Chapter 3:- Markets

A market is a set of actual or prospective customers who need or want a product.

Active channels
↳ Directly discuss with customers.

Passive channels
↳ On base of observation of previous product and experience.

Things already available are assumptions.

UML Diagrams

Models, Views, Diagrams

When the program is not running
↳ static

Use Case is static diagram (Black white diagram, Provide Association)

1. Use Cases → Step by step to perform a function
2. Actor → Primary actor
3. Relationship → Association

1. Use Cases
   - Name
   - Primary Actor
   - Pre-Condition
   - Goal
   - Trigger
   - main success Scenario
   - Alternative flow

   Qualities / Exceptions

Use case diagram just give overall view of diagram

Trigger
↳ When use case should be initiated.
eg. When student click subject registeration button.

Main process Scenario
Step by step scenario

User oriented Scenario	User + System oriented Scenario
① Student will open browser	① Browser is opened.
② Stud open Lms.uet.edu.pk	② Website is opened
③ Stu sign into.	③ Stud is sign in
④ Stu select subject reg	④ Stud has selected sub reg

Qualities
↳ Non-functional Req
eg Security

Exceptions.
↳ Problems and it's Solutions
e.g No internet is available
Sol:- Take hotspot

Relations:-

Extend ← extend → alternative, secondary flow ~~exceptions~~

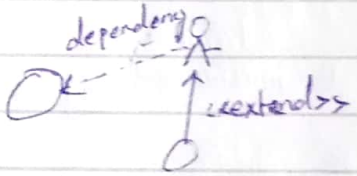
Include ← include → exceptions

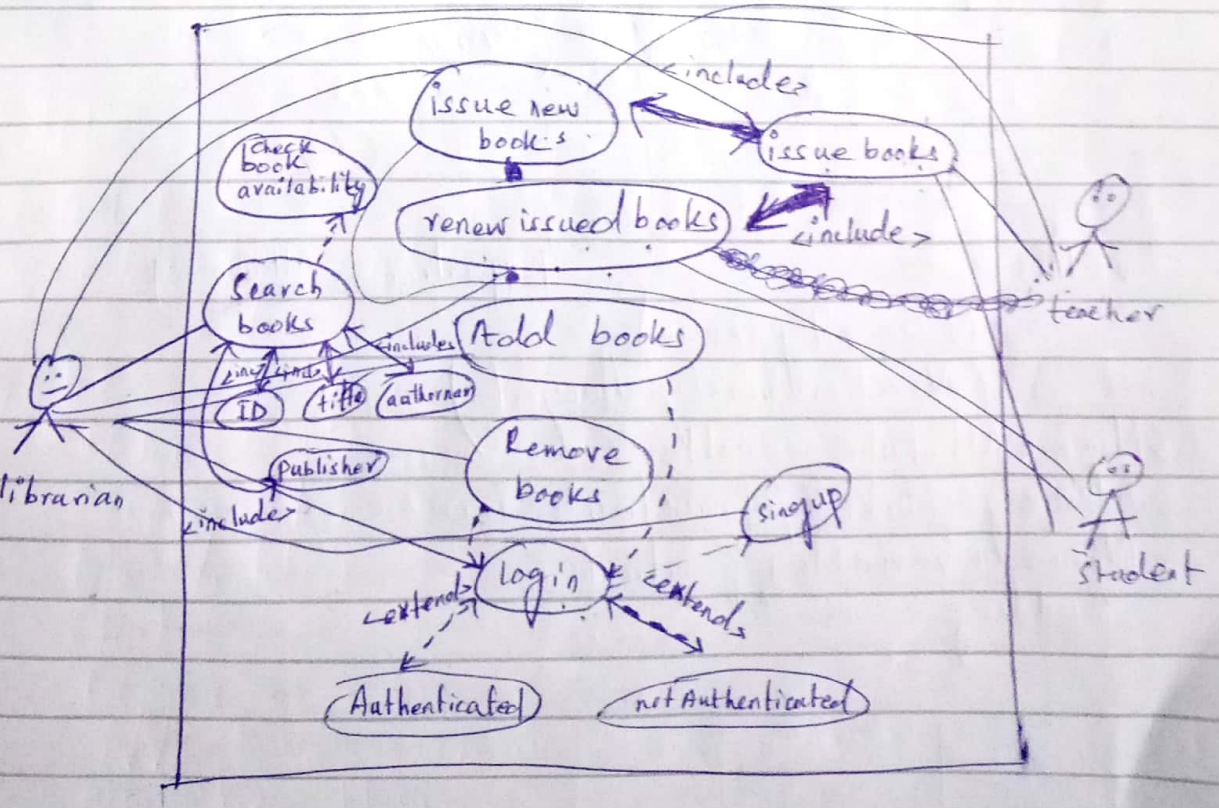
Dependency - - - → different ways to perform a task

In test case you test your usecases.

University library Management System.

issue new books
renew issued books
Adal books
Remove books
login
Search books
books availability





University's Library

Issues Books :-

Name Issues Books

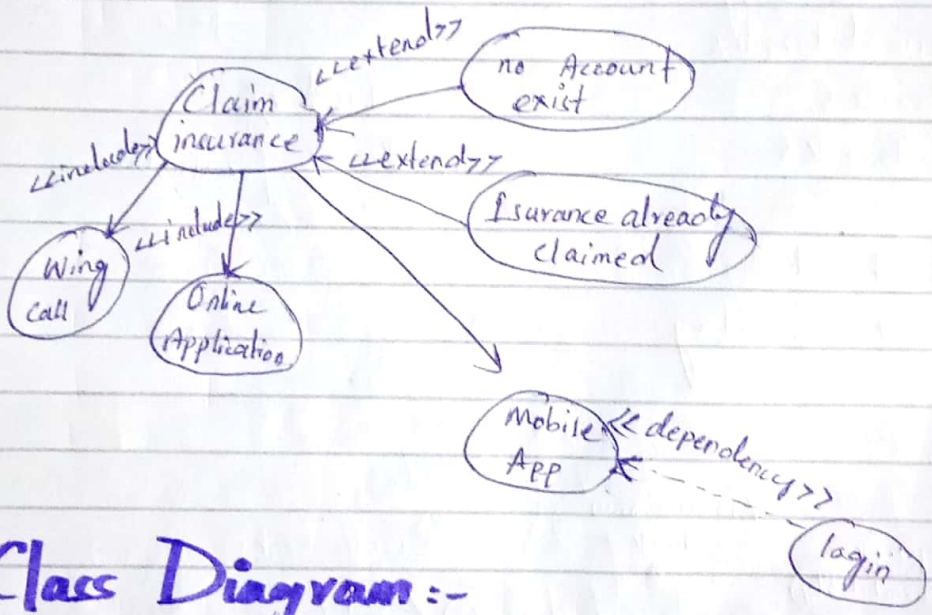
primary actors Student, teacher

pre-condition (1) Already issued books must be less than 5 books.

(2) Librarian checks that how many books have been issued to students.

(3) Student has to show his/her student ID Card.

(4) Librarian checks the availability of books.



Class Diagram :-

→ Class

Properties → Attributes / variable

→ Name only (noun) (1)
(Never or rarely changed)

→ Name + Attribute / variable (2)

→ Name + operation / function (3)

→ Name + variable + function (4)

Access modifiers :-

+ → public
- → private
→ protected
underlined → Static

Static object will be created only one time but final object will be created again again whenever it is called.

Student
ID stuID
Depart name string
FName string
Age int
Address string
Email string
Login( )
registeration( )

Interface :-

Interface is implemented private, protected properties cannot be defined

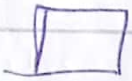
Abstract class :-

Abstract class is extend and private, protected properties can be defined.

If all properties are public then such abstract class is interface.

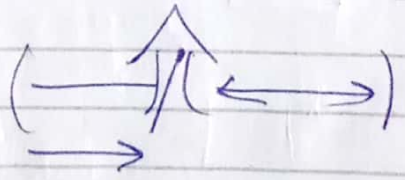
Interface is not class

Relationship between Classes :-

i) Association :- both means same

→ binary
→ unirarly

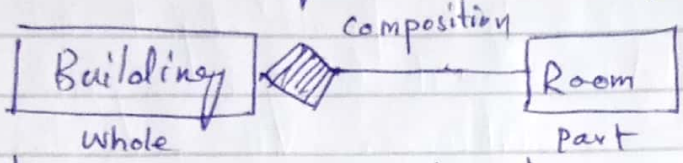


ii) Composition ◆] Part-whole relationship
iii) Aggregation ◇]

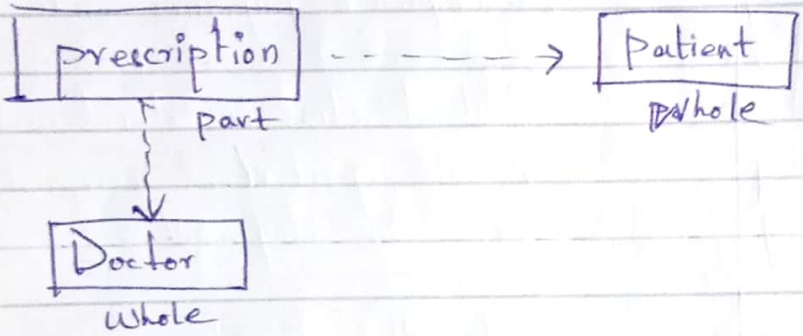
iv) Inheritance/Generalization ↑

v) Dependency - - - →

After removing part class whole class will be changed in composition



After removing part class whole class will be removed and would not exist.



Multiplicity :-

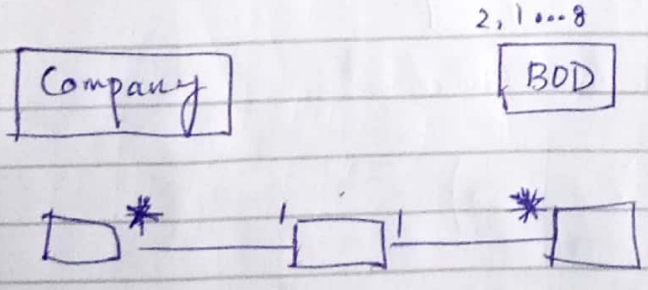
1 —— * 0 - - - ∞

1 —— n 1 - - - ∞

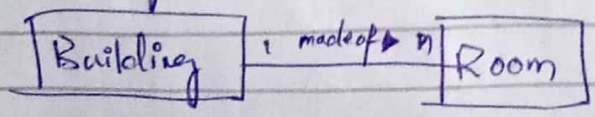
1 - - - 1

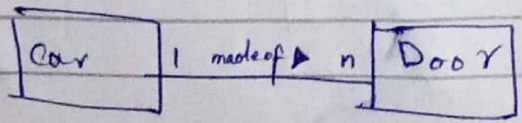
1 - - 2 —— 2 → range

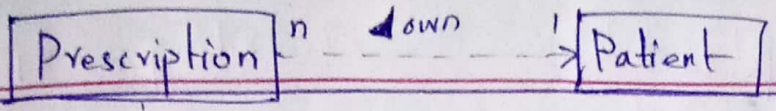
1 - - - 8, 2 - - - *

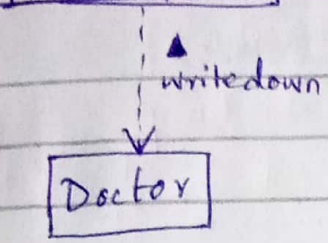


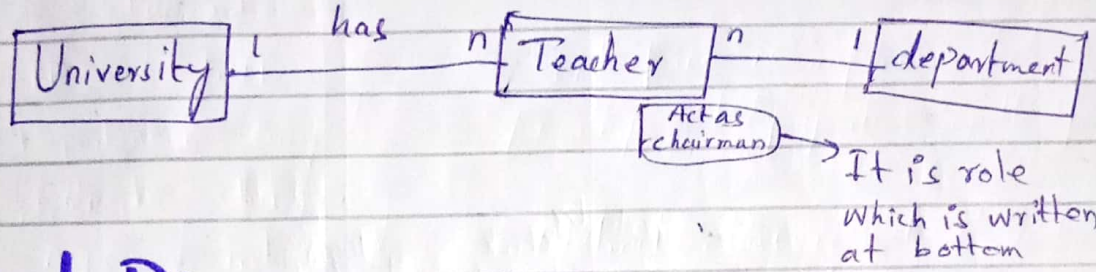
Naming the Class → To increase readability





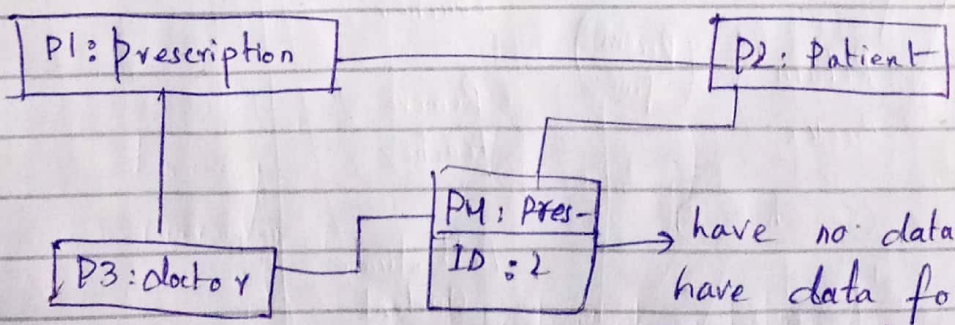






Object Diagram :- static (snapshot of Diagram)

→ Remove Relations (Aggg, comp, Dep, Inheritance)
→ Convert all relations to Association.
→ Remove multiplicity names of relation
→ Remove data types, Access modifier.
→ Convert class names to object by underlining.
→ Asign unique keyword.
→ Mention data in variable.
→ You can make multiple obj of a class.

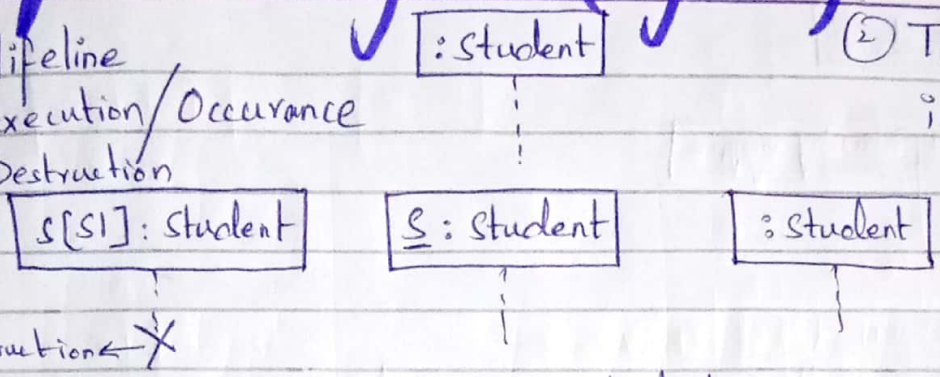


have no datatype but have data for variable.
There can be multiple objects of a class.

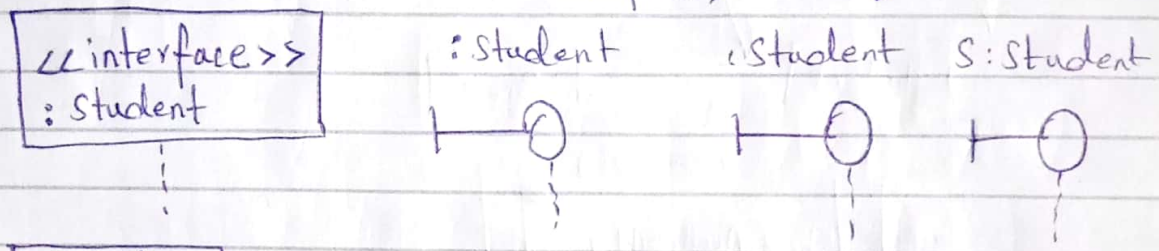
Sequence Diagram :- (Dynamic)

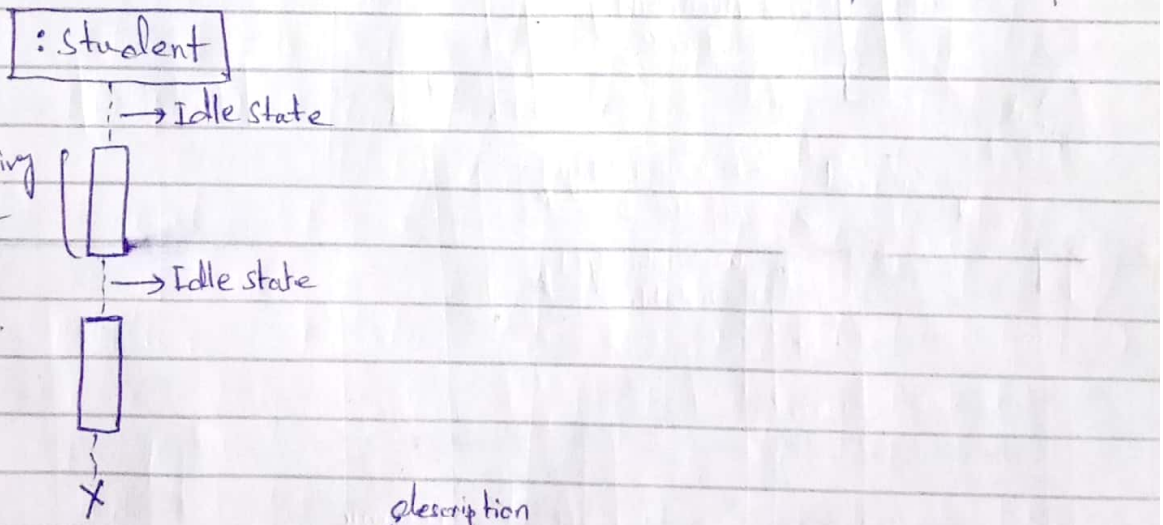
① Dynamic
② Timing is important

1- lifeline
2- Execution / Occurance
3- Destruction

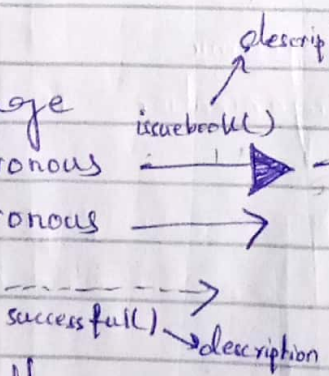


life line of S of class Student using selector [s1]

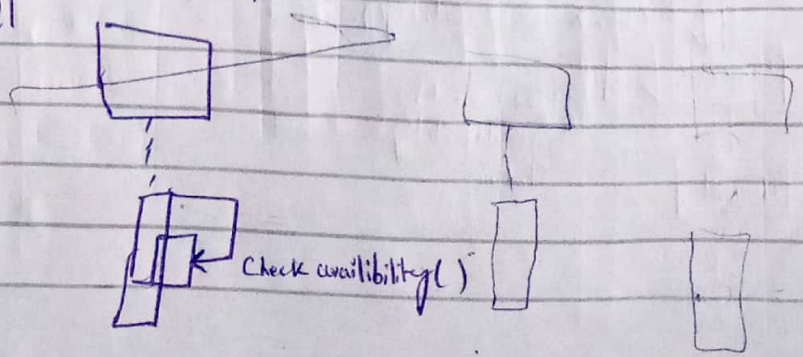




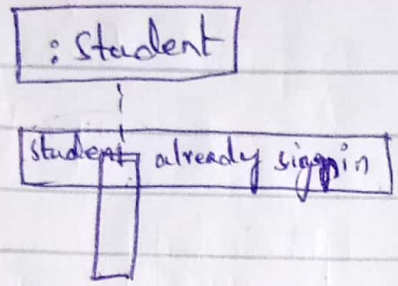
④ Message
Synchronous → There must be reply otherwise next message will not forward.
Asynchronous →
Reply - - - - - →



⑤ Self Call

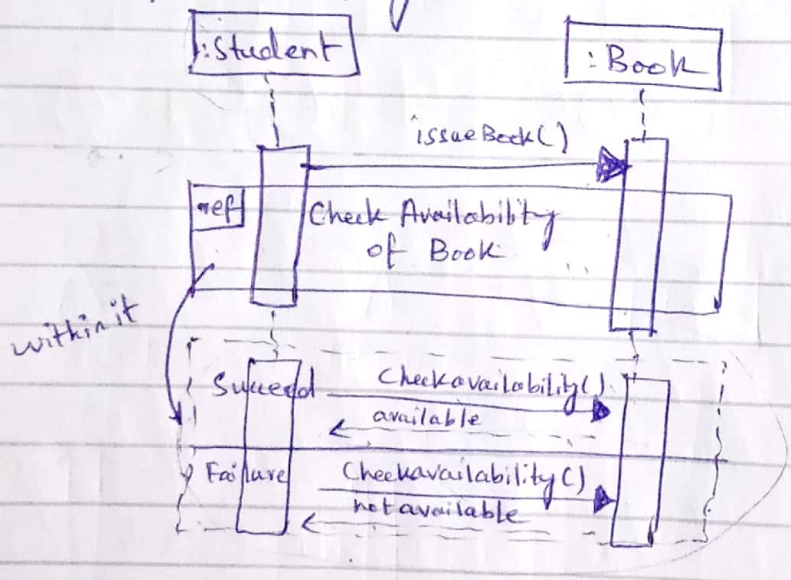


⑥ State Invariant (Condition)



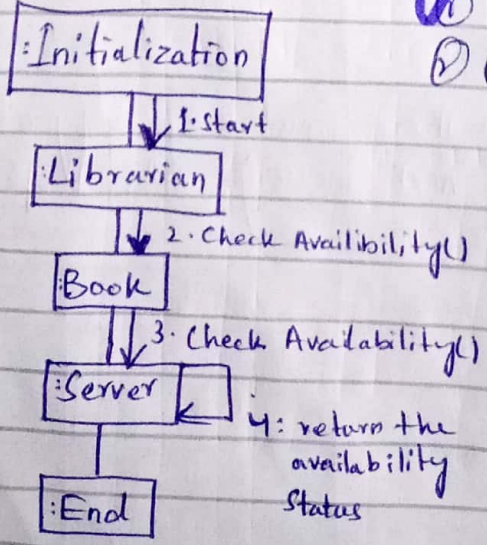
⑦ Interaction Use:

To call a sequence diagram in another Sequence diagram.

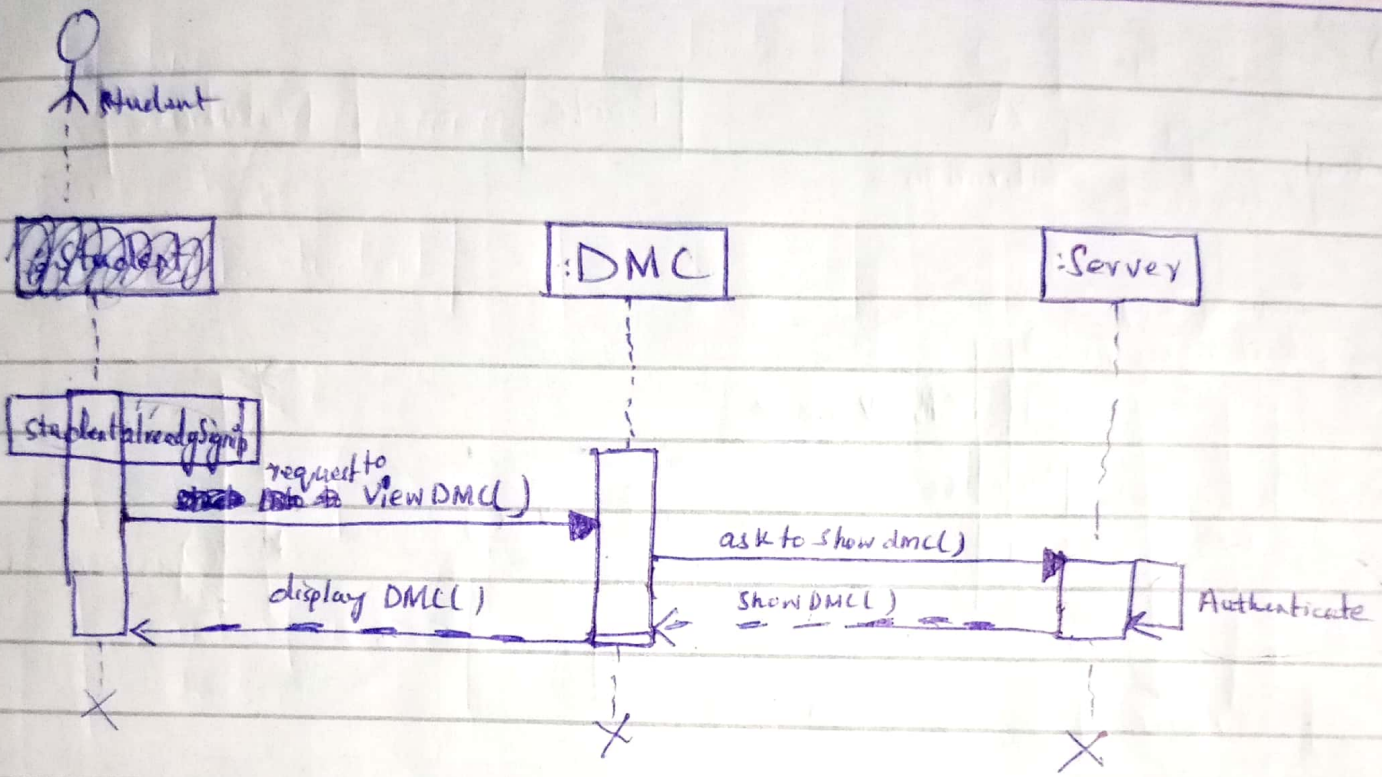


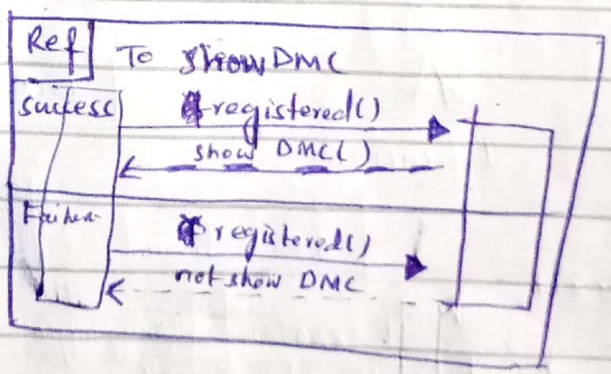
Collaboration Diagram:-

① Dynamic
② Organization is important



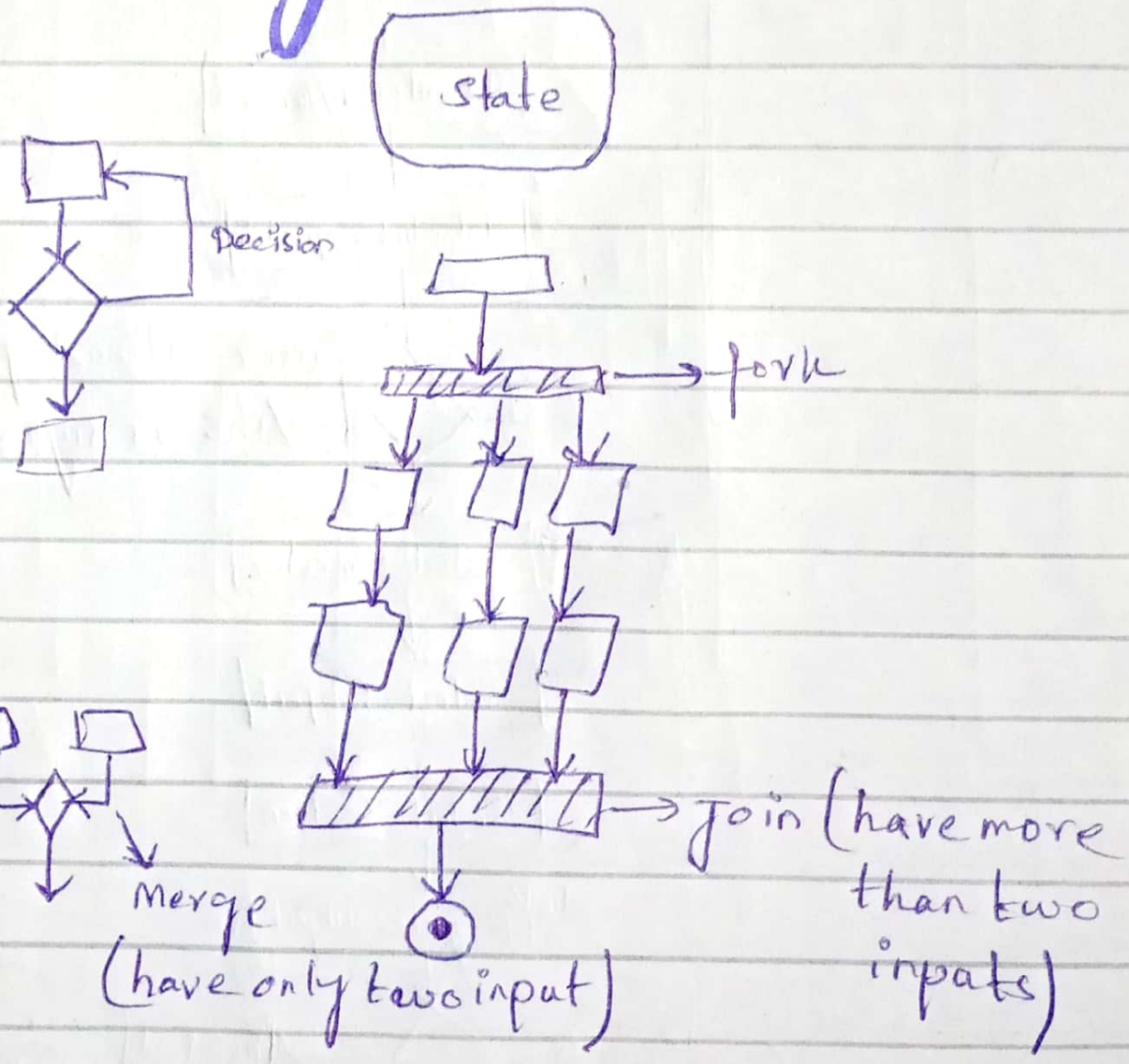
View DMC on LMS

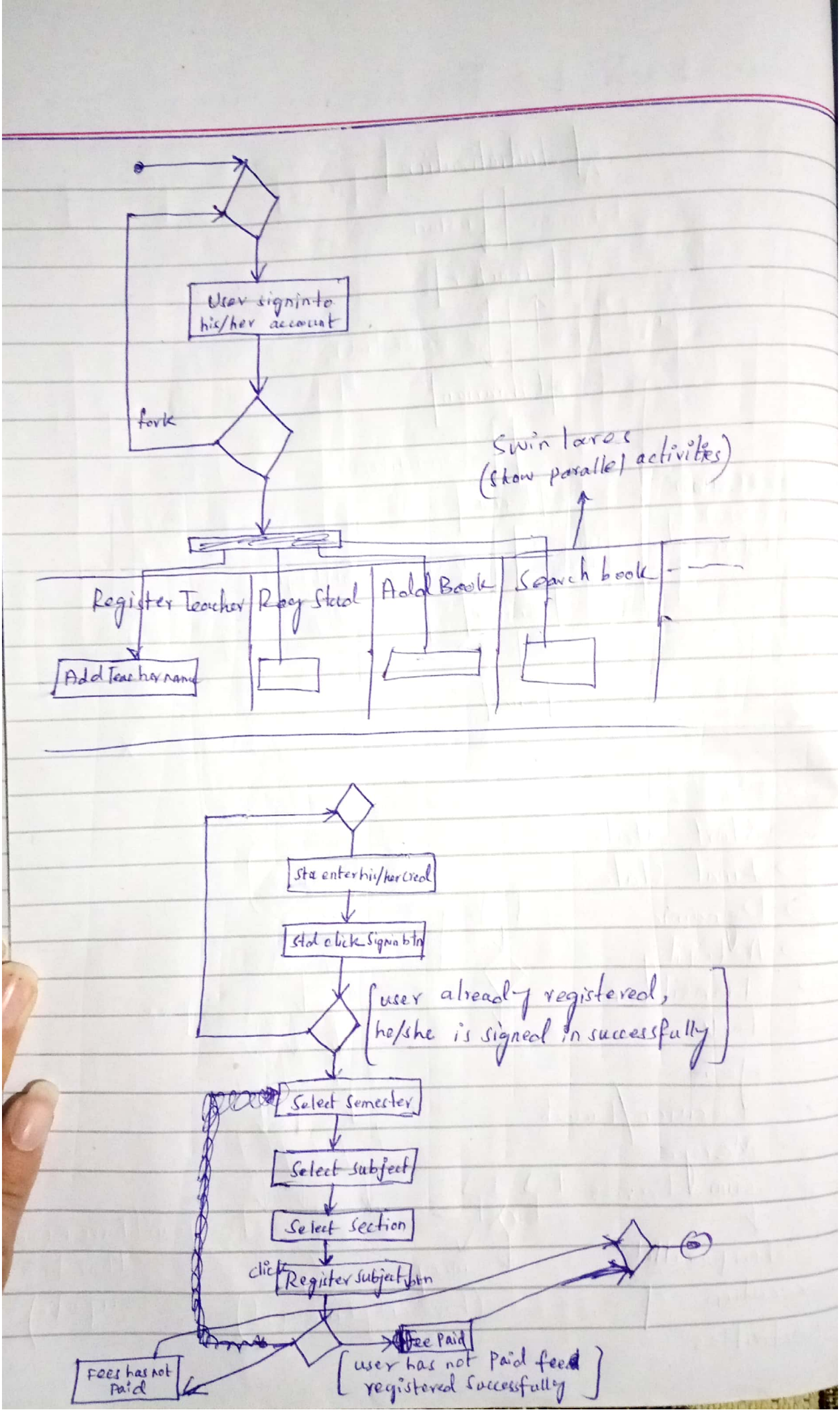




State Machine Diagram:-

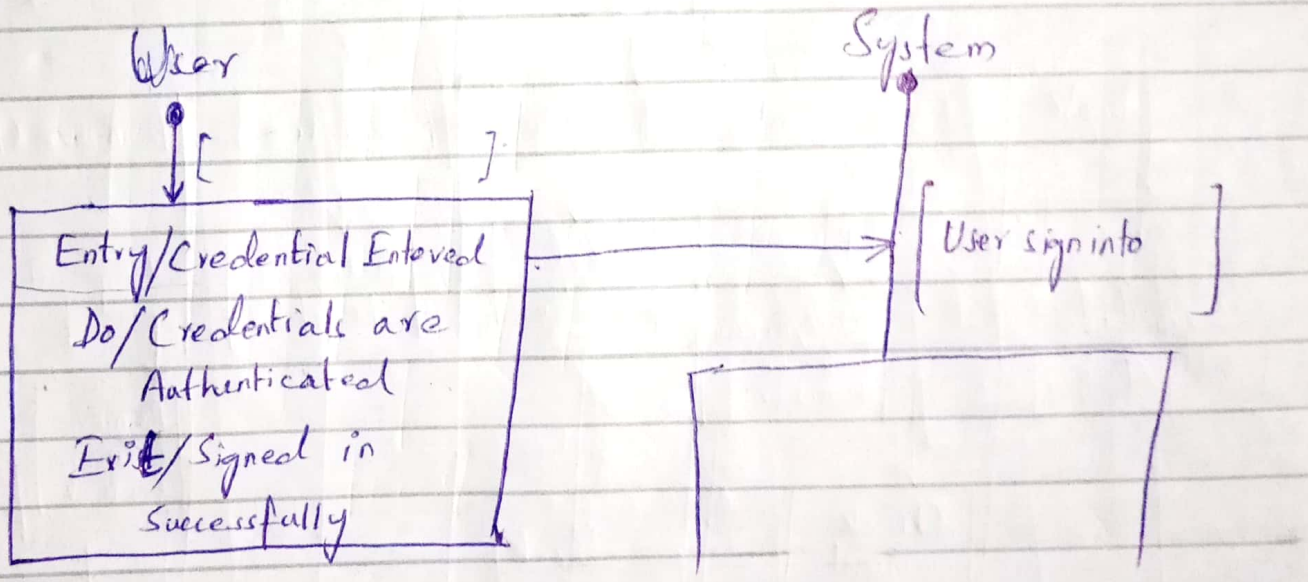
- → Start State
- → Final State
- → Dynamic
- → Activity
- → Transition
- → Fork
- → Join
- → Decision / Branch
- → Merge
- → Swim Lares
  - ↓
  - Show parallel execution of Activities





State Machine Diagram:-

→ Start State
→ Final State
→ State has
→ [a name / Do/Activity]
→ Entry/Activity
→ Exit/Activity

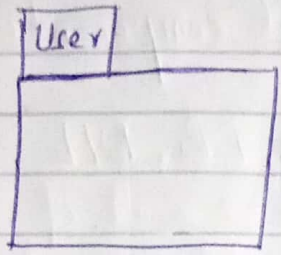


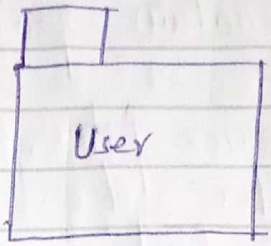
Total are 10 diagrams

Apache (website) server
mySQL (Database) server

Package Diagram:-

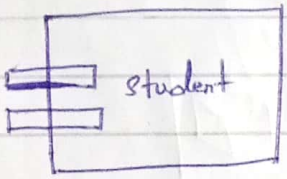
→ To simplify the view of the program for developers.



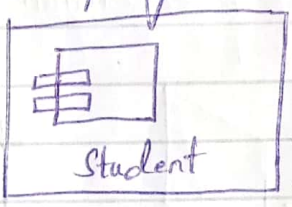


Component Diagram:-

→ Show structure of the program. (Show Software)

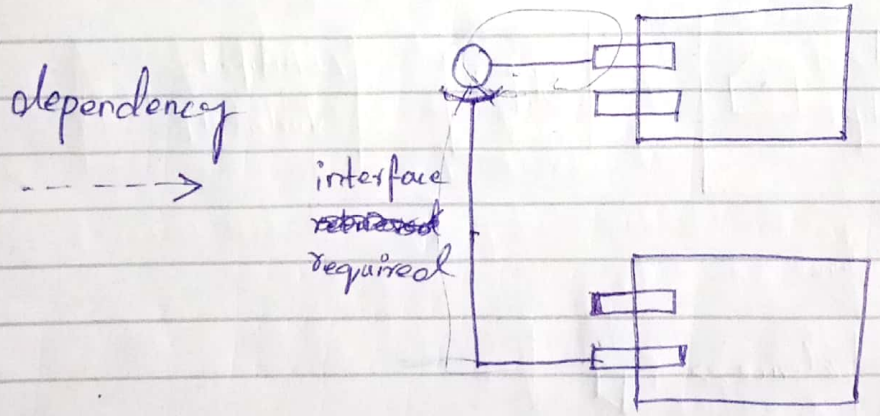


interface



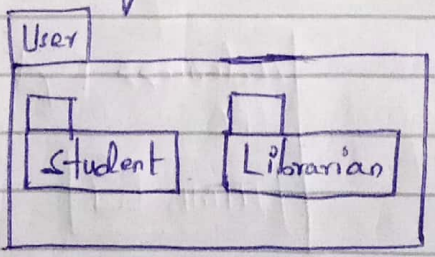
Interface provided

dependency



In package diagram:-

→ A package can contain other pakages.



→ Dependency (- - - - - - →)

Deployment Diagram :-

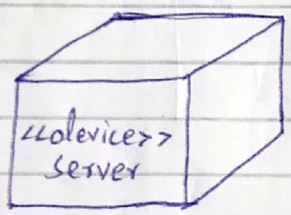
→ Physical architecture of hardware and software.

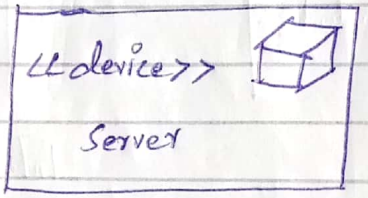
→ Can contain pakages and components.

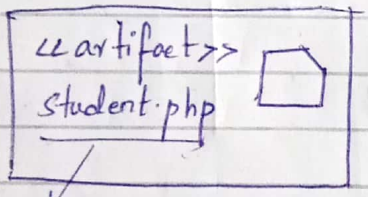
→ Artifact :-
Source files, Executables, DB, DLL, files etc.

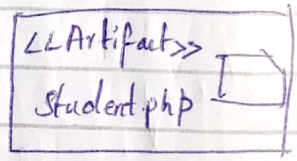
→ Artifact Instance :-
Object of the artifact.

→ Node :-
Hardware on which artifact deployed of execution environment.









To show instance