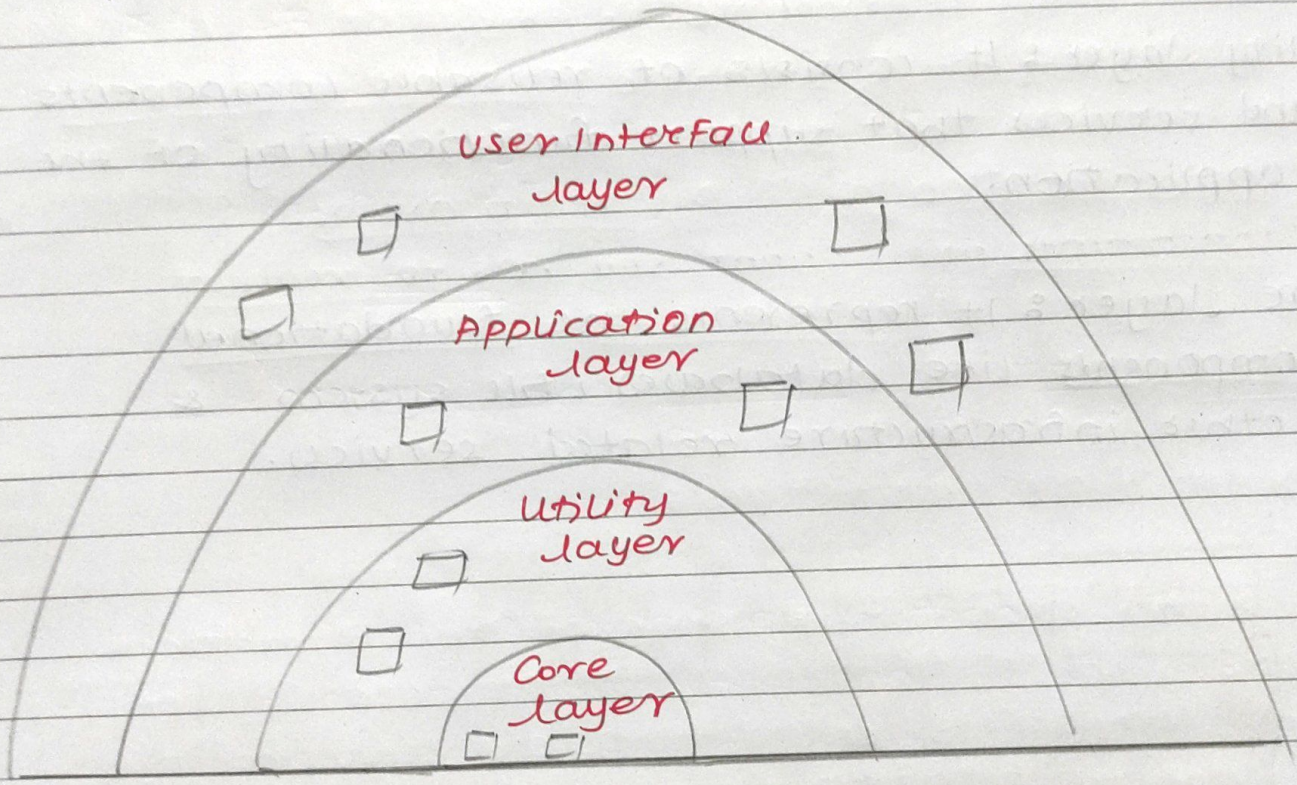
UNIT - IV

* Explain layered system architecture with neat diagram.

→

- Layered architecture is composed of different layers. Each layer is intended to perform specific operations so machine instructions set can be generated. Various components in each layer perform specific operations.



- The Outer layer is responsible for performing the User Interface operation while the components in the inner layer perform Operating system interfaces.

- The components in intermediate layer perform utility services and application software functions.

- User Interface layer: This is outermost layer of the software architecture, responsible for presenting info to user and receiving input.

- Application layer: It contains the core business logic and functionalities of the s/w. It encapsulates rules & algorithms that govern the behavior of the system.

- Utility layer: It consists of reusable components and services that support functionality of the application.

- Core layer: It represents the foundational components like databases, file system & other infrastructure related services.

* Explain the following architectural styles with merits/demerits :

(i) Data-centered architectures :

- In this architecture the data store lies at the centre of the architecture and other components frequently access it by performing add, delete and modifying operations. The client s/w requests for the data to central repository. Sometime the client s/w accesses the data from the central repository without any change in data or without any change in actions of s/w actions.

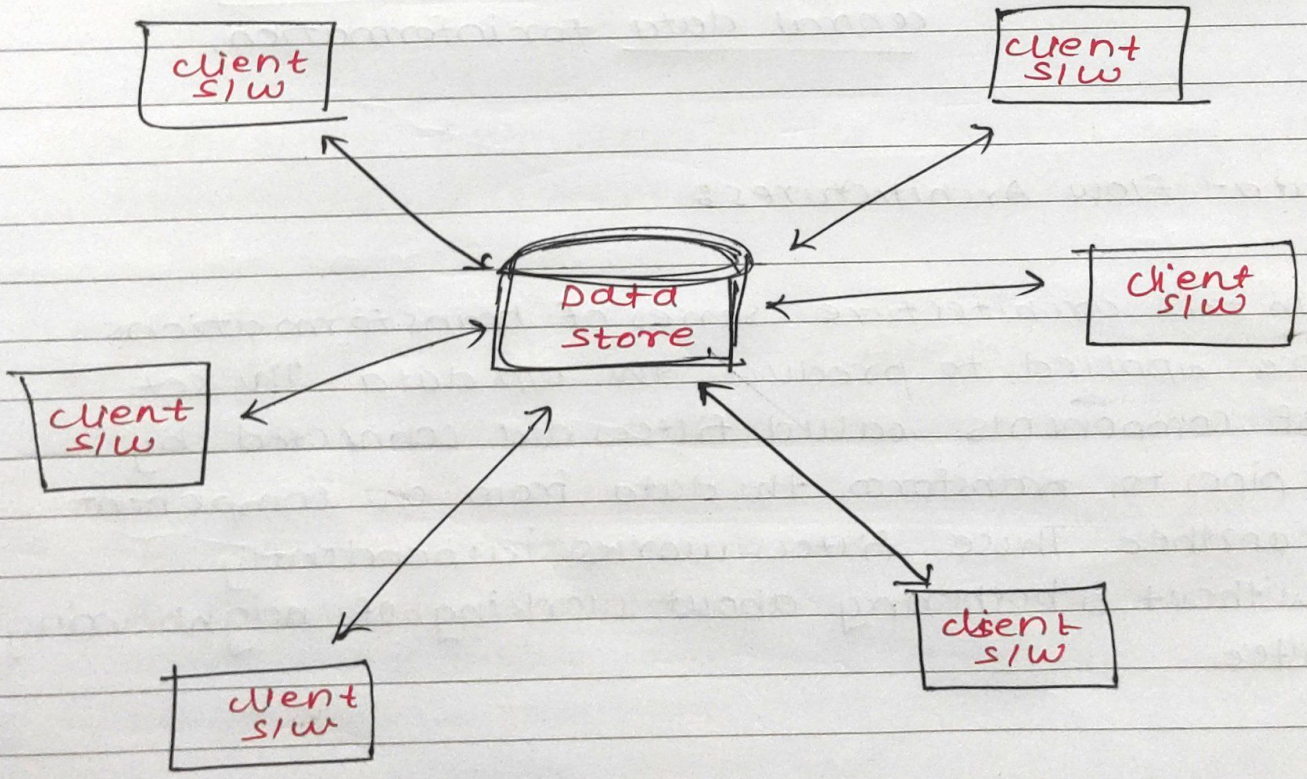


fig. Data centered architecture

- Data centered architecture posses the properety of interchangeability. It means any component from the architecture can be replaced by a new component without affecting the working of other components

- In data centered architecture data can be passed among the components. In this,
  Components are : Database elements such as tables, queries
  Communication are : by relationships
  Constraints are : Client s/w has to request central data for information.

(ii) Data-flow Architectures:

⇒

- In this architecture series of transformations are applied to produce the o/p data. The set of components called filters are connected by pipes to transform the data from one component to another. These filters works idependently without a bothering about working of neighbouring filter

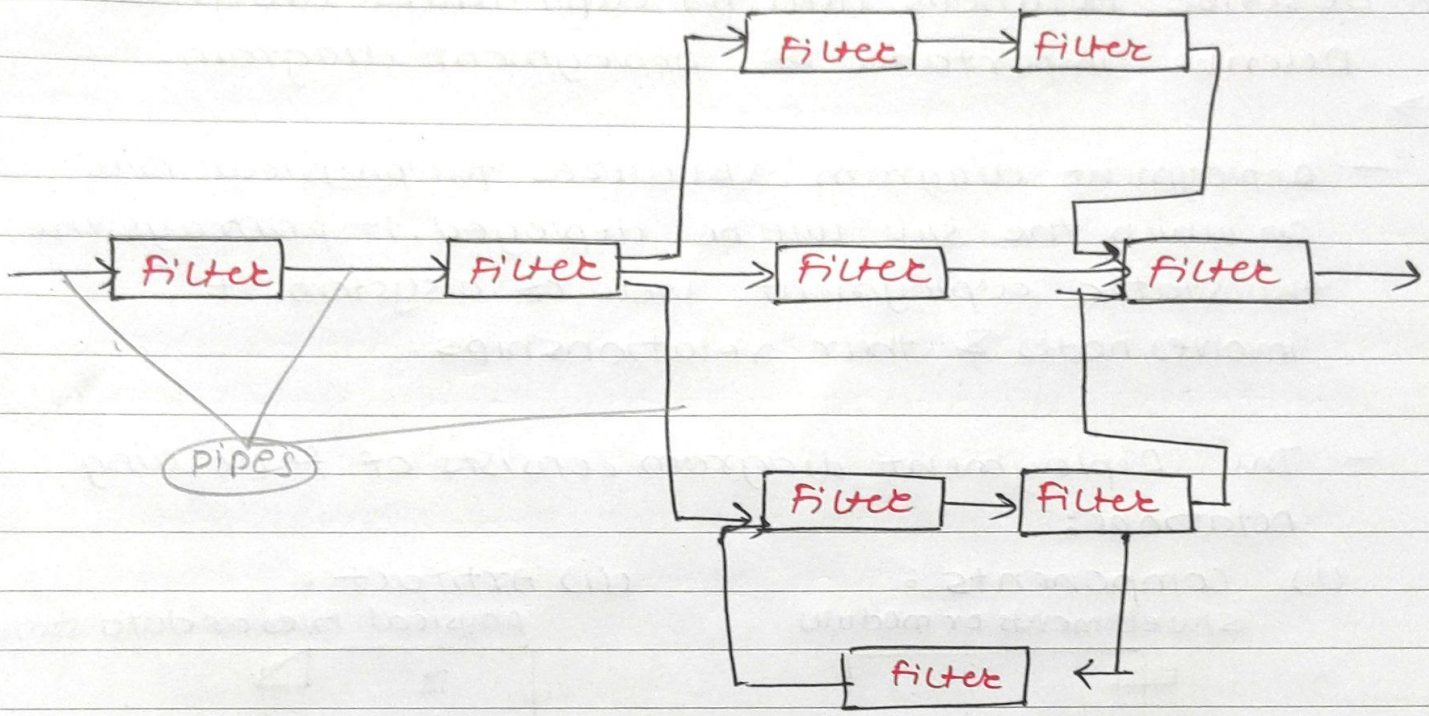


fig. pipes & filters

If the data flow degenerates into a single line of transforms, it is term as batch sequential.

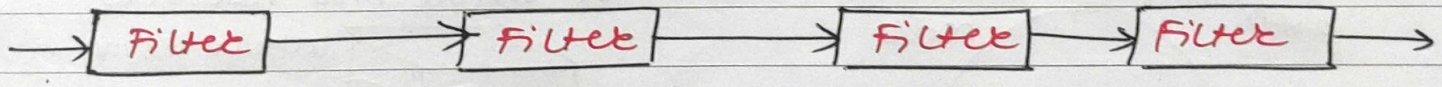


Fig. Batch sequential

In this pattern the transformation is applied on the batch of data.

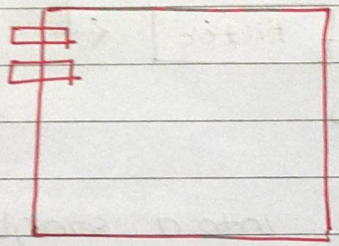
* Describe notations used for deployment diagram
Describe importance of Deployment diagram.

→

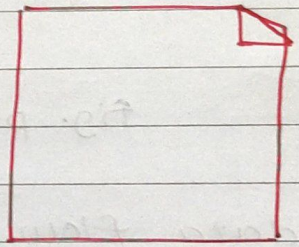
- Deployment diagram visualizes the physical h/w on which the s/w will be deployed. It portrays the the static deployment view of a system. It involves nodes & their relationships.

- The Deployment diagram consists of following notations:

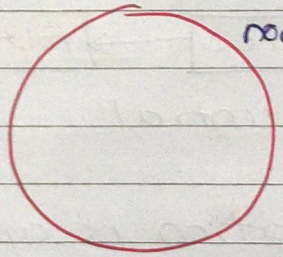
(i) components: s/w elements or modules



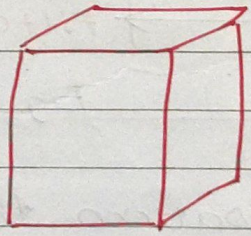
(ii) Artifact: physical files or data stores



(iii) Interface: communication channel b/t nodes



(iv) Node: represents physical or virtual h/w resources where comp. are deployed.



* Importance of deployment diagram :

- Visualization of Deployment architecture :-
  They offer a visual representation of how s/w components are deployed across h/w nodes or servers.

- Communication Tool :
  It serves as a means of communication among developers, architects & stakeholders. They help convey the deployment strategy.

- Resource planning and Optimization :
  Deployment diagram gives a brief overview of how resource planning is done.

- Deployment automation :
  It serves as a blueprint for automating deployment processes. They provide necessary information to configure deployment scripts, and containerization platforms, etc.

* Explain Data-flow architecture and Layered Architecture in details.

* Data flow Architecture.
- Data flow architecture focuses on the flow of data through a system.
- It is centered around the flow of data through a system and the transformations that occur to that data as it moves from input to output.
- It tell's how data moves between processing components, known as filters or transforms connected by pipes
- This architecture is often used in system that process data stream.

* components :
1. Filter's :- These are independant component's that process input data and produce output data. Each filter performs a specific transformation or or computation on the data.

2. piepes :- These are connecters that pass data from one filter to another. They serve as channels through which data flows.

* characteristics :-

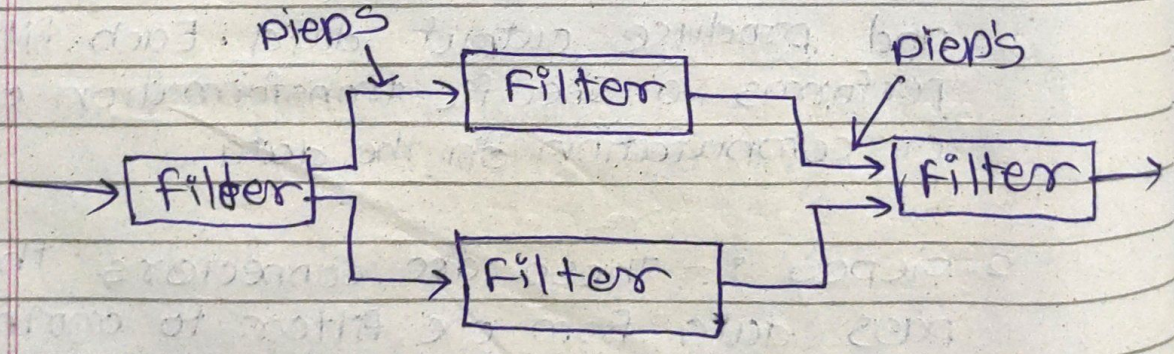
1. Modularity :
   Each filter is a self contained module, making the system to understand develop and maintain.

2. Reusability :
   Filter's can be reused accross different system if the data format remain's consistent.

3. concurrency.
   - filter's can often run in parallel

4. flexibility.
   - It is easy to add, remove and replace filters without affecting the overall system.

* Diagram :-

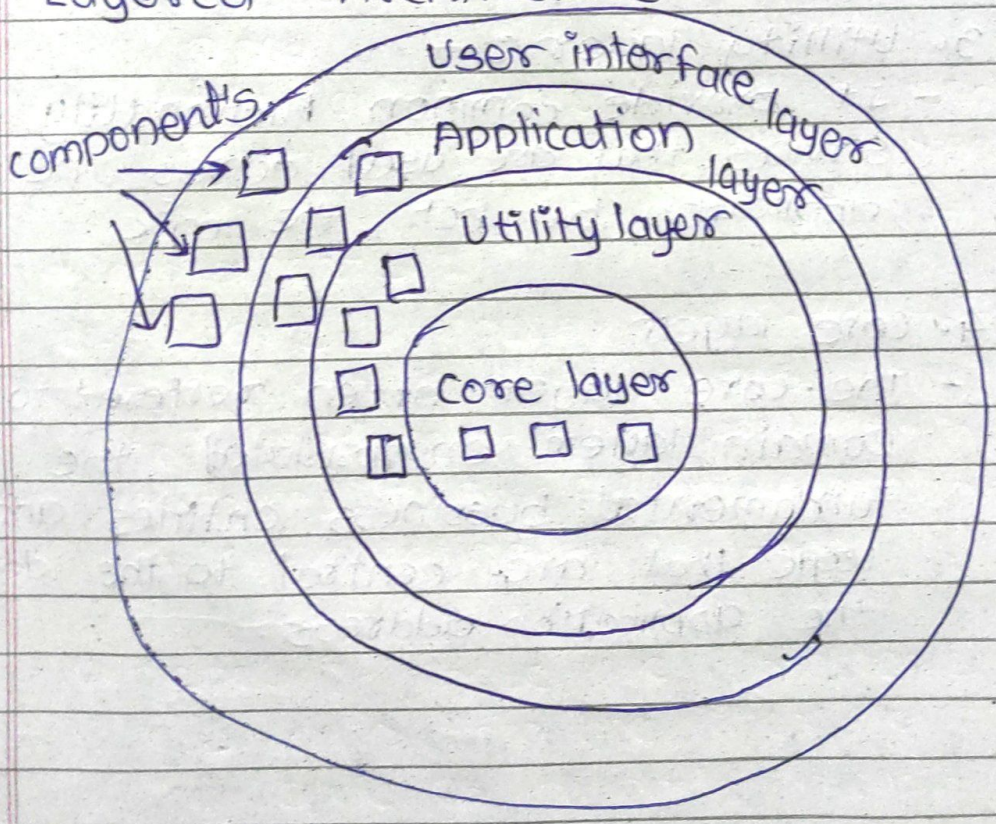


- A pipe and fiter pattern has a set of components called filter's connected by pipes that transmit data from one component to the next.

Example system's :-

1. unix piepes and filters
   In which command (filter's) are connected with lines.

* Layered Architecture.



- Layered Architecture is a design pattern that organize the system into hierchical layer's each with specefic role and responsibility.

1. User interface layer :-
- The (UI) layer is responsible for presenting information to the user and handling user interaction
- It's a layer that user directly interacts with.

2. Application layer :-
- The Application layer sometimes reffered to as the services layer or Bussiness logic layer contain's the core fuctionality and bussiness rule's of the application.

3. Utillity layer :-
- It provide common fuctionality and services that are used across the application but not the logic

4. Core layer
- the core layer often reffered to the Domain layer encapsulated the fundamental bussiness entities and logic that are central to the domain the application address.

Page No.
Date

* Explain golden rule's for User Interface Design
- User interface (UI) design is craial for ensuring that user's can interact with a software application effectively and efficiently.

1. consistency
- Use uniform element's across the interface such as font's, colors, button's and layout's.

2. Feedback
- provide immediate and clear feedback for user actions.

3. Visibility
- Keep user's informed about what is going on through appropriate timely feedback.

4. User control and freedom.
- Undo and Redo
Allow user's to easily revers's their actions to correct mistakes.

5. error prevention
Design interface that minimize the change of user error's.

6. Help
- provide Help for better under-standing.

* what question needs to be answered in order to develop a project plan according to W^5HH principle. ?

The W^5HH project Managment framework that help in planning and managing software engineering project by addressing key queostions.

1. why is the system being developed ?
   - understand the goal, Aim and puspose of project. (puspose :-)

2. what will be done ?
   - Requirement :- Define the scope of the project the feature and functionalities that need to be developed.

3. when will it be done ?
   - schedule. :- Establish a timeline for the project, including milestones deadline and delivery dates.

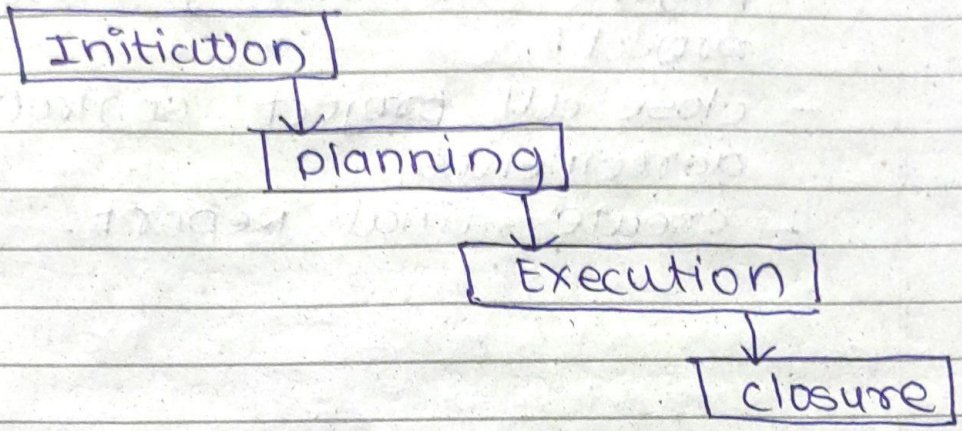
4. who is responsible for a fuction ?
   - Responsibillities :- identify the project, stakeholders, team members and their roles and responsibility

5. where are they organizationally located ?
   - Locations :- Determine the place where team is located.

6. How will the job be done technically and managerially.
   - Approach : define methodologies, technologies tools and processes that will be used in project.

7. How much of each resource is needed
   - Resource : Estimate the resource required including budget, person's, equipment and time.

* Explain the project managment life cycle.



1. initiation
   - First you need to identify a business need, problem or opportunity.
   - Define the project at a high level and gain approval to proceed.

2. planning
   - Develop a detailed project plan ~~that~~ outlines the scope, schedule cost, quality comunnication risk, and resource.
   - plan for project communication and user managment.

3. Execution.
   - complete the work defined in the project plan to meet project objectives.
   - Execute tasks according to the project plan.

4. closing
- finalize all activities deliver the project and formally close the project.
- close all project contracts and agreements.
- create final report.

Unit IV :-

① What is software architecture? Why architecture is important? What is use of Architecture decision description template?

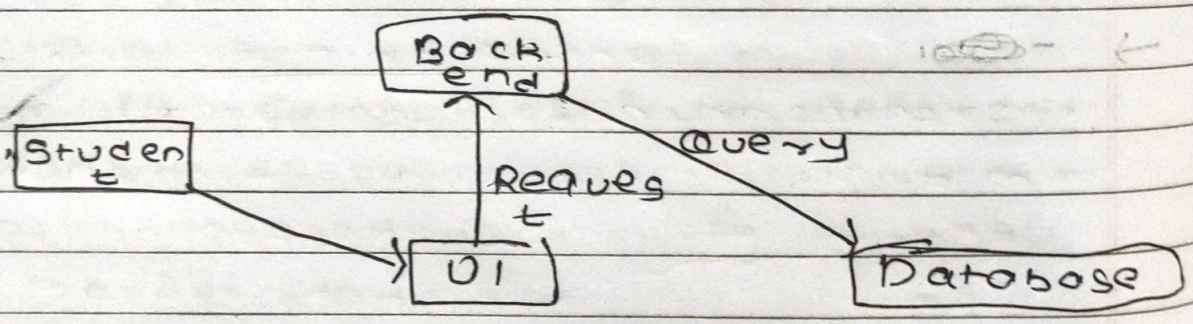
→

*Architecture :

- It is a structure of system which consist of different components external visible properties of these component and inter-relationship among these component

For example :

Architecture of attendence Application



* Why architecture is imp?

- ① It give clear idea about the system need to be built
② Some early decision is taken due to which it increase performance and operation remains under control.

* ADD (Architecture Design Decision) Description template used in software architecture to document and communicate important decision during design process.
- This table typically includes the decisions, their context, dependencies, consequences etc.

(2) ~~Question~~
How to translate requirement model to design model?

→

(1) Understanding requirement:-
Take the requirements from user and understand it properly in order to provide functionality

(2) Identify design element:-
Break down requirement into smaller component such as module, function, classes.

(3) Define architecture:-
Determining overall structure of system including how component will interact.

④ Refine details:
Elaborate on each component specifying their attribute, method and behaviour in more detail.

⑤ Consider Constraints:
Account for technical, operational and environmental constraint during design phase

⑥ Validate design:
Review design against requirements to ensure all requirement are satisfied

⑦ Iterate and Refine:
Iterate on design, incorporating feedback and making adjustment as neccessary until it meets Project needs.

DATE:

Q. What is test case design? Write the various approaches for test case design.

→

Q. Explain any four design concepts with appropriate example.

→

1) Abstraction
2) Refinement
3) Modularity
4) Architecture

1) Abstraction:

- Abstraction involves simplifying complex systems by focusing on high-level concepts while hiding the lower-level details.

- The goal of abstraction is to reduce complexity & make the system more understandable.

e.g. An abstract class in OOP defines a template for other classes. It outlines methods & properties without implementing the details.

2) Refinement:

- Refinement is the process of elaborating & detailing the design from high-level abstraction to more detailed levels.

- It ensures that system design progressively becomes more specific & detailed.

e.g.

3) Modularity :

- Modularity refers to diving a system into distinct, manageable, & independent modules that can be developed, tested.

- It enhances maintainability, scalability & reusability of the system. Individual modules can be developed & updated without affecting entire system.

e.g. A web application might be divided into modules such as user authentication, data processing etc.

4) Architecture :

- Architecture refers to high-level structure of a system, defining the organization of its components, their relationships, & the principles guiding its design & evolution.

- It provides a blueprint for system construction, ensuring that the system meets its functional &

PAGE NO.
DATE:

non-functional requirements,

-e.g. The architecture of a microservice based application defines how services interact, communicate, & are deployed independently.

Q. What is design pattern? How patterns can be used in design?

→ - Design patterns are reusuable solution to common problems encountered in software design.

- They provide templates or blueprints for solving specific design problems in a structured & efficient way.

Use of patterns in design:

1) Promote Reusuability:
By following established patterns, developers can leverage pre-defined solutions, saving time & effort.

2) Encourage Scalability:
Patterns help in creating designs that are flexible & adaptable to changes, making it easier to scale the system as requirements evolve.

PAGE NO.
DATE:

3. Improve maintainability:

Using patterns can lead to cleaner, more organized code that is easier to understand & maintain over time.

4) Enhance Collaboration:

Design patterns provide a common language for developers to communicate & understand each other's designs, promoting collaboration within a team.

Q. Explain object oriented view of component level design with suitable example.

→

PAGE NO.
DATE:

Q. Differentiate betⁿ Following :-

i) Cohesion & coupling in context of software design.?

Aspect	Cohesion	Coupling
1. Definition	Degree to which elements within a module belong together.	Degree of interdependence between modules.
2. Desired state	High cohesion	Low coupling
3. Change propagation	Changes within a module have limited impact if cohesion is high.	Changes in one module may necessitate changes in other modules if coupling is high.
4. Focus	Internal organization of a single module	Relationship & dependency between multiple modules.
5. Impact of High	Enhances readability, maintainability, & reusability.	Increase module independence & system flexibility.