USN: 01fe18bcs211

Roll No: 419

1a) Software engineering ethics

Software engineering includes wider responsibilities than simply the application of technical skills.

Software engineers must be honest and ethically responsible for the product. They should behave as responsible individuals if they are to be respected as individuals.

Ethical behaviour is more than simply upholding the law but also involves following a set of principles that are morally correct.

Engineers should not exploit their technical knowledge for inhuman activities.

Issues of proffesional responsibility

* Confidentiality

Engineers should normally respect the confidentiality of thier employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.

* Competence

Engineers should not misinterpret thier level of competence They should not knowingly accept work which is out of their competence

USN: 01fc18bcs211

Roll No: 419

* Intellectual property rights

→ Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright etc. They should be careful to ensure that the intellectual property of employers and clients are protected

* Computer misuse.

→ Software engineers should not use their technical skills to misuse other people's computer. Computer misuse ranges from relatively trivial (game playing on an employer's machine) to extremely serious (dissemination of virus). Injecting viruses or malware in the system causes harmful effects.

* The data of the company should not be leaked.

1b) Functional requirements :-

1) Camera shall capture images

2) Camera shall detect the person

3) Camera shall detect the person and classify further as wearing mask or not.

4) Doors open if the person is wearing the mask.

5) Doors remain closed if the person is not wearing mask.

6) Warning message "Wear Mask" is displayed on screen if person is not wearing mask.

Non-functional requirements:

* The camera should be working 24/7.
* The camera should detect a person in 20seconds of her/his entry in the place.
* The camera should open the door before 10seconds after detecting that the person is wearing mask.
* The camera should be of 64MP.

All the requirements are according to priority

2a) Principles of Agile Methods.

1) Incremental delivery: the product is delivered in increments for the users or customers. The feedback or inputs are taken from them and if any changes to made it is incorporated in next increment depending on it's priority, cost and schedule.

2) Customer involvement: The customer is also part of the development team he is fully involved in the process from giving requirements prioritezing them writing the acceptance tests in the testing phase and also be involved in the testing and validation phases. They evaluate iterations.

3) People not process: This is done from pair programming, customer involvement and collective ownership of the code where everyone is responsible for all parts of code.

The skills of team members should be recognized and exploited. Team members should be left to work on their own will without a prescriptive process.

4) Embrace change: Expect the system ~~Do~~ requirements to change and the system should accomodate to the change. The system should be designed to evolve and meet business needs

5) Maintain Simplicity: It is the key to keep activities as simple as possible to avoid complexities so that incorporating changes will be easier with simplicity. "Refactoring" is one such approach of continous code refinement to keep it simple, clean, maintainable. We tidy up all the parts of codes. Replace inplace function definitions with calls to methods included in header. Renaming attributes and methods.

The design is also kept simple. "KIS" method is used for designing where nothing more or less than the user story is designed. The simplicity is extra efforts but helps in smooth development of software

USN: afc18bcs211

Roll No: 419

2b) (i) Incremental development

Merits:

→ This process model is suitable for the application as the requirements are not stable.

→ As it is universty application for examination the requirements might change or new feature needs to be added owing to the variety of stakeholders (students, lecturers, HOD's etc)

→ As online exams is a must in pandemic and the learning process can't stop the quick delivery is needed to the University.

→ Highly feature- driven

→ As it is a new approach to study or academics the client or university owners also can't produce or come up with complite and correct requirements.

→ A team 7-10 developers can work on this system and quick delivery is possible.

→ If there is no enough expertise or human resources to develop it from scratch then this model is not useful as it involves highly iterative approach.

Demerits:-

* Risk of failure is more.
* Too many requests for changes or features can degrade the system being developed
* The different stages of development are interleaved so there is no exact demarkation between them.
* If there is no mush resources with university and time duration for the delivery is short than this not suitable If the system is not needed for long time.

ii) Integration and configuration model.

→ This is suitable for quick delivery of the software as pandemic has approached. It makes use of some existing software or components and don't develop from scratch. Cost is reduced.

→ Owing to the current pandemic situation gathering resources like humans and cost or money is very difficult. So this is best suited if resources are less.

→ less human efforts is needed and risk of software failure is very less. as we are using the already existing and working components.

→ Google forms can be used for objective approach.

→ Exam.net can be used for subjective approach.

→ Web cams can be used through MS teams as proctoring is needed.

→ Team of H-S developers can be involved in the development process.

→ All the available softwares can be integrated to build the required system.

Demerits:

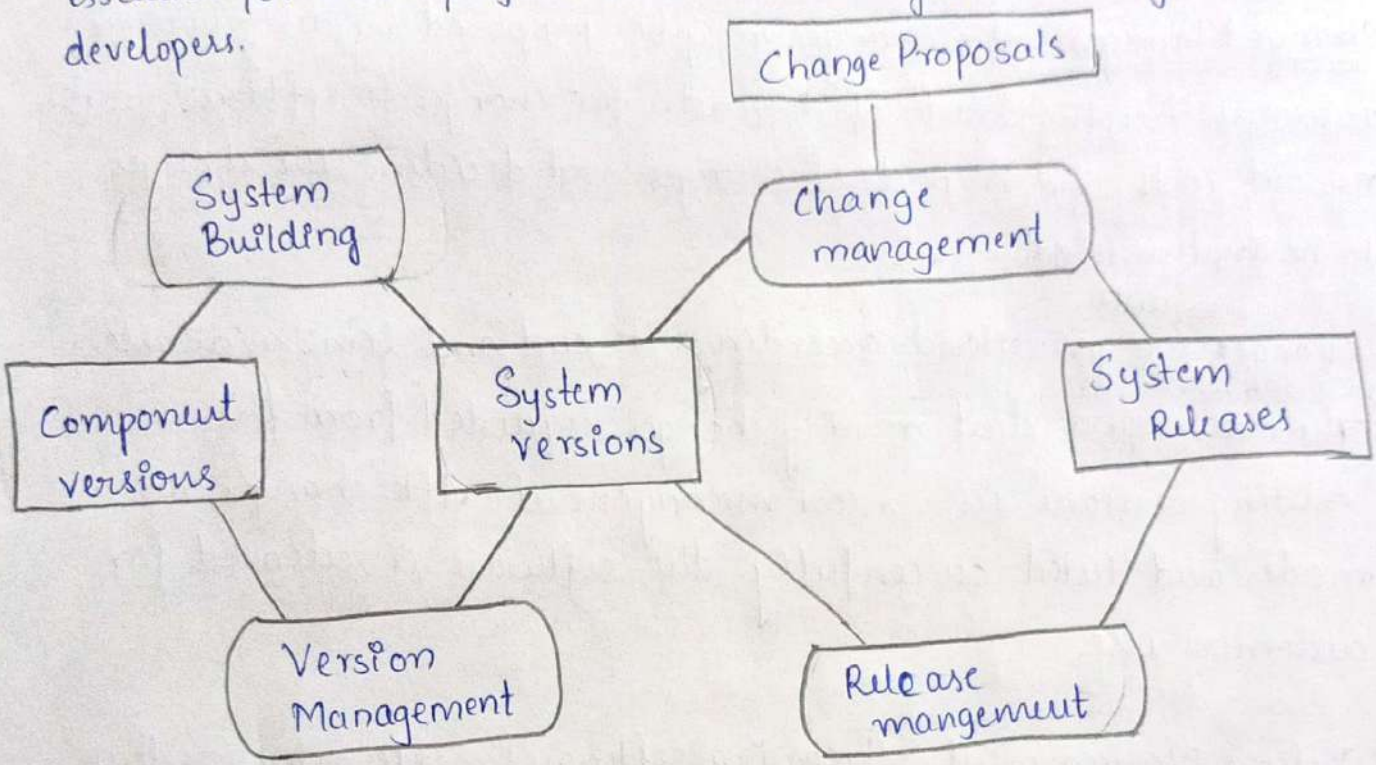
→ If we reuse some components or software then we might be forced to modify the requirements as per the reused system or software.

→ There might be integration issues during the development process.

→ The feedback mechanism and customer involvement is not as robust as incremental.

→ Requirement compromises are inevitable and so might not meet all the user requirements

5)* Configuration management (CM) is concerned with the policies, processes and tools for managing changing software systems. Without CM we may loose track of what changes and component versions have been incorporated into each system version. CM is essential for team projects to control changes made by different developers.



Configuration Management Activities

* Version management: Keeping track of the multiple versions of the system components and ensuring that changes made to components by different developers do not interfere with each other.

* System building → Process of assembling and linking all the programs components and libraries, data and compling these to create executable system. It is a complex process. Every version of components and versions of components used in system that constitute system version are recorded and hence it communi-

cates with version management. Developers develop system just for testing before giving to client, so it's important to keep track of that systems as well.

So system building is linked with version management.

Change Management : Changes may be proposed by developers or customers. Keeping track of requests for changes to software, working out costs and impact of changes and deciding the changes to be implemented.

Changes requests are prioritised according its cost and time available. Then developers start making changes requested from the existing versions in version management. After changes are made and tested successfully the software is released for customer's use.

Release Management : Preparing software for external release and keeping track of the system versions that have been released. So that if exact replacement or copy of version is requested by customers then it can be produced.

CM tools are very important in agile development as the components and systems are changed everyday.
In development phase the development team is responsible for managing system configuration and new functionalities can be added. In testing phase, the bugs are fixed, perfomance improvements are done. No new functionality is added. In release phase when the software is released to customers use.

USN: 01fe18bcs211

Roll: 419

1b continued)

Functional requirements:

* An alert system shall be configured to notify illegal entry
* Application shall handle multiple people entering the department at same time.
* Application shall be powered up all day.

Non-functional requirements :-

* The application should not cost more than 2 hrs of out-of-service state per week.
* Downtime should not be more than 2 minutes per hour.
* Application results should be reliable during night and dim areas as well.

USN: 01fe18bcs211

Roll: 419

1a continued)

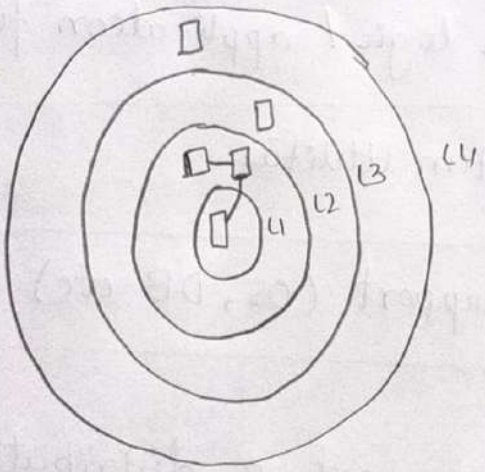
* Privacy → handling, storing, sharing user data only under the circumstance and for the purposes that the user sets.

* Transparency → Transparent decision making procedure of intelligent systems, publically available ethics policies

01fe18bcs211
H19

8a) Layered architecture:

* Here the functionalities are separated
* Used to modul the interfacing of subsystems.
* Organizes the system into set of layers each layer provides a set of functionalities or services.
* A layer depends on its adjacent layers for services.
* Diagram:



Layers interact with each other interfaces between them.

* Supports incremental development of sub-systems in different layers.
* Used when:
  1) Work is distributed among different teams
  2) When new functionality is to be added on top of existing
* Demerits:
  1) Machine dependend layers are not independent
  2) When layer interface changes, the adjacent layers get affected

01fe18bcs211
4/19

* It is often artificial to structure systems this way.

Example: Operating System

User Applications
User Interface
User Interface management Authentication and authorization.
Core business logic / application functionalities
System Utilities
System support (OS, DB etc).

Merits:

* It is beneficial when work is distributed as every layer can be changed independently.
* It is best suited for multi-level security system where secured or critical assets are kept in innermost layers
* Redundant facilities can be provided in each layer to increase dependability of the system.

Demerits:

* Performance can be a problem because of multiple levels of interpretation of a service request as it is processed at each layer.
* Ex: Operating system has different layers as show where user applications depends on services of UI and UI depends on UI management that includes authorization & authentication

01fe18bcs211

H19

* Providing a clear separation between layers is often difficult and a high-level layer may have to interact with lower-layers rather than through the layer immediately below it.

8b)

Requirements:

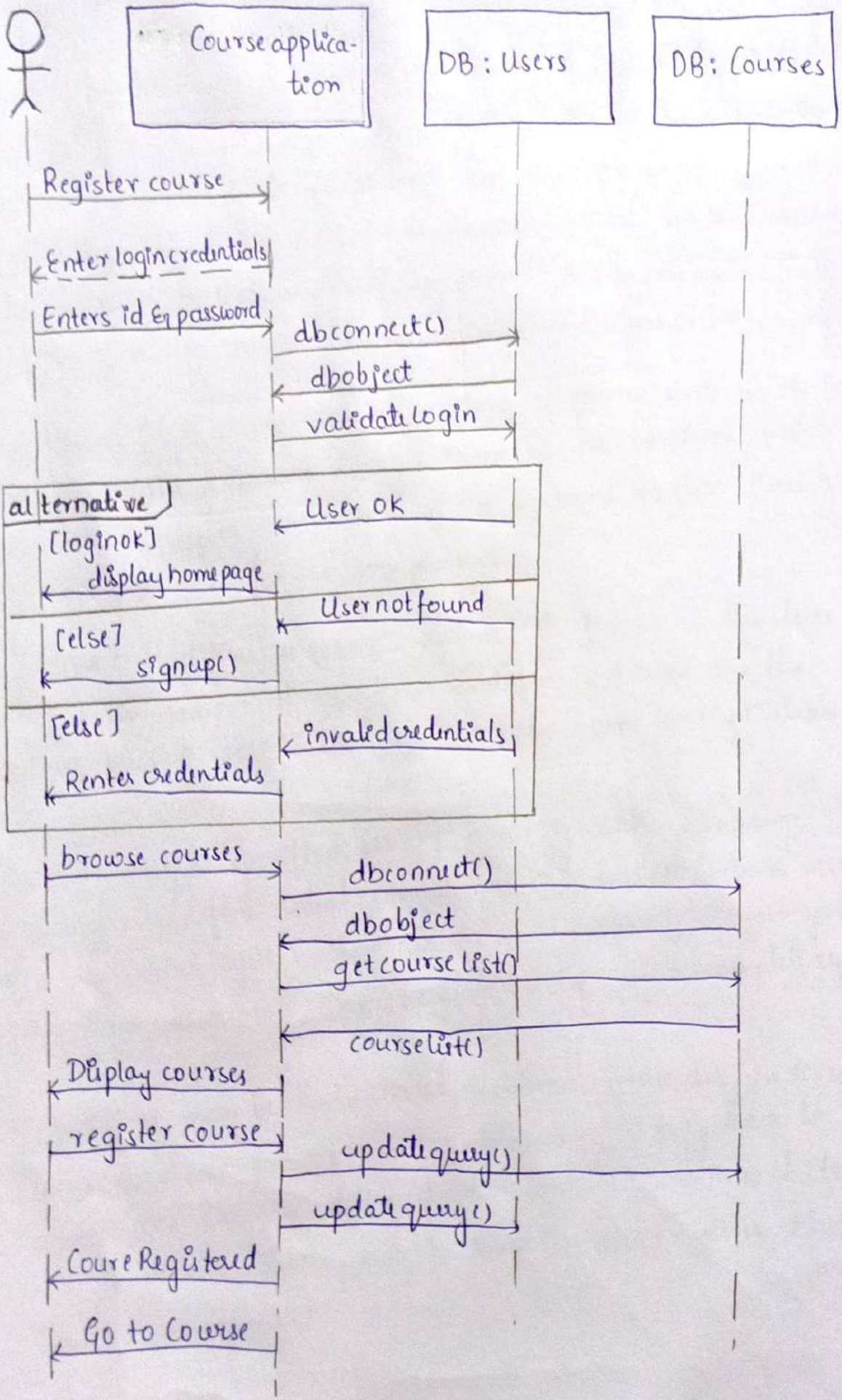
1) Register course
2) Login / Signup
3) Unregister course
4) Save Course

Higher is important one is register course

Classes: 1) Users: DB
2) Courses: DB
3) Coursera application.

Course application has UI that user interacts with. A database for maintaing user info such as credentials, courses registered, courses completed etc.

Course database that has list of courses according to different categories.



10) * Version Management: It a the process of keeping track of different versions of software components or configuration items and the systems in which these components are used. It also involves ensuring that changes made by different developers to these versions do not interfere with each other. Therefore version management can be thought of as the process of managing codelines and baselines.

Codelines: It is a sequence of versions of source code with later versions in the sequence derived from earlier versions. Codelines normally apply to components of the systems so that there are different versions of each component.

Baseline: It defines a specific system. The baseline therefore specifies the component versions that are included in the system plus a specification of libraries used, configuration files etc.

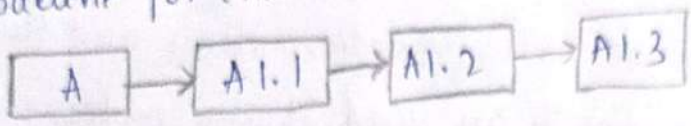
Baselines may be specified using a configuration language, which allows you to define what components are included in a version of a particular system. They are very important because you often have to recreate the specific version of a complete system.

Productlines may be instantiated so that there are individual system versions for different customers. You may have to recreate the version delivered to a specific customer if, for example, that customer reports bugs in their system that have to be repaired

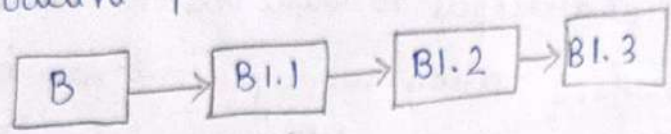
01fe18bcs211
#19

Consider we have four components A, B, C, D

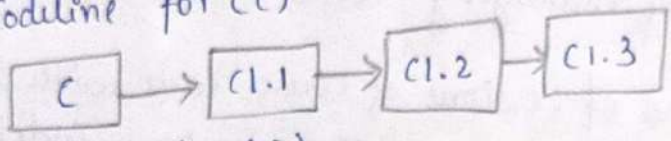
Codeline for (A)



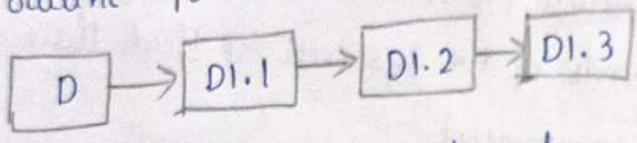
Codeline for (B)



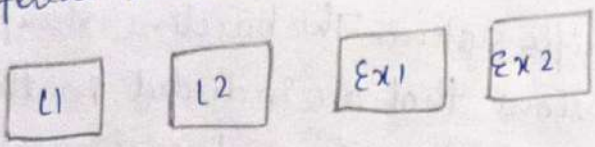
Codeline for (C)



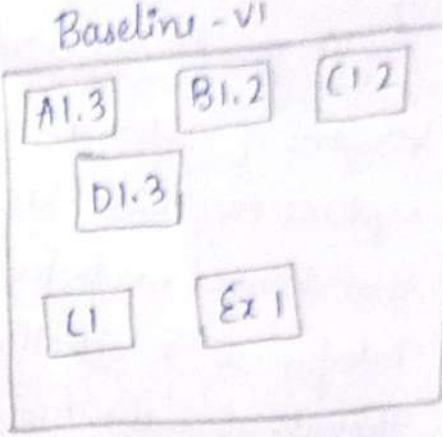
Codeline for (D)



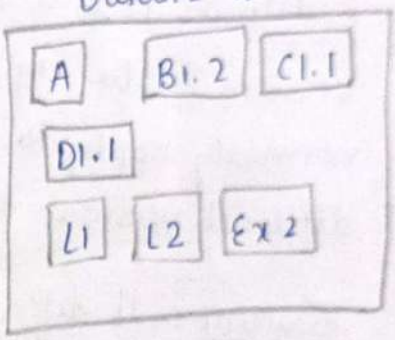
Different libraries and external component



Baseline - V1



Baseline - V2



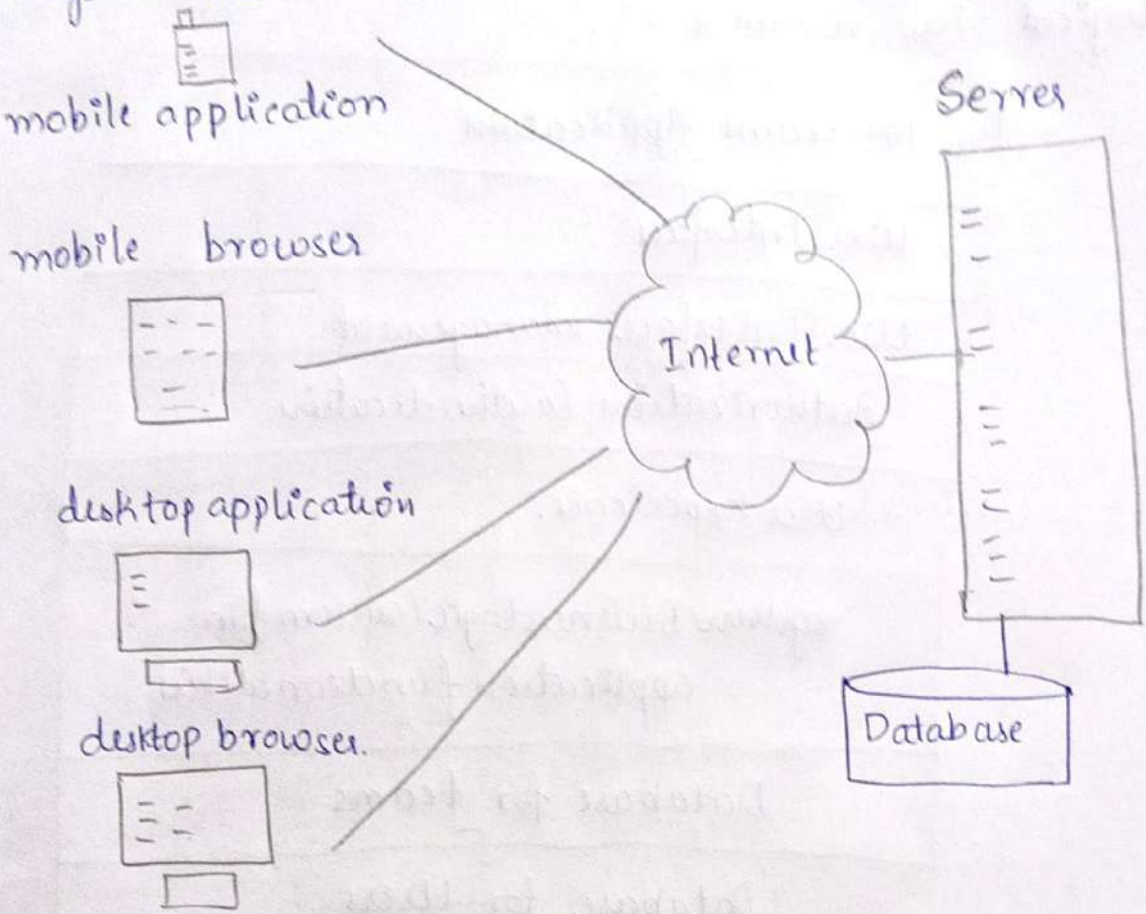
Using different components versions we build baseline (set of components considered for system)

Mainline - product line to be developed and released.

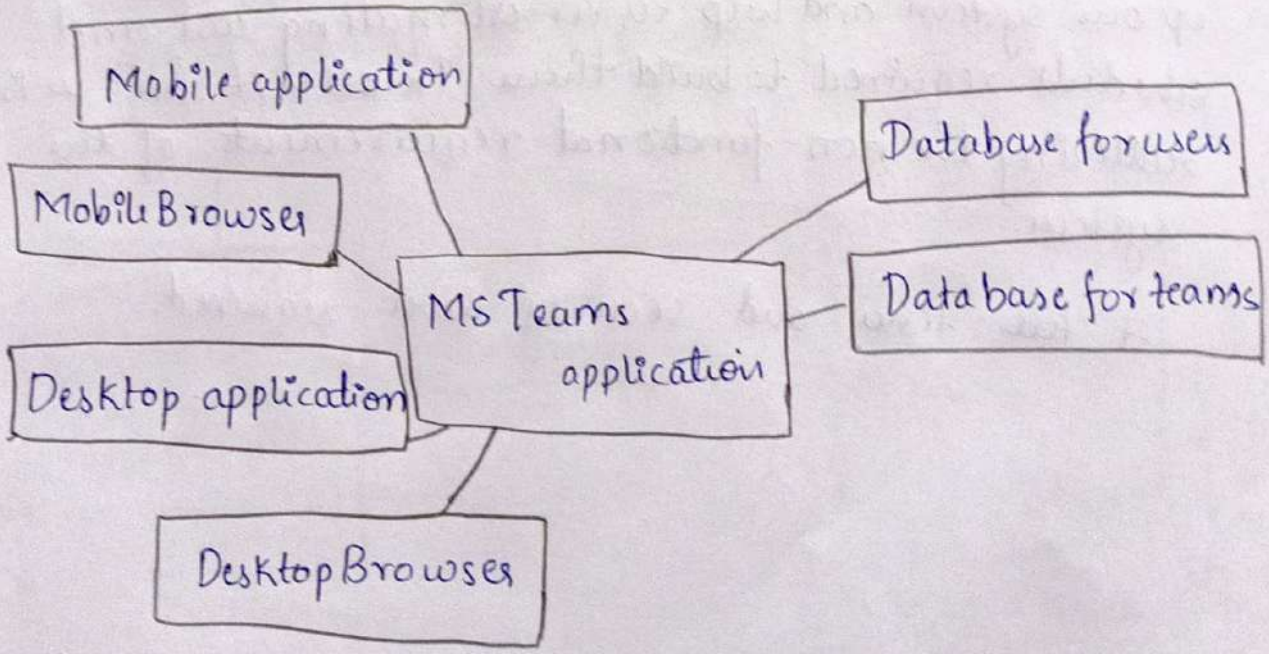
01fe18bcs211

3119

(6b) Physical view



logical view

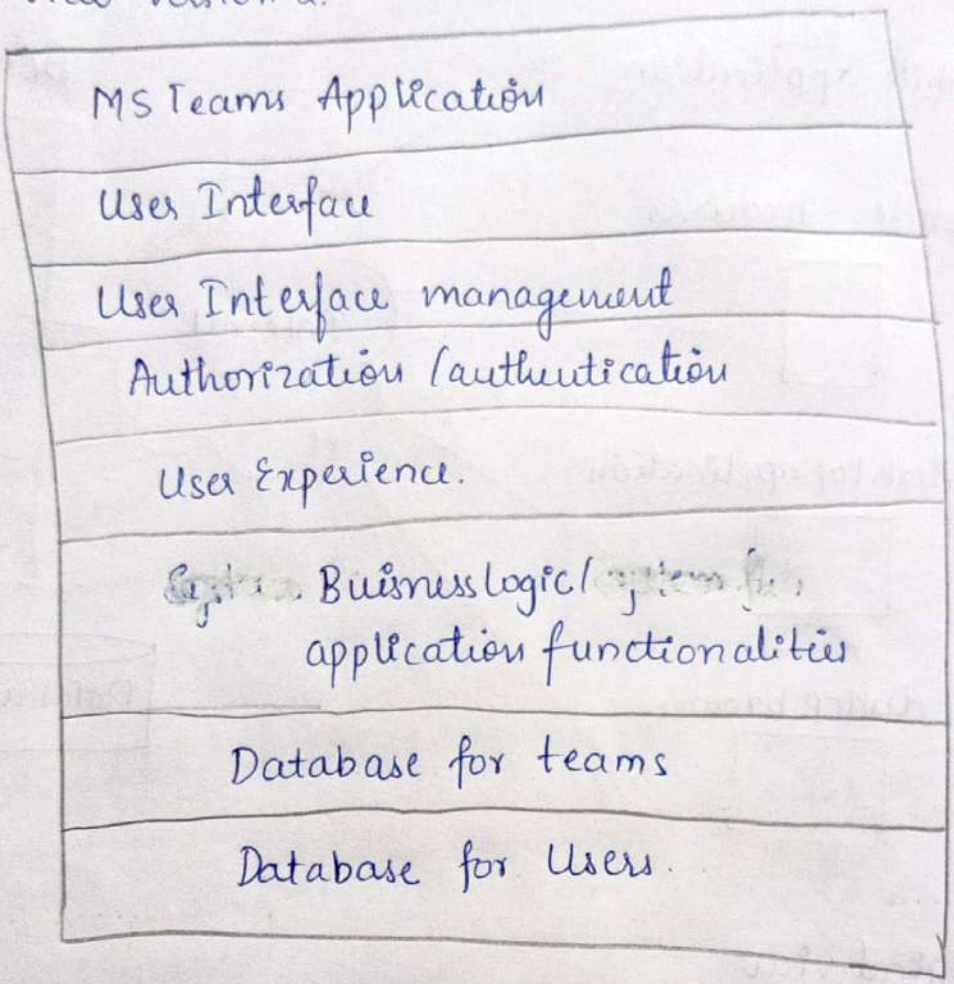


01fe18bcs211

4119

-OR-

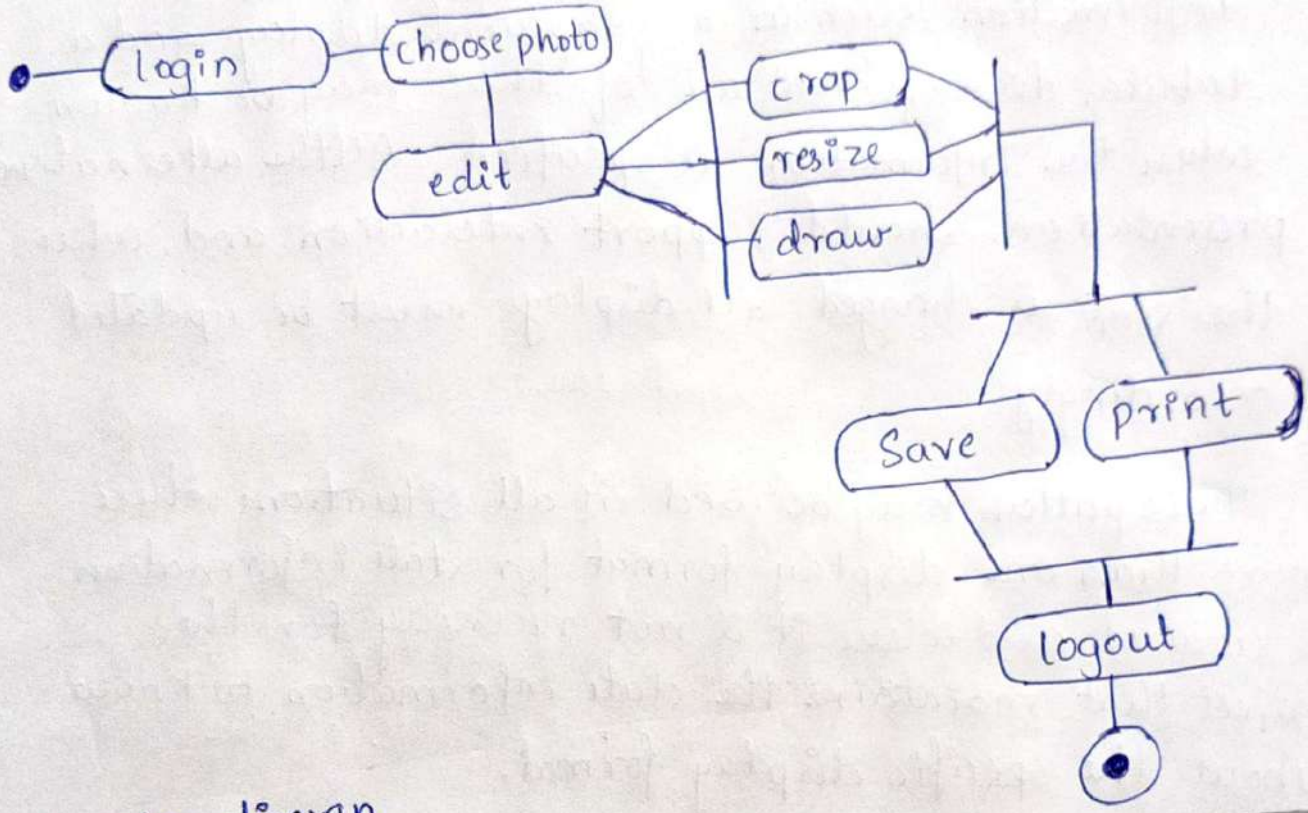
logical view version 2.



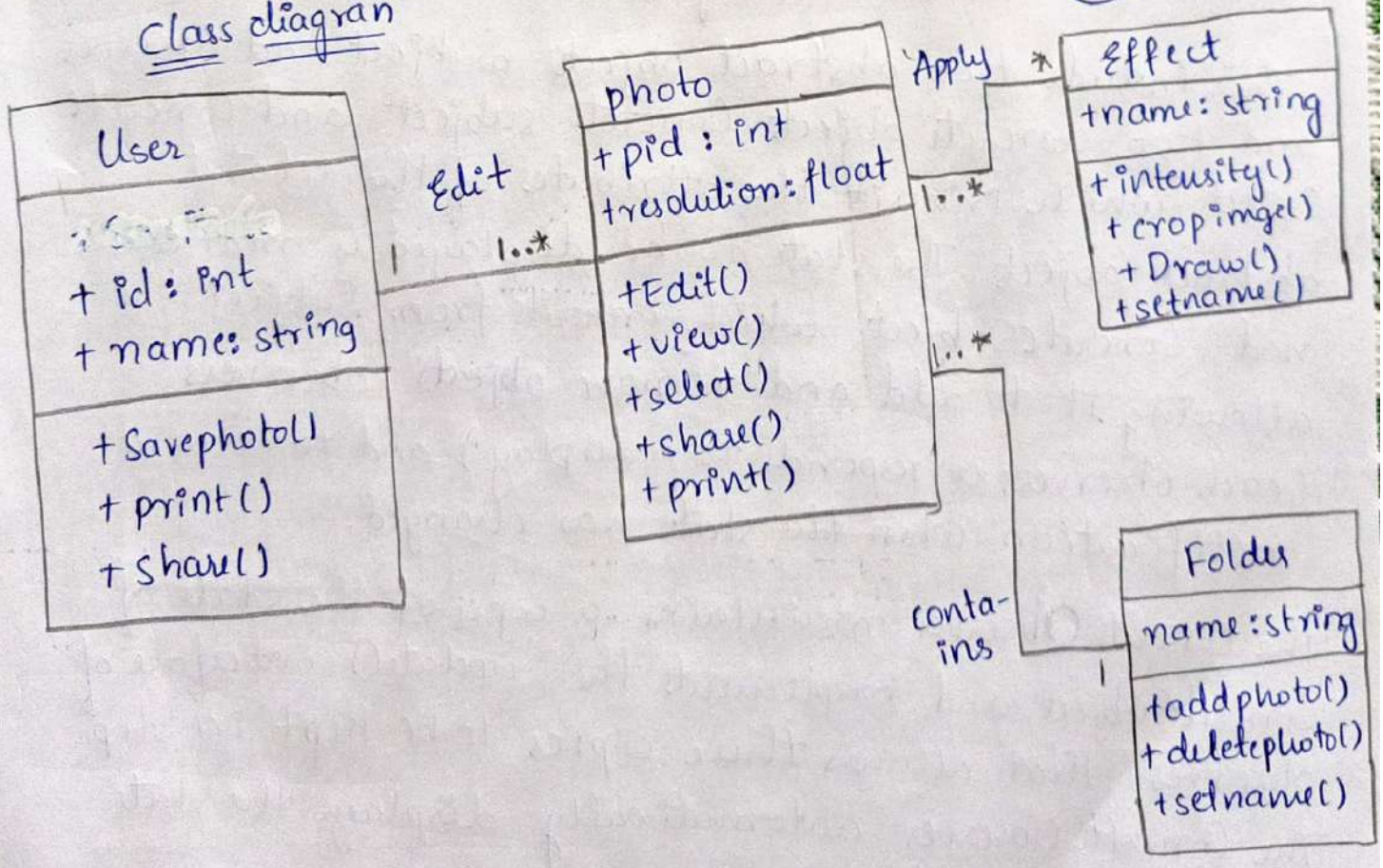
This views makes us realize what all components make up our system and help us in estimating cost and schedule required to build them. In all it help us in realizing the non-functional requirements of the system.

It tells time and security level required.

7b) State diagram



Class diagram



7a) It is used in many situations where you have to provide different or multiple displays of state information, such as a graphical display and a tabular display. Not all of these may be known when the information is specified. All the alternative presentations should support interaction and, when the state is changed all displays must be updated accordingly.

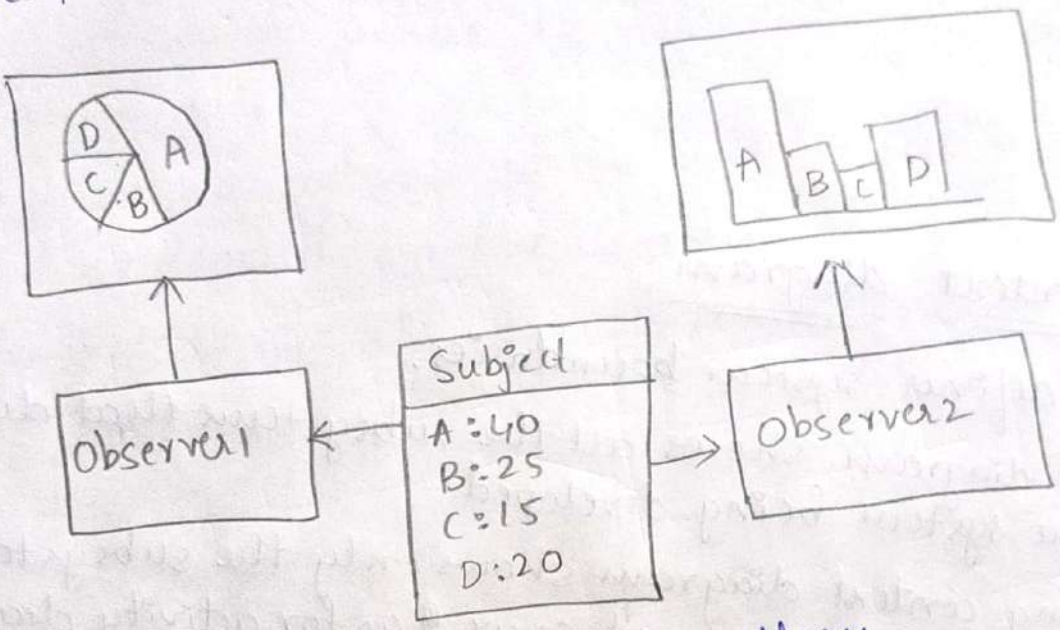
This pattern may be used in all situations where more than one display format for state information is required and where it is not necessary for the object that maintains the state information to know about the specific display format.

This include two abstract objects Subject and Observer and two concrete objects, Concrete subject and Concrete Object, which inherit the attributes of the related abstract objects. The state to be displayed is maintained ConcreteSubject which inherits from Subject allowing it to add and remove observers (each observers corresponds to a display) and to issue a notification when the state has changed.

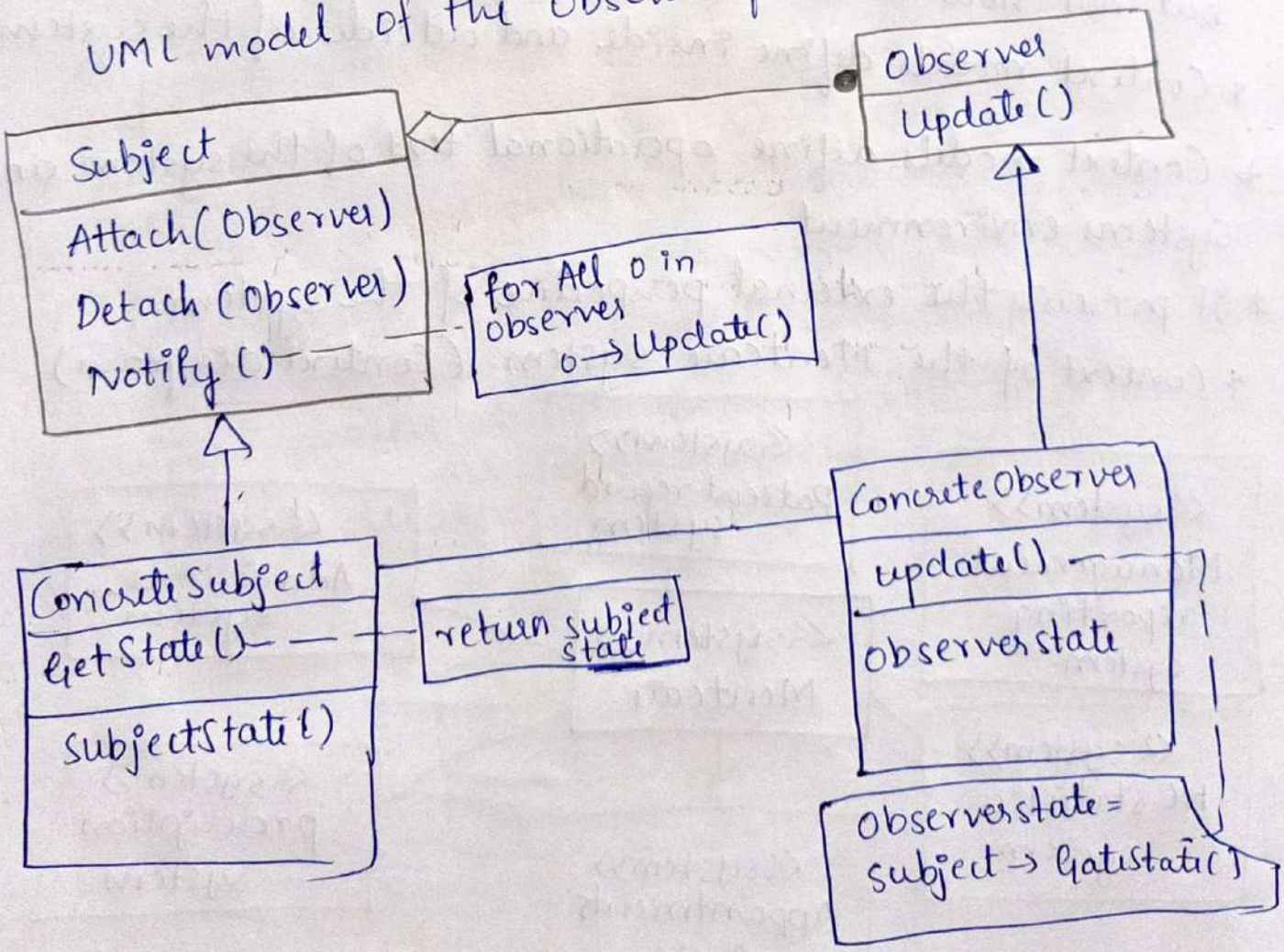
The Concrete Observer maintains a copy of the state of ConcreteSubject and implements the update() interface of observer that allows these copies to be kept in step. The ConcreteObserver automatically displays the state and reflects changes whenever the state is updated

01fe18bcs211
419

Example:
Multiple display of marks (One pie chart
One - histogram)

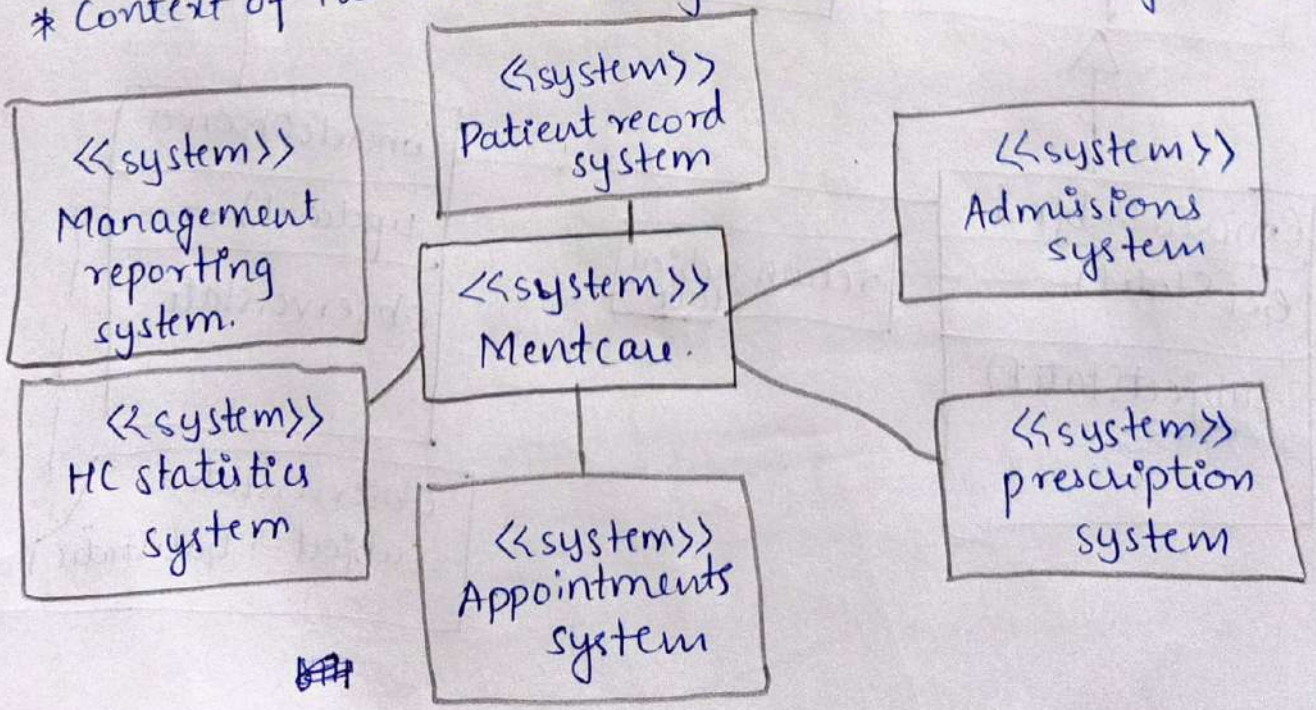


UML model of the Observer pattern.



6a) Context diagram

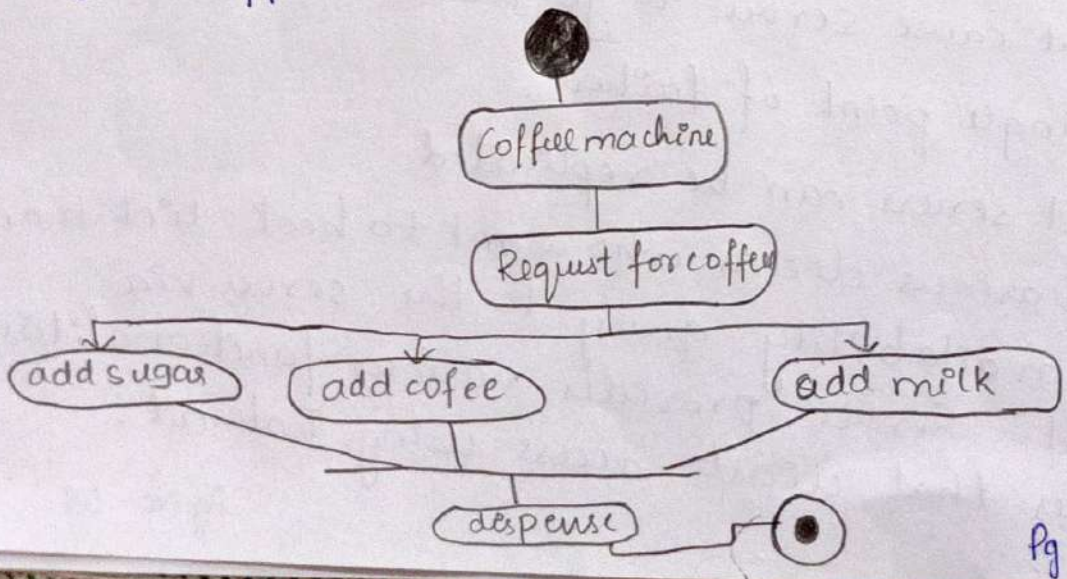
* It defines system boundaries.
* Context diagram shows all the subsystems that depends on the system being developed.
* Drawing context diagram shows only the subsystem but not how it's developed so we go for activity diagram.
* Context models define inside and outside of the system.
* Context models define operational text of the system and systems environment
* It presents the external perspective of the system.
* Context of the Mentcare system. (Context diagram)



~~01fe18bcs211~~

* It is a political judgement
* Social and organizational concerns may affect the decision where to position the system boundaries
* System boundaries show other systems that are used or depend on the system being developed.
* The position of system boundary effect the system requirements.
* There might be a pressure that increase/decrease the influence or workload of different parts of organization.
* Context models can be represented using activity diagrams which is modelling the process. Process models reveal how the system being developed is used in broader business process. Activity digram defines the set of activities that define system

Ex: Coffee dispenser



USN: 01fe18bcs211

Name: Sneha Majjigudda