Requirement Analysis of a Product

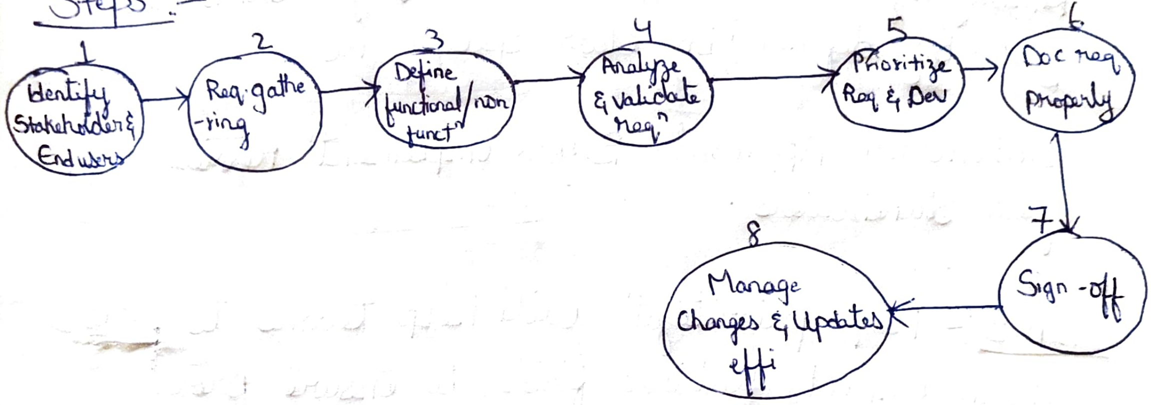
*. Validating / Identifying product needs before starting its development

*. Gathering info from stakeholders & end users.

*. It helps reduce project cost, as fixing issues early is cheaper then correcting them later.

*. It should be conducted before starting a project to have a clear roadmap.

Steps :-


1
Identify Stakeholder & End users
2
Req gathe-ring
3
Define functional/non funct^n
4
Analyze & validate req^n
5
Prioritize Req & Dev
6
Doc req properly
7
Sign-off
8
Manage Changes & Updates effi

Step 1:- Identify who will be the stakeholder i.e., the person responsible for providing input about the product functionalities, product goal & how the product will work.

Step 2:- Requirement gathering from the stakeholders to get input & understand the goal of product.

Like - Group Interviews, Surveys & Questionaire

Step 3:- Once req is defined the ~~requirements~~ teams will be group together. Two primary types

Functional Req:- funct^n the product should have like

• System should allow users to reset their p/w by email.

• System should allow users to update their profile photo

Non functional :- This req is related to system performance, security & usability.
Eg - Appl^n should load within 2 sec on high speed

Step 4 :- Analyze the requir^n and align with pro goal.

Methods :-

Feasibility Analysis - Checking req, can be implemented actuality.

Ambiguity Testing - ~~avoid~~ remove vague terms like fast by defining in clear specific no.

Stakeholder Approval - Before implement^n review req. from stakeholders.

Step 5 :- Req. prioritizat^n will help teams to focus on high impact features first, to ensure project delivery on time.

Step 6 :- Proper document^n helps prevent miscommunicat^n track progress & streamline future updates.
MS word and Google Docs to manage doc.

Step 7 - Once req^n are finalized get signed agreements from stakeholders. Everyone should agree on final req.

Step 8 :- Manage Change & Updates

Object Oriented Analysis & Design (OOAD).

* Way to design Software by thinking of everything as objects similar to real life-things.
* Based on Concepts of OOP & systematic approach to design & develop Software systems.

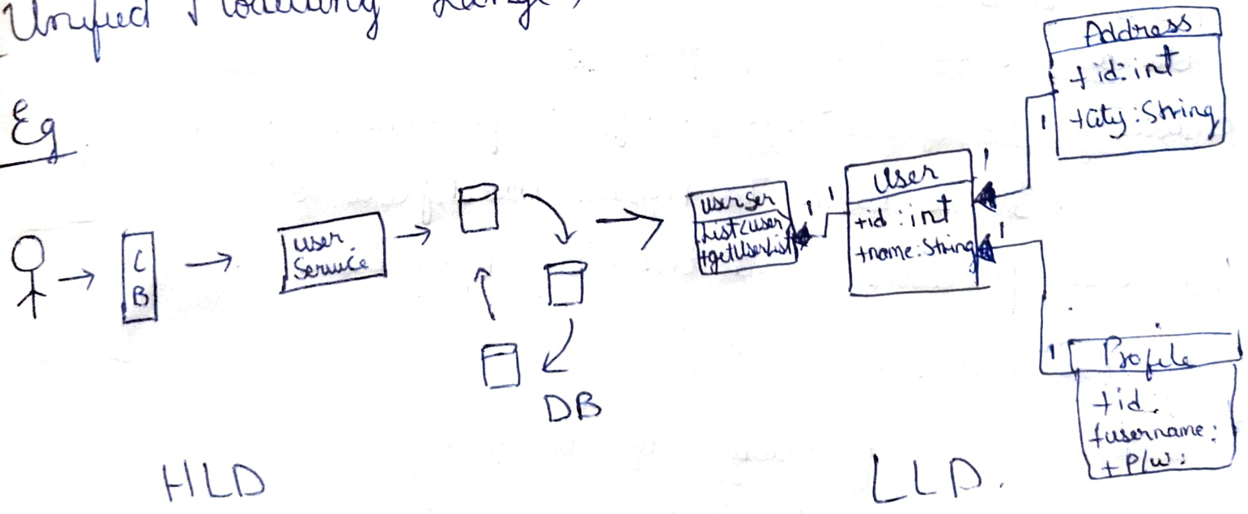
HLD - High Level Design ($\overset{\text{User Service}}{\uparrow}$ Component selectⁿ).
LLD - Low Level Design

HLD - Selectⁿ of platforms, diff tools & database design. It gives overview of Component & gives input to create LLD.

LLD - It is created based on HLD. Describes class diagrams with methods & relations between classes & program.

* To transform HLD into LLD we use UML Diagram (Unified Modelling Langu).

Eg



UML Diagrams

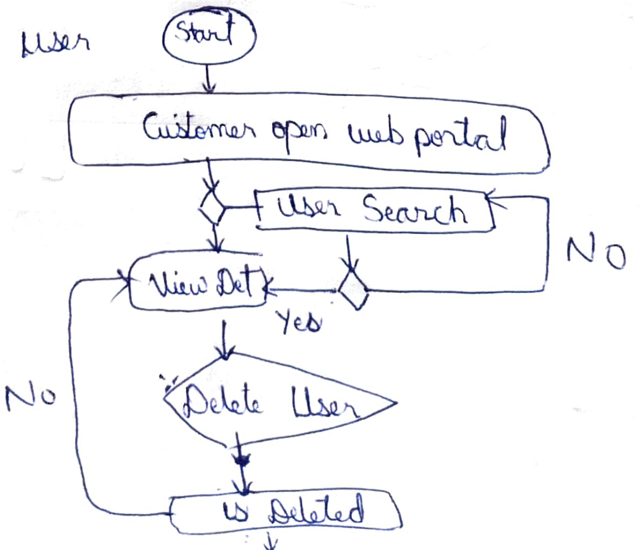
Structural UML Diagram	Behavioural UML Diagram
Class diagram	Sequence
object	Use Case
Package	Activity
Component	State
Composite Structure	Communicatⁿ
Deployment	Interactⁿ overview
Profile	Timing

UML – Collectⁿ of diagrams, it will help to understand the behaviour & structure of the system being designed. Graphical presentatⁿ of Component.

Use Case Diagram – It will show what does the system do from the user point of view & it gives high level functional behaviour.
Actors: users or roles
Use cases: action they perform.

Activity Diagram – Shows flow of control for a system. It is used to model workflow or business processes and internal operations.

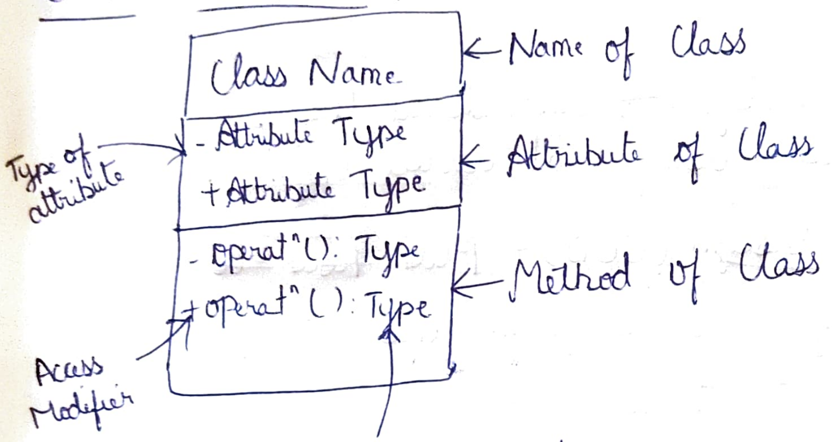
Eg – Delete a user



Sequence Diagram - It will show the interactⁿ step by step.

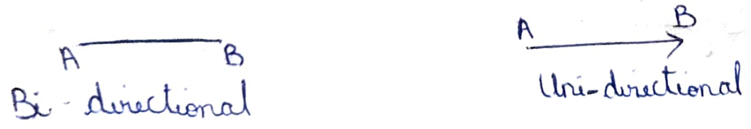
State Diagram - Shows states of an object and how it change based on actions. (object lifecycle).
Eg → Initial → Pending → Confirmed → Cancelled

Class Diagram.



+ → public
- → private
# → protected
N → package

*. Association → when two classes in a model need to communicate with each other, there must be a link betⁿ them. This link can be represented by associatⁿ.



(i). Aggregratⁿ (Dependent)
(ii). Compositⁿ (~~In~~dependent)
(iii) Generalizatⁿ
(iv). Dependency [A<·······B]
(v). Multiplicity

SOLID

S - Single Responsibility Principle
O - open closed Principle
L - Liskov Substitutⁿ Principle
I - Interface Segregatⁿ Principle
D - Dependency Inversion Principle.

Steps in OOAD

1. Object - Oriented Analysis → Understand the problem and model the system using real-world objects.

2. Object - Oriented Design (OOD) - Plan how to build system using class diagram, interactⁿ & responsibilities.

OOAD is a method of designing a system by thinking in terms of object - just like real world objects.

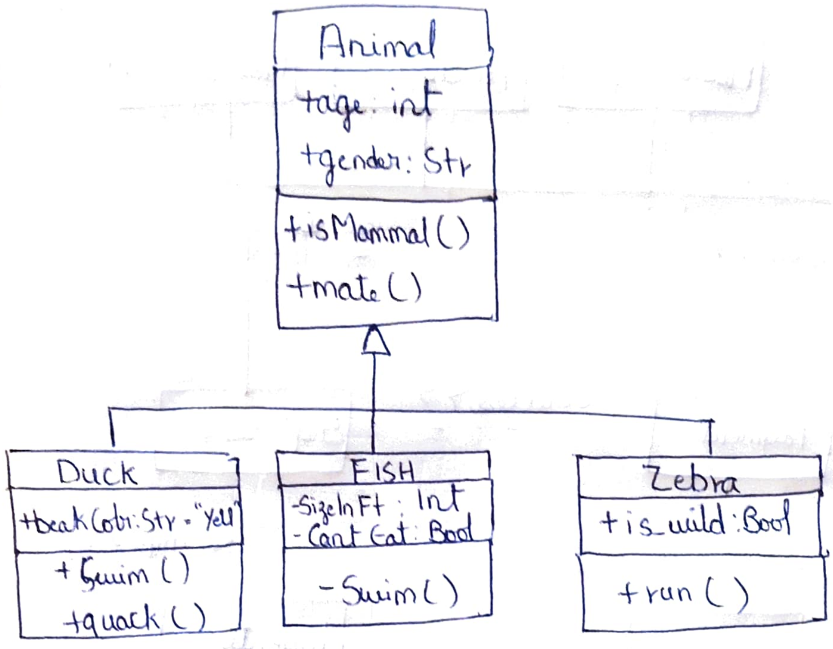
Component Diagram (System Parts)

It will show different components / modules of system & how they interact. Gives modular view of system.

Deployment Diagram (Physical Setup)

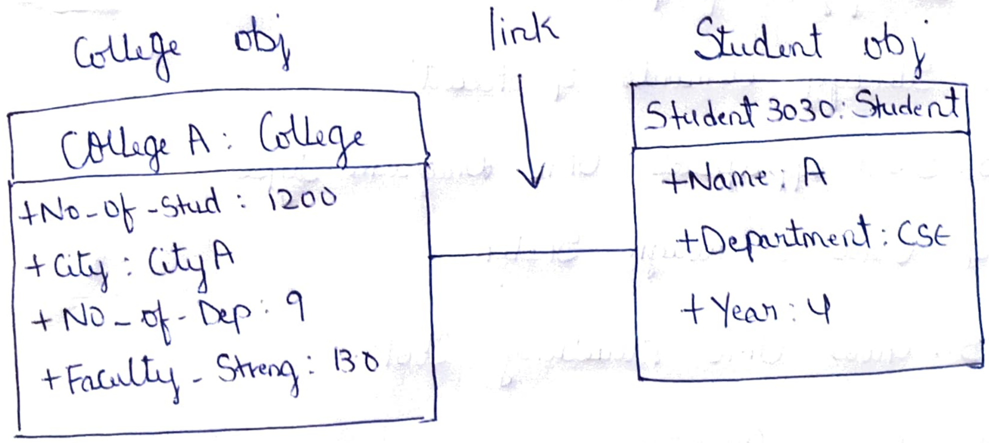
Shows where software will run - like servers, databases ~~de~~ device

Class Diagram



Object Diagram

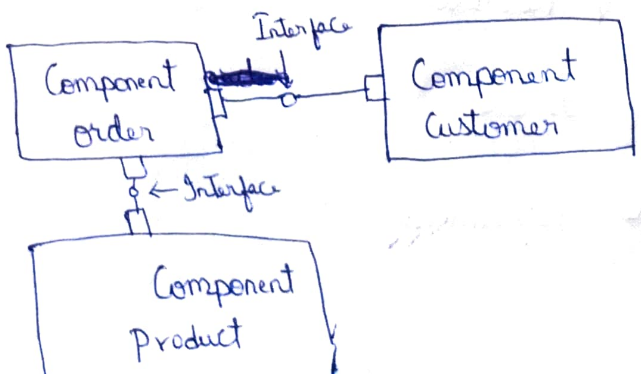
It uses a link & 2 objects.



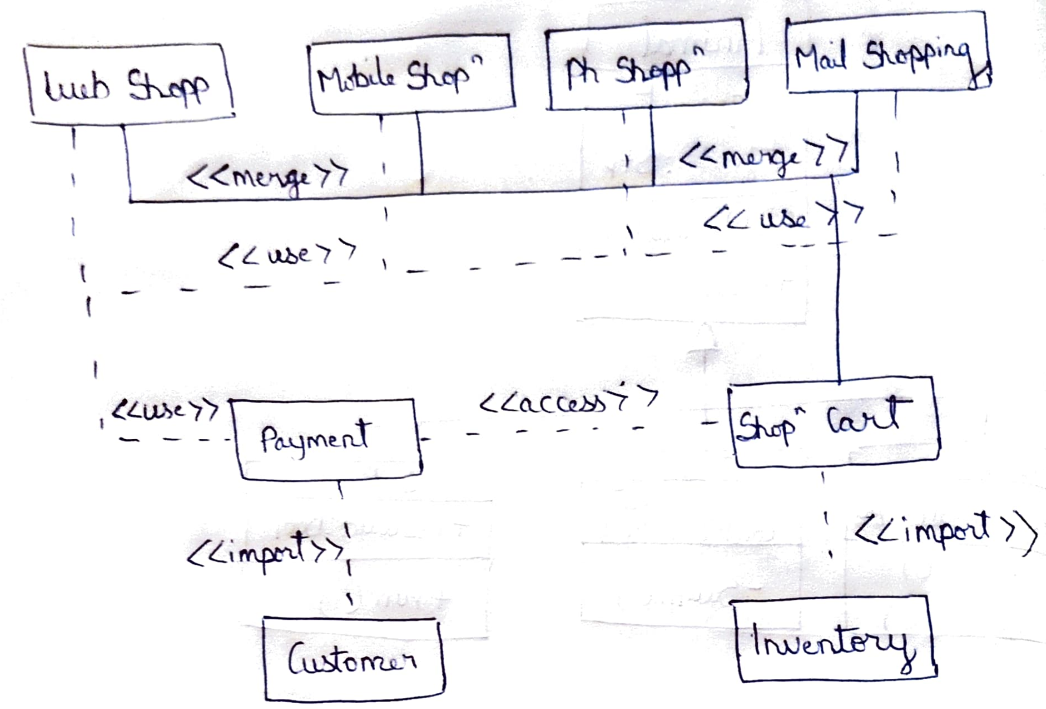
* An obj of class Student is linked to an obj of class College.

Component Diagram

Online Store



Package Diagram



Steps to Create UML Diagram

Step 1: Identify the purpose

Step 2: Identify elements & Relatⁿ

Step 3: Select appropriate UML Diagram Type

Step 4: Create a rough Sketch

Step 5. Choose UML Modelling Tool.

Step 6: Create diagram

Step 7. Define Element properties

Step 8. Add annotations & Comments.

Step 9: Validate & Review

Step 10: Refine & Iterate

Step 11. Generate Documentatⁿ.