Date: 20-5-22

## Static View

A view when the system is not in Running state is called static views.

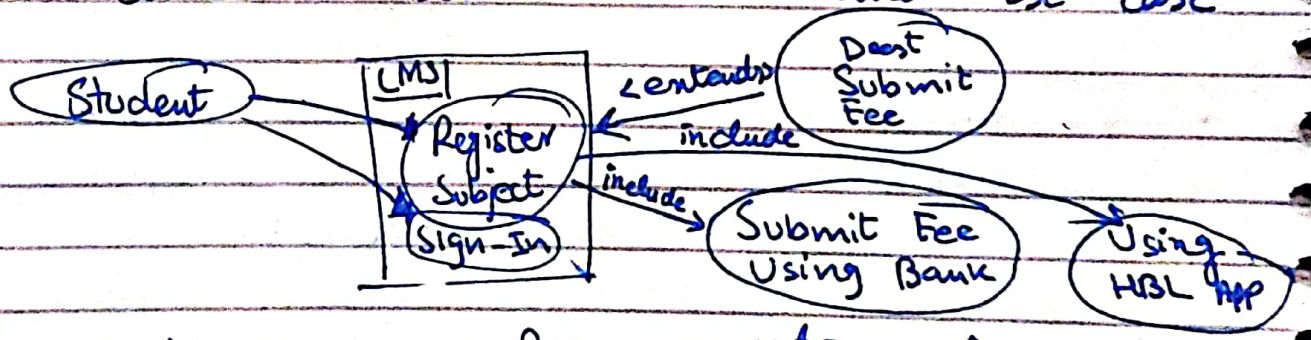
- System exists but does not do any thing.

## Dynamic View

A system in running state is called dynamic state/ view.

## Use Case Diagram (dynamic)

1. Actor: User
2. Use case: what he want to do
3. Relationship: Connection between user and use case




- Actor
- Action

<extends>

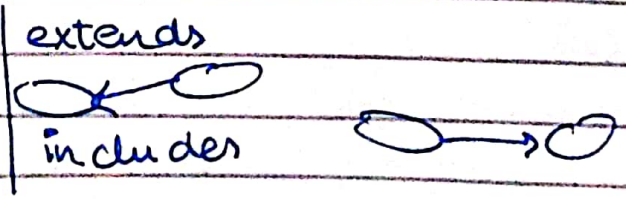
- Describe the problems
- Exceptions

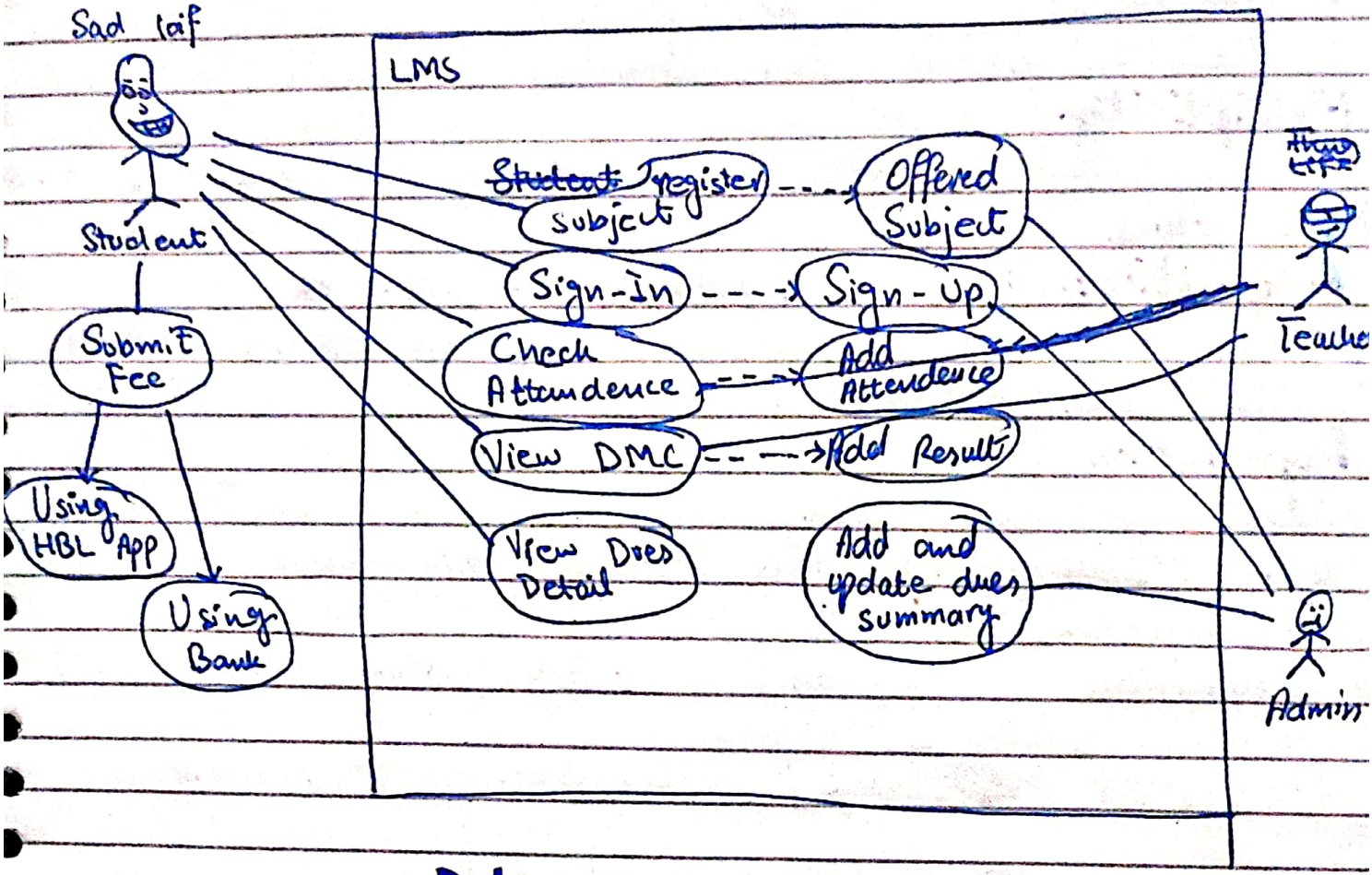
<includes>

- Alternate Flow

Extra functionality

- Pre-conditions for use case






Date: 23-5-22

## Class diagram

### - Static diagram

- Name only
- Name + attributes
- Name, attributes and operations
- Name + operati

Class Name →

| Student |
| --- |
| ID : string |
| Name: String |
| F.name : " |
| DOB : date |
| email: email |
| Address: string |

| Sign in ( ) |
| --- |
| View DMS ( ) |
| Subject Reg. ( ) |
| View Attendance ( ) |

← Attributes

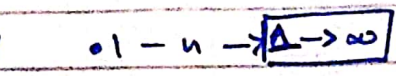
| |
| --- |
| + Public |
| # Protective |
| - Private |

- Object diagram


- Relation : Association
  - No operations
  - No multiplicity
  - No naming

- Underline " " for static
- Italic for *Abstract class Name*

## Multiplicity

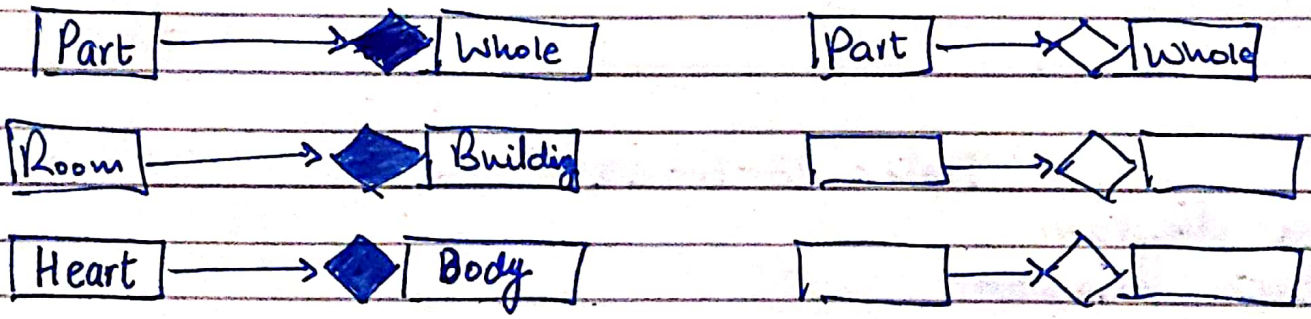
- 1 – 1
- 1 – 
- 1 – n 
- Rang 1 — 0..3

## Part whole

**Composition:** When the part class ↑ class exist independly then it is called Composition (seperated from whole)

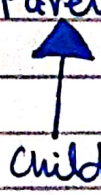
**Aggregation** When the part class seperated from whole class cannot exist whole as a class is called aggregation

- Association "—" "⟷"
- Composition — has — Aggregation




## Intevitance

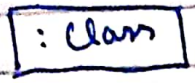
- Parrent Class and Child class
- Class class can access all the public and parivate members of parent class.
- Parent class cannot access the child class.



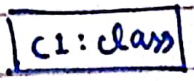
Date: 27-5-22

## Sequence Diagram

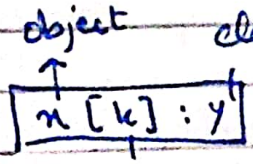
- Dynamic state - Every class is passing messege to other class




Class diagram

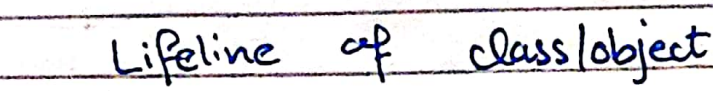



Objet diagram

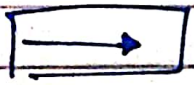



- Interface

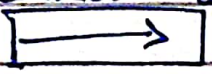

diagram



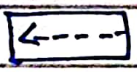

### Messeges

① Syndronous  (must reply)

- Without reciving the reply of first messege the class cannot send messege.

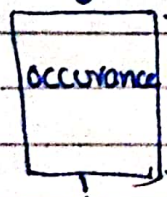
② Asyndronous 

- Without taking care of reply the class can send reply

③ Reply  • Reply : can be both sides Both Syndronous and Asyn ... can have reply

### Destruction

- After the lifetime a object needs to be distroy




→ execution start

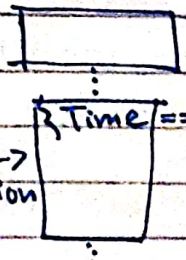
→ end

destruction

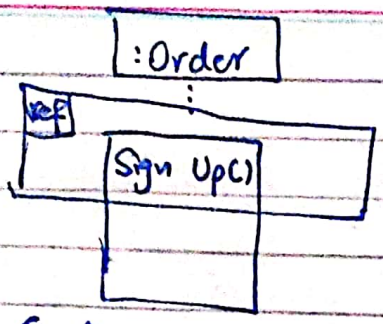
**Main Reason**

- Communication
- Time

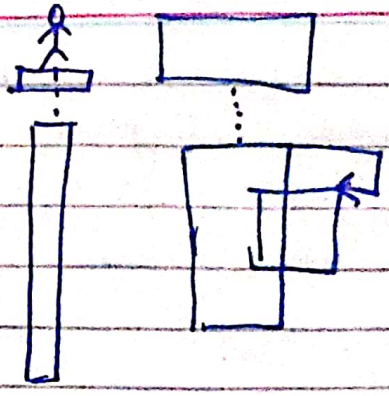
we can use → execution condition




the class still remain in m.m




• We are using ref. diagram to execute it first and compulsory



**Gate**

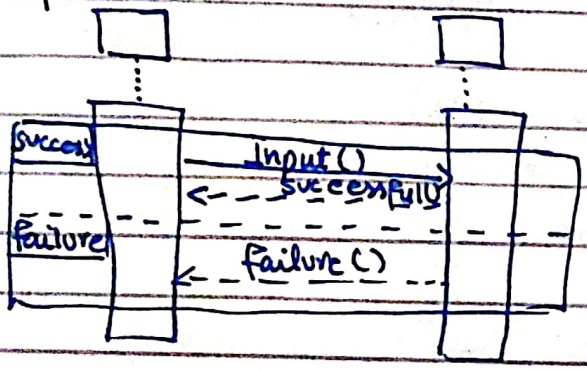
Starting of sequence diagram

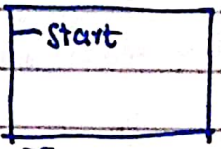
●→

If no boundry exist

**Control Structure**

if - else



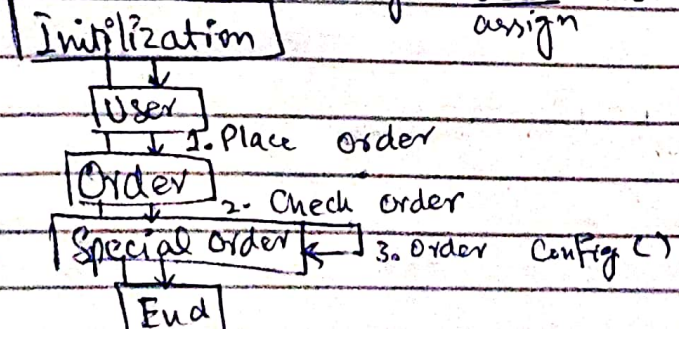

If boundry

→ • Start
• Begin

## Date : 3-6-22

**Collaboration Diagram** } (Dynamic) Interactive diagram → both diagram interact with each other by sending msg.

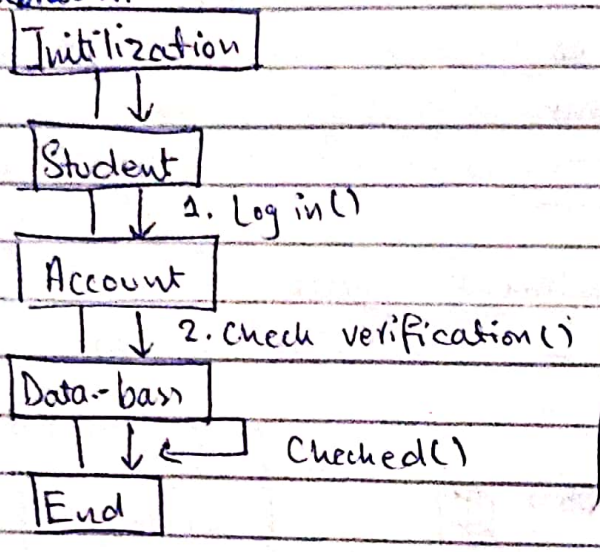
- → Sequence diagram
- → Vertical representation / Horizontal
- → Must represent the "end" box and "initilization"
- → For readability add numbers to the messeges. (add = assign)



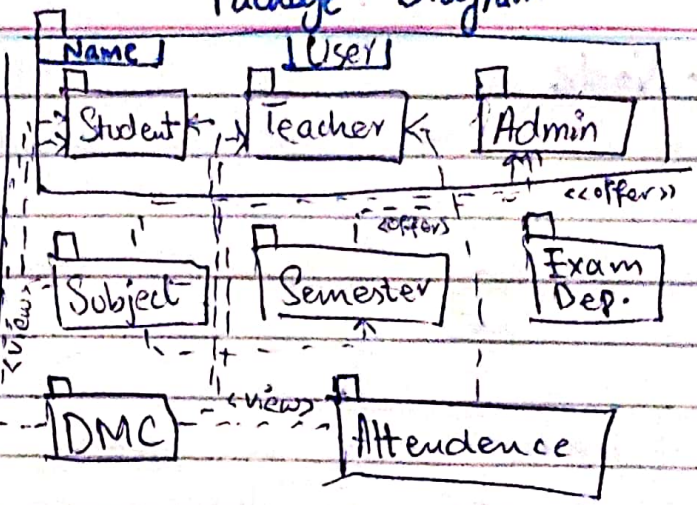

Main Reason

Structure

## Example: LMS Login Collebration




## Pachage Diagram




## Pakage diagram

- Name can be written anywhere

→ Combination of related ~~diagram~~ classes
→ State: static
→ can be combination of classes and pakages
→ To increase readability of the program
→ Dependency: change in

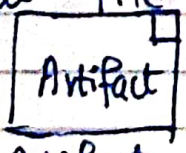
**Date: 6==6-2022**

## Deployment Diagram (Static)

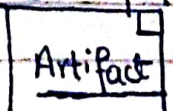
→ How can we deploy a system in real ~~world~~. world.
→ Use images and emgies for better detail.

- **Artifact**

→ Real world entity, Sourse file, .dll files, DB table output file .etc.



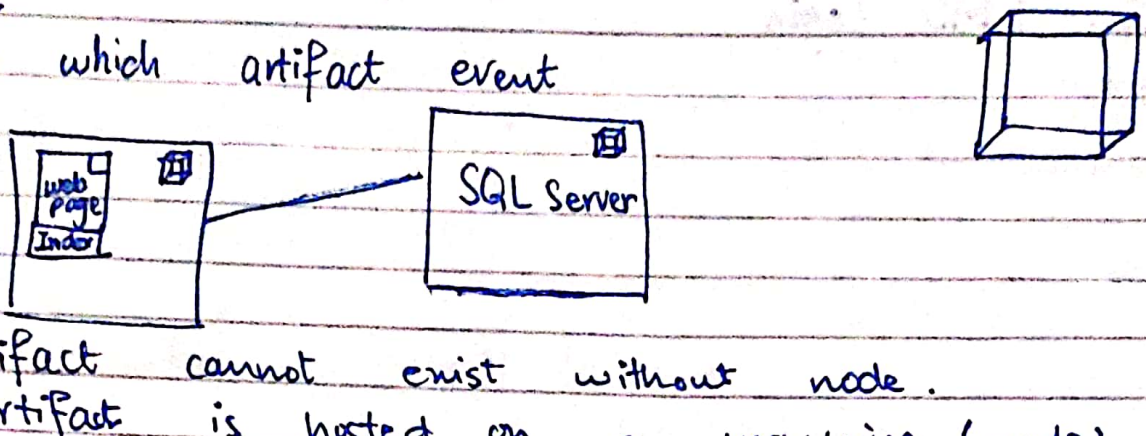

Artifact



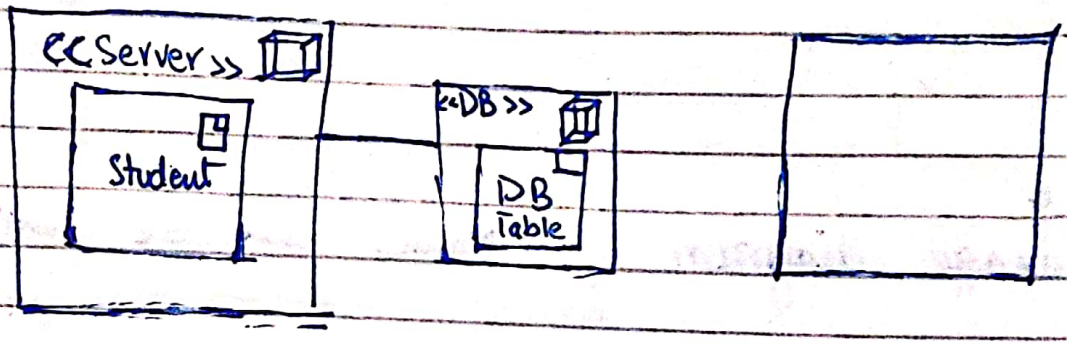

Artifact Instance
Artifact Object

- **Node**

On which artifact event

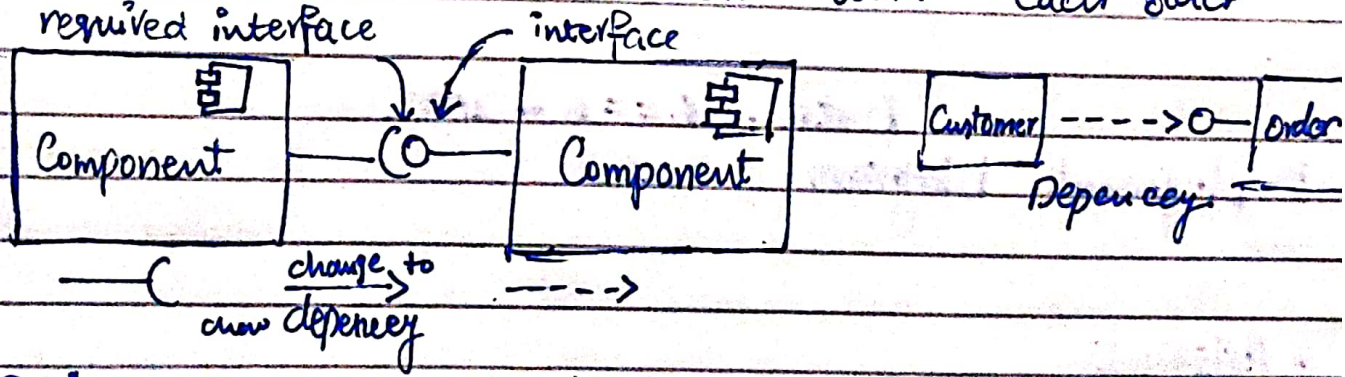



→ Artifact cannot exist without node.
→ Artifact is hosted on a machine (node).




## Component Diagram

→ what are the component and how they are related and communication with each other

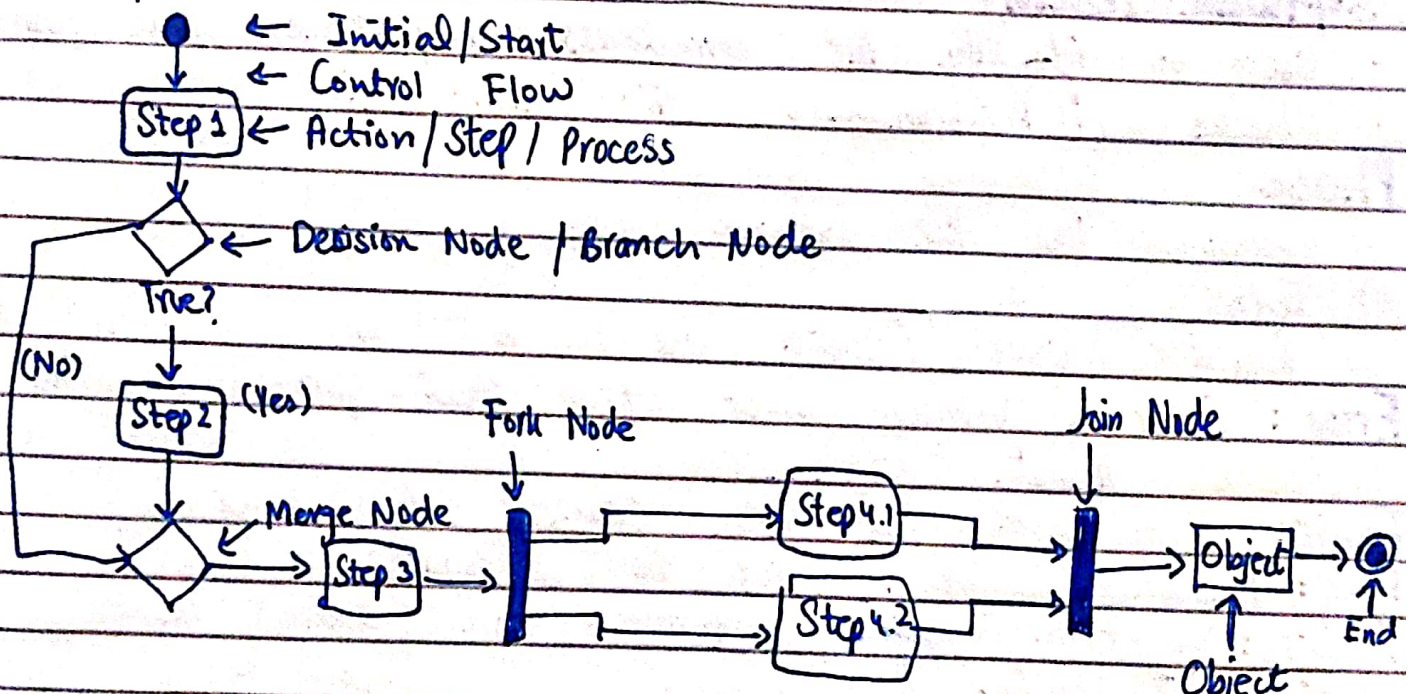



## Activity Diagram (Dynamic)

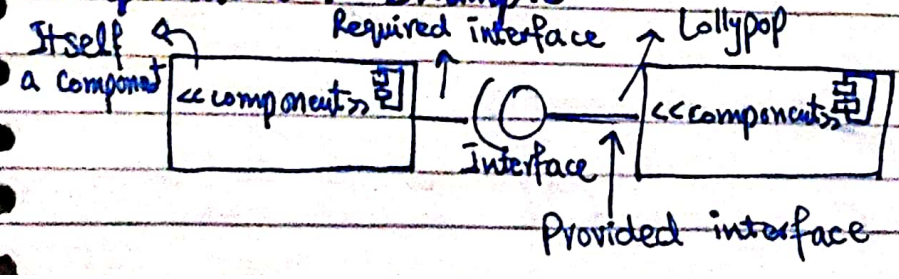
- Fancy flowchart - displays the flow of activities involved in a single process.
- State
  → the work done by system

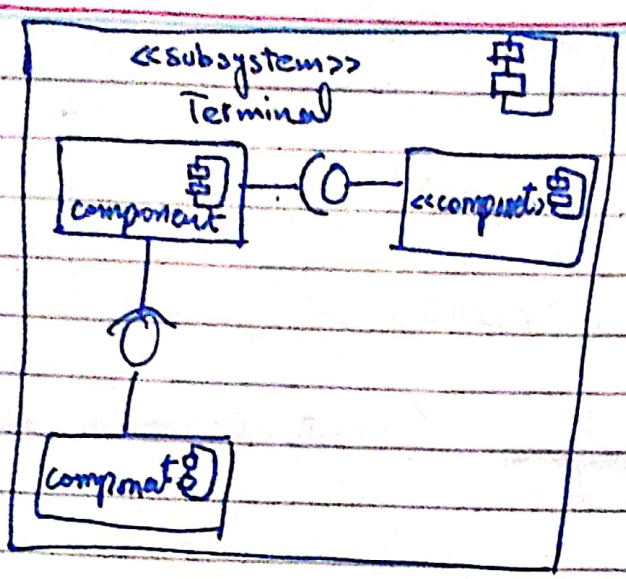
- Start ●
- End ◉
- Activity ▢
- Flow Control →
- Branch → ◇ → (1 input two outputs)
- Merge → ◇ (2 inputs 1 output)
- Fork ↓ ═══ ↓↓↓↓ (1 Input, Many output)
- join ↓↓↓↓ ═══ ↓ (1 output, many input)

Example:




## • Component diagram Explained + Example






Date: 13-6-2022

## Software Testing

- Used to identify the correctness, completeness and quality
- Checks (i) Performance (ii) Security (iii)

## Phase

→ Delevoper test the software
→ Testor " "
→ Registor User (Beta user) " "

**Error:** Error is human ~~error~~ action that produces the incorrect result.

**Fault:** State of Software ceused by error

**Bug:** Presence of error in software

**Failure:** Deviation of Software from explict result.

**Testing life cycle:**

Project initialization
↓
System study
↓
Test plan
↳ Test cases → Design → Execute → Report defects → Report → Analysis → Regresion Testing → Test again n again

After to find the bug, it is resolved and tested again.

SAR → Self Assermant Report at the end of the year

- Top down aproch
- Bottom up approch

(Front + Back end)

**Level testing** → Test everything seperately.

(i) **Unit** testing (Front end) Only outside (GUI)

(ii) **Integration** (Combine the module and then test them)

↓ Combine Unit testing

(iii) **System** → Test the whole system

see inside the system ↑ white box test

Black box

Gray box ↓ How the buttons are working

(First front end then back end)

page

Stubs → If one system is created and one is in development state which is important for testing for 1st module/system then a dummy is created for the testing of 1st module.

Sign in → Sign up page (depends)

↓ completed

Unit Testing

↓

Integretion testing → we create dummy data/page

(still developing ↑ requires)

**Date : 11-6-2022**

**Type of Testing**

1. Unit Test
2. Integration Test
3. System Test
4. Usibility Test (GUI, Look n Feel, Speed)
5. Performance Test (Load, Stress, Data) (with large data)
6. Security Test (Unauthorized Access, Virus
7. Smoke Test (Test system inputing with small data)

Rigix → Format

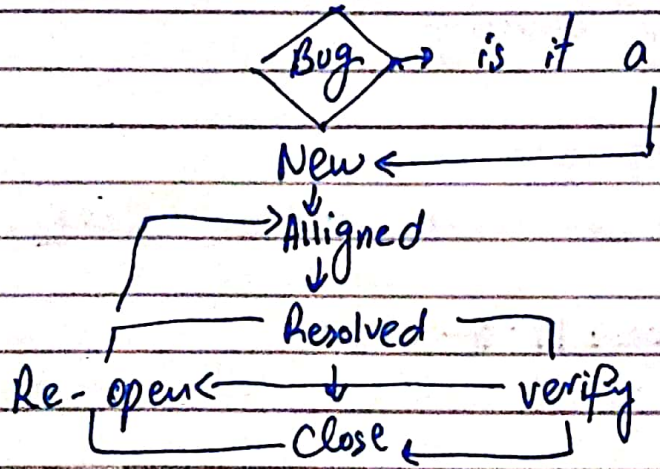
Tester inside the team/group (experts)

⑧ Alpha (Test on developers side, Controlled condition)
⑨ Beta (Outside software house, Uncontrolled conditions Registered/Authorized users)
⑩ Acceptance (Testing user-level requirements i.e wireframe)
⑪ Regression (Test a function if bug identify, Correct it, test the whole system again)
⑫ Monkey Test (No particular guide like test the system randomly)

during SDLC → Specification, Guidelines, inspection)

Verification : is the product right
Validation : is it the right product
After SDLC → fullfill the requirement

## Bug Life Cycle




| Functional Testing | | Non-Functional Test | |
|---|---|---|---|
| - Installation | | - Performance | |
| - Usability | | - Stress | |
| - Smoke | - Destructive | - Security | |
| - Regression | - Recovery | - Accessibility | |
| - Monkey | - Acceptance | - Load testing | - Data |

Date: 20-6-2021

## Unit Level : Log in Page

→ Email should be corrected and password shall have 8 characters

- Sigin
- Sigup

| Test data | Expected Result | Actual Result | Result |
|---|---|---|---|
| abc@gmail.com | Accepted | Accepted | Pass |
| @gmail.com | Rejected | Rejected | Pass |
| abc@gmail | Rejected | Rejected | Pass |
| 12@gmail.com | Rejected | Accepted | Fail |

These two must be same for pass result

- Password

| Test data | Expected Result | Actual Result | Result |
|---|---|---|---|
| 12345678 | Accepted | Accepted | Pass |
| 1234 | Rejected | Rejected | Pass |
| abc123 | Rejected | Rejected | Pass |
| abc!123? | Rejected | Accepted | Fail |

some special characters are not allowed

| | | | |
|---|---|---|---|
| abcd | Rejected | Rejected | ~~Pass~~ |
| abc_1#2 | Rejected | Accepted | Fail |

## Integration Level

→ Sign in Module

| Test data | Expected Result | Actual Result | Result |
|---|---|---|---|
| User ~~sign~~ sign in but doesn't exist | User has Sign-in | Signed-in | ~~Fail~~ Pass |

| | | | |
|---|---|---|---|
| Use doest exist sign-in | Jump on sig-up | Jumped | Pass |
| User enter wrong usernam | Does not sig-in | Does not sig-in | Pass |
| User enter wrong password | " | " | Pass |
| User does not enter anything | Sign in Failed | Sign in Faild | Pass |

## System Level

Test case ID : 001 Created by : Zeeshan Description : Sig-in
QA Test Log : Ali reviewed testing performed by Zeeshan
Prerequisite : i) Internet ii) Account exist Test : Pass

| Sr # | Steps | Expected | Actual | Pass / Fail / Not executed / Susp |
|---|---|---|---|---|
| 1 | Enter email, pas | credentials | As expated | Pass |
| 2 | Click sig-in button | Logged-in | As expected | " |

Post Conditions : User is signd-in , able to search products