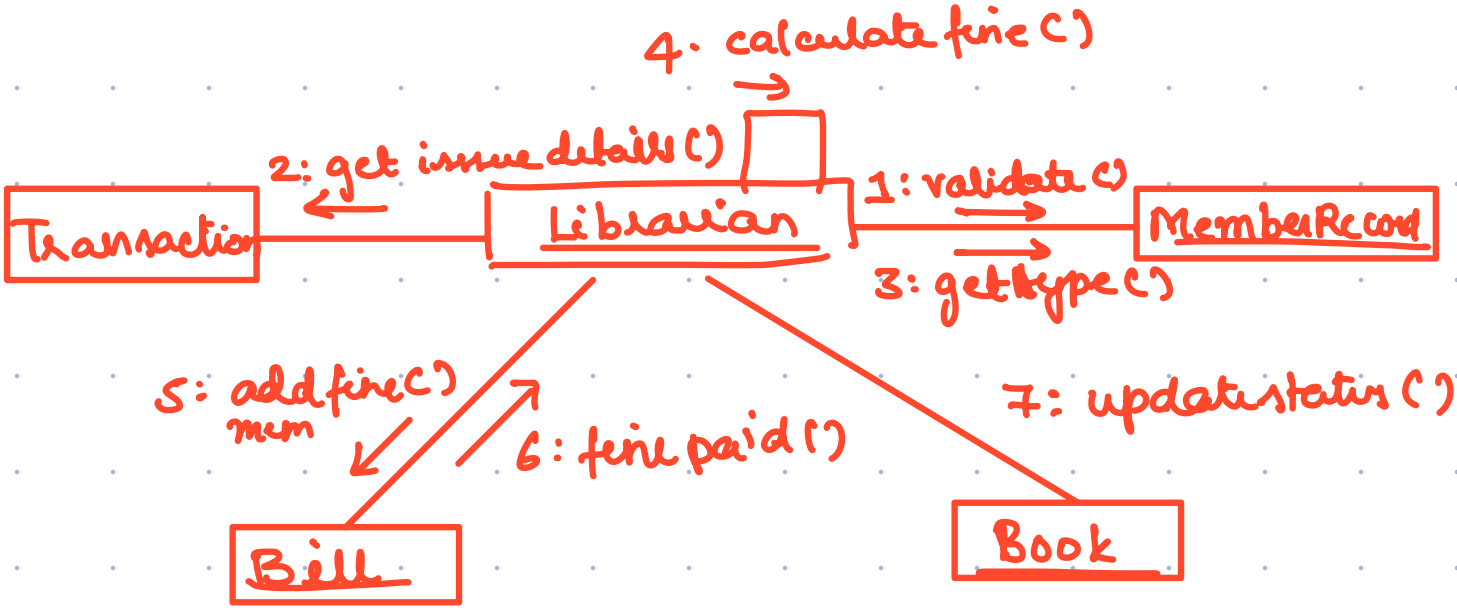
Structural + Behavioural

Flow of messages b/w objects

—————— solid link b/w object

→ message (labelled)

Prefixing message with numbers to indicate sequence



State Chart

→ used to model how an object changes in its lifetime.

→ how the behaviour changes in diff use case executions