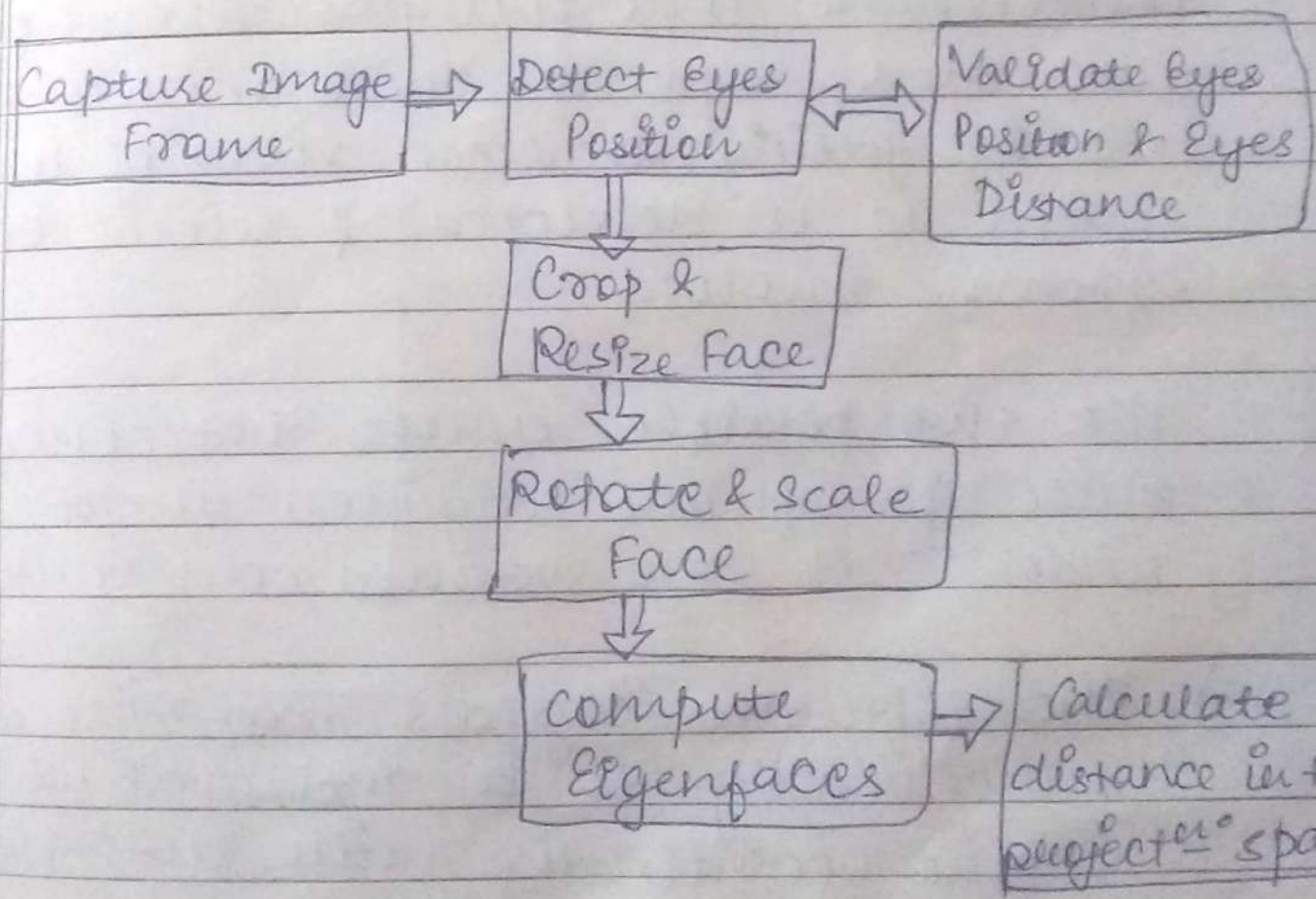
CHAPTER 4 : SYSTEM MODELLING

1). DFD for face Detection App System.



3). Explain why it is important to model the context of a system that is being developed. Give two examples of possible errors that could arise if software engineers do not understant the system context.

It is important to model the context of a system because context models show the environment that includes several other automated systems. These models will help in getting a clear view of the system to be developed & several other systems involved.

The two possible errors that could arise if software engineers do not understand the systems context are:

a). The software engineers may miss some functionality to be included which require coordinating with automated systems.

b). The system design & development may be deviated from the actual functionality as some requirements may not be known properly.

5). What is State Diagram & when is it used?

- A state diagram is a diagram which is used to represent the condition of the system or part of the system at finite instances of time.
- It is a behavioral diagram & it represents the behavior using finite state transitions.
- It shows how the system reacts to internal & external events.
- It shows system states & events that cause transitions from one state to another.

Uses :-

- We use it to state the events responsible for change in state (we do not show what processes cause those events).
- We use it to model the dynamic behavior of the system.
- To understand the working of obj/classes to internal or external stimuli.

6). Using the UML graphical notation for object classes, design the following object classes & identify attributes & operations using your experience.

- A telephone
- A printer for a personal computer
- A personal stereo system

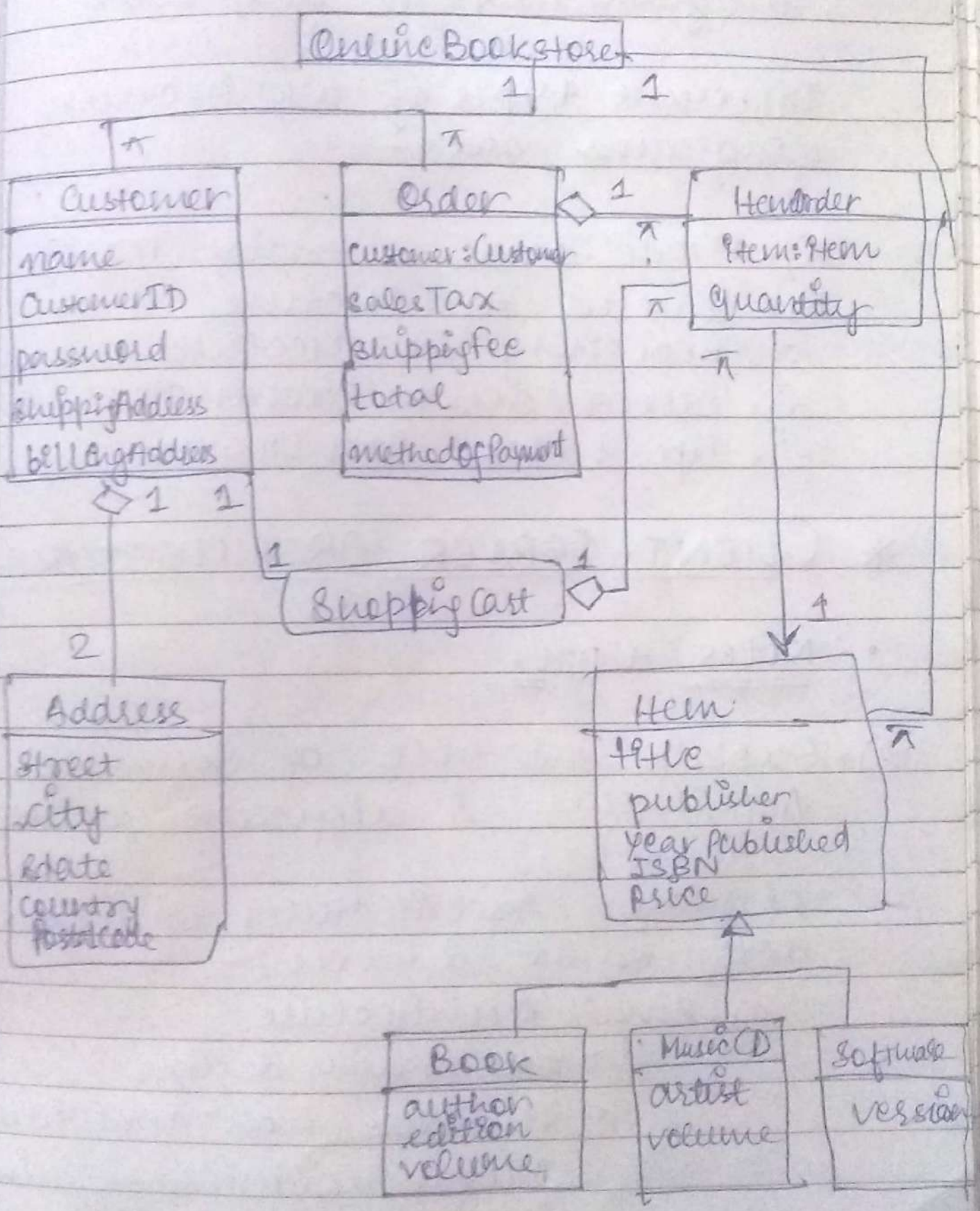
Telephone
number : integer serviceProvider : string keys : integer [0-9] specialKeys : {*, #, R}
dialCall() receiveCall() redialLastCall() autoAnswer() msgPlayback()

PC Printer
model : string type : {color, b&w} copies : integer status : {busy, free}
queueJob() printJob() fetchJob() cancelJob()

Stereo System
model : string type : {mini, big} radio : boolean cdSystem : {cd, MP3, cd/MP3} equalizer : {pop, rock, classic} track : integer currentStation : float
play() pause() rewind() forward() changeStation() changeEqualizer()

Impressions
Date
Page

Q1) Class diagram for scenario: Online bookstore sells books, CDs & software....



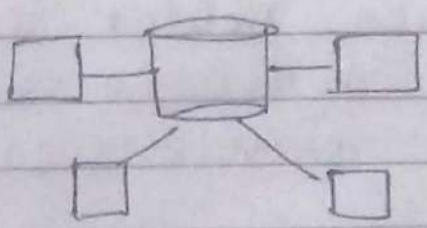
Chapter - 05(Lp)
Architectural Design

- Architectural design provides abstract view of the system by using different architectural models or the views & helps to provide a communicatⁿ betⁿ developer & client.

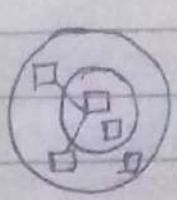
- This hides the internal details of the contents.

- It provides info about
  - set of components / modules.
  - structure
  - interactⁿ betⁿ components.

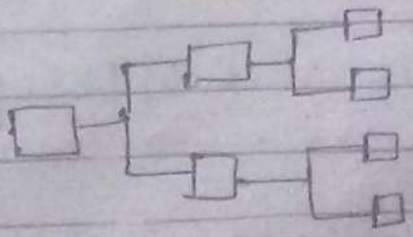
- Can be organised in different ways, such as :-
  - Data Centered



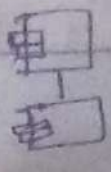
  - layered Architecture



  - Data flow



  - Component Org (Deployment Rep. Arch.



Quiz In data centered architecture, all components access information from database. True

* Levels of Architecture :-

(1) Small architecture
↳ program design
• how individual programs interact with each other.

(2) large architecture
↳ program design
↳ interaction with other system components
↳ modules of system components.

* Advantages of Architectural Design :-

(1) Stake holder communication
(2) It provides System Analysis → FR & NFR
(3) It provides large scale Reuse

* Video 2

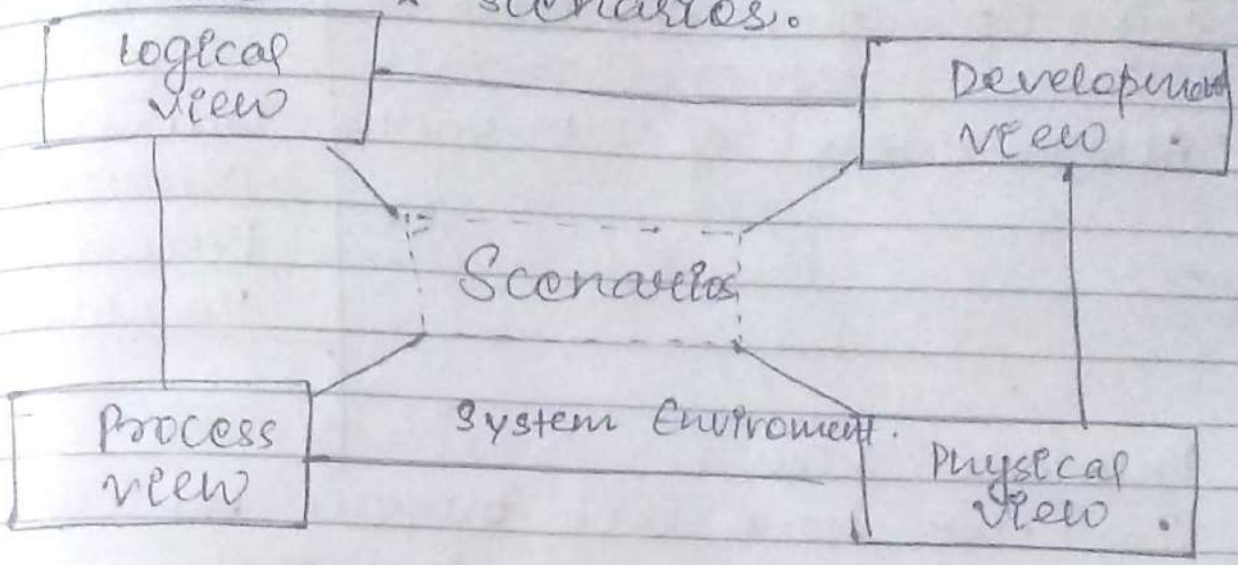
* ARCHITECTURAL VIEW

• what views/perspectives are useful for design? → multiple views.

• what notation? → simple block diagram
(• Box & connecting lines)
• UML notations. (OOP)

* 4+1 Architectural Views :-

↳ To design a system we have 4 basic model & 1 for use cases & scenarios.



* Logical View :-

It provides key abstractions in terms of objects & classes.

* Process View :-

It tells how at runtime different processes interact each other.

* Development View :-

It provides all details related to implementation & deployment of systems.

* Physical View :-

It provides all information regarding the mapping between Software & Hardware Components.

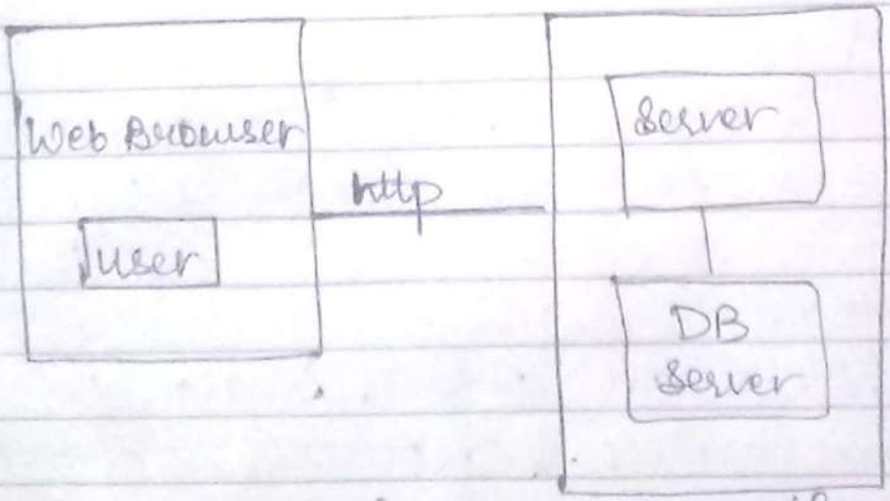
Examples:-

① Logical View:
E.g: Operating System (OS)

Abstract level of representation →

User Applications
I/O
Drivers
Memory
Processes
Hardware

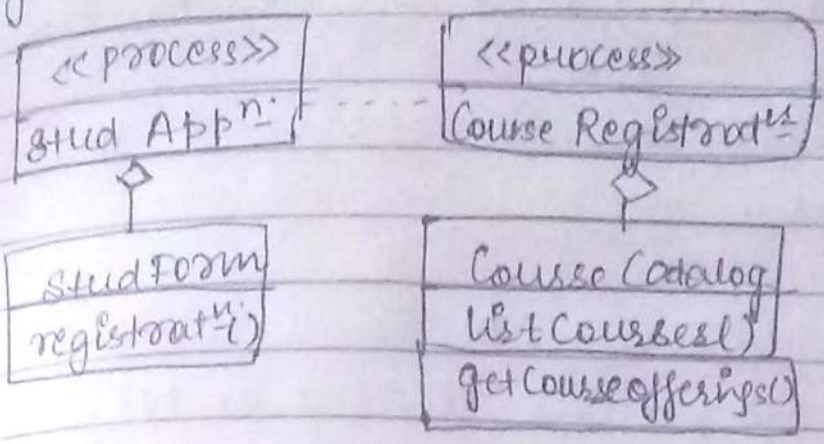
② Physical View:
E.g: • We have Web browser which allows user to browse.
• It can connect to the server.
• It checks to the availbility, connectivity.



• The protocol for connection b/w user & server may be http.
• The server can be of the type DataBase (DB) server which provides data security.

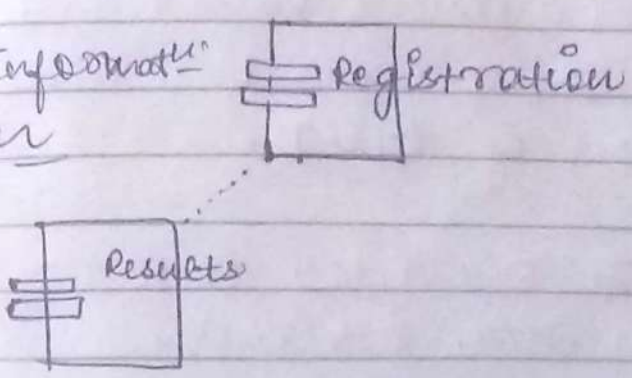
③ Process View:

E.g: Student Information System



④ Development View:

E.g:
Student Information System



* Video 3: Architectural Patterns

* Architectural patterns provide way of presenting, sharing, reusing & documenting system components.

* Different patterns: used for :-

① Model-View Controller } separate the
② Layered Architecture } fuctionalities
③ Repository Architecture → Data sharing among components
④ Client-Server Architecture → Data shared at diff loc
⑤ Pipe-filter Architecture → Data processing applications.
↳ provides implementation details also.

* Video4: Model-Viewed Architecture (MVC)

MODEL- VIEW CONTROLLER ARCHITECTURE

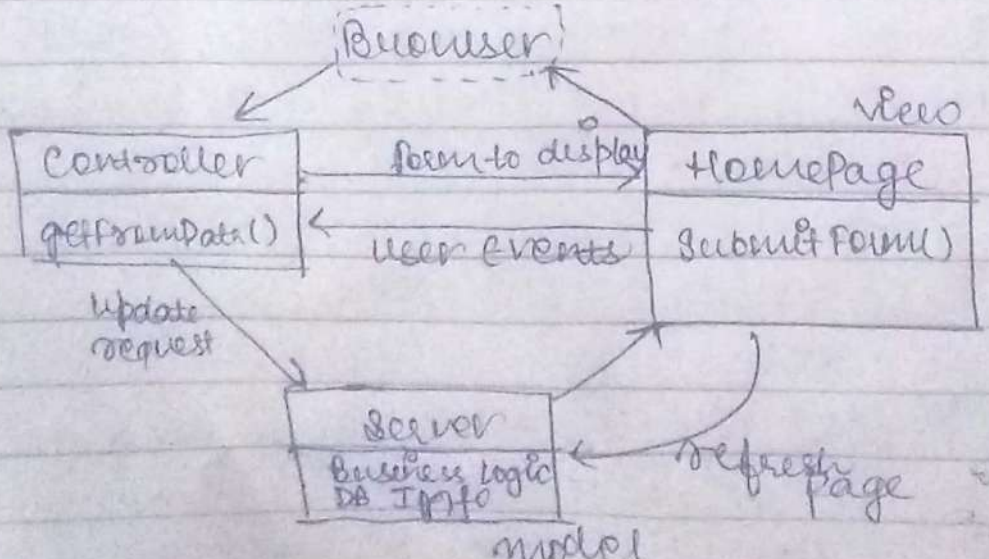
3 stages/components to represent the architecture.

① Model :- It provides business logic & operations related to DB.

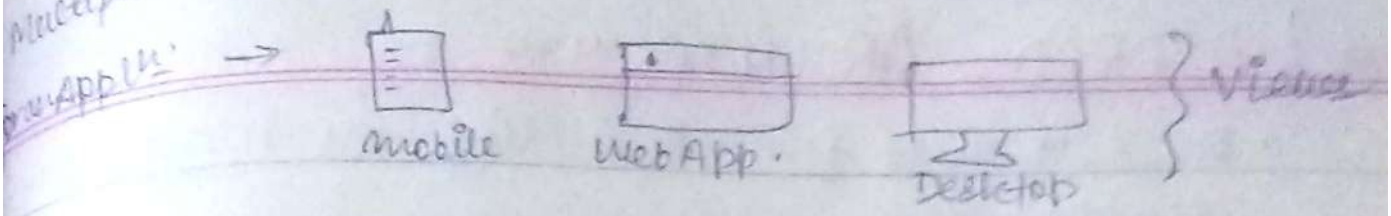
② View :- It provides User Interaction views (different views) ∴ mainly concerned with different types of GUI.

③ Controller :- It provides interaction b/w model & views.

E.g: WEB APPLICATION for student information



Multiple views



Controller Applicat^n.

Business logic
DB

} server

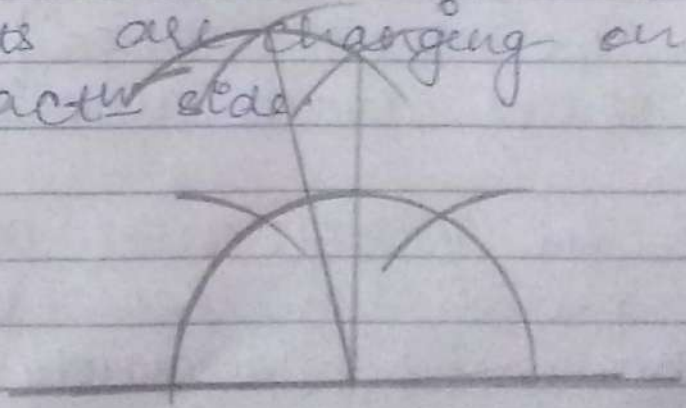
*Adv :-

① MVC is advantageous when we have multiple views to be considered for a web application.
↳ or M or C

② Web logic can be modified Independently without affecting the other components.

*When to use this MVC ?

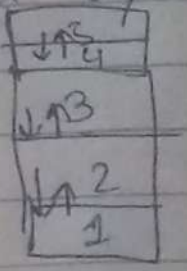
• Used when we have mult. views to be considered & also when the requirements are changing on the user Interact^n side.



* Video 5 : (Layered Architecture)

* LAYERED - ARCHITECTURE :-

- It seperates functionalities as independent components.
- Each of the functionalities are represented using different layers.
- Each layer relies (can interact) on facilities / services provided by layer below it.



- There is an interface b/w each of the layer but the services can be provided by any of the layer.

* Adv :-

- Any layer can be implemented independently & is changable without affecting the layer beside it.
- However, when the interface changes then we need to modify the subsequent layers accordingly.

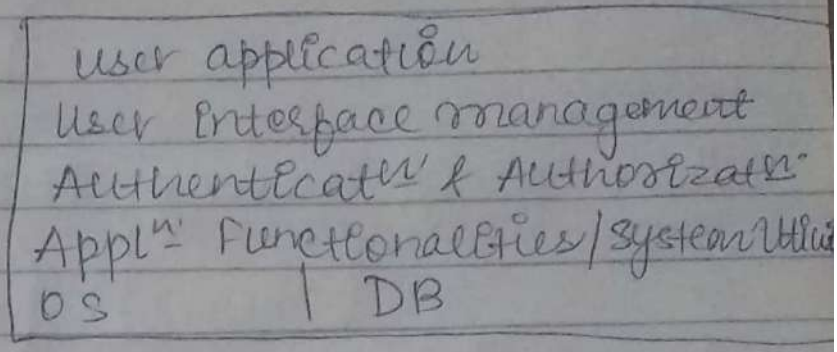
* Uses :

- Used for incremental development of project i.e. each layer can be incremented separately & modified as per the user feedback.

• Used when new functionalities need to be developed on existing one.

• Development process is distributed.

E.g OS



* Disadv :-
• machine dependent layers are not independent.

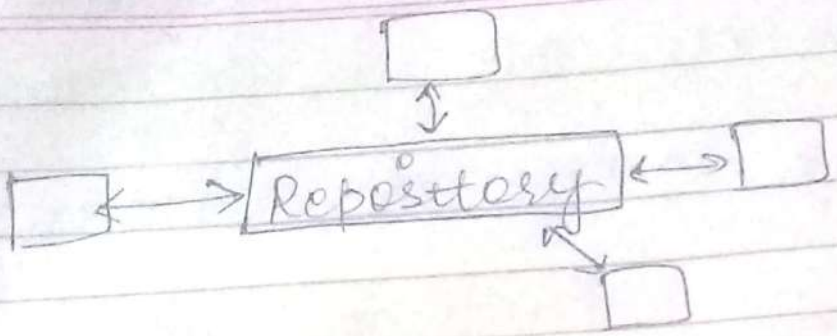
* Video 6 : (Other Architectures)

* Repository Architecture :-

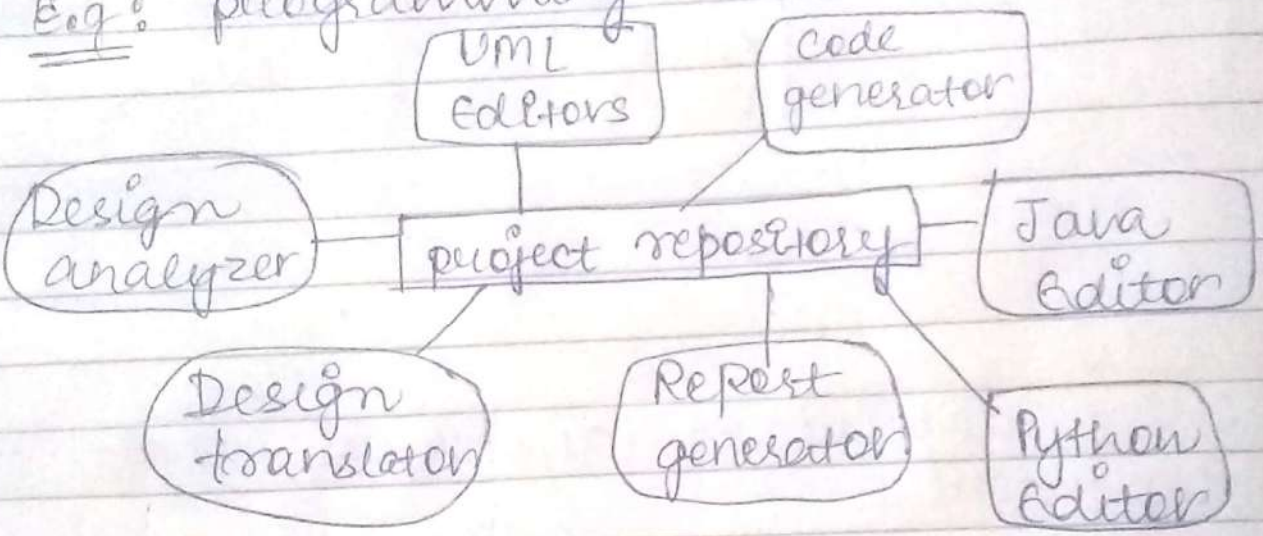
• It is used when data generated is volumetrically large & need to be shared among components.

• Used for data driven applications.

• Data is stored centrally.



Eg: programming IDE



Disadv :- single point failures for services.

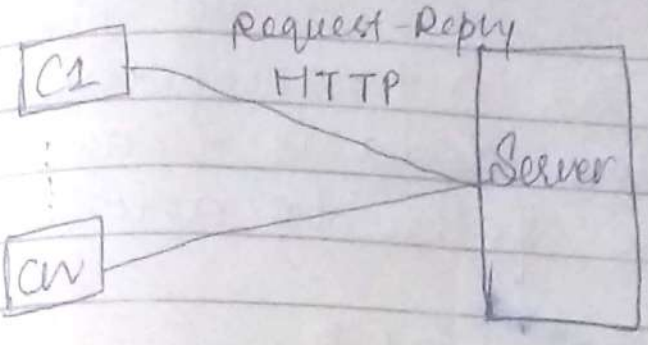
* Client Server Architecture :-

Functionalities can be accessed through client server request.

This architectural pattern is used when
- data is shared from different locations
- functionalities are " " " ".

• Here, we have server that stores business logic & data/services.

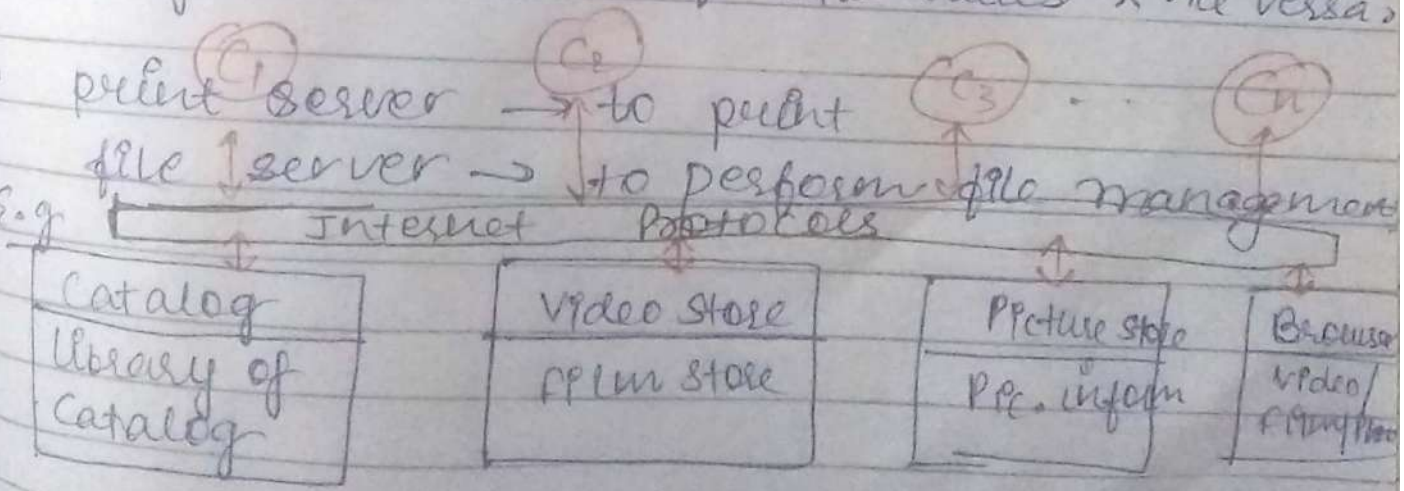
• clients/users access the service.



• clients communicate with the server using different Request-Reply protocols, which provides comm. b/w client & server.

• Here, we can use distributed servers which provide different functionalities. It makes the system safe & secure when one of the system fails.

• Server programs can be changed irrespective of the client functionalities & vice versa.



* Data Processing Applications :-

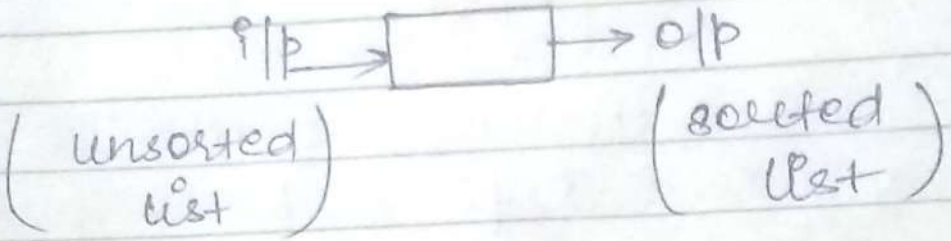
1). Runtime processing of data

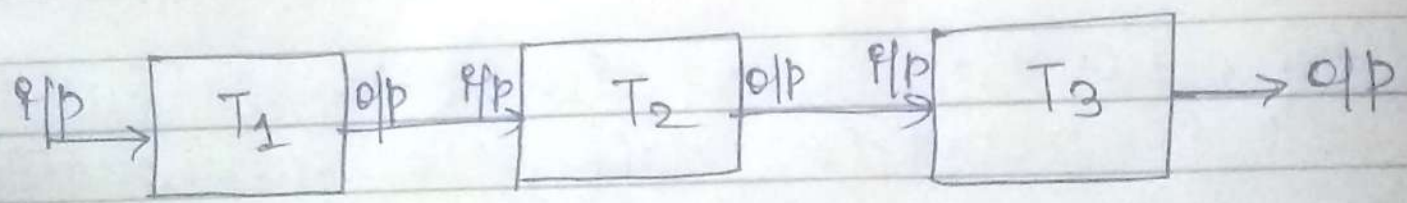
PIPE & FILTER

- Dataflow → pipe
- Data processing unit → transform or filter

⇓

We call it as a transform because the data is transmitted from one form to another.





- Each box represents abstract definition for the task to be performed.

CHAPTER-5
ARCHITECTURAL DESIGN

1). List different types of architecture diagrams. Explain any one.

Different types of architecture diagrams are :-

a). Model-View-Controller (MVC)
b). Layered Architecture
c). Repository Architecture
d). Client-Server Architecture
e). Pipe & Filter Architecture

Client-server architecture

- Distributed system model which shows how data and processing is distributed across a range of components.
  - Can be implemented on a single computer.
- Set of stand-alone servers which provide specific services such as printing, data management, etc.
- Set of clients which call on these services.
- Network which allows clients to access servers.

The Client–server pattern

Name	Client-server
Description	In a client–server architecture, the functionality of the system is organized into services, with each service delivered from a separate server. Clients are users of these services and access servers to make use of them.
Example	Figure 6.11 is an example of a film and video/DVD library organized as a client–server system.
When used	Used when data in a shared database has to be accessed from a range of locations. Because servers can be replicated, may also be used when the load on a system is variable.
Advantages	The principal advantage of this model is that servers can be distributed across a network. General functionality (e.g., a printing service) can be available to all clients and does not need to be implemented by all services.
Disadvantages	Each service is a single point of failure so susceptible to denial of service attacks or server failure. Performance may be unpredictable because it depends on the network as well as the system. May be management problems if servers are owned by different organizations.

2). Explain the role of Software Architecture & different views.

Software Architectures can be designed at 2 levels :-

a). Small Architecture

↳ program design

- It tells how individual programs interact with each other.

b). Large Architecture
- ↳ program design.
- ↳ interactⁿ with other system components
- ↳ modules of system components.

* Role of Software Architecture :-

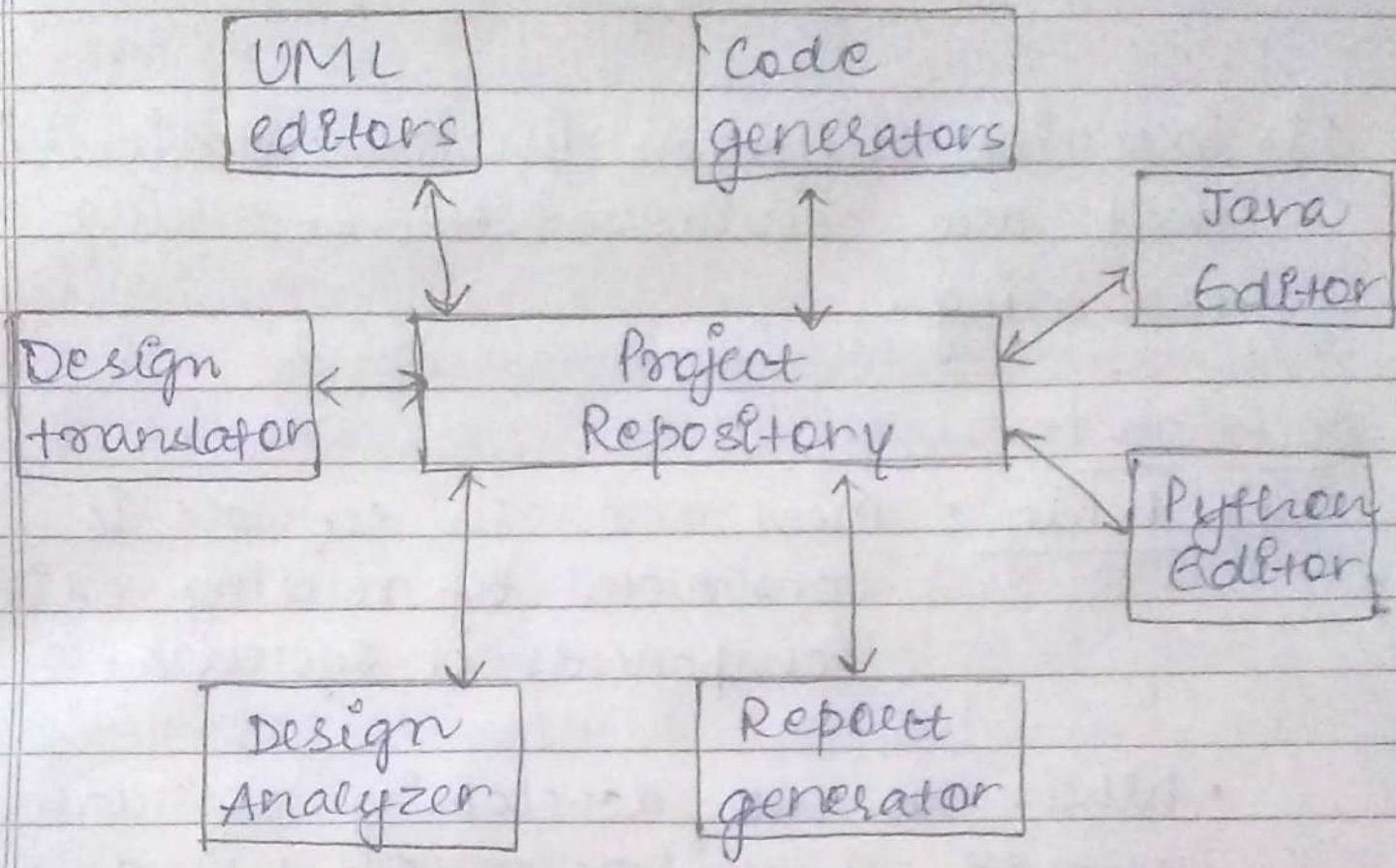
- It provides an abstraction to manage the system complexity & establish a communication & coordination mechanism among components.
- It defines a structure solⁿ to meet all technical & operational requirements.
- It involves a set of significant decision about the organization related to software development.
- These decisions may comprise of :-
  - Selectⁿ of structural elements & their interfaces by which system is composed
  - Behavior
  - Architectural decisions aligned with business objectives.
  - Architectural styles guide the organization.

3). Explain Repository architecture with suitable ~~of~~ example.

- This architecture is used when data generated is voluminous & need to be shared among components.
- It is used for data driven application.
- In this architecture, the sub-systems must exchange data. This may be done in two ways:-
  - Shared data is held in a central database or repository & may be accessed by all sub-systems.
  - Each sub-system maintains its own database & passes data explicitly to other sub-systems.
- When large amounts of data are to be shared, the repository model of sharing is the most commonly used architecture as it has an efficient data sharing mechanism.

Impressions
Date
Page

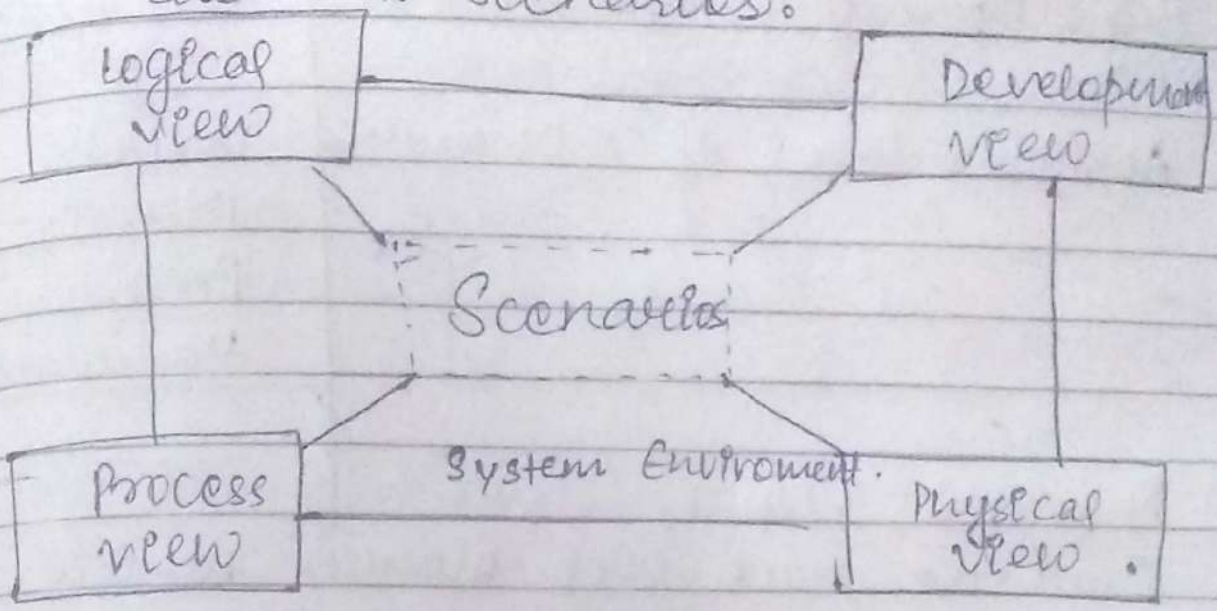
E.g: repository architecture for an IDE:-



4). With a suitable diagram, explain different views for architecture diagrams.

4+1 Architectural Views :-

↳ To design a system we have 4 basic model & 1 for use cases & scenarios.



* Logical View :-
It provides key abstractions in terms of objects & classes.

* Process View :-
It tells how at runtime different processes interact each other.

* Development View :-
It provides all details related to implementation & deployment of systems.

* Physical view :-
It provides all information regarding the com: between Software & Hardware Components.

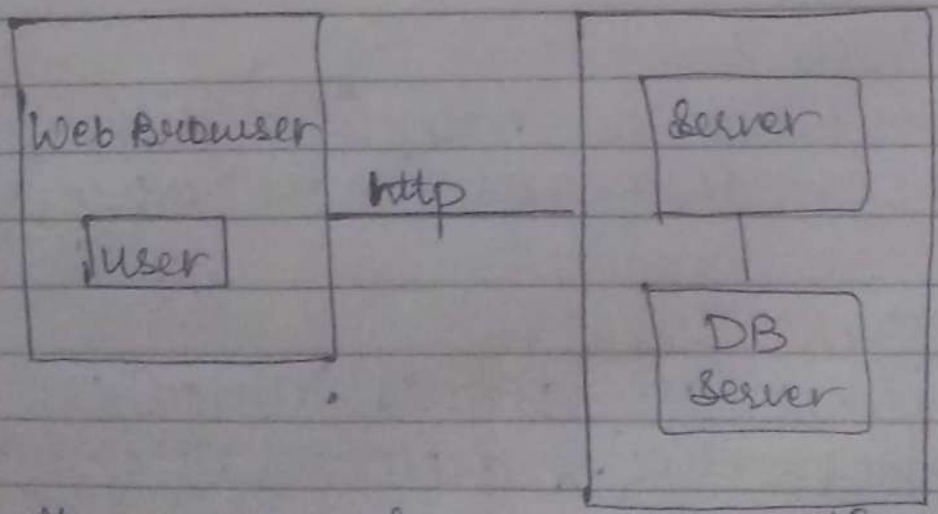
Examples :-

① Logical View :
E.g : Operating System (OS)

Abstract level of representation →

User Applications
I/O
Drivers
Memory
Processes
Hardware

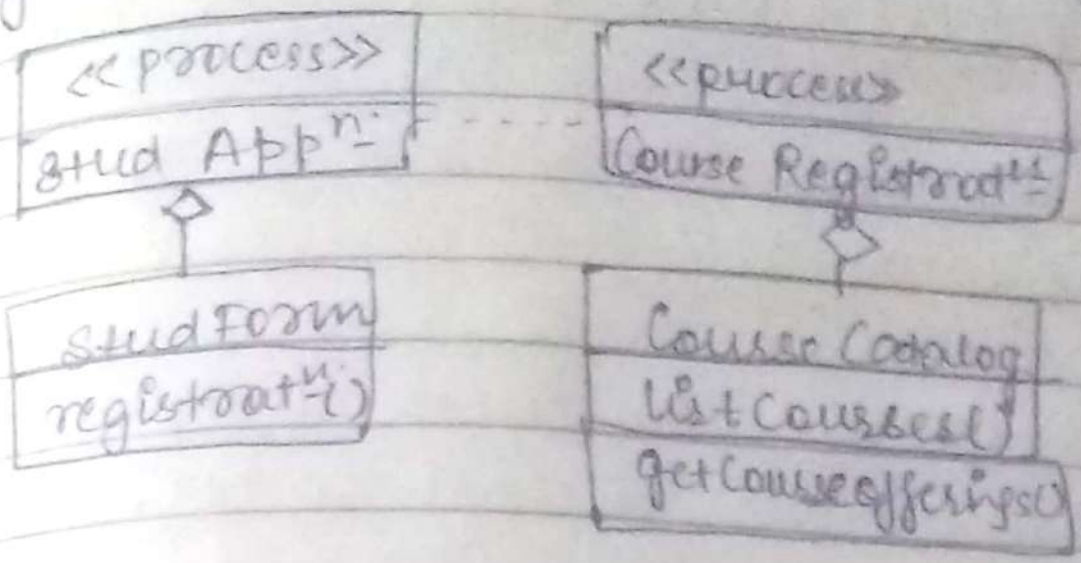
② Physical View :
E.g : • We have Web browser which allows user to browse.
• It can connect to the server.
• It checks to the availability, connectivity.



• The protocol for connection b/w user & server may be http.
• The server can be of the type DataBase (DB) server which provides data security.

(3) Process View :

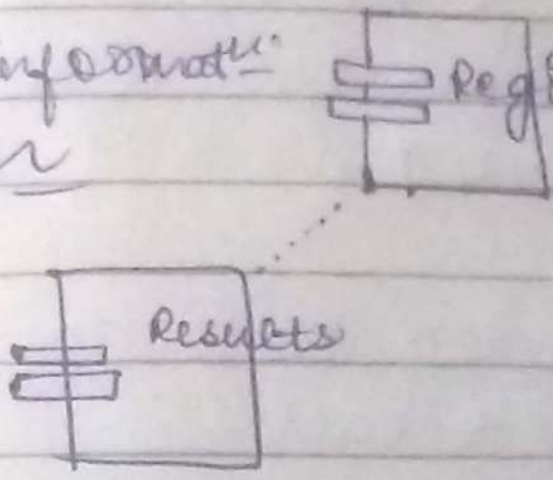
E.g : Student Information System



(4) Development View :

E.g :

Student Information System



Chapter 6
Object Oriented Design & Implementation

1). Describe different implementation issues that are considered ~~usin~~ during software engineering.

Solution :-

1). Reuse : Most modern software is constructed by reusing existing components or systems.

- When we are developing software, we should make as much use as possible of existing code.

2). Configuration Management :

- During the development process, many different versions of each software component are created.

- If we don't keep track of these versions in a configuration management system, we are liable to include the wrong versions of these components in our system.

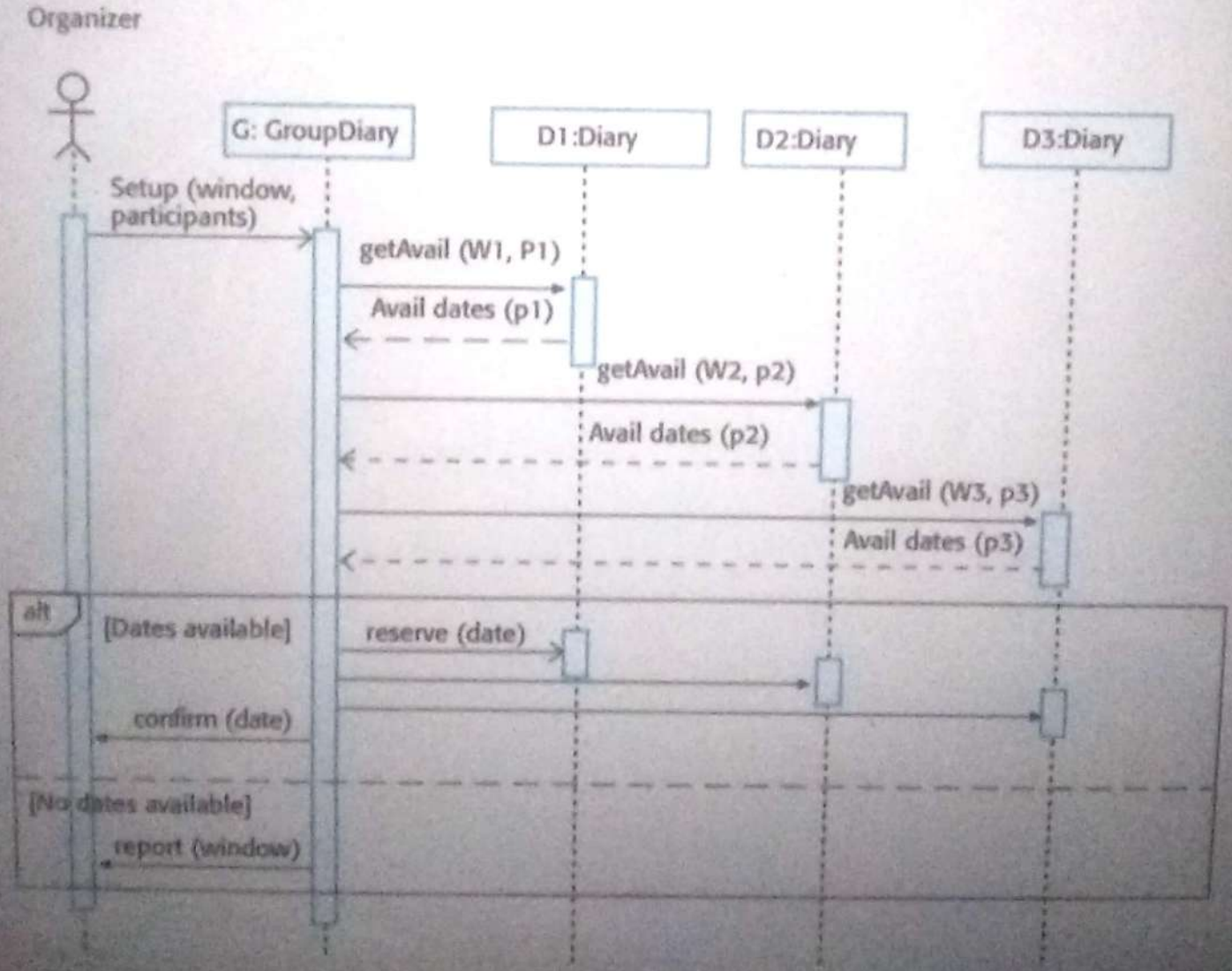
3). Host Target development:

- Production software does not usually execute on the same computer as the software development environment.
- Rather, ~~you~~ we develop it on one computer and execute on a separate comp.
- The host & target systems are sometimes of the same type but, often they are completely different.

2). Draw a sequence diagram showing the interactions of objects in a group diary system when a group of people are arranging a meeting.

Assumes there are 3 participants in the meeting, one of whom is the meeting organizer. The organizer suggests a 'window' in which the meeting should take place and the participants involved. The group diary communicates with the diaries of the participants in turn, modifying the window accordingly as there availability is known. So, if the organizer suggests a window of 18th-19th June, the group diary consults the organizer's diary (D1) and finds availability on these days. D2 is then contacted with that availability, not the original window. If there are no mutually available dates in the window, the system reports this to the organizer. Otherwise, a date is selected, entered in all diaries and confirmed to the organizer.

Ans.



3). Describe different ways of identifying objects of an object oriented application.

There are various ways about how to identify object classes in object-oriented systems:-

a). Use a grammatical analysis of a natural language description of the system to be constructed.
- Objects & attributes are nouns.

b). Use tangible entities (things) in the application domain such as aircraft, roles such as manager or doctor.

c). Use a scenario-based analysis where various scenarios of system use are ~~identification~~ identified & analyzed in turn.

- As each scenario is analyzed, the team responsible for the analysis must identify the required objects.