Chapter 5 : Agile development :

Agile means flexible. So agile is all about flexibility.

What is Agility :

⇒ Agility encourages team structures and attitudes that make communication more facile.

⇒ It emphasizes rapid delivery of operational software and deemphasizes the importance of intermediate work products.

⇒ It adopts the customer as a part of the development team.

⇒ It recognizes that planning in an uncertain world has its limits and that a project plan must be flexible.

The Agile model :

⇒ Agile SDLC model is a combination of iterative and incremental process model.

⇒ Agile methods break the product into small incremental builds.

⇒ These builds are provided in iterations.

⇒ Each iteration typically lasts from about one to three weeks.

⇒ Every iteration involves cross functional teams working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing.

⇒ At the end of the iteration a working product is displayed to the customers and important stakeholders.

When to use agile model:

⇒ When new changes are needed to be implemented. The freedom agile gives to change is very important. New changes can be implemented at very little cost because of the frequency of new increments that are produced.

⇒ To implement a new features, the developers need to lose only the work of a few days, or even only hours, to roll back and implement it.

⇒ Unlike waterfall model in agile model, very limited planning is required to get started with the project. Agile assumes that the end users need are ever changing in a dynamic business and IT world.

⇒ When a highly qualified and experienced team is available and the project size is small.

Advantages:

⇒ Customer satisfaction by rapid, continuous delivery of useful software.

⇒ Customers, developers and testers constantly interact with each other.

⇒ Working software is delivered frequently.

⇒ Is a very realistic approach to software development.

⇒ Promotes teamwork and cross training.

⇒ Resources requirements are minimum.

⇒ Suitable for fixed or changing requirements.

⇒ Delivers early partial working solution.

⇒ Good model for environments that changes steadily.

⇒ Easy to manage.

⇒ Given flexibility to developers.

Disadvantages:

⇒ In case of some large software deliverable, it is difficult to assess the effort required at the beginning of the software development life cycle.

⇒ Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for newbie programmers.

⇒ Not suitable for handling complex dependencies.

⇒ More risk of sustainability, maintainability and extensibility.

⇒

Chapter-4: Principle that guide practice

4.1: Software engineering knowledge:

⇒ Software practitioners এরা মনে করে, Java, C++, HTML এর knowledge এর মতো software engineering টাও একটা exclusive knowledge.

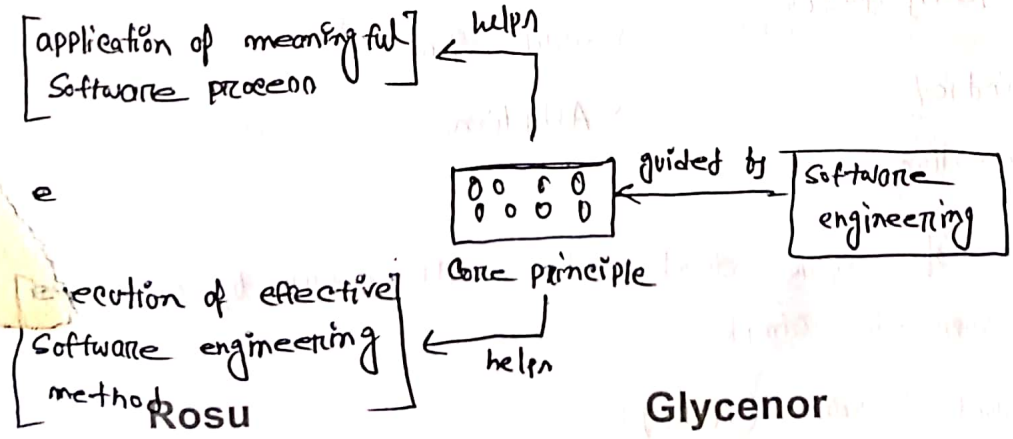
⇒ বিবেচনা করা হয় SE knowledge টার just 3 years half life আছে, কারণ আমাদের আজকে SE সম্পর্কে যা শিখতে হবে, দুই বা তিন বছর পর তা অপ্রাসঙ্গিক হয়ে যাবে,

⇒ কিন্তু SE এ কিছু principle যা কখনোই old হবে না, বড়ো একজন professional programmer কে lifetime support দিবে,

⇒ Core principal গুলো একজন SE কে কোন model, method, tools apply করতে হবে, সে সম্পর্কে অবগত রাখে,

4.2: Core principles:

Software Engineering is guided by a collection of core principles that help in the application of meaningful software process and execution of effective software engineering method.



Rosu Glycenor Insul

Core principle এর কাজ

Process level

→ Establishes a philosophical foundation

→ Performs framework & umbrella activities

→ Navigates the process flow

→ Produces a set a of software engineering products.

Practice level

→ Establishing a collection of values and rules.

→ Analyze a problem.

→ Design a solution

→ Implement and test the solution.

→ Deploy the software in the user community.

Principles that guide process:

The following set of core principle can be applied to the framework.

(i) Be agile: Being agile vs Doing agile

→ thinking process, approaches quickly করতে হবে

→ এটা mindset

→ transformation

→ quickly step নিতে হবে,

→ এটা practice

→ Adoption

The basic targets of agile development should govern by approach.

keep technical approach simple.

keep work product concise (সংক্ষিপ্ত)

make decision locally.

(ii) focus on quality at every step:

Exit condition for every process activity, action.

(iii) Be ready to adapt.

(iv) Build an quality-ful team.

(v) Establishing management of communication and coordination.

(vi) Manage change.

(vii) Annen tink.

(viii) Create work products, that provide value to others.

Principle that guide practice:

High quality operational software, যার কিছু function & feature আছে, সব গুলো stakeholder (র needs fulfill করে এমন software বানাতে কিছু core principles adopt করতে হয়, যা আমাদের technical work কে guide করবে।

(1) Devide and conquer: Stated in a more technical manner, analysis and design should always emphasize separation of concern.

(2) Understand the use of abstraction: An abstraction is a simplification of some complex element of a system used to communication, meaning in a single phrase.

(3) Strive for consistency: A familiar context make software easier to use.

Rosu Glycenor Insul

④ Focus on transformation of information:

⑤ Build software that exhibits effective modularity.

⑥ Look for patterns.

⑦ When possible, represent the problem and solution from a number of different perspective.

⑧ Remember that some one will maintain the software.

Principles that guide each framework:

① Communication principle:

→ Listen
→ Prepare before you communicate
→ Some one should facilitate the activity.
→ Face to face communication is best.
→ Take note and document decisions.
→ Strive for Collaboration
→ Stay focused
→ If something unclear, draw a picture.
→ Agree & Move on
Can't Agree: Move on
If features & function isn't clarified: Move on
→ Both parties win in negotiation.

Planning principle:

→ Project scope বুঝা,

→ Planning এ সব stakeholder দের ডুকানো,

→ Planning is ~~Heuristic~~ Iterative.

→ Estimate based on what u know.

→ Risk consideration.

→ Be realistic.

→ Adjust granularity → level of detail. as u define the plan.

→ Describe how u accomodate change

→ Define how u ensure quality.

→ Track the plan.

Deployment principle:

→ Customer এর expectation manage করা,

→ A complete delievery package

→ Software deliver হলে এর support regime ঠিক হবে,

→ Appropriate instruction must be provided

→ Bug should be fixed.

Chapter : 5

Requirement Engineering :

Inception :

Random public কে analyze করে তাদের problem খুঁজে বের করা - এই problem indentify করা, problem analyze করা, understand করা, Problem এর সকল aspect বুঝা, যাতে solution পেতে বেশি problem না হয়,

Problem এর nature and scope define করা হয়,

Solution এর idea চেয়া, examine, loopholen খুঁজা, solution কাদের জন্য, এদের economical benefits কি, business value কি - সব analyze করা হয়,

Elicitation :

client কি চায়, তা জানা, কখন কোন problem তৈরি হয়.

→ Problem of scope : client unnecessary requirement বলে যার ফলে software team confused হয়ে যায়, এতে system boundary ill defined হয়,

→ Problem of understanding : client নিজেই confused, সব details mention করতে না,

→ Problem of volatility : Over the time, client তার requirement change করতেই থাকে,

Elaboration: Inception ও Elicitation এ যেসব Info এর জন্ম base করে Elaboration এ details এ কথা হয়।

Elaboration task আমাদেরকে Refined Requirement model develop করতে help করে, এটা আমাদের বলে End User কিভাবে System এর সাথে interact করবে,

Negotiation: Budget কম But requirement বেশি, resource কম Demand বেশি কোন time কম দিয়ে কাজ করাতে চায়, Unrealistic problem/requirement identify করা লাগে, এভাবে win win situation create করি,

Specification: functional/Non-functional requirement কে specify করা, Specification যেকোন written document, a set of graphical models, a form of mathematical model, a collection of usage scenarios হতে পারে, specification খুবই flexible

Validation: work product কে validate করা হয়, এখানে specification কে examine করা হয়, ambiguous nature identify করা, requirement এ inconsistency identify করা, error identify করা requirement এ priority assign করা,

requirement management: কিভাবে বারবার requirement change হয়, এগুলোর requirement গুলোকে monitor, track, control করা হয়।

5.4: ~~use case~~:

Stack holders:

Anyone who benefits in a direct or indirect way from the system which is being developed.

Stackholders are: Business operations managers, product managers, marketing people, internal and external customers, end users, consultants, product engineers, software engineers, support and maintanance ~~engin~~ engineers,

5.4: use case:

→ Captures the functionality of system from user perspective.

OOP তে UML diagram এ use case diagram use করা হয়,

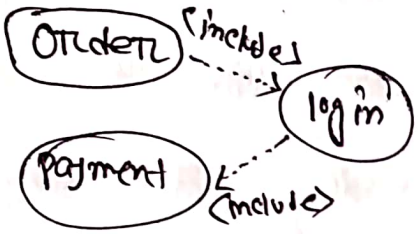
UML - Unified Modelling Language.

Actor: যে system এর সাথে interact করে,

use case: functionality / operation.

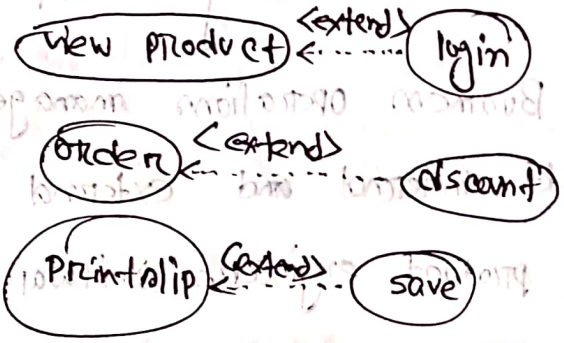
link → Actor এর সাথে use case এর link.

Arrow - generalization.



login ছাড়া order payment possible না।

include ও obiously



login ছাড়া view product এর ও যায়।

extend optional

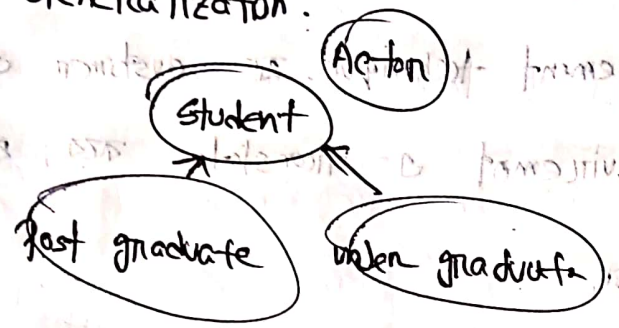
Requirement engineering এ participants:

→ Software engineer

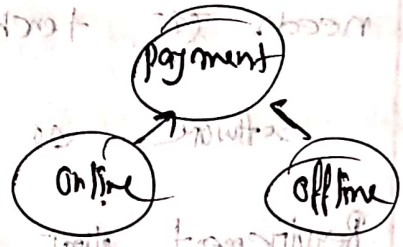
→ Manager

→ Customer/User

Generalization :



use case



Student post graduate/under graduate দুইটি হতে পারে,

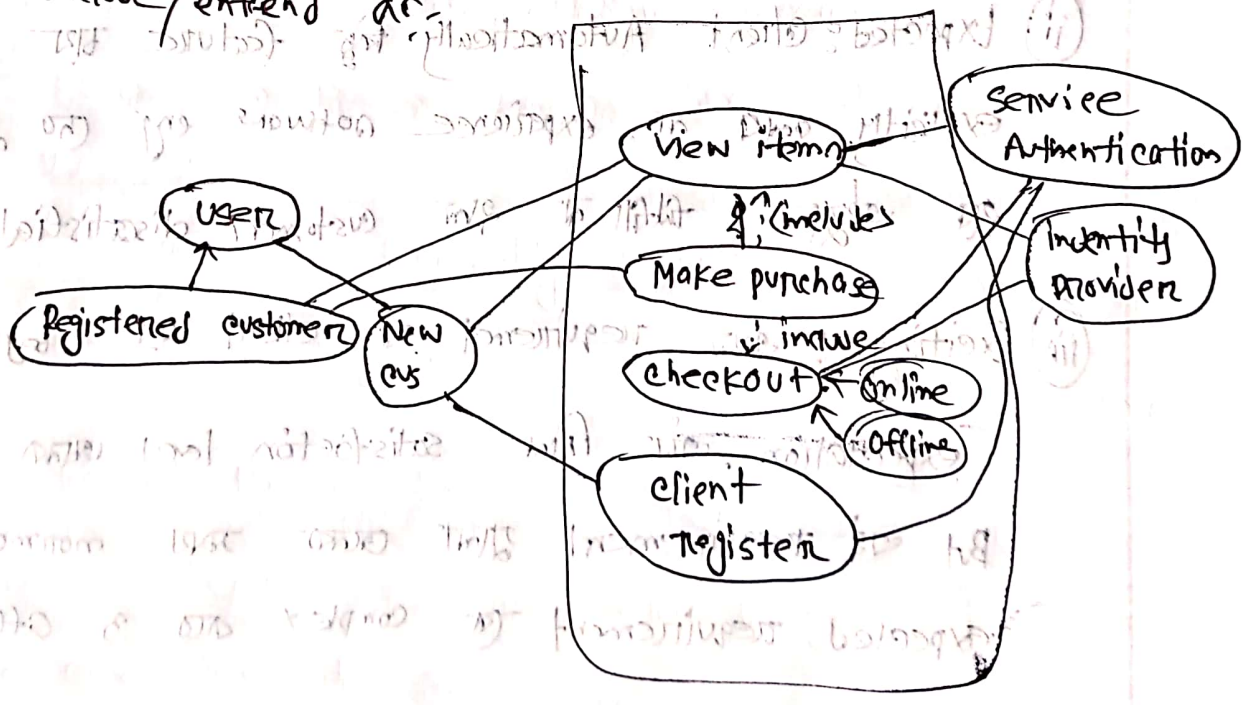
payment online/offline দুইটি হতে পারে,

Online Shopping

(i) Actor থাকবে দুইটি,

(ii) process/function

(iii) include/extend এর,



5.3.2: Quality function Deployment:

এটি একটি quality management technique. যা customer এর need কে technical requirement এ translate করে, যার Software এর ভালো লাগে,

Requirement গুলো meeting, report, document থেকে নেয়া হয় এবং requirement table বানানো হয়।

Requirement ৩ প্রকার -

(i) Normal: Basic Rule যা meeting এর মাধ্যমে stackholder থেকে সরাসরি নেয়া হয়, এগুলো implement হলে Stackholder satisfy হবে।

(ii) Expected: Client Automatically কিছু feature চায় যা তারা হয়তো explicitly বলবে না, experience software এর দেখে বুঝে নিতে হবে এইগুলো fulfill না হলে customer dissatisfied হবে,

(iii) Exciting: এমন requirement যা client এর beyond the expectation. এটি দিলে satisfaction level অনেক high হবে, But এই requirement গুলো অনেক সময় normal ও expected requirement কে complex করে ও affected করে।

5.3.4:

Elicitation work product:

Requirement elicitation এর জন্য depend করে কোন work product দিবে। system এর size এর উপর depend করে change হবে, Work product দিতে পারে-

→ A statement of need and feasibility.

→ Bounded system of work

→ List of customer who present in the req. elicitation.

→ System or technical environment

→ A set of usages scenario.

5.4: Element of requirement Model.

Representation of different mode use করার different perspective থেকে দেখা সম্ভব হয়।

Chapter - 6

Rules of thumb:

→ The model should focus on the requirements that are visible within the problem or business domain

→ Each element of the requirement model should add to an overall understanding of software requirements.

→ Delay consideration of infrastructure and other non-functional models until design.

→ Minimize coupling throughout the system.

→ Be certain that the requirement model provides value to all stack-holders.

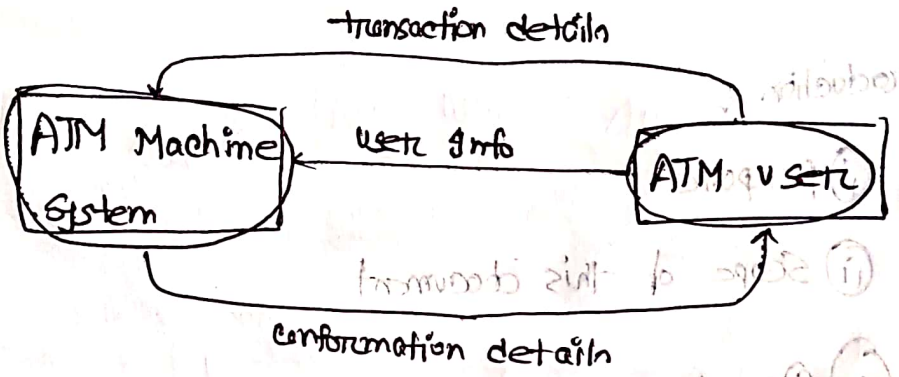
→ Keep the model simple as it can be.

* Write down the structure of a requirement document.

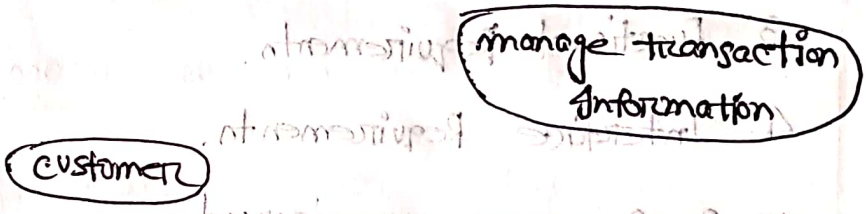
⇒

1. Introduction :
   - (i) purpose
   - (ii) Scope of this document
   - (iii) Overview
2. General description
3. Functional Requirements.
4. Interface Requirements.
5. Performance requirements.
6. Design constraints
7. Non-functional attributes
8. Preliminary scheduled and Budget
9. Appendices.

DFD → Data Flow Diagram.

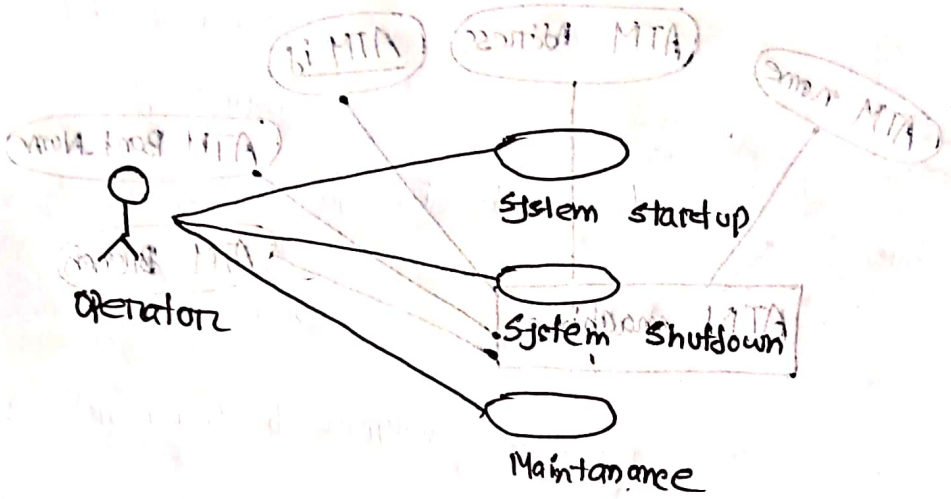


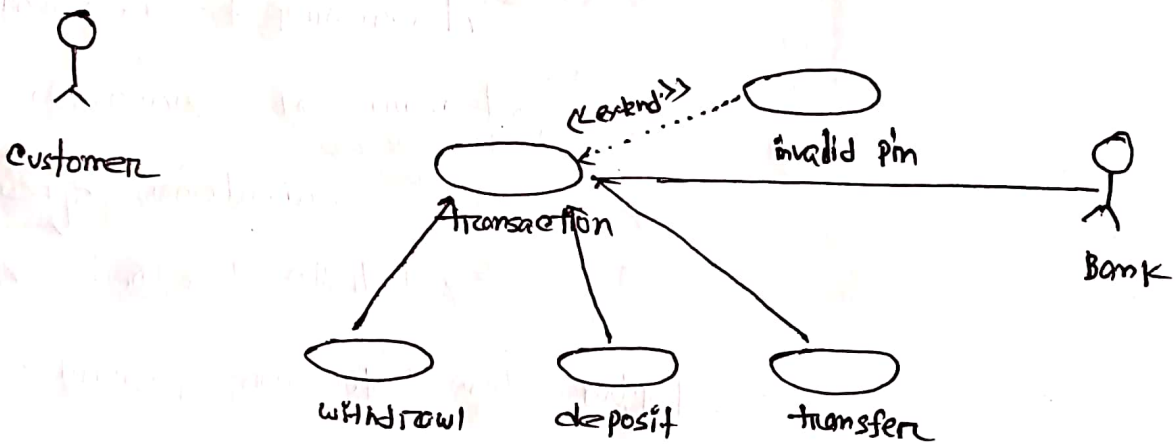
level-0



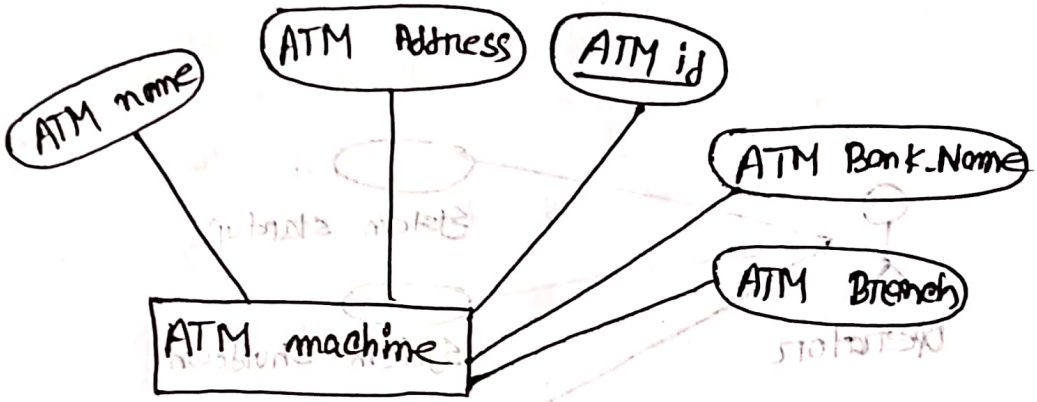
ATM A Machine

Use case

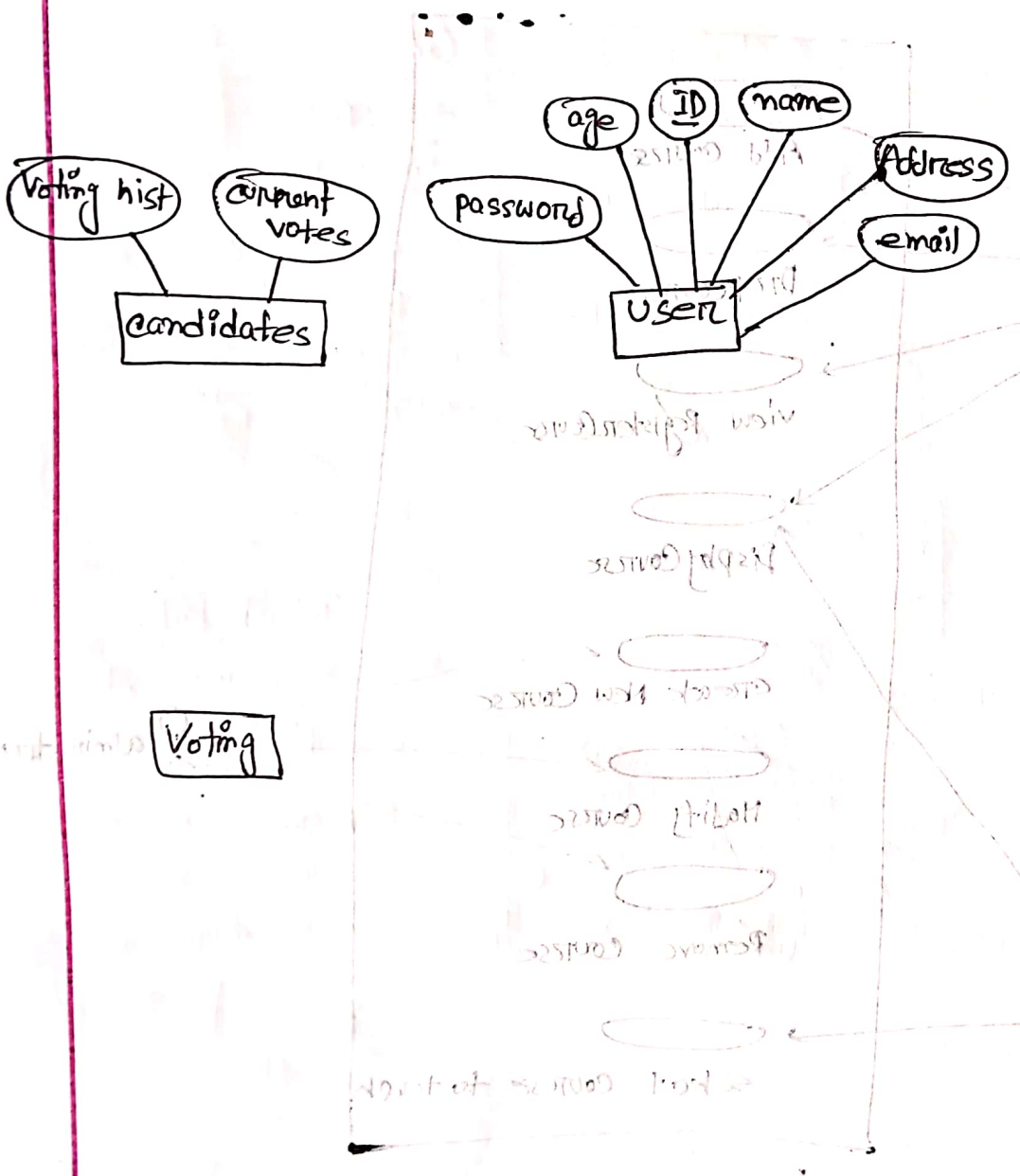




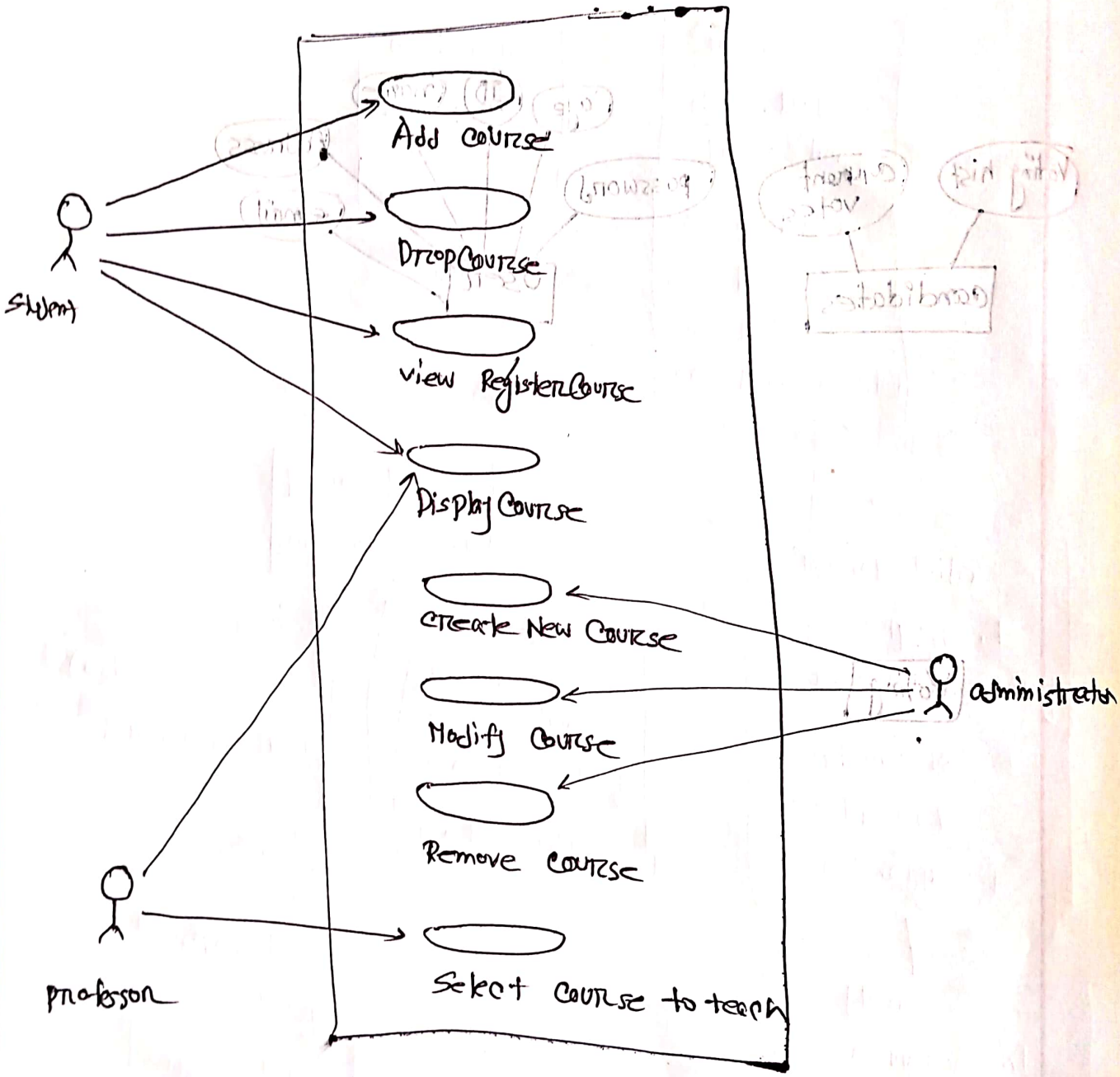
ER-diagram:


ATM name
ATM Address
ATM id
ATM Bank_Name
ATM Branch
ATM machine

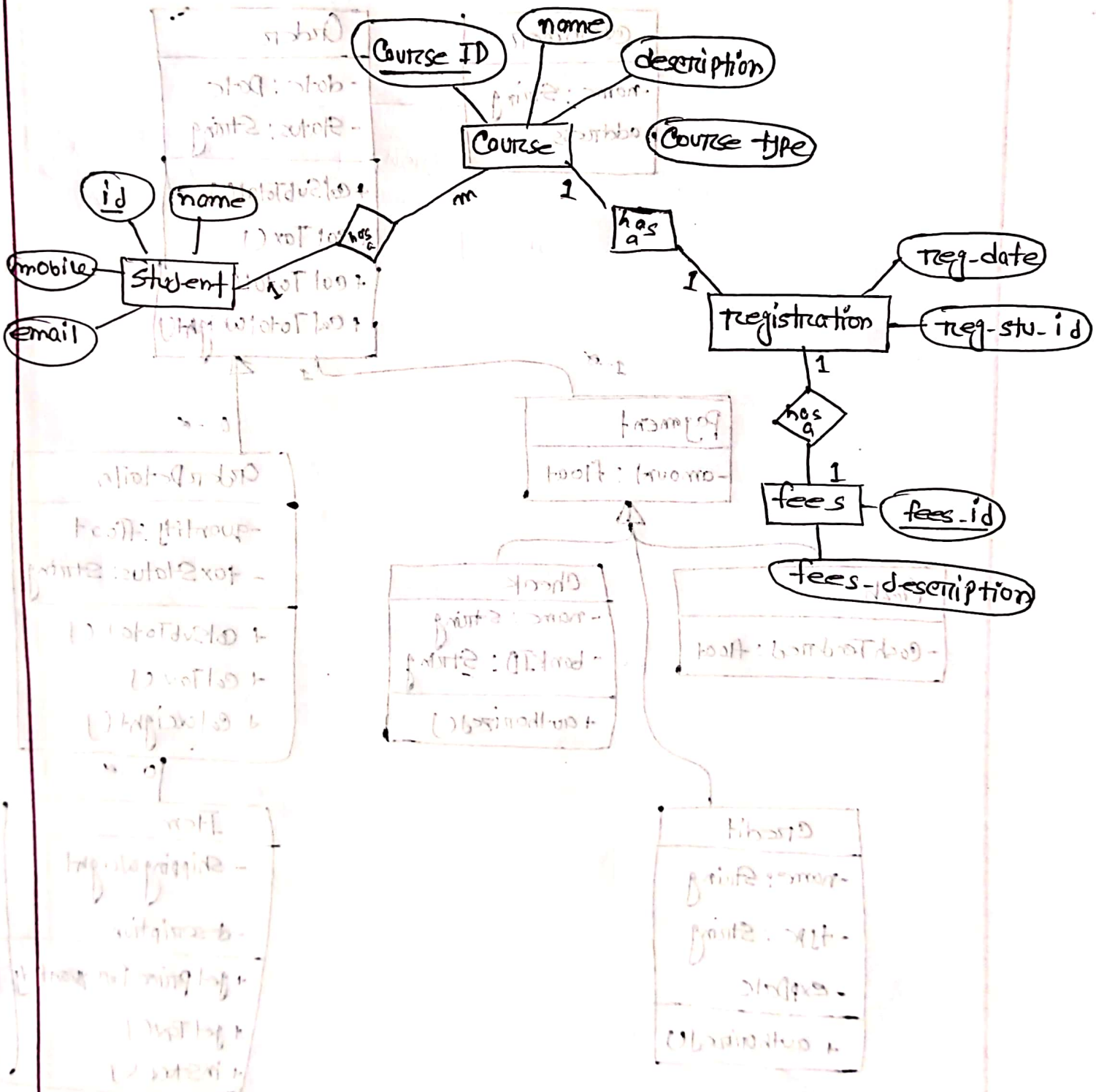
Voting system ERD-



Student Registration use case diagram:

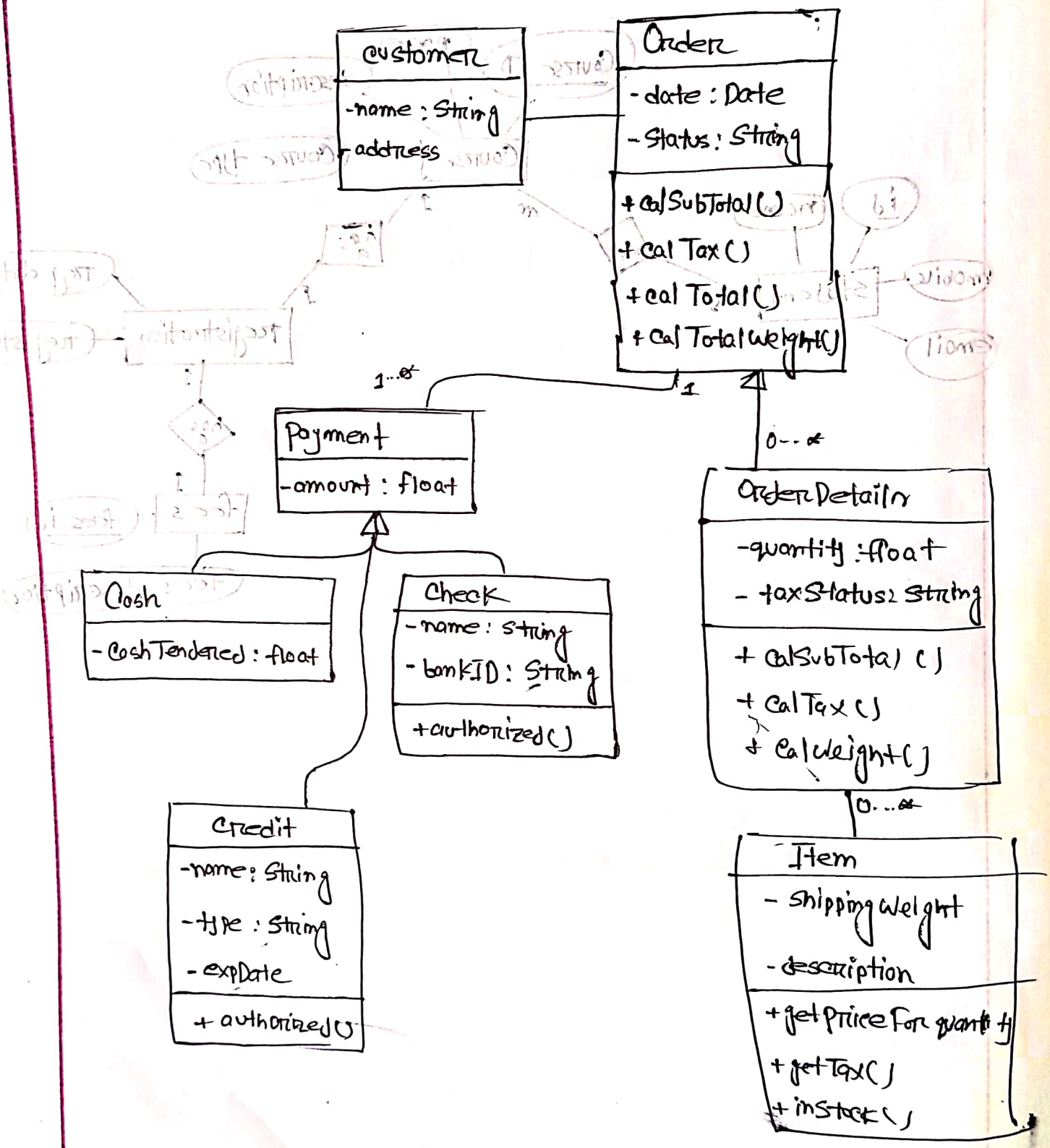


ER - diagram


Course ID
name
description
Course
Course type
id
name
has
m
1
has
mobile
Student
1
email
1
registration
reg-date
reg-stu-id
1
has
1
fees
fees_id
fees-description

Constraints

Online order system, Class diagram.



Chapter-6

Requirement Modeling

Date

RM

what is it ?

→ Requirement modeling is a combination of text and diagrammatic forms to depict requirements in a way that is relatively ~~under~~ - easy to understand and more important straight forward to review for correctness, completeness and consistency.

why important ?

to validate software requirements, you need to examine them from a number of diffenent point of view.

Requirement modeling from three prespective

1. Scenario - based - models
2. Data / information models
3. class - based models

How Do I ensure that I have done it right?

Requirement modeling work products must e reviewed for correctness, completeness, nd consistency. They must reflect the eeds of all stakeholders and establish a foundation from which design can be conducted

Requirement analysis

Date

- results in the specification of software's operational characteristic
- indicates software's interface with other system elements
- ~~of~~ establishes constraints that software must meet
- ~~to~~ allow a software engineer to be called as a analyst

Type of RM

① Scenario based modeling
(based on point of view of acton)

② Data model
(depict information domain for problem)

③ Class-oriented model :-

(represent object-oriented classes)

④ Flow-oriented model

(represent functional element of system)

⑤ Behavioral models

(depict how software behave as consequence of external events)

Objective of Requirement Model

Date

- to describe what customer requires
- to establish a basis for the creation of a software design
- to define a set of requirements that can be validated once the software is built.

Analysis Model → is a bridge between system description and the design model

- is a model that should describe what the customer wants, establish a basis for design, and establish a target for validation.

Basic guidelines that can help us as we do requirement analysis work?

Rules of thumb → followed creating on a analysis model.

1. The model should focus on the ~~problems~~ requirements that are visible within problem on bussiness domain
2. Each element of the requirements model should add to an overall understanding of software requirements
3. Delay consideration of infrastructure and other non-functional model until design.
4. minimize coupling throughout the system

Date

(5) Be certain the requirements models provides value to all stakeholders.

(6) Keep the model as simple as it can be

Domain analysis :-

- software domain analysis is the identification, analysis and specification of common requirements from a specific application domain, typically for reuse on multiple projects within that application domain.

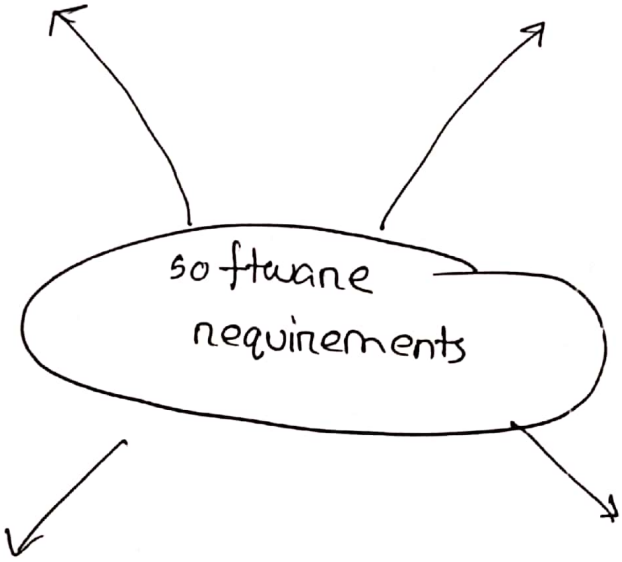
- The intent is to identify common problem solving elements that are applicable to all applications within the domain

elements of analysis model

(what different point of view can be used to describe the requirement model)

Scenario based models
- use case
- user stories

class model
- class diagrams
- collaboration diagrams



Behavioral Model
- state diagram
- sequence diagram

Flow Models
- DFDs
- Data Models

Date :

why should build models ?
why not just build the system itself?

- we can construct models in such a way as to highlights on emphasize, certain critical features of a system, while simultaneously de-emphasizing other aspects of the system.

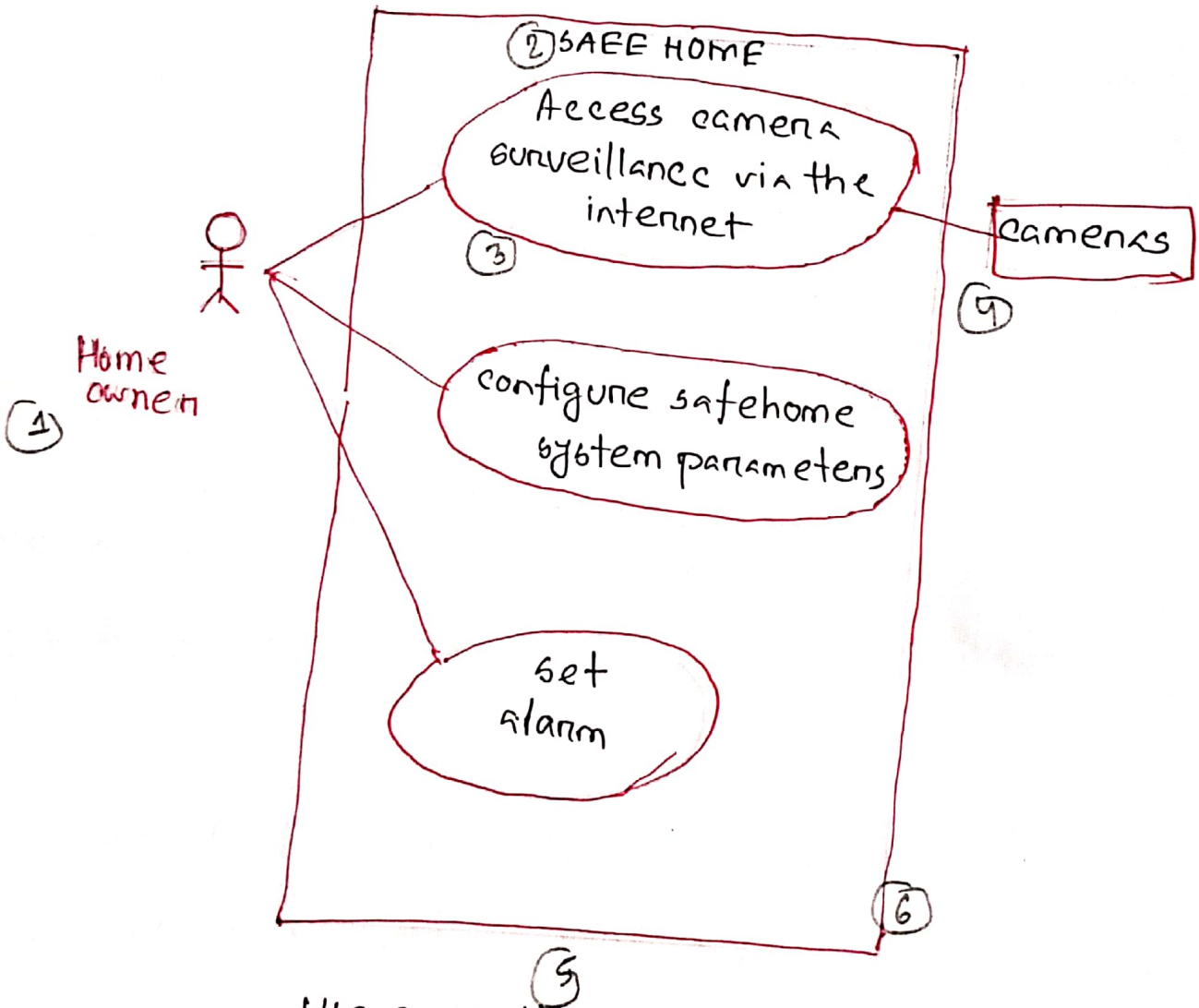
what is use cases? xxx

- are simply an aid to defining what exists outside the system (actors) and what should be performed by the system.

- captures the interactions that occur between producers and consumers of information and system itself.

Scenario base modeling

1. creating a Preliminary Use case → primary
2. Refining a preliminary Use case
3. writing a formal Use case



Use case diagram for safehome

How do I examine alternative courses of action when I develop a use case? (Refining) use case

- Can the actor ~~task~~ take some other action at this point?
- Is it possible that the actor will encounter some error condition at this point? If so, what might be?
- is it possible that actor will encounter some other behavior at the point (behavior that invoked by some event oustide actor's control)

elements

UML MODEL

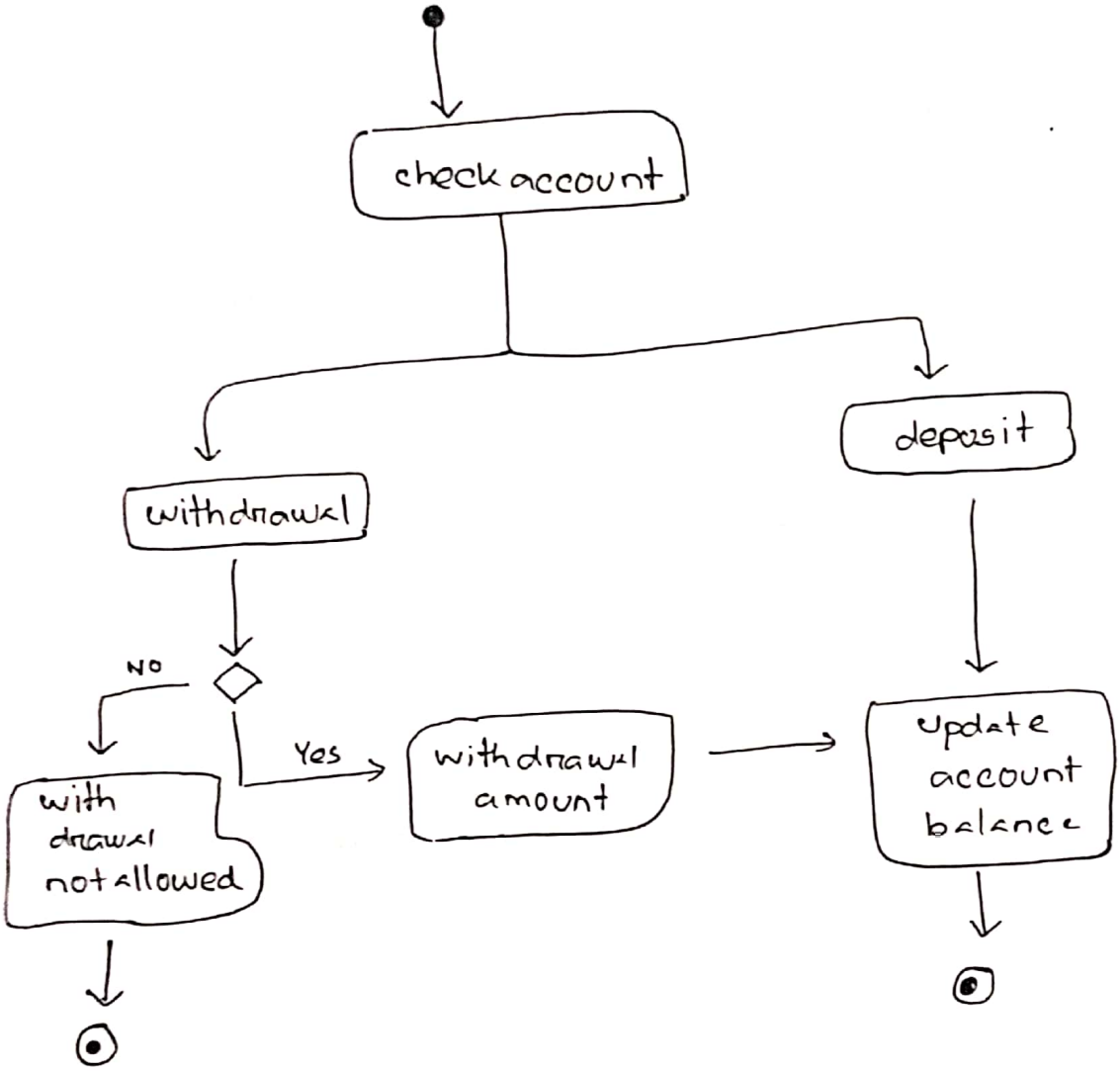
UML = Unified Modeling Language

UML activity diagram :-

→ Represents the actions and decisions that occur as some function is performed

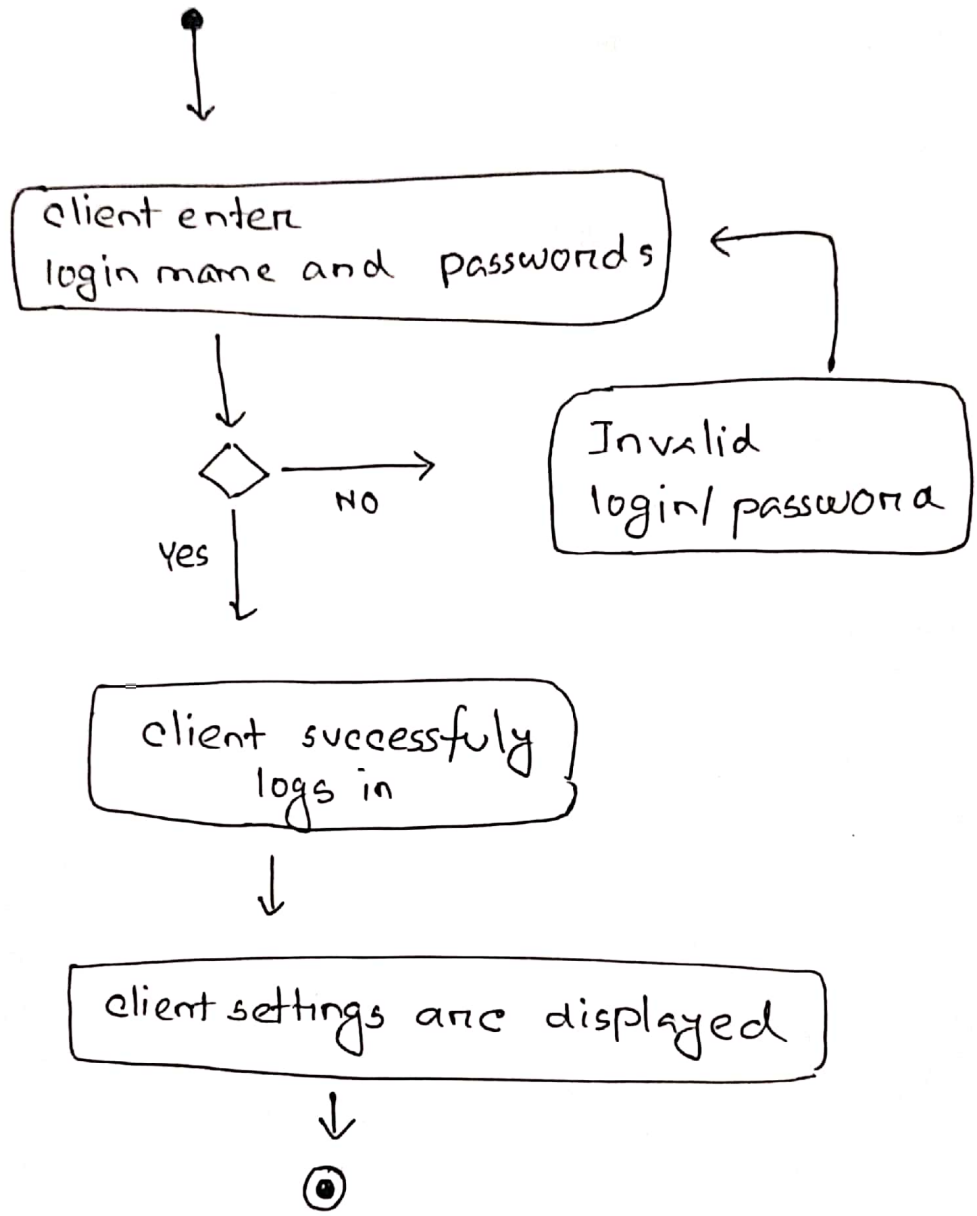
→ supplements the use case by providing a graphical representation of the flow of the interaction within a specific scenario.

Activity diagram for a banking system

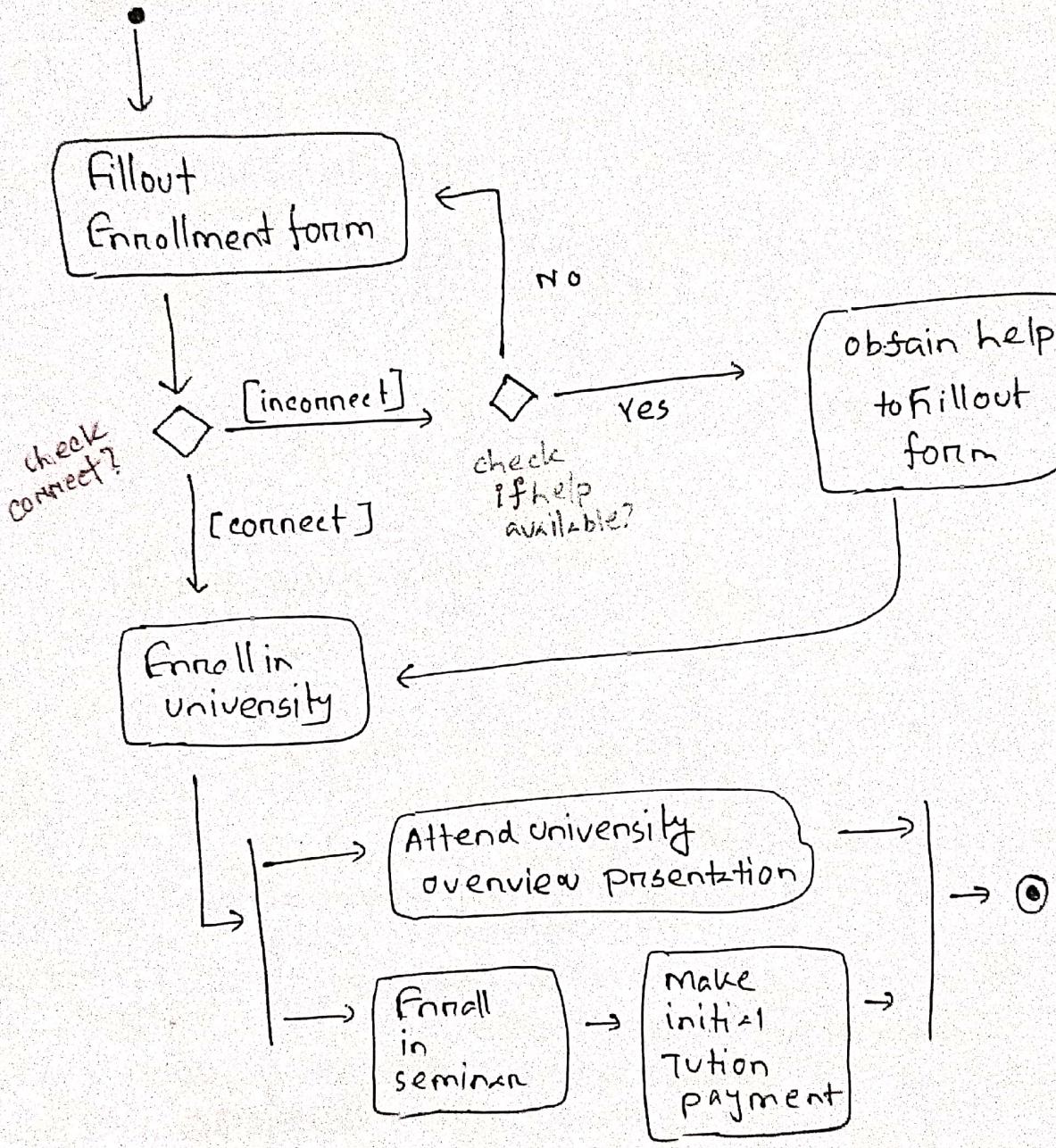


Bestron™

Activity diagram of login page



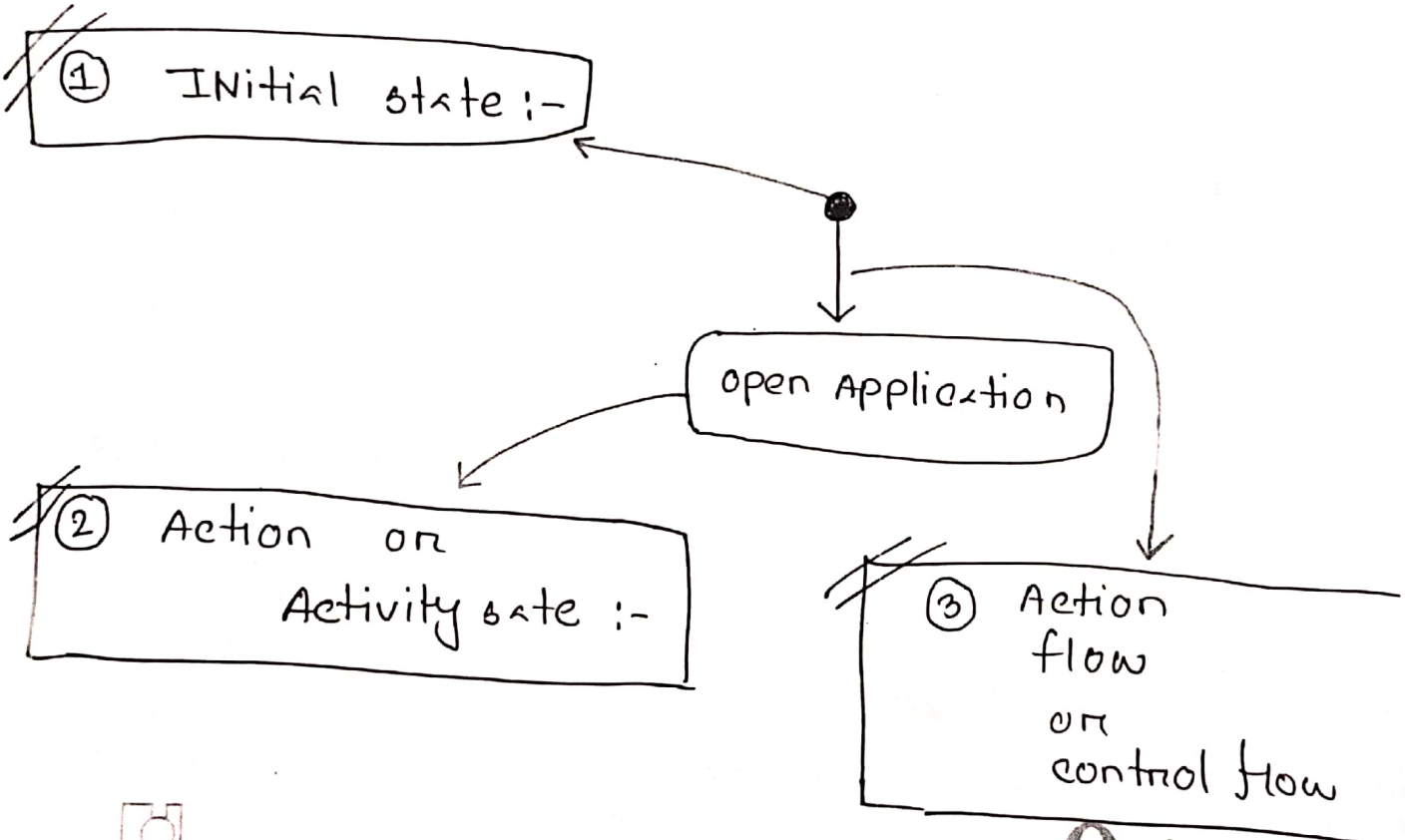
Activity diagram of student enrollment



→ refers to the steps involved in the execution of a use case

→ illustrate the flow of control in a system

Activity diagram notation :-

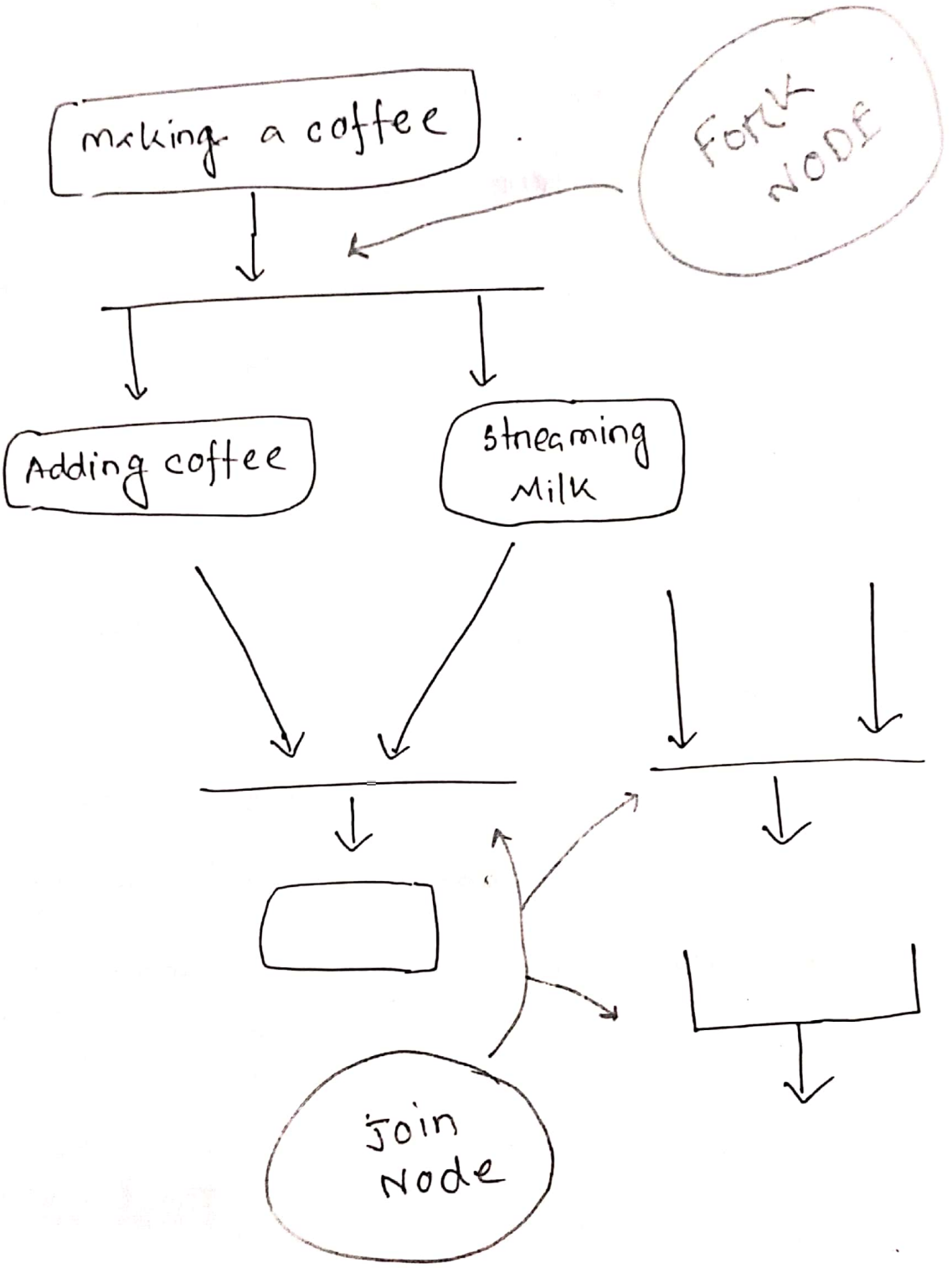


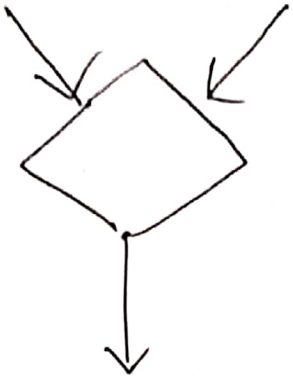
Activity diagram

→ is basically a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

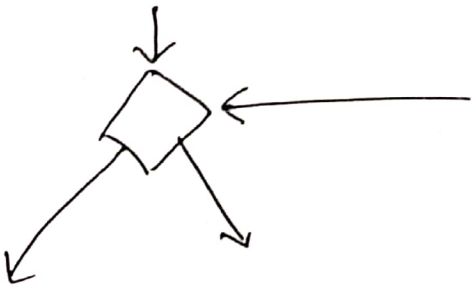
Swimlane diagram

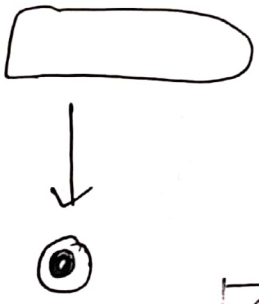
→ is type of flowchart, it diagrams a process from start to finish, but it also divides these steps into categories to help distinguish which departments or employee are responsible for each set of actions.





Merge Node



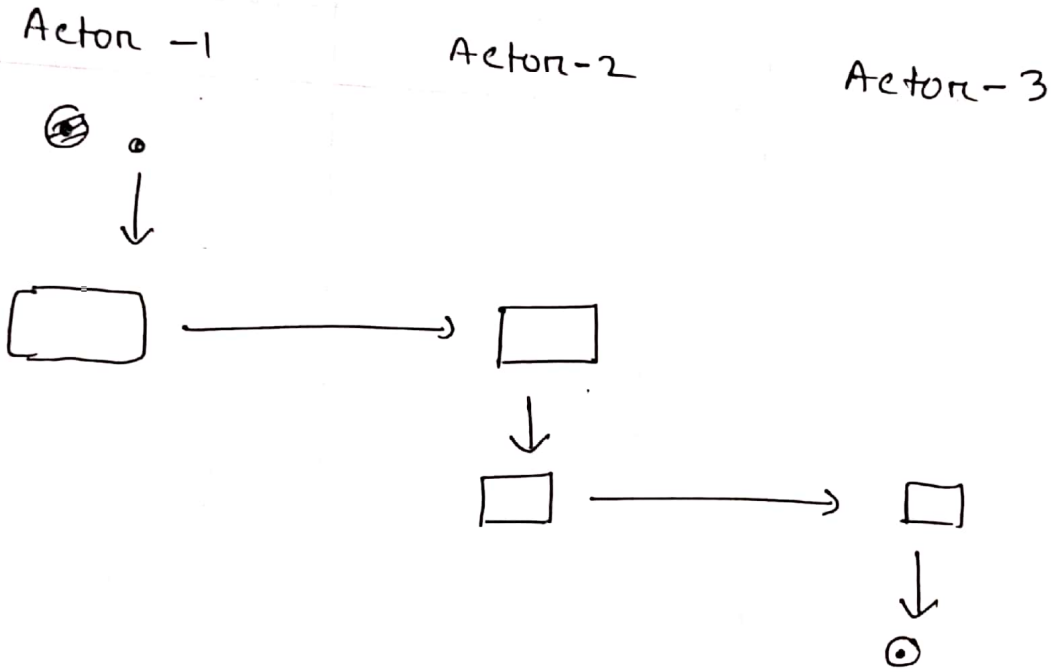


END NODE

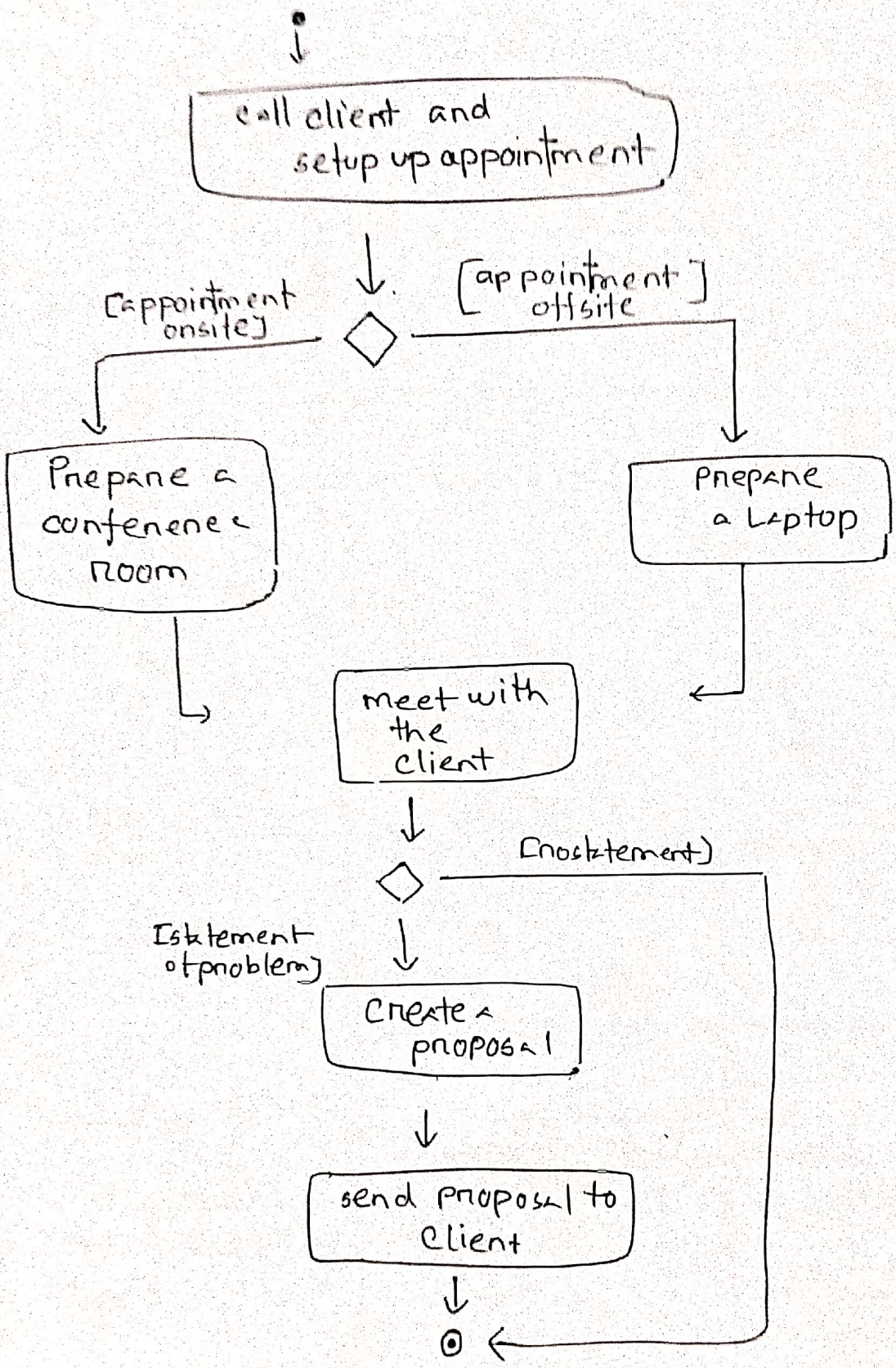
Swinlane Activity Diagram

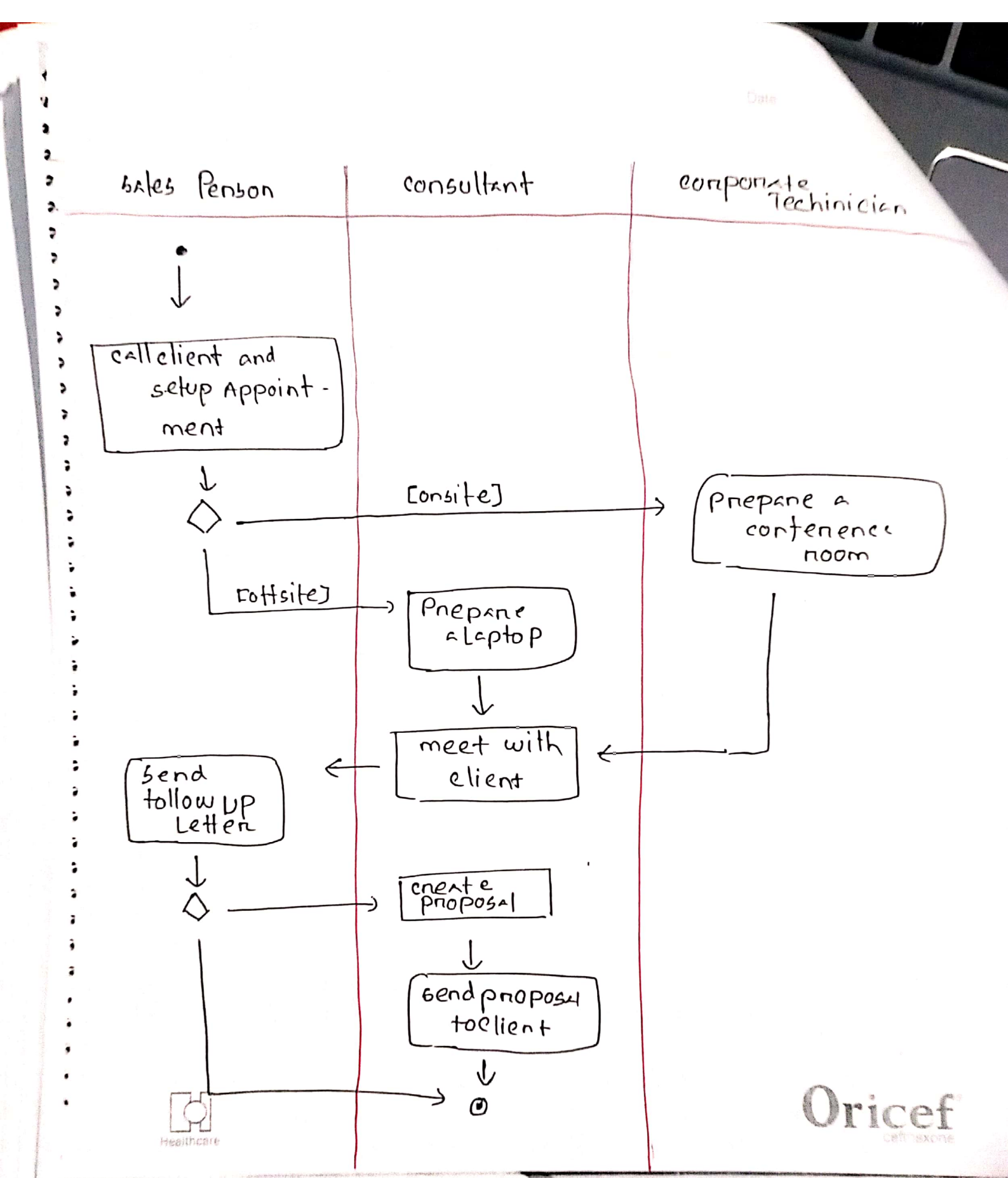
Date

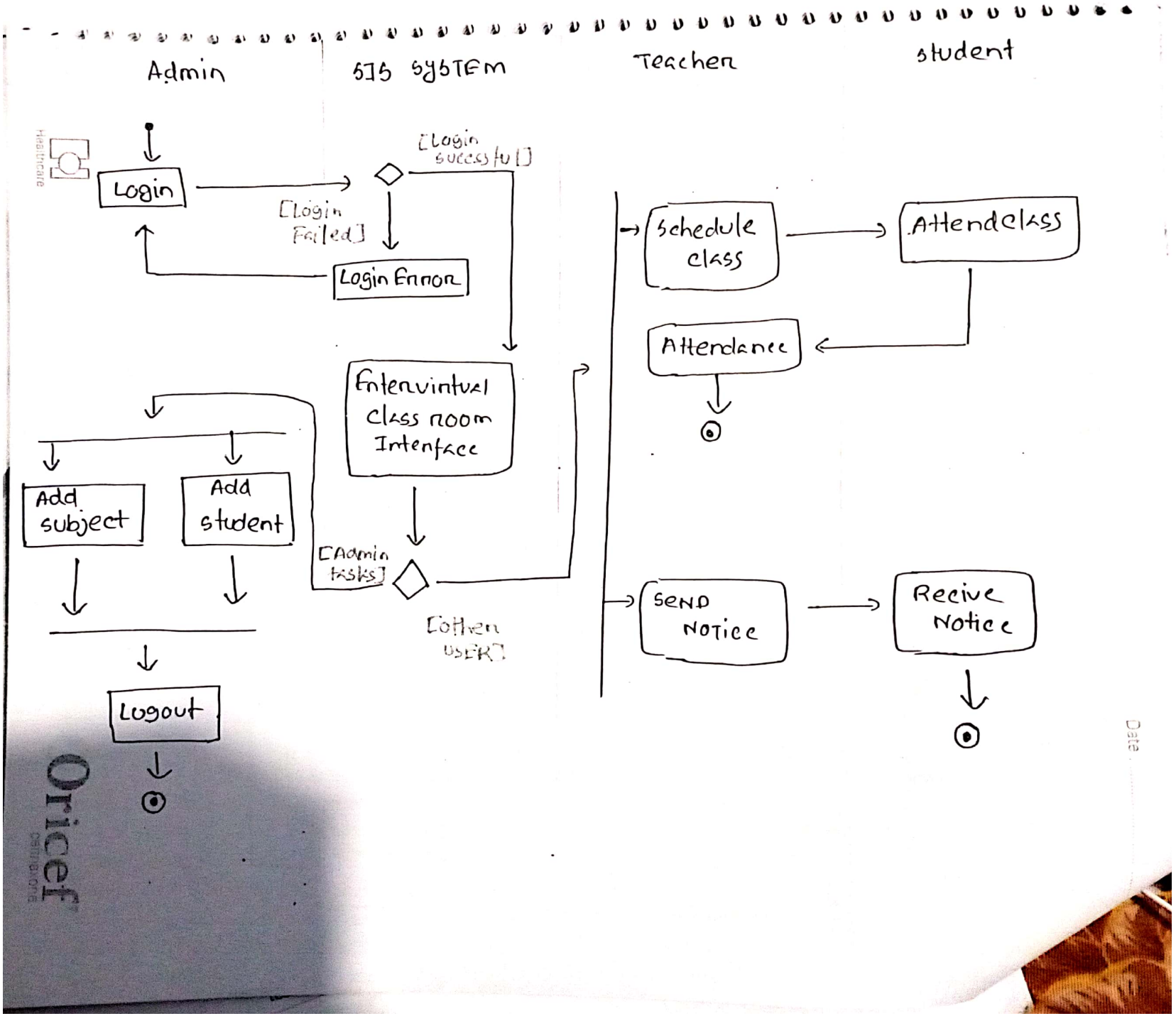
A swimlane is a way to group activites performed by the same actor on an activity diagram on to group activites in a single thread.



the business process for meeting a new client




Sales Person
consultant
corporate Technician
call client and setup Appoint-ment
[onsite]
Prepare a conference room
[offsite]
Prepare a Laptop
meet with client
send follow up Letter
create proposal
send proposal to client
Healthcare
Oricef


Admin
SIS SYSTEM
Teacher
student
Login
[Login successful]
[Login Failed]
Login Error
Enter virtual class room Interface
Add subject
Add student
[Admin tasks]
[Other USER]
Logout
schedule class
Attend class
Attendance
SEND NOTICE
Receive Notice

Data modeling

When to choose Data Model ?

- If software requirements include the need to create, extend or interface with a data-base or if complex data structure must be constructed and manipulated.

① Data object :-

- is a representation of composite information that must be understood / is processed by software

↓

something that a number of different properties or attributes

Example

- an external entity (anything that produces or consumes information)
- a thing (report/display)
- an occurence (telephone call)
- event (an alarm)
- a role (salesperson)
- an organizational unit (accDuting department)
- a place (warehouse)
- a structure (a file)

Anything that can be defined as a set of attributes

— Data object

② Data Attributes:-

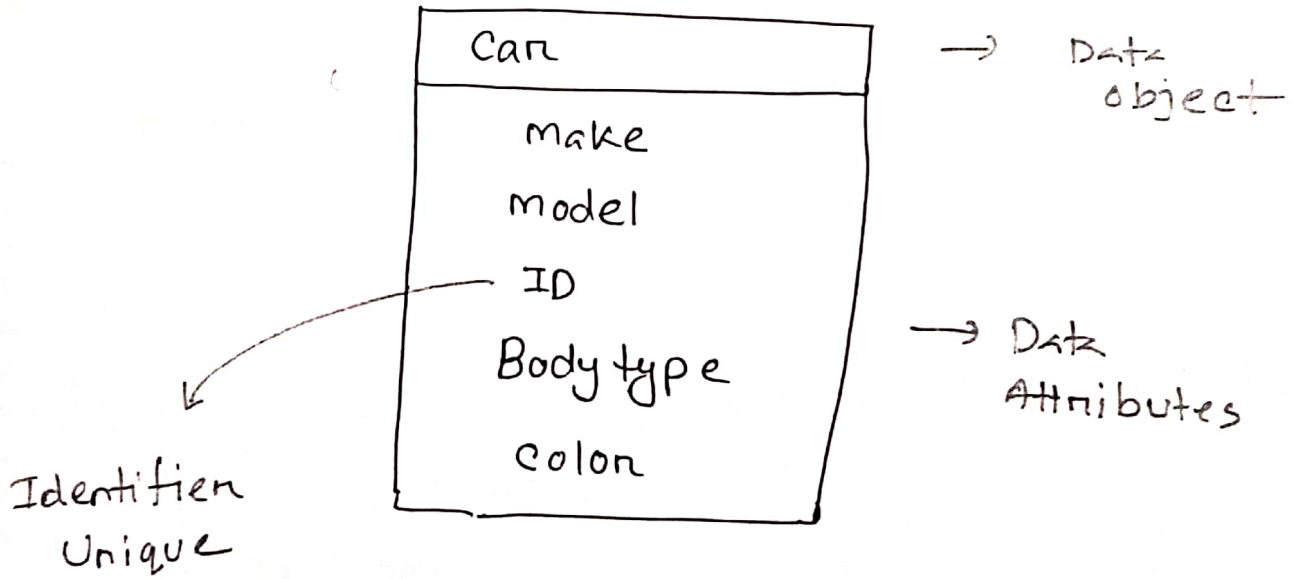
Date

- define the properties of a data object and take on one of three different characteristic

① Name instance of the data object

② Describe the instance

③ make refenence to another instance in another table.



Data object and object oriented classes are same ??

→ NO.

Data object defines a composite data item that is, it incorporates a collection of individual data items / attributes and gives the collection of items a name (the name of data object)

object oriented class encapsulates data & attributes but also incorporates the operations / method that manipulate the data implied by those attributes.

③ Relationship

- Data objects are connected with one - another

ERD - Entity Relationship Diagram

- is a graphical representation that depicts the relationship between data objects.

Class Based Modeling

- represents the objects that the system will ~~manipulate~~ manipulate, the operations / methods that will be applied to the object to effect manipulation. and relationship between the objects

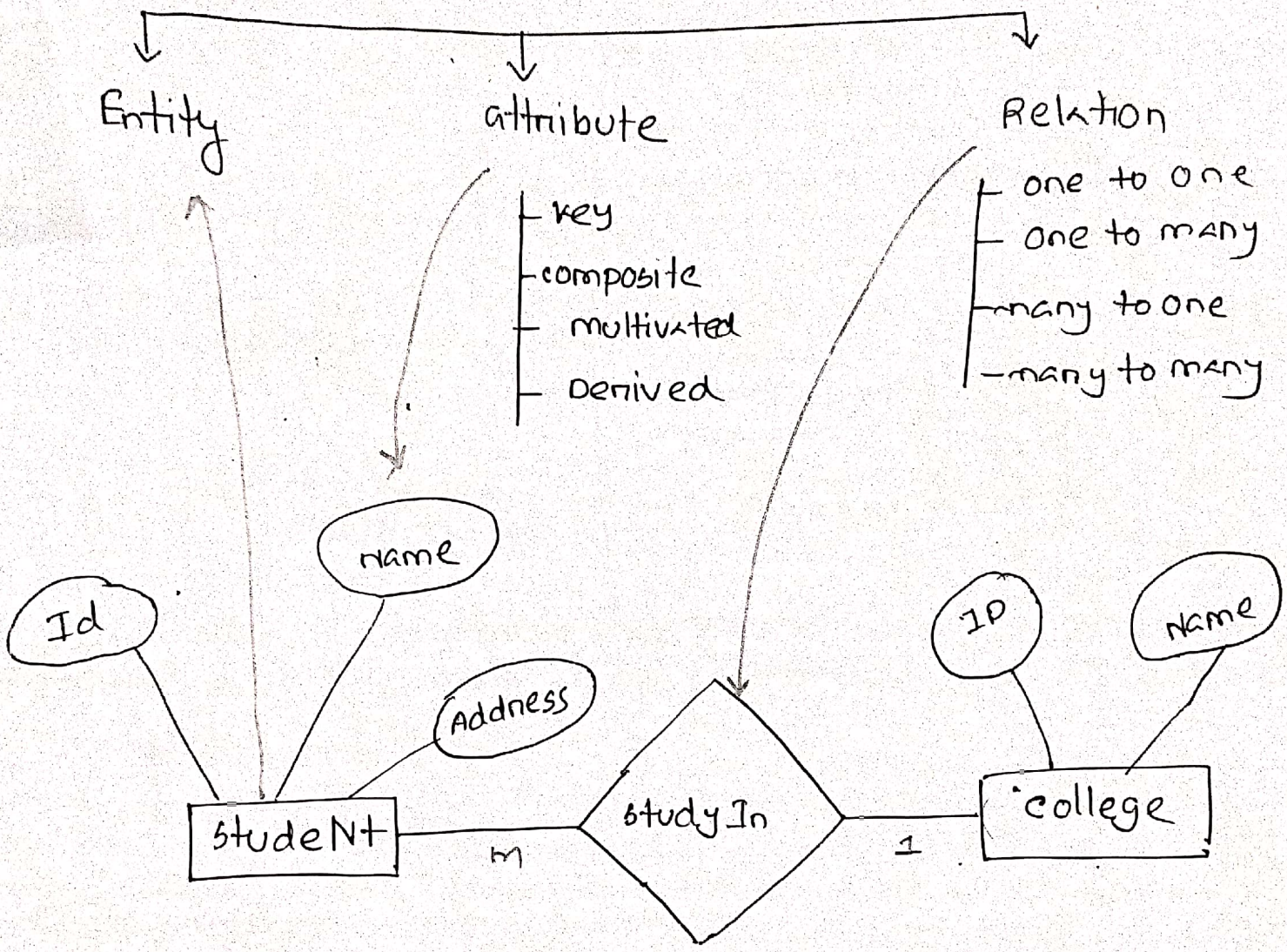
(1) ~~Inde~~ Identifying ~~Ac~~ Analysis classes

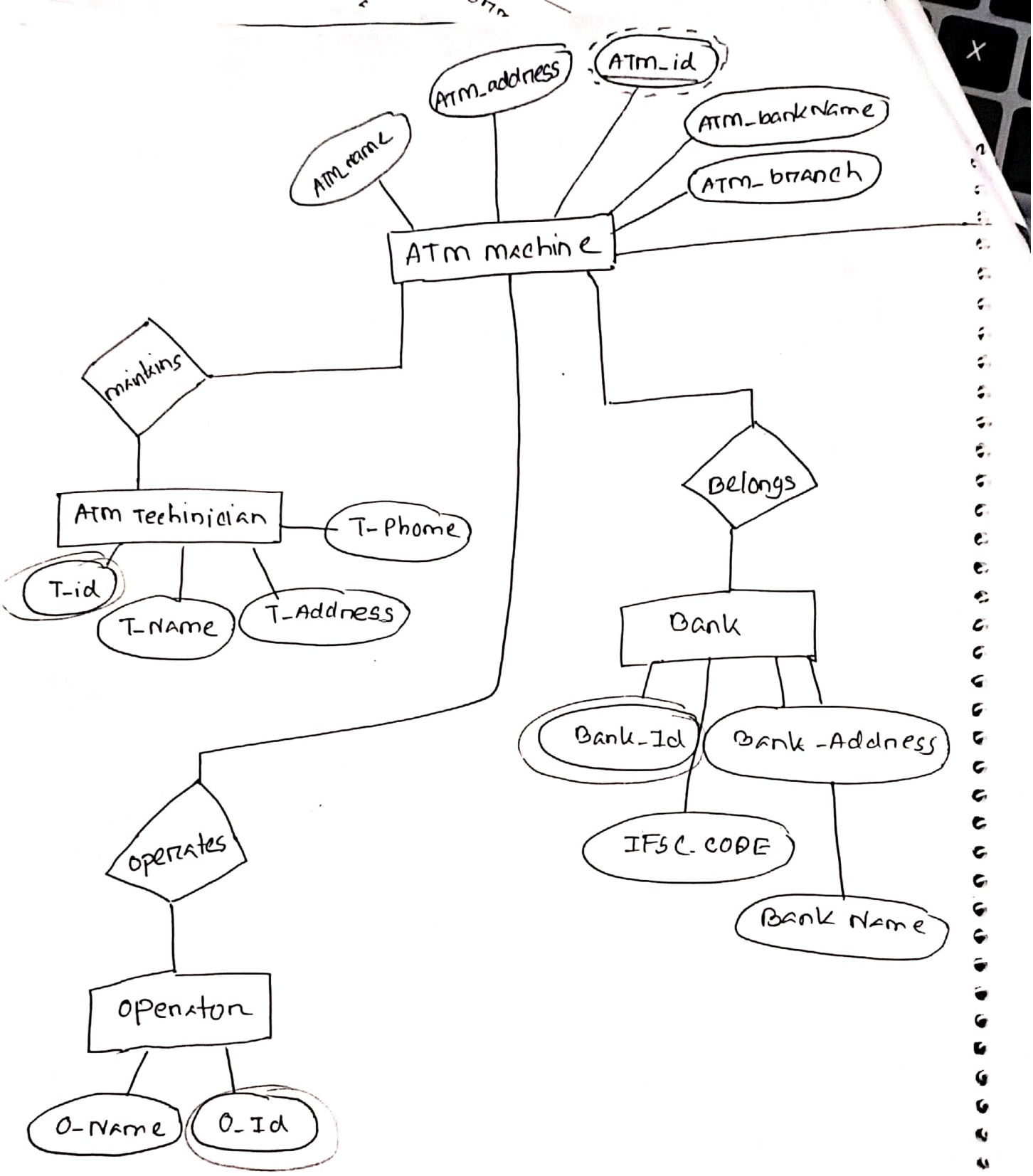
(2) Specifying Attributes

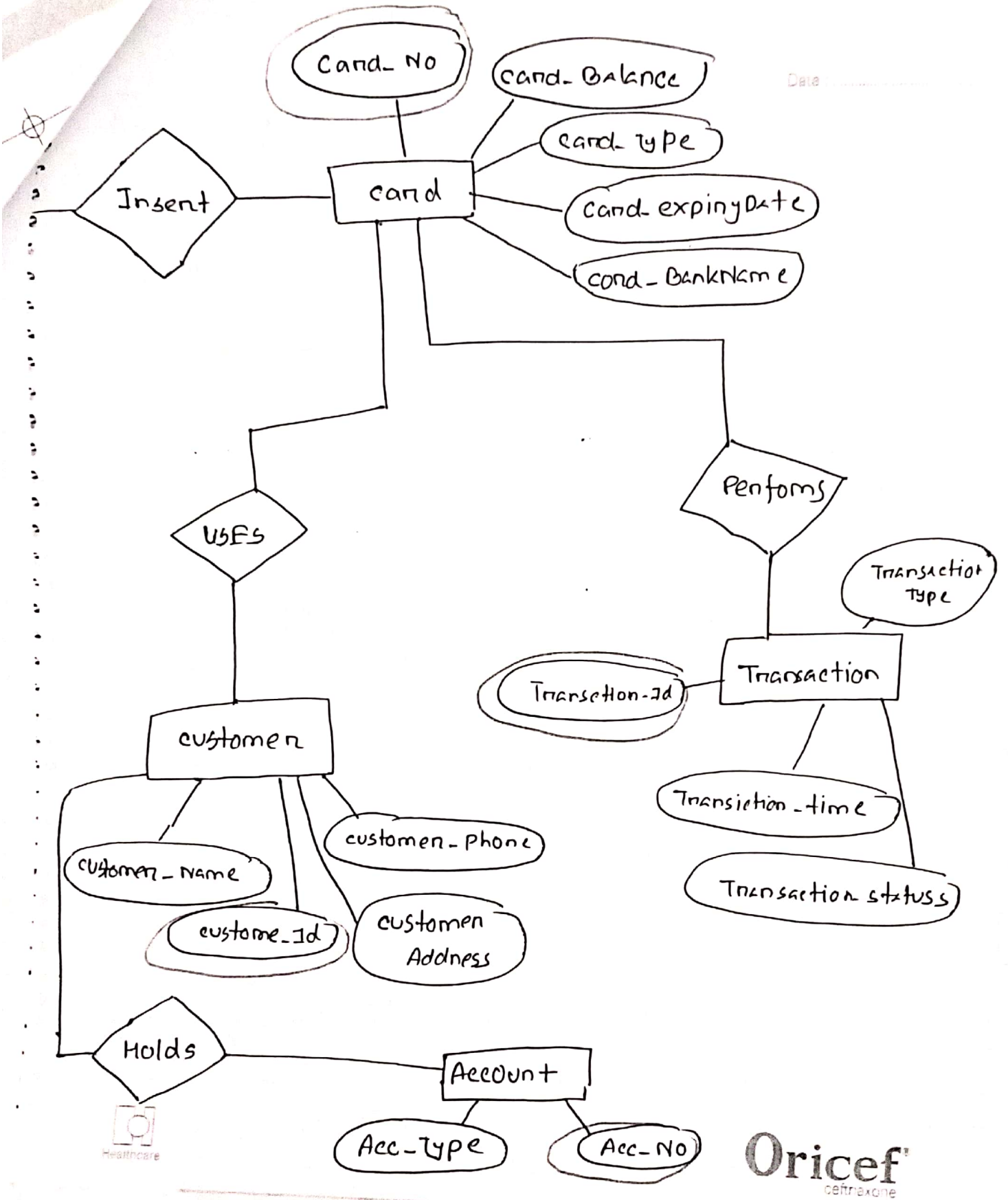
(3) Defining operations

ER - Entity Relationship Model

Date




ATM_address
ATM_id
ATM_bankName
ATM_name
ATM_branch
ATM machine
maintains
Belongs
Atm Technician
T-Phome
T-id
T-Name
T-Address
Bank
Bank_Id
Bank-Address
IFSC_CODE
Bank Name
operates
Operator
O-Name
O_Id


Card_No
card_Balance
card_type
card_expiryDate
cond_BankName
Insert
card
USES
Penforms
Transaction type
Transaction
Transaction_Id
customer
Transiction_time
Transaction_status
customer_Phone
customer_Name
customer_Id
customer Address
Holds
Account
Acc_Type
Acc_No

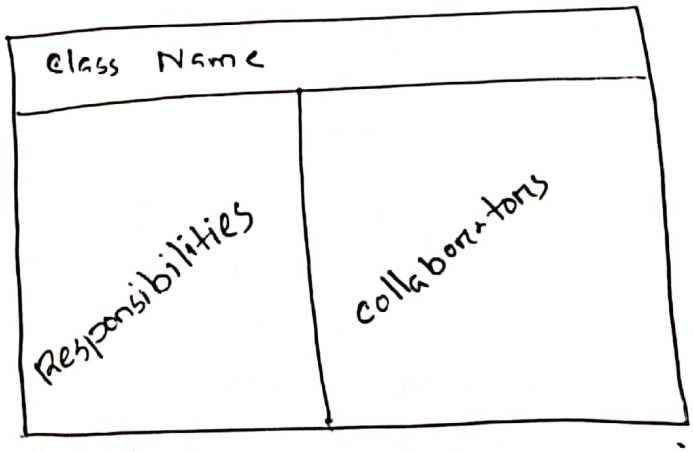
CRC

class Responsibility Collaboration

CRC model is really a collection of standard index cards that represents classes.

Class card contains

1. Attributes
2. methods
3. Responsibilites
4. Collaborators



Bestron™

Responsibility :-

- You should ask yourself what a class does as well as what information you wish to maintain about it

- identify a responsibility for a class to fulfill collaboration with another class

Collaborators

it must collaborate (work) with other classes to get the job done. Collaboration will be in one of two forms : a request for information or a request to perform a task. To identify the collaborators of a class for each responsibility ask yourself "does the class have ability to fulfill this responsibility ?" If not then look for a class that either has the ability to fulfill the missing functionality or the class which should fulfill it.

tudent	
Enroll in seminar Drop a seminar Request transcripts	seminar Transcript Enrollment

Transcript	
Determine average mark	student seminar Professor Enrollment

Professor	
Provide information Seminar Instructing	Student Seminar

Seminar	
Seminar Number Waiting list Enroll students Add students Drop students	Student Professor