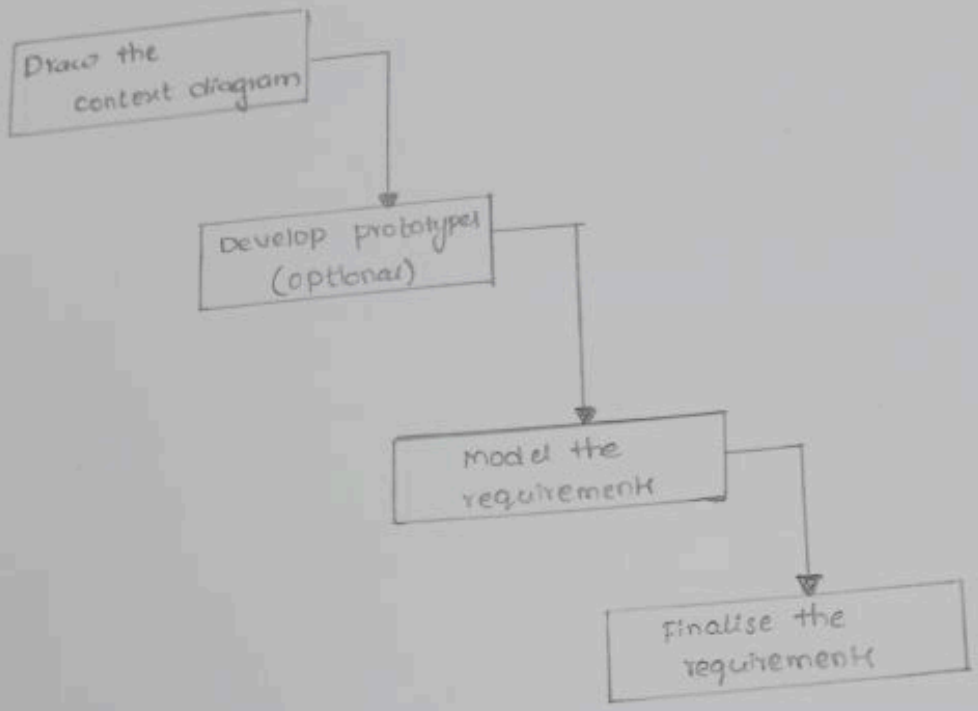
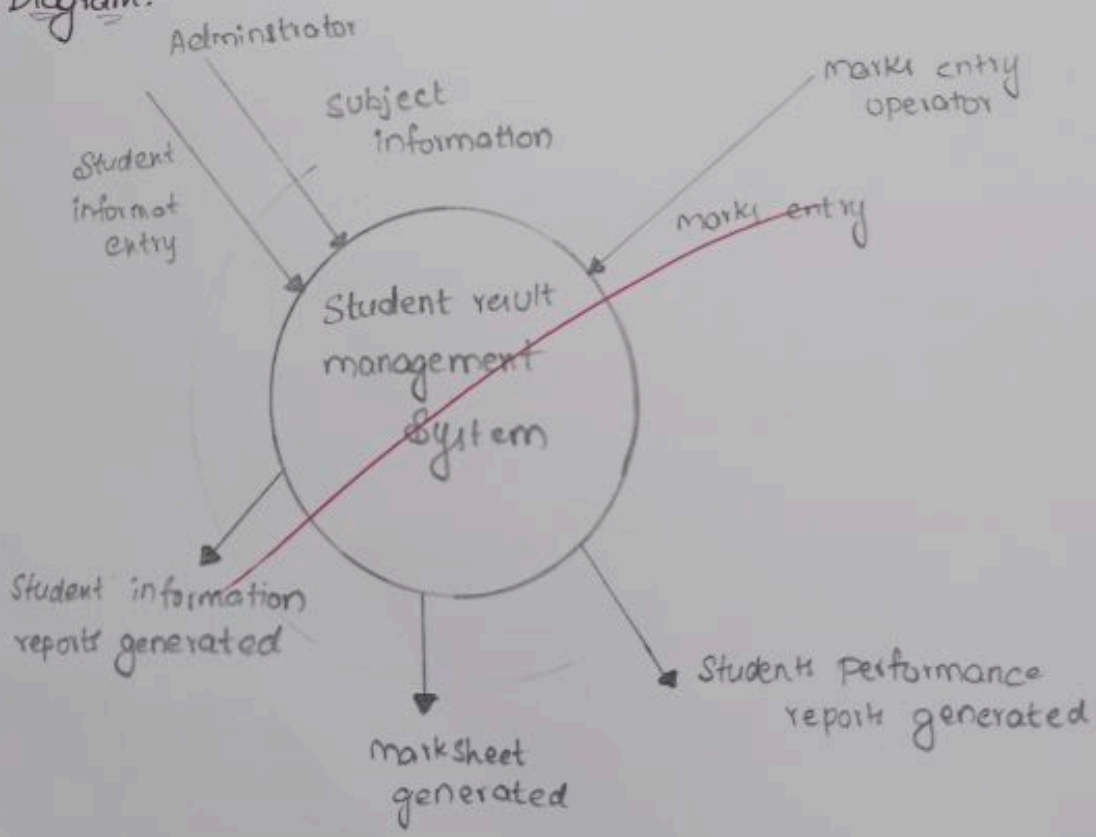
INDEX

Sl. No.	Date	Name of the Experiment	Page No.	Marks (10)	Remark Signatu
1.	19/2/2022	Requirement Analysis and SRS	1-3		
2.	19/2/2022	ER diagram, dataflow diagrams, CFD	4-6		
3.	26/2/2022	Use case, Activity, class diagram	7-9		
4.	26/2/2022	sequence and interface to class diagram	10-11		
5.	5/8/2022	prototype model of product	12-14		
6.	19/3/2022	Course management system	15-16	10	
7.	26/3/2022	Easy Leave	17-18		
8.	16/4/2022	E-bidding	19-20		
9.	16/4/2022	electronic cash counter	21-22		
10.	25/4/2022	Book Bank	23-25		
11.	16/5/2022	credit card processing	26-27		

Steps of Requirements Analysis:



Context Diagram:



Experiment - 01

Aim: Do the Requirement Analysis and prepare SRS

Required Analysis.

Required Analysis is significant and essential activity after elication. we analyze, refine and scrutinize the gathered requirements to make consistent and unambiguous requirements. this activity reviews all requirements and may provide a graphical view of the entire System.

i) Draw the context diagram:

the context diagram is a simple model that defines the boundaries and interfaces of the proposed system that interact with the System.

ii) Development of a prototype (optional):

One effective way to find out what the customer wants is to construct a prototype, something that looks and preferably acts as part of the System they say they want.

Some projects are developed for general market. this prototype should be built quickly and at a relatively low cost. Hence it will always have limitations. this is an optional activity.

iii) Model requirements.

this process usually consists of various graphical representation of the functions, data entities, external entities and the relationships between them. the graphical view may help to find incorrect, inconsistent, superfluous requirements.

iv) Finalise the requirements

After modeling the requirements, we will have abetter understanding of the system behaviour. The inconsistencies and ambiguities have been identified and corrected. Now, we finalize the analyzed requirements, and the next step of document these requirements in a prescribed format.

Software Requirement Specifications:

The production of the requirements stage of the software development process is Software Requirements Specifications (SRS) is a format report, which acts as a representation of Software that enables the customer to review whether it (SRS) is according to their requirements.

Following are the features of a good SRS document.

1. Correctness: the SRS is complete if and only if, it includes

2. Completeness: User review is used to provide the accuracy of requirements stated in the SRS

1. All essential requirements, whether relating to functionality performance, attributes.

2. full labels and references to all figures, tables in the SRS and definitions of all terms.

3. Consistency: the SRS is consistent if and only if, no subset of individuals requirements described in it conflict.

i) the specified characteristics of real-world objects may conflicts.

a) the format of an output report may be described in one requirement as tabular but in another as textual.

ii) there may be a reasonable or temporal conflict between the two specified actions.

for example:

a) one condition may state that "A" must always be follow "B", while other requires that "A" and "B" occurs.

4. Unambiguousness:

SRS is unambiguousness when every fixed requirement has only one interpretation. In case there is a method used with multiple definitions, the requirements report should determine the implicatio-ns in the SRS so that it is clear and simple to understand.

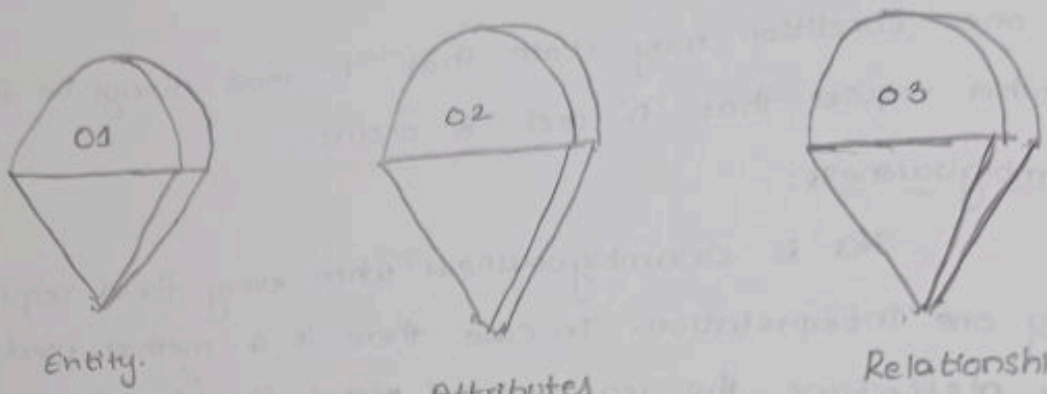
5. Ranking for importance and stability:

the SRS is ranked for importance and stability if each requirement in it has an identifier to indicate either their significance or stability of that particular requirement.

6. Modifiability:

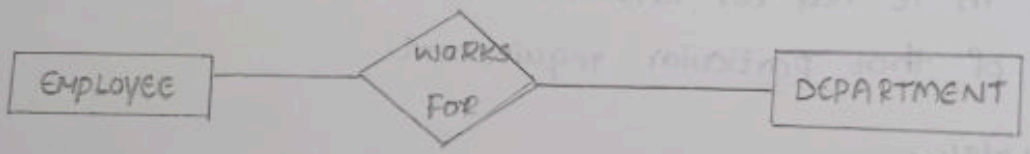
SRS should be made as modifiable as likely and should be capable of quickly obtain changes to the system to some extent. modifications should be perfectly indexed and cross-referenced.

Component of ER Diagram.

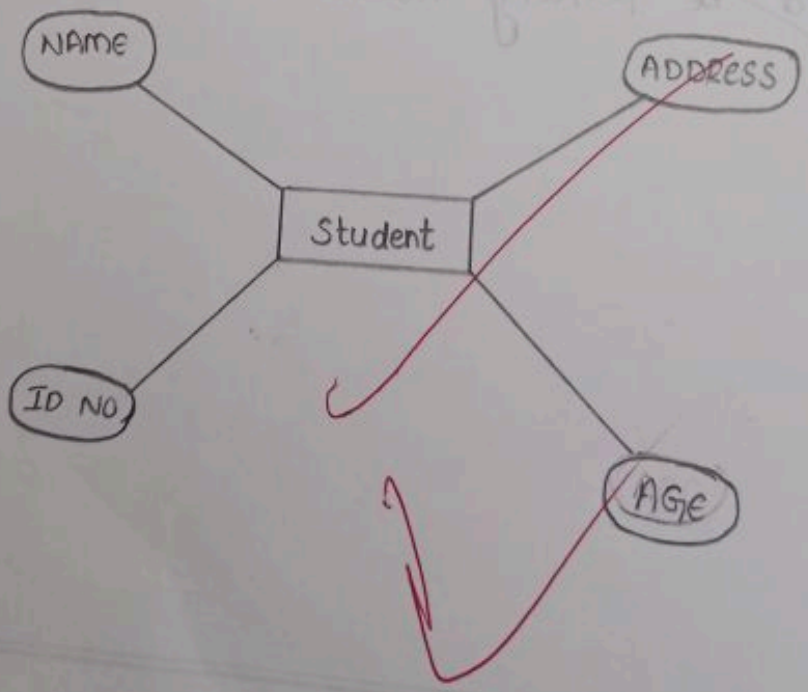


Entity. Attributes Relationship

Entity:



Attributes.



Experiment - 2

ER diagrams, Dataflow, CFD

Entity - Relationship diagrams:

It is a data modeling method used in Software engineering to produce a conceptual data model of an information System.

Purpose of ERD:

→ It serves as a documentation tool.

→ Database analyst gains a better understanding of the data.

Component of a ER-diagram:

Entity:

→ It is an real world entity that can be merely identifiable.

→ It is an rectangle with in a ER diagram.

Attribute:

→ Entities are denoted utilising their properties known as attribute.

for eg: A Student name cannot be numeric

→ It has to be alphabetic.

Types of Attributes:

→ Key Attribute

→ Composite Attribute.

→ Single-valued Attribute

→ Multi-valued attribute

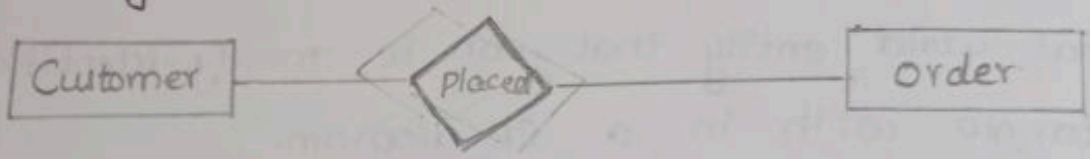
→ Derived Attribute.

Relationships in ERD:

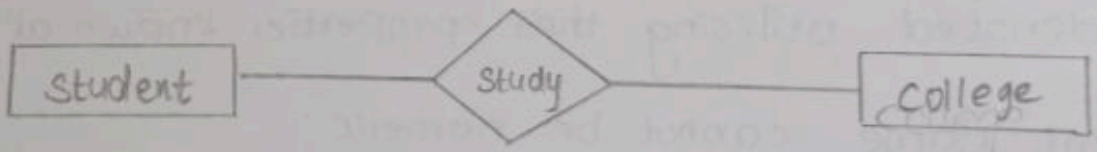
Types of cardinalities:

1. one to one:

2. One to Many:



3. Many to one:



4. Many to many.

Relationships;

The association among entities is known as realtionship. Relationships are represented by the diamond-shaped box.

for eg: an employee works_at a department, a student enrolls in a course. Here, works_at and Enrolls are called relationships.

Data Flow Diagrams.

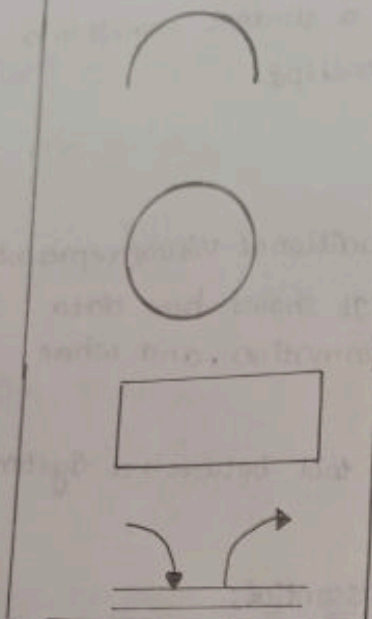
A Data flow Diagram (DFD) is a traditional visual representation of the information flows within a system. It shows how data enters and leaves the system, what changes the information, and where data is stored.

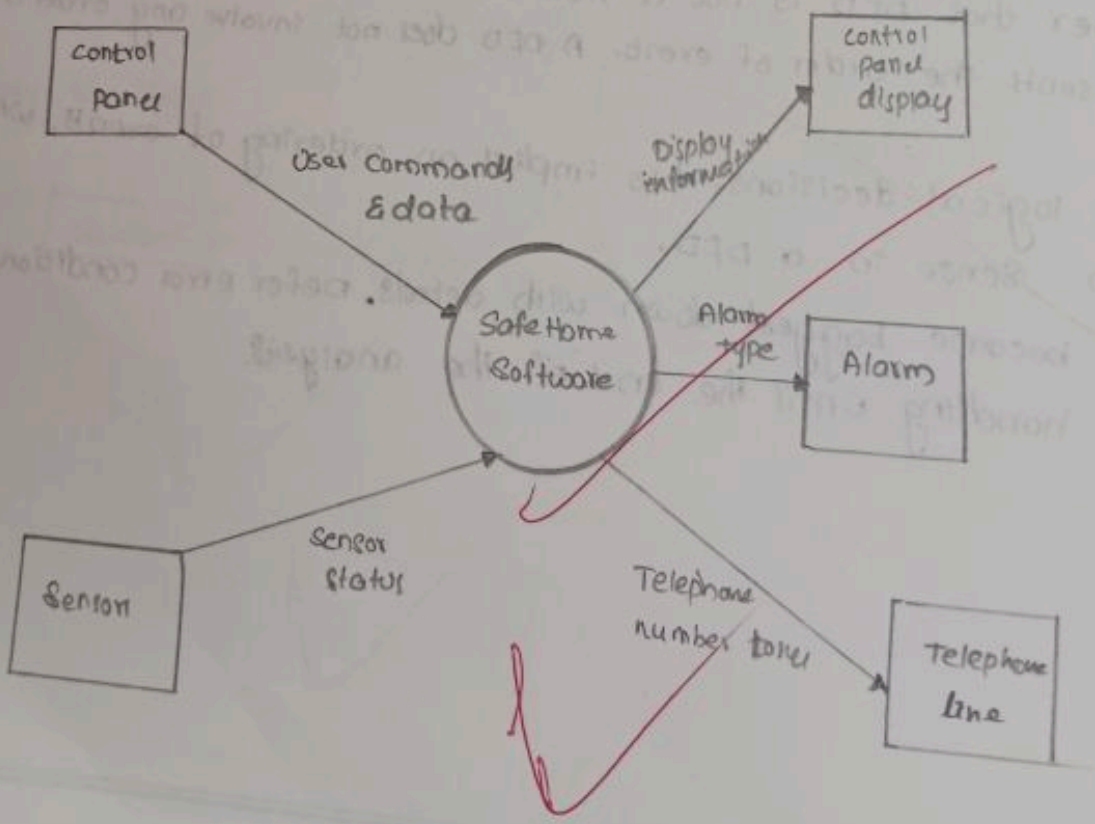
It may be used as a communication tool between a system analyst and any person.

The following observations about DFDs are essential:

1. All names should be unique. this makes it easier to refer to elements in the DFD.
2. Remember that DFD is not a flow chart. Arrows is a flow chart that represents the order of event. A DFD does not involve any order of events.
3. Suppress logical decisions. this implies an ordering of events, which makes no sense in a DFD.
4. Do not become bogged down with details. Defer error conditions and error handling until the end of the analysis.

Symbols for Data Flow Diagrams.

Symbol	Name	Function
	Dataflow	used to connect processes to each other, to sources for sinks; the arrow head indicates direction of data flow
	Process	performs some transformation of input data to yield output data.
	Source of Sink (External Entity)	A source of system inputs or sink of system outputs
	DataStore	A repository of data; the arrow heads indicate net input and net output to store



Levels in Data Flow Diagram (DFD):

The DFD may be used to perform a System or Software at any level of abstraction. DFDs may be partitioned into levels that represent increasing flow and functional detail. We will see primarily three Levels in the dfd, which are : 0-level DFD, 1-Level DFD, 2-level DFD.

0-Level DFDM:

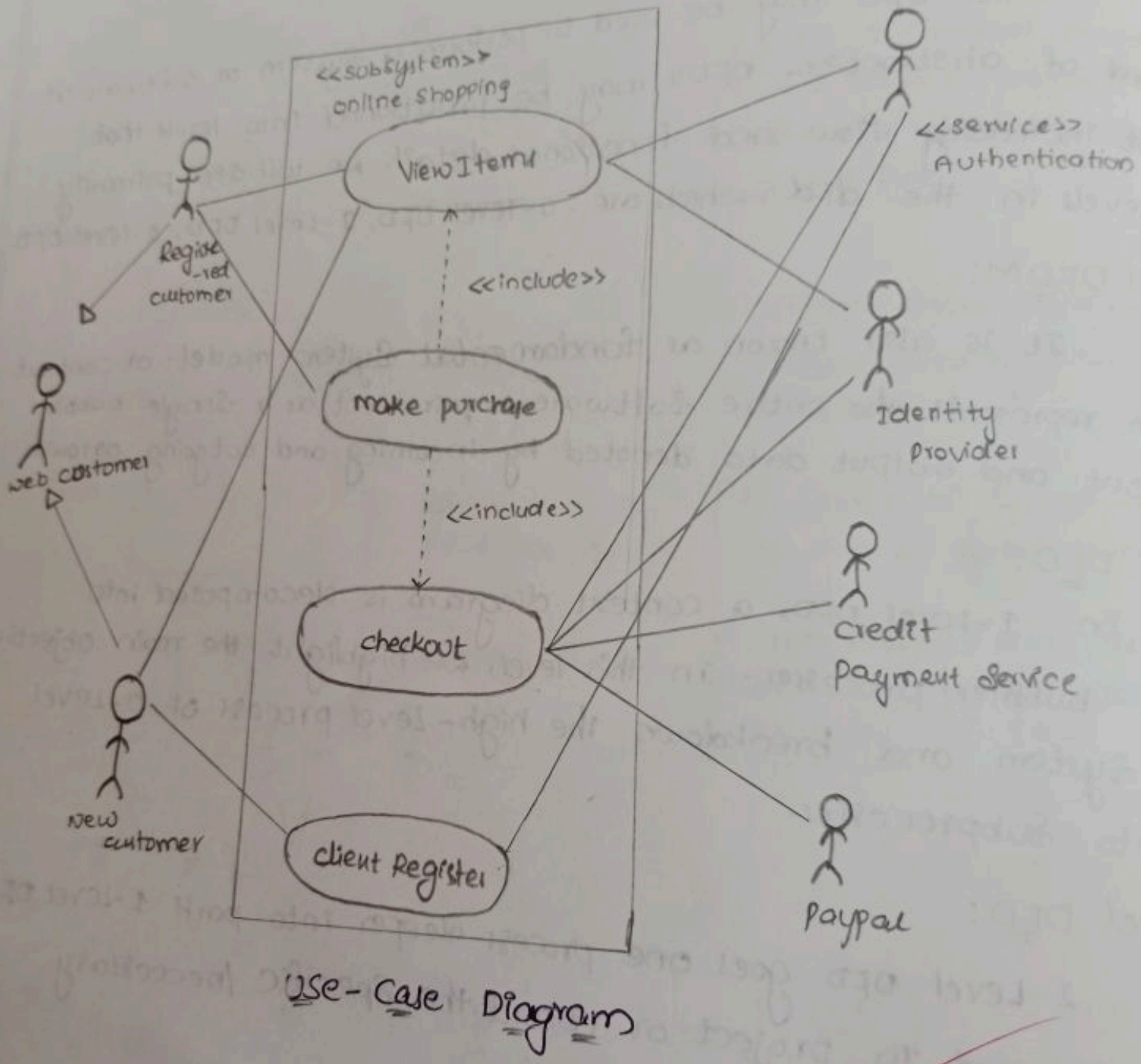
It is also known as fundamental System model or context diagram represents the entire Software requirement as a single bubble with input and output data denoted by incoming and outgoing arrows.

1-Level DFD:

In 1-Level DFD, a context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main objectives of the System and breakdown the high-Level process of 0-Level DFD into subprocesses.

2-Level DFD:

2-Level DFD goes one process deeper into parts 1-Level DFD. It can be used to project or record the specific /neccessary detail about the System's functioning.



Use-Case Diagram

Experiment - 3

Software designing use case diagrams, Activity diagrams. Build and test class diagrams.

Use case diagram:

In UML, use-case diagrams model the behavior of a system and help to capture the requirements of the system. User-case diagram describe the high-Level functions and scope of a system these diagrams also identify the interaction between the System and its actors.

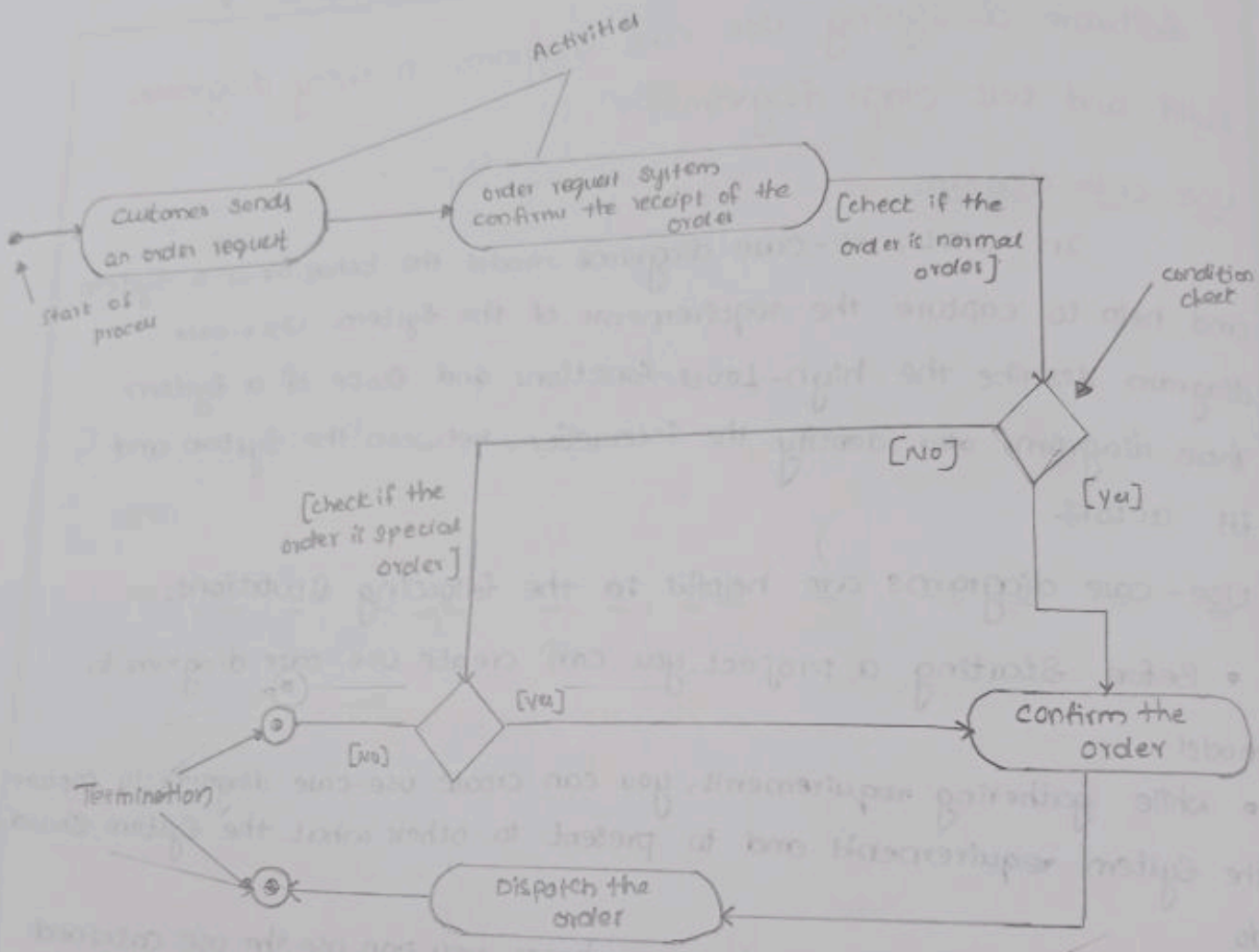
Use-case diagrams are helpful in the following situations:

- Before starting a project, you can create use-case diagrams to model.
- While gathering requirements, you can create use-case diagrams to capture the system requirements and to present to other what the system should do
- During the analysis and design phases, you can use the use cases and actors from your use-case diagrams to identify the classes that the system requires
- During the testing phase, you can use use-case diagrams to identify tests for the system.

Model elements in use-case Diagrams:

- Actors.

An actor represents a role of a user that interacts with the system that you are modeling, the user can be a human user, an organization, a machine or another external system.



Activity Diagram

- Subsystems:

In UML models, subsystems are a type of stereotyped component that represent independent, behavioral units in a System.

- Relationships in use-case diagrams:

In UML, a relationship is a connection between model elements. A UML relationship is a type of model element that adds semantics to a model by defining the structure and behaviour between the model elements.

Activity Diagram:

Activity diagram is another important diagram in UML to describe the dynamic aspects of the System.

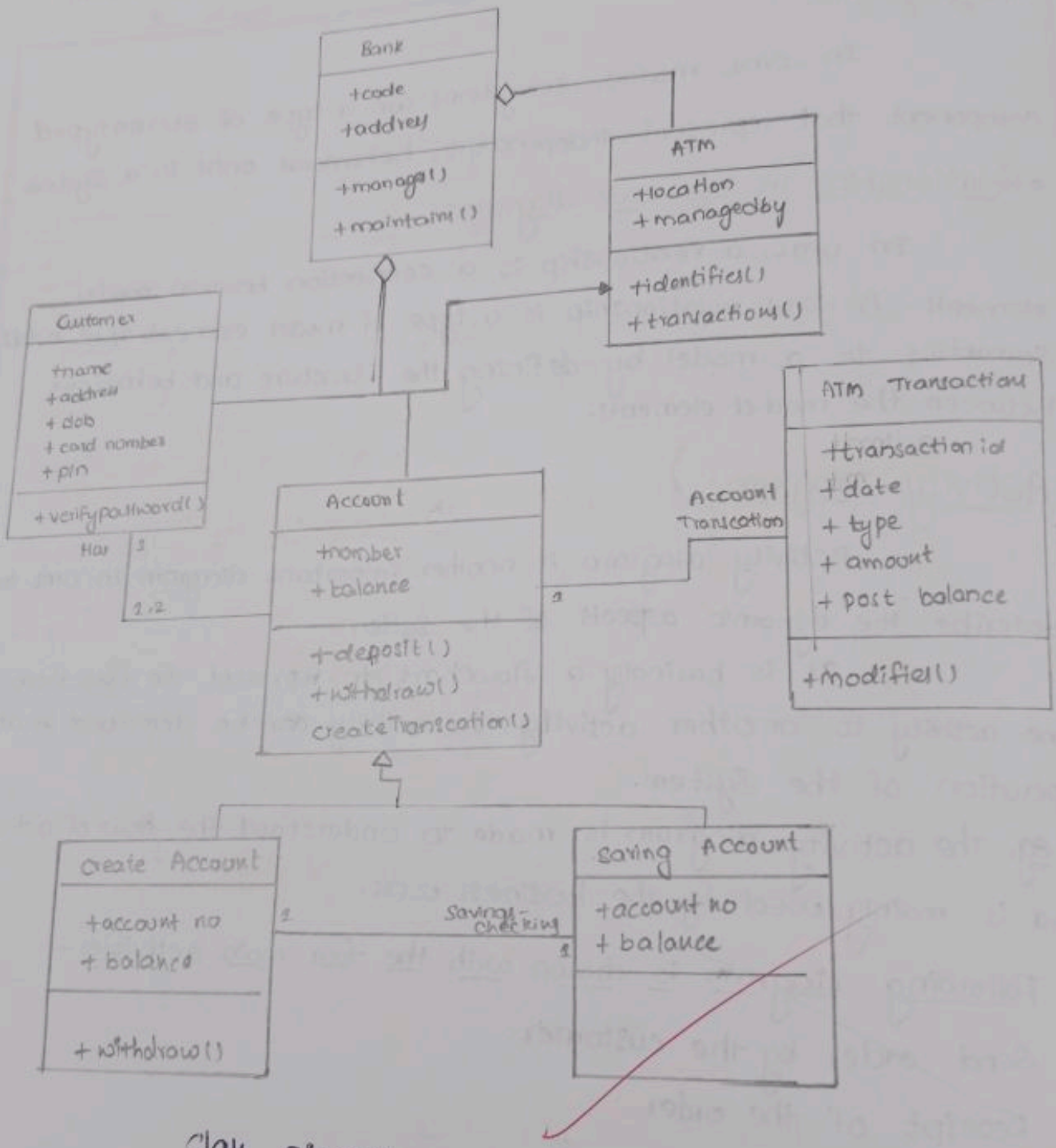
It is basically a flowchart to represent the flow from one activity to another activity. the activity can be described as an operation of the System.

eg. the activity diagram is made to understand the flow of activities and is mainly used by the business users.

Following diagram is drawn with the four main activities -

- Send order by the customer.
- Receipt of the order
- Confirm the order
- Dispatch the order.

After receiving the order request, condition checks are performed to check if it is normal or special order. After the type of order is identified, dispatch activity is performed and that is marked as the termination of the process.



Class Diagram.

This diagram is used to model the activities which are nothing but business requirements. the diagram has more impact on business understanding rather than on implementation details.

Activity diagram can be used for -

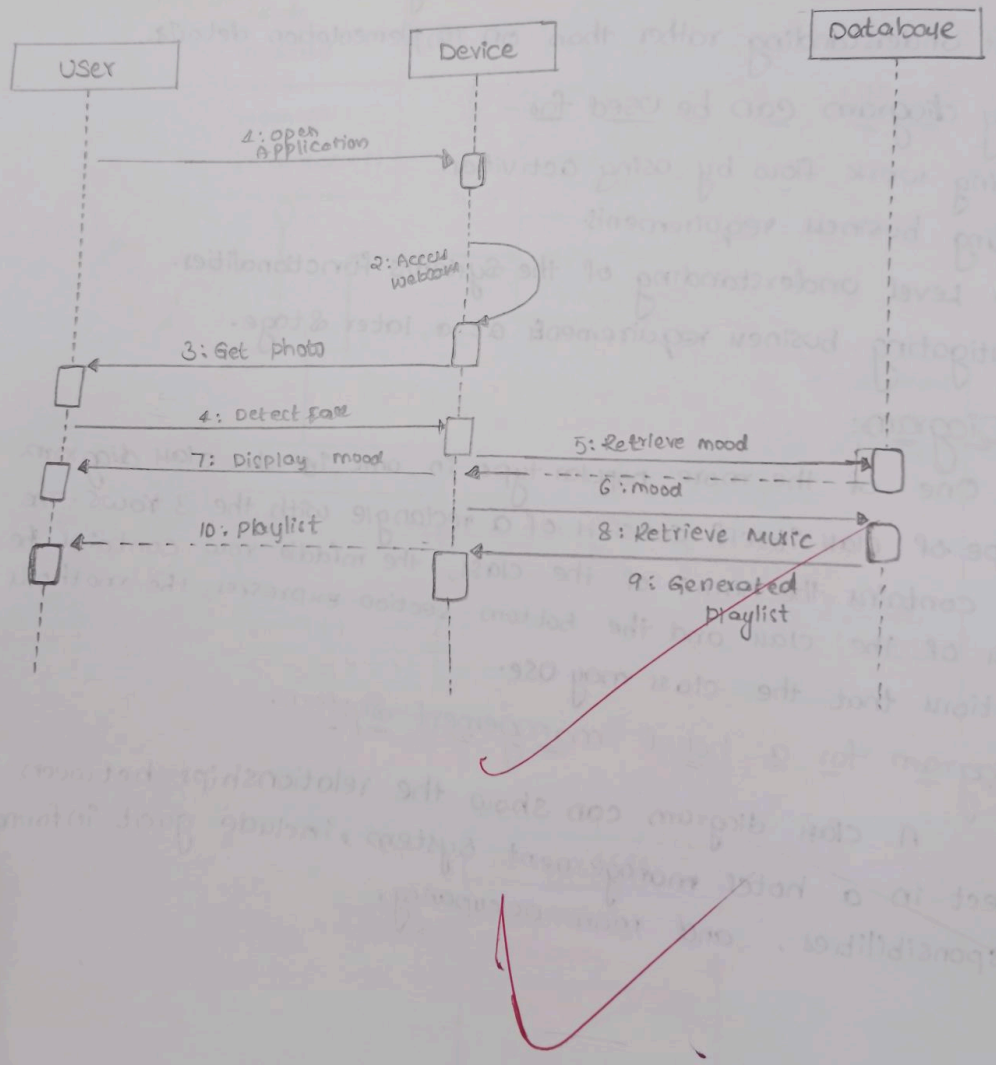
- Modeling work flow by using activities.
- modeling business requirements
- High Level understanding of the System's functionalities.
- Investigating business requirements at a later stage.

Class Diagram:

One of the more popular types in one is the class diagram. the shape of class itself consists of a rectangle with the 3 rows. the top row contains the name of the class, the middle row contains the attributes of the class and the bottom section expresses the methods or operations that the class may use.

Class Diagram for a hotel management System.

A class diagram can show the relationships between each object in a hotel management system, include guest information, staff responsibilities, and room occupancy.


User
Device
Database
1: open application
2: Access webcam
3: Get photo
4: Detect face
5: Retrieve mood
6: mood
7: Display mood
8: Retrieve music
9: Generated playlist
10: playlist

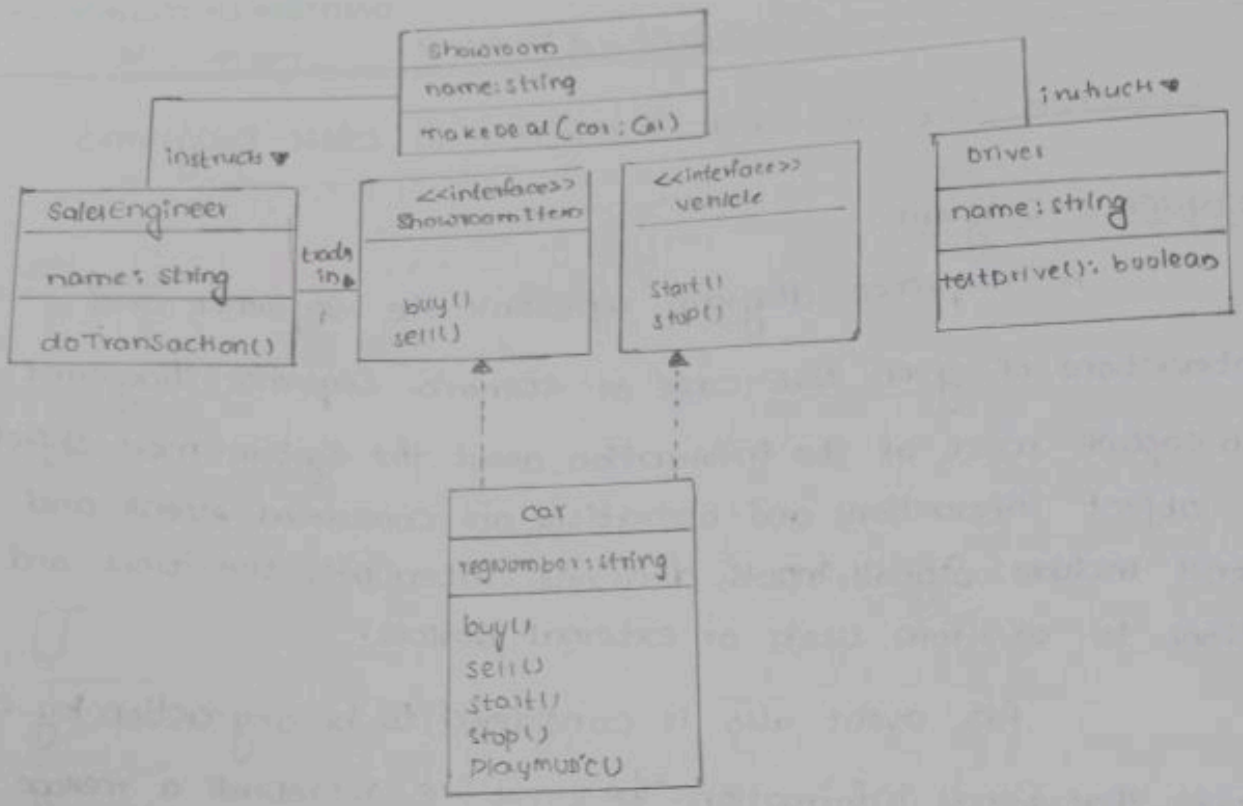
Experiment - 4

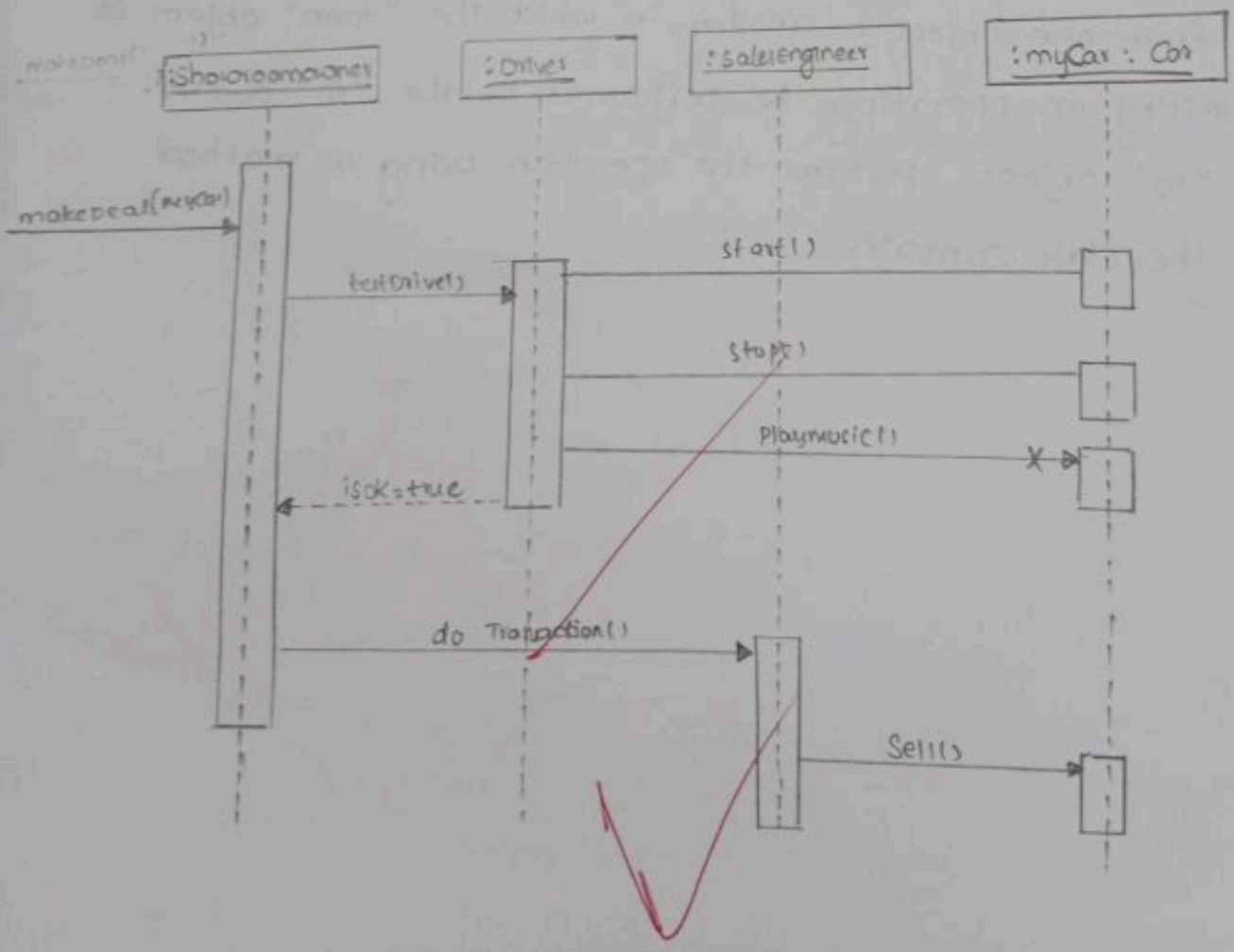
Sequence Diagrams and Interfaces to Class Diagrams.

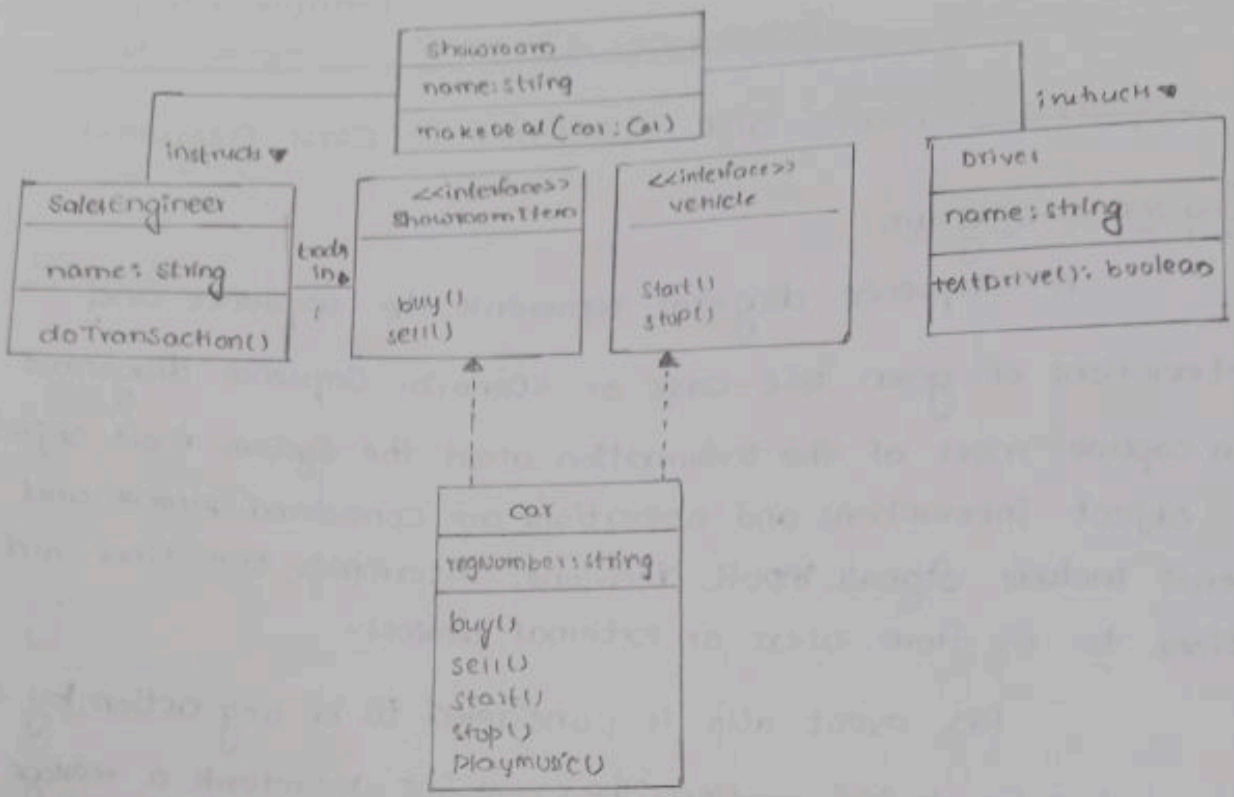
Sequence Diagram:

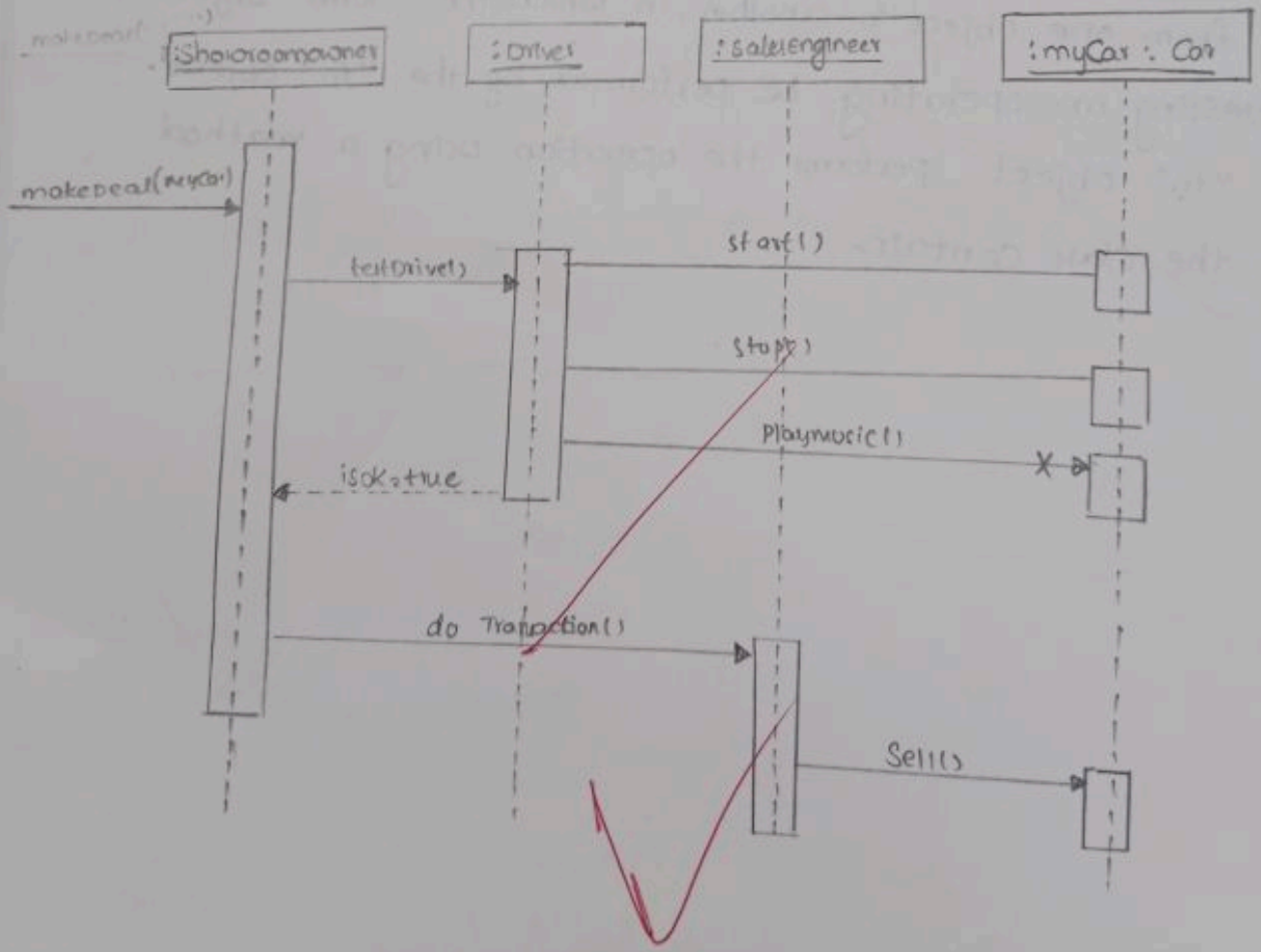
A sequence diagram represents the sequence and interactions of given use-case or scenario. Sequence diagrams can capture most of the information about the system. most object to object interactions and operations are considered events and events include signals, inputs, decisions, interrupts, transitions and actions to or from users or external devices.

An event also is considered to be any action by an object that sends information. The event line represents a message sent from one object to another, in which the "form" object is requesting an operating be performed by the "to" object. The "to" object performs the operation using a method that the class contain.


Showroom
name:string
makeDeal(car:Car)
instructs
tests
SaleEngineer
name: string
doTransaction()
<<interface>>
ShowroomItem
buy()
sell()
<<interface>>
vehicle
start()
stop()
Drives
Driver
name:string
testDrive(): boolean
car
regNumber:string
buy()
sell()
start()
stop()
playMusic()

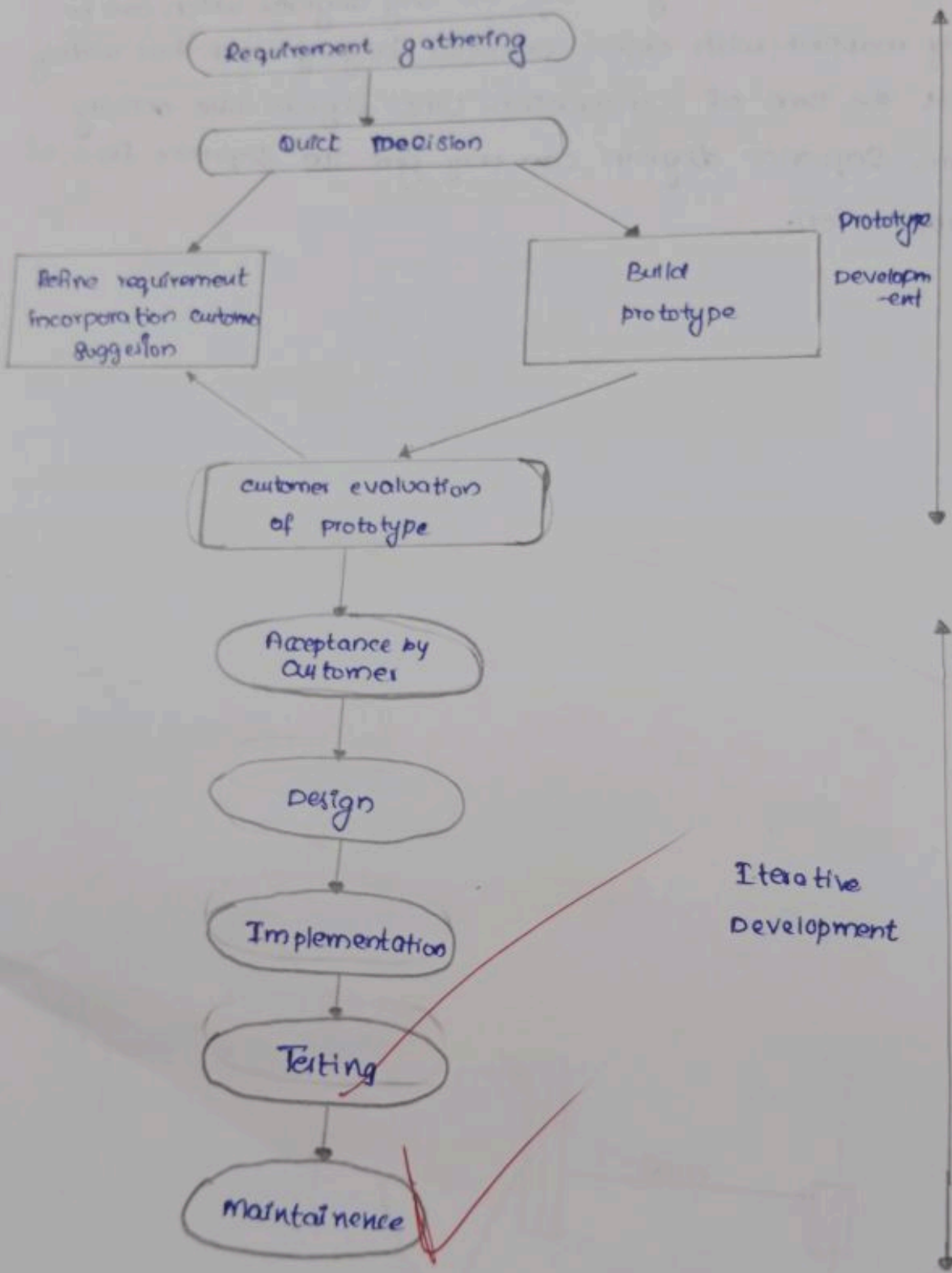

:Showroomowner
:Driver
:SaleEngineer
:myCar : Car
makeDeal(myCar)
testDrive()
start()
stop()
playMusic()
isOK=true
doTransaction()
sell()


Showroom
name: string
makeDeal(car: Car)
instructs
SaleEngineer
name: string
doTranSaction()
trades in
<<interface>>
ShowroomItem
buy()
sell()
<<interface>>
vehicle
start()
stop()
instructs
Driver
name: string
testDrive(): boolean
Car
regNumber: string
buy()
sell()
start()
stop()
playMusic()


:ShowroomOwner
:Driver
:SaleEngineer
:myCar : Car
makeDeal(myCar)
testDrive()
start()
stop()
playMusic()
isOk=true
doTransaction()
Sell()

Interface to class Diagram:

Class diagram are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction. UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application.


Requirement gathering
Quick Decision
Refine requirement incorporation cutome suggestion
Build prototype
Prototype development
customer evaluation of prototype
Acceptance by customer
Design
Implementation
Testing
Maintainence
Iterative Development

Experiment - 5

Prototype model of the product.

Prototype Model:

The prototype model requires that before carrying out the development of actual software, a working prototype of the system should be built. A prototype is a toy implementation of the system. A prototype usually turns out to be a very crude version of the actual system, possible exhibiting limited fonctional capabilities, low reliability and inefficient performance as compared to actual software.

Steps of prototype model:

1. Requirement Gathering and Analyst.
2. Quick Decision
3. Build a prototype
4. Assessment or User Evaluation.
5. Prototype Refinement.
6. Engineer product.

There are four types of models available.

A) Rapid throwaway prototyping -

This technique offers a useful method of exploring ideas and getting customer feedback for each of them. In this method, a developed prototype need not necessarily be part of the ultimately accepted protype. Customer feedback helps in preventing unnecessary design faculty and hence, the final prototype developed is of better quality.

B) Evolutionary prototyping-

In this method, the prototype developed initially is incrementally refined on the basis of customer feedback till it finally get accepted. This is because developing a prototype from scratch for every iteration of the process can sometimes be very frustrating for the developers.

C) Incremental prototyping-

In this type of incremental prototyping, the final expected product is broken into different small pieces of prototypes and being developed individually. In the end, when all individual pieces are properly developed.

D) Extreme prototyping-

this method is mainly used for web development. It is consists of 3 sequential independent phases:

D.1) In this phase a basic prototype with all the existing static pages are presented in the HTML format.

D.2) In this 2nd phase, functional screens are made with a simulated data process using a prototype service layer.

D.3) this is the final step where all the services are implemented and associated with the final prototype.

Advantages of prototype model:

1. Reduce the risk of incorrect user requirement.
2. Good where requirements are changing/ uncommitted.
3. Regular visible process aids management.
4. Support early product marketing
5. Reduce maintenance cost.
6. Errors can be detected much earlier as the System is made side by side.

Disadvantage of prototype Model:

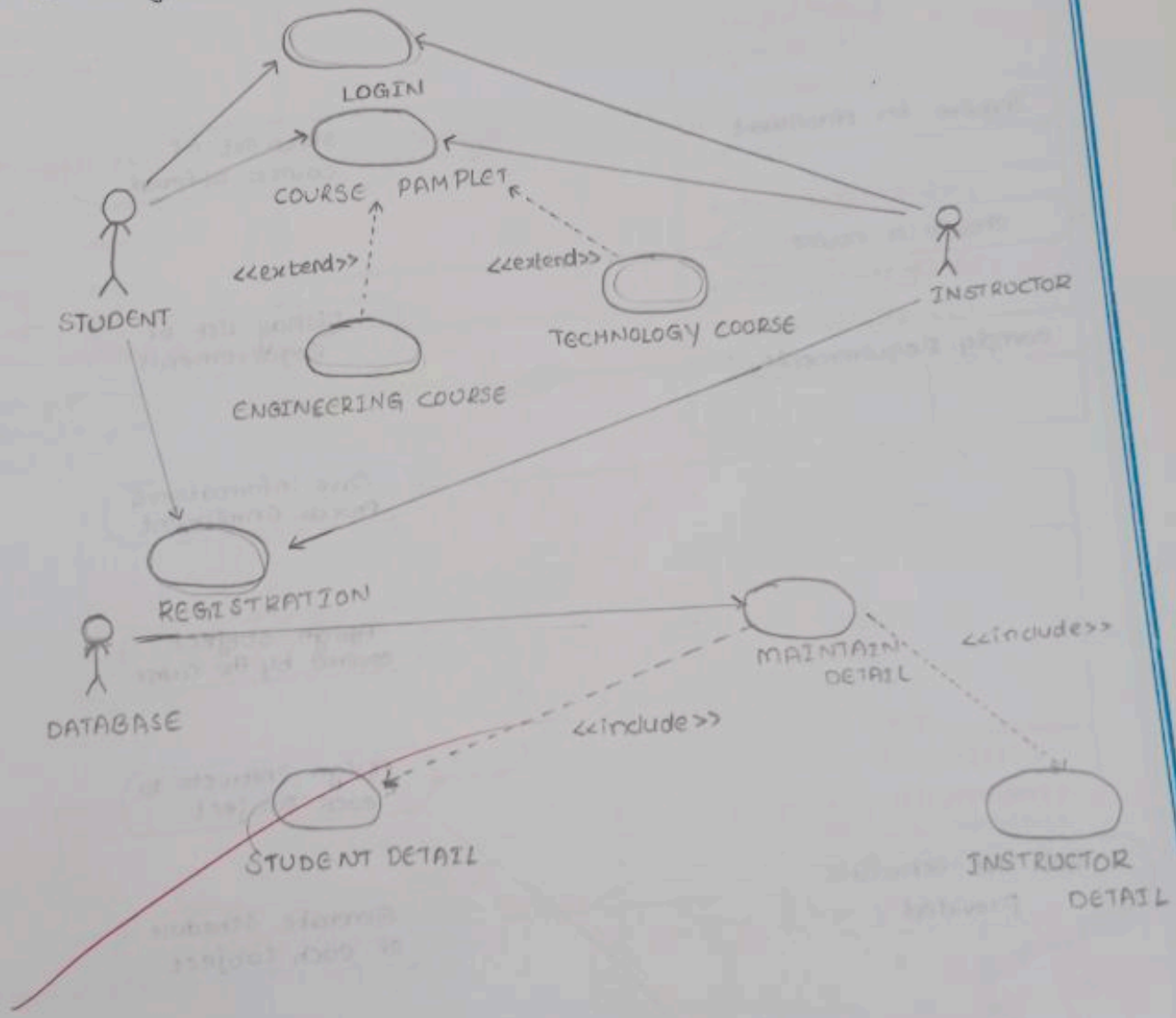
1. An unstable / badly implemented prototype often becomes the final product
2. Require extensive customer collaboration.
   - costs customer money
   - needs committed customer.
   - Difficult to finish if customer withdraw.
   - may be too customer specific, no broad market
3. Difficult to know how long the project will last
4. Prototyping tools are expensive
5. Special tools & techniques are required to build a prototype.
6. It is a time-consuming process.

Reg. No. : .....

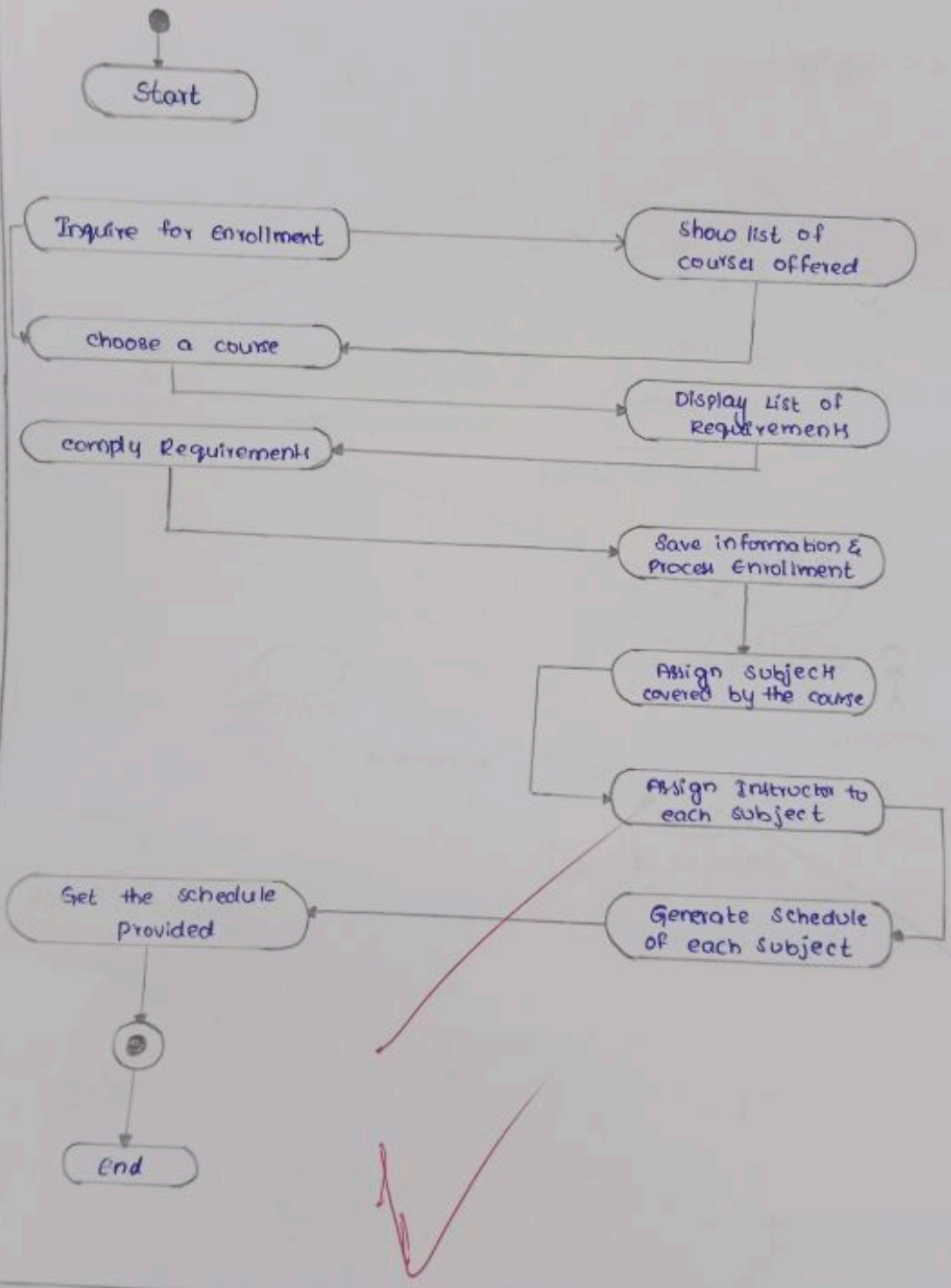
Experiment - 6

Course Management System (cms):

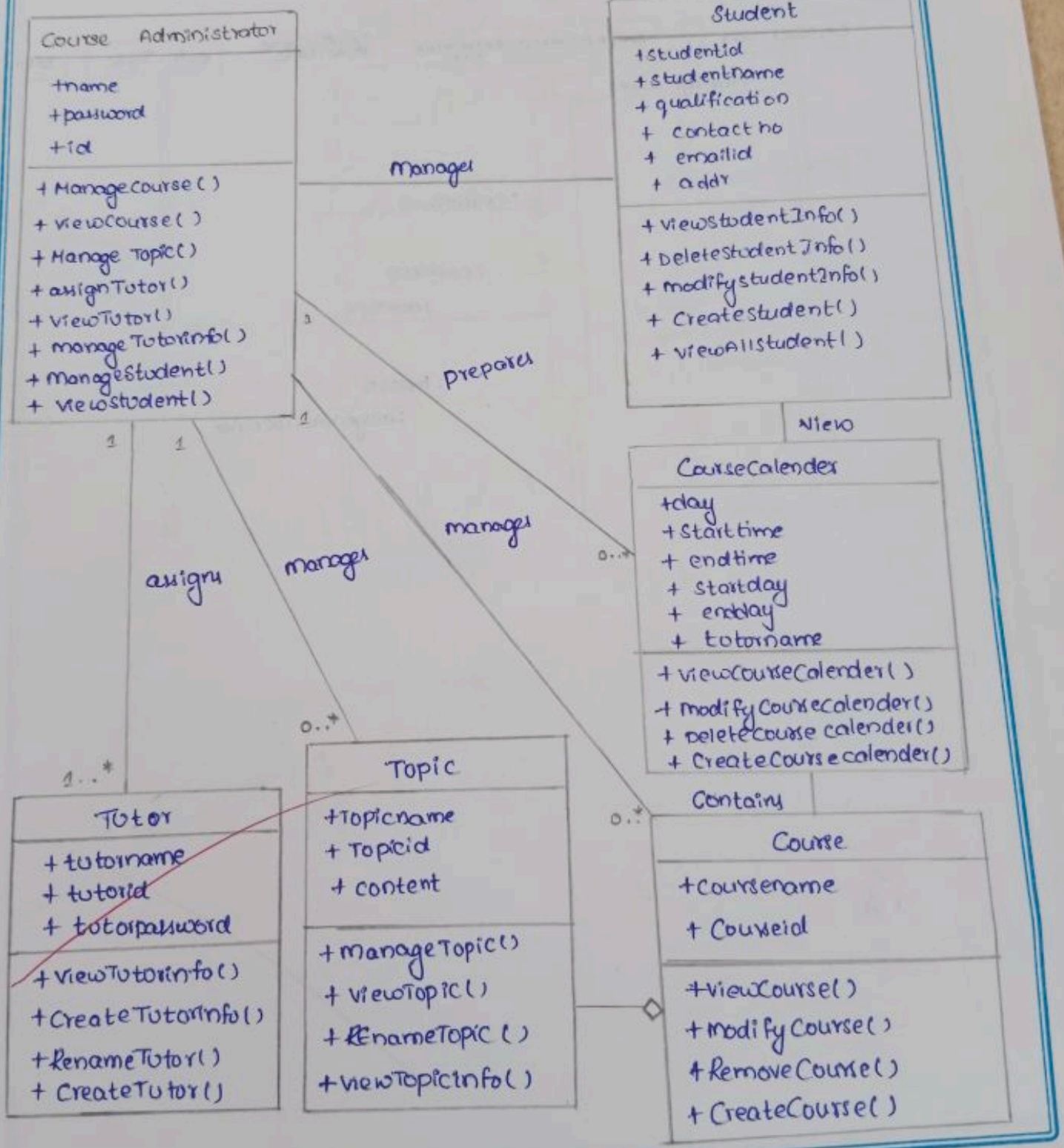
Use case diagram:



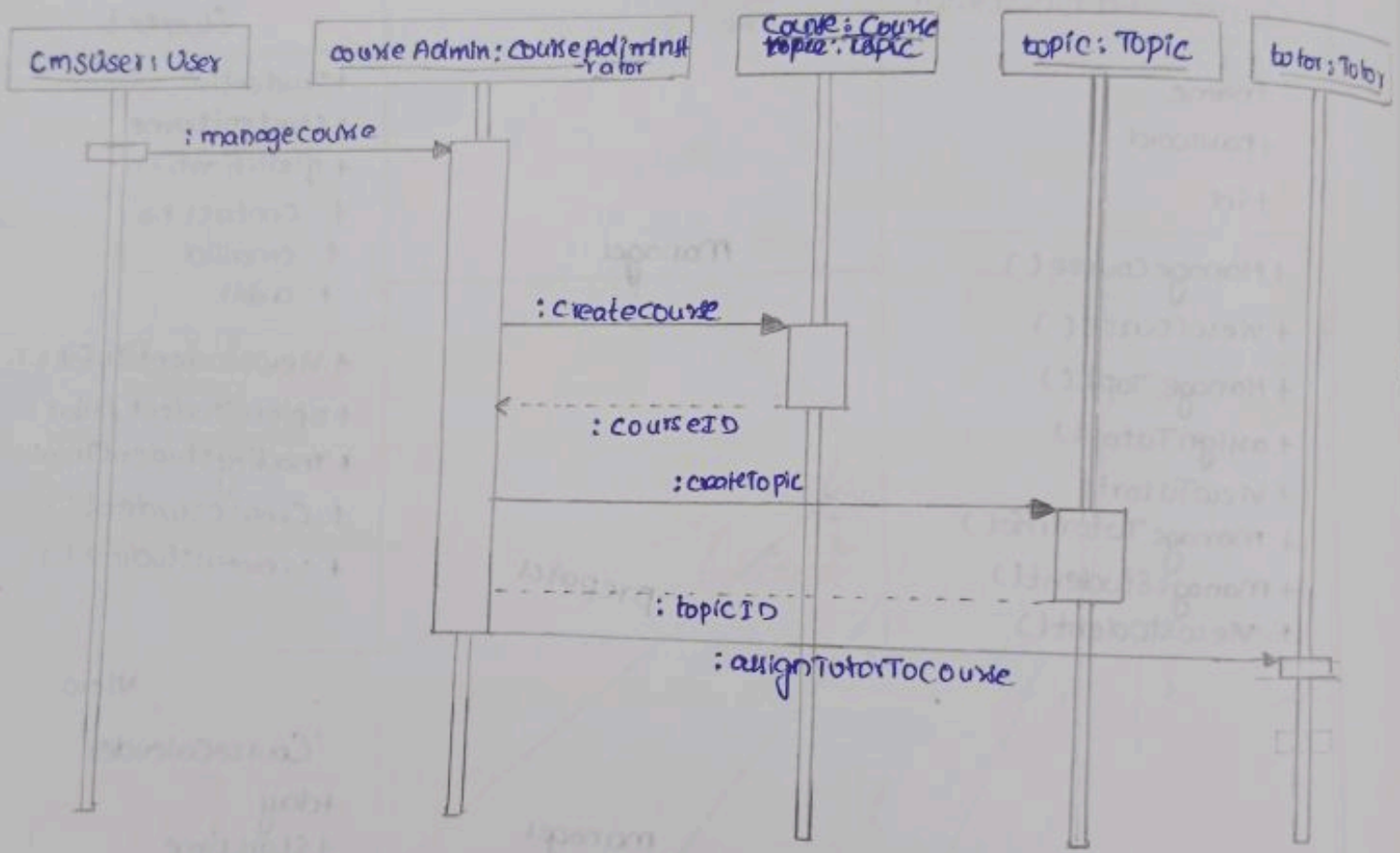
Activity Diagram:



Class Diagram:



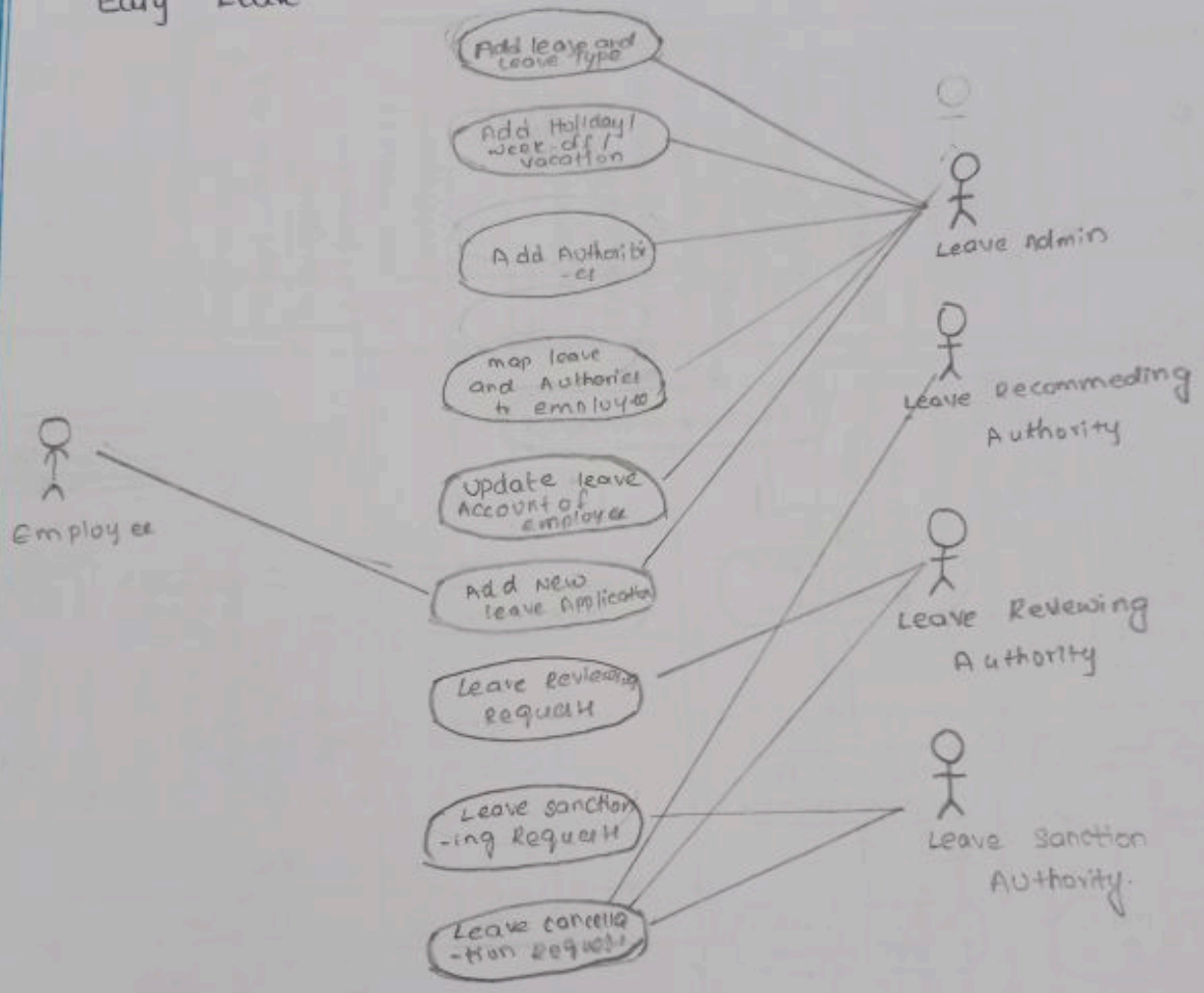
Sequence:


cmsUser: User
course Admin: CourseAdminist rator
course: Course
topic: Topic
topic: Topic
tutor: Tutor
: manageCourse
: createCourse
: courseID
: createTopic
: topicID
: assignTutorToCourse

Reg. No. : .....

Experiment - 7

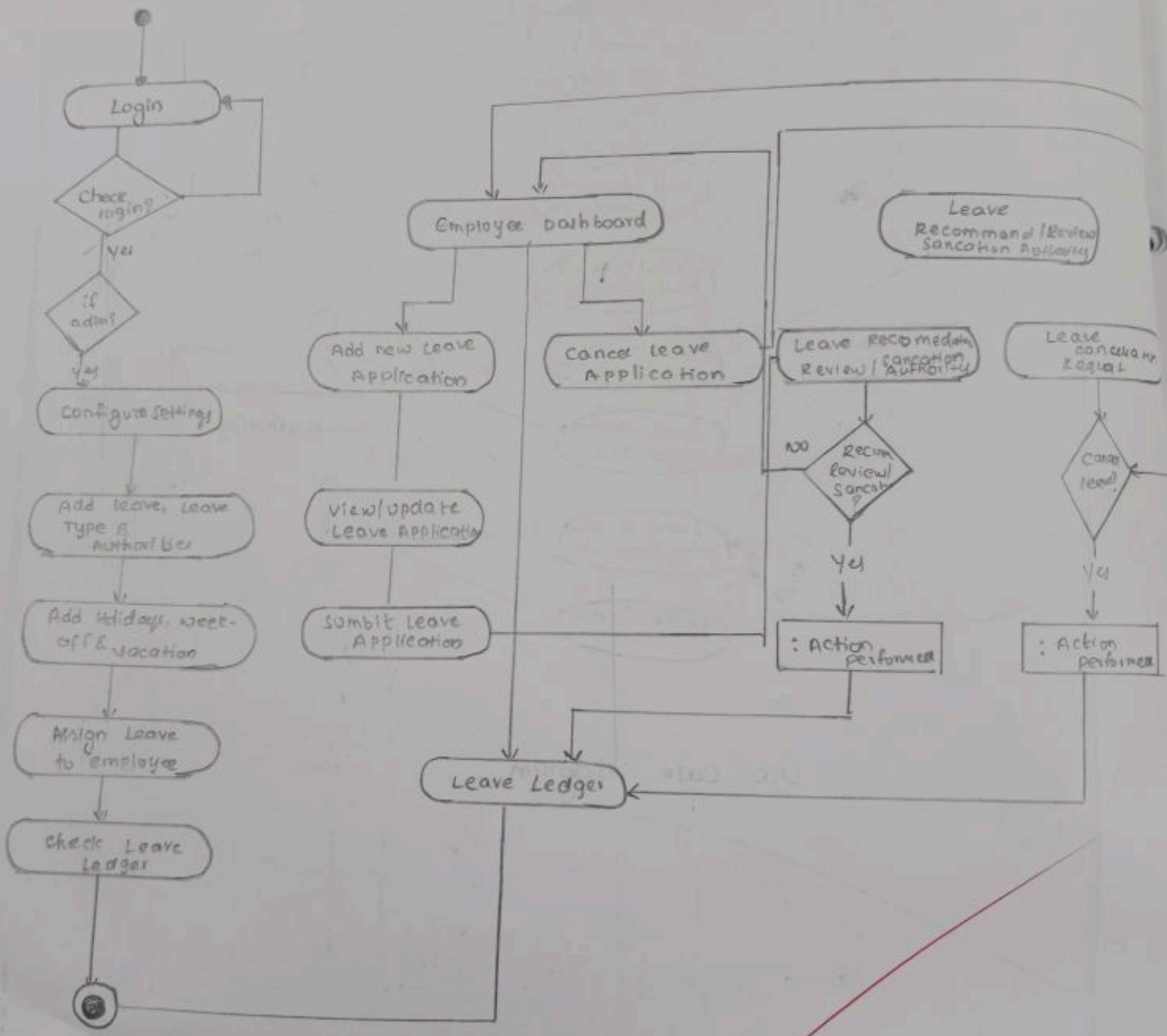
Easy Leave



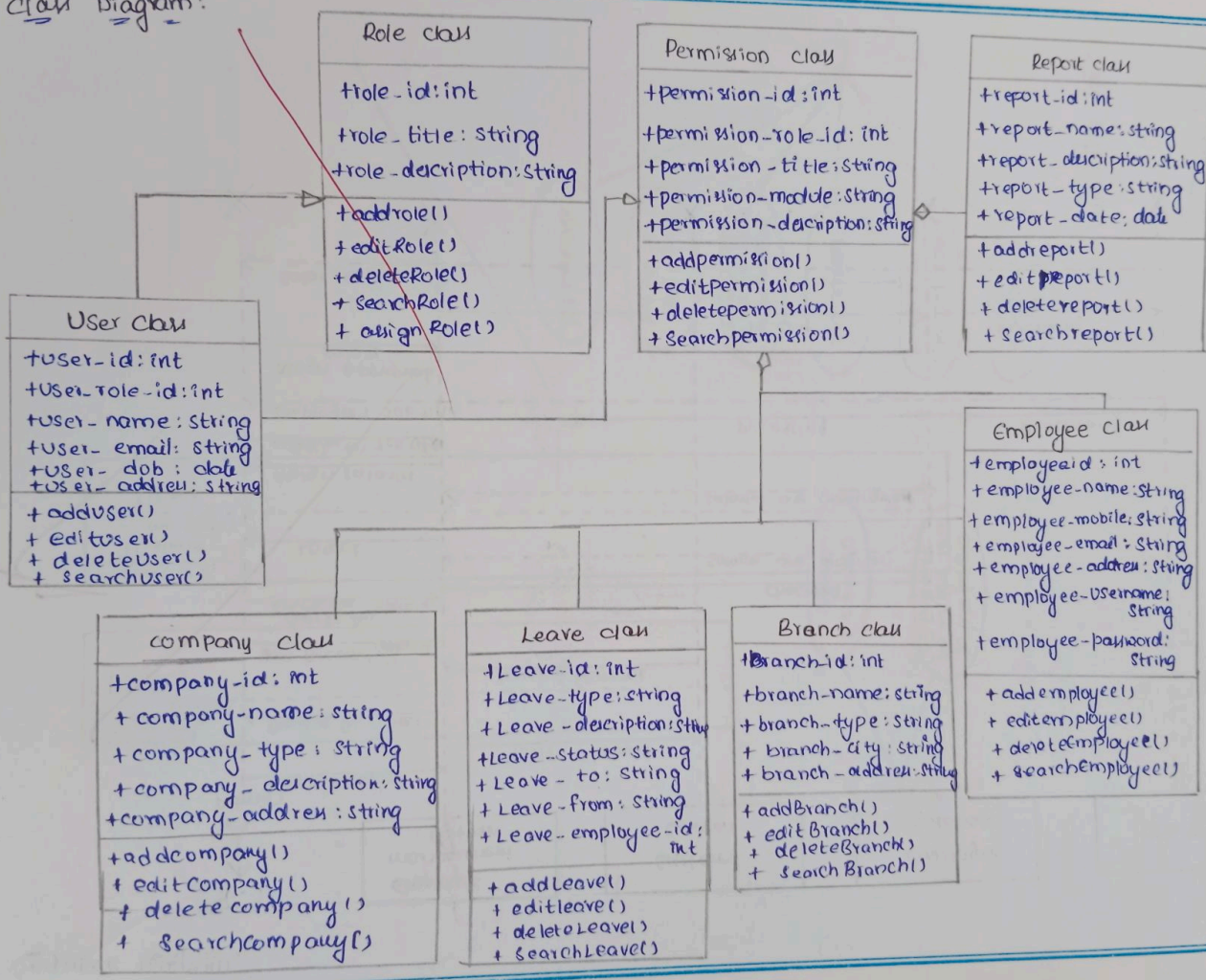
Use Case Diagram

Activity Diagram:

Admin



Class Diagram:

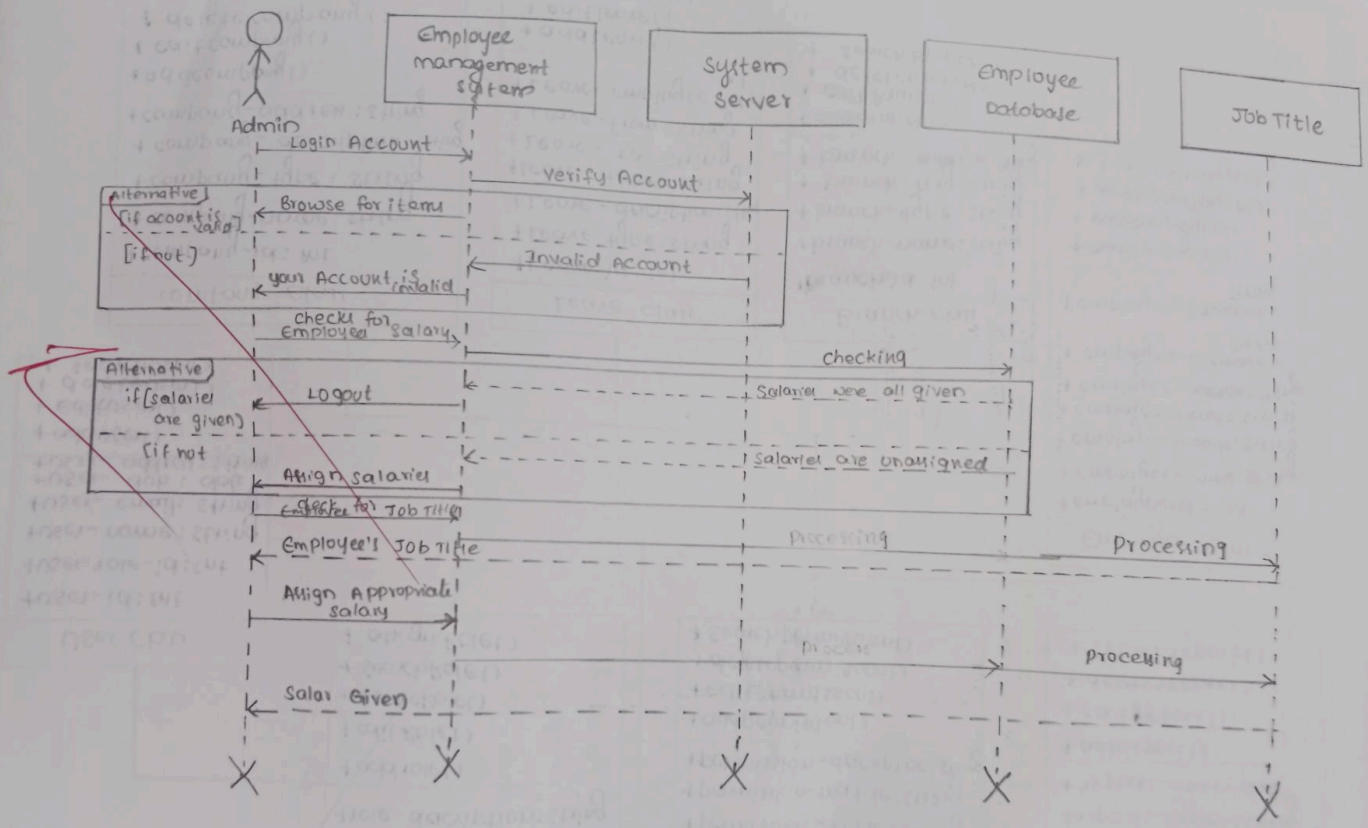


Reg No.: .....

INSTITUTE OF TECHNOLOGY

Page No.: 18

Sequence Diagram:



NRI
INSTITUTE OF TECHNOLOGY

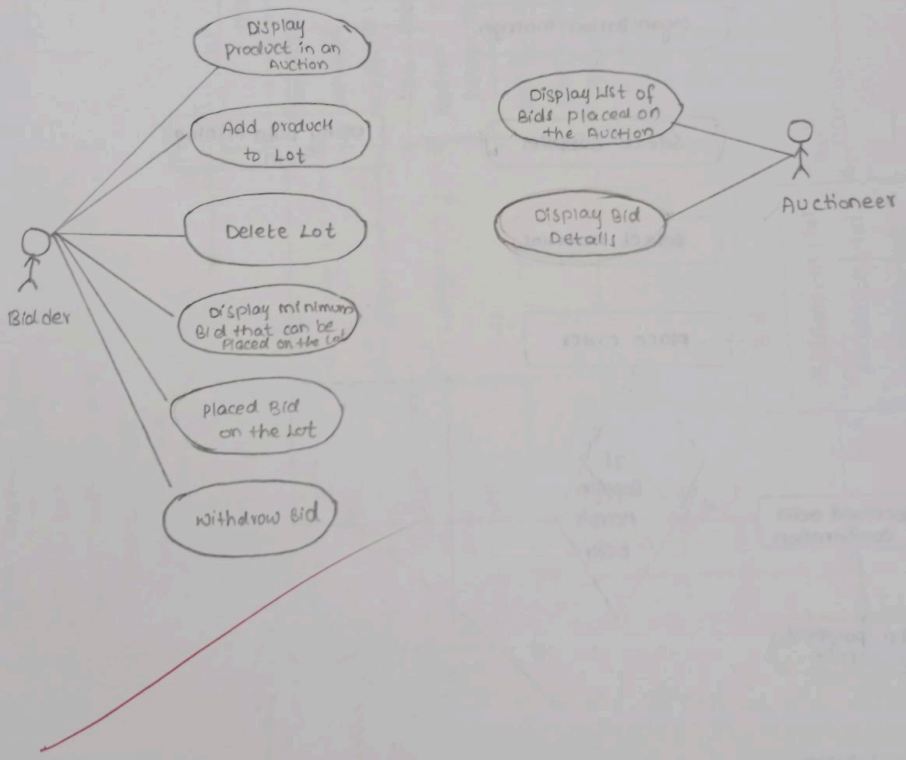
Reg. No. : .....

Experiment - 8

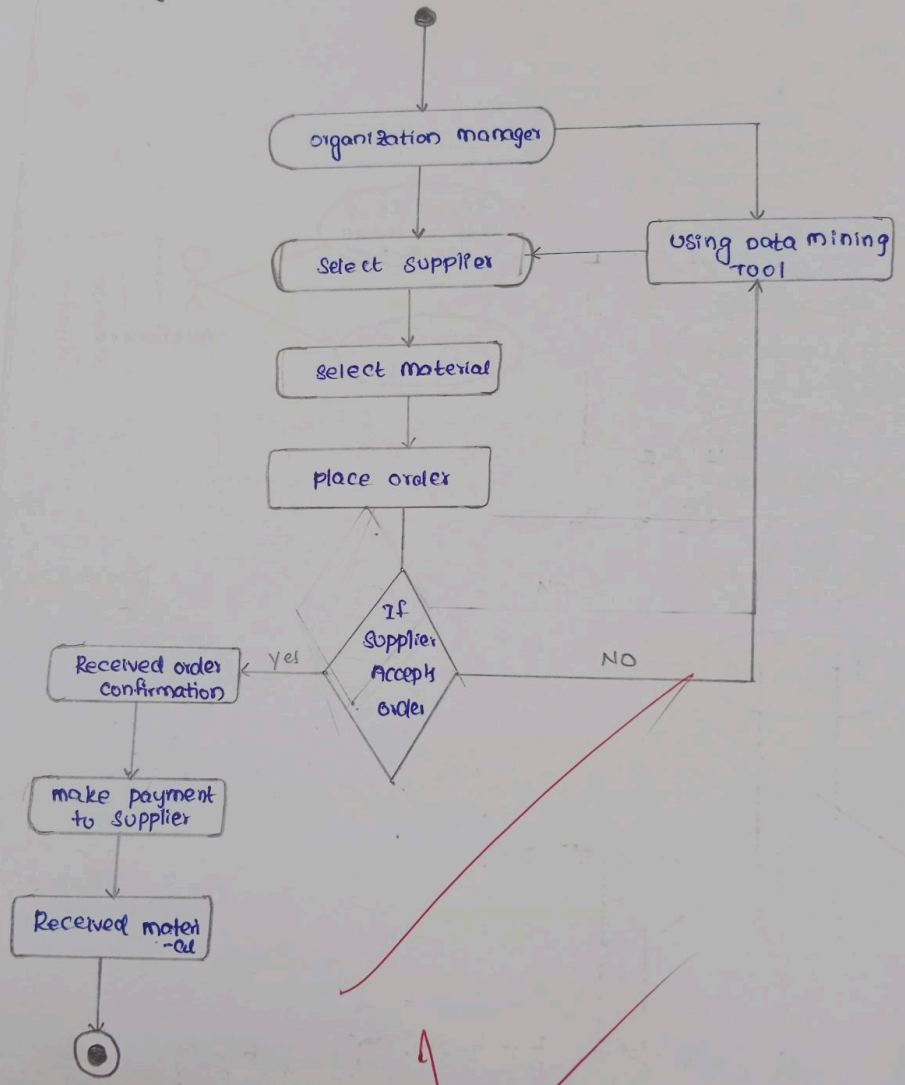
Page No. : 19

E-Bidding.

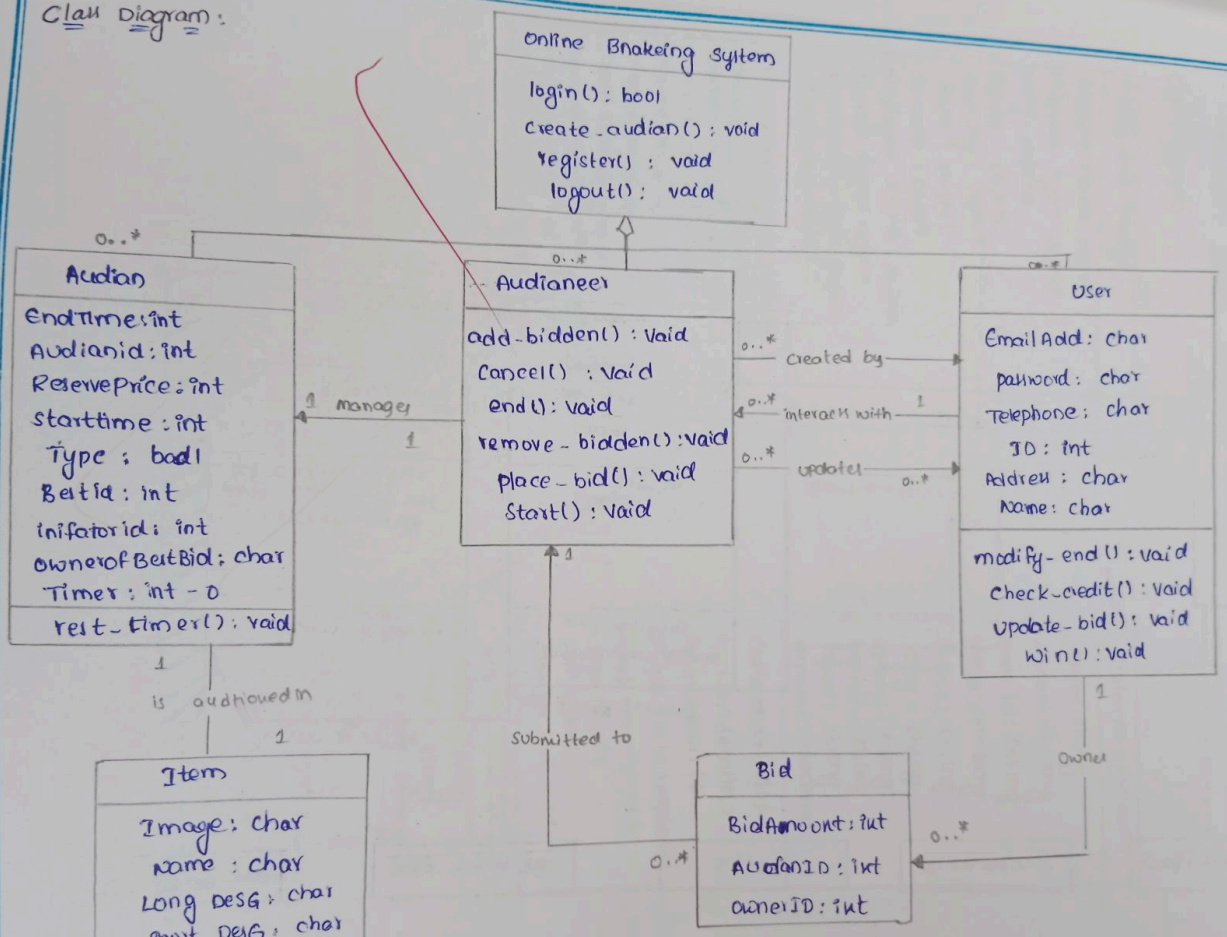
Class Diagram:



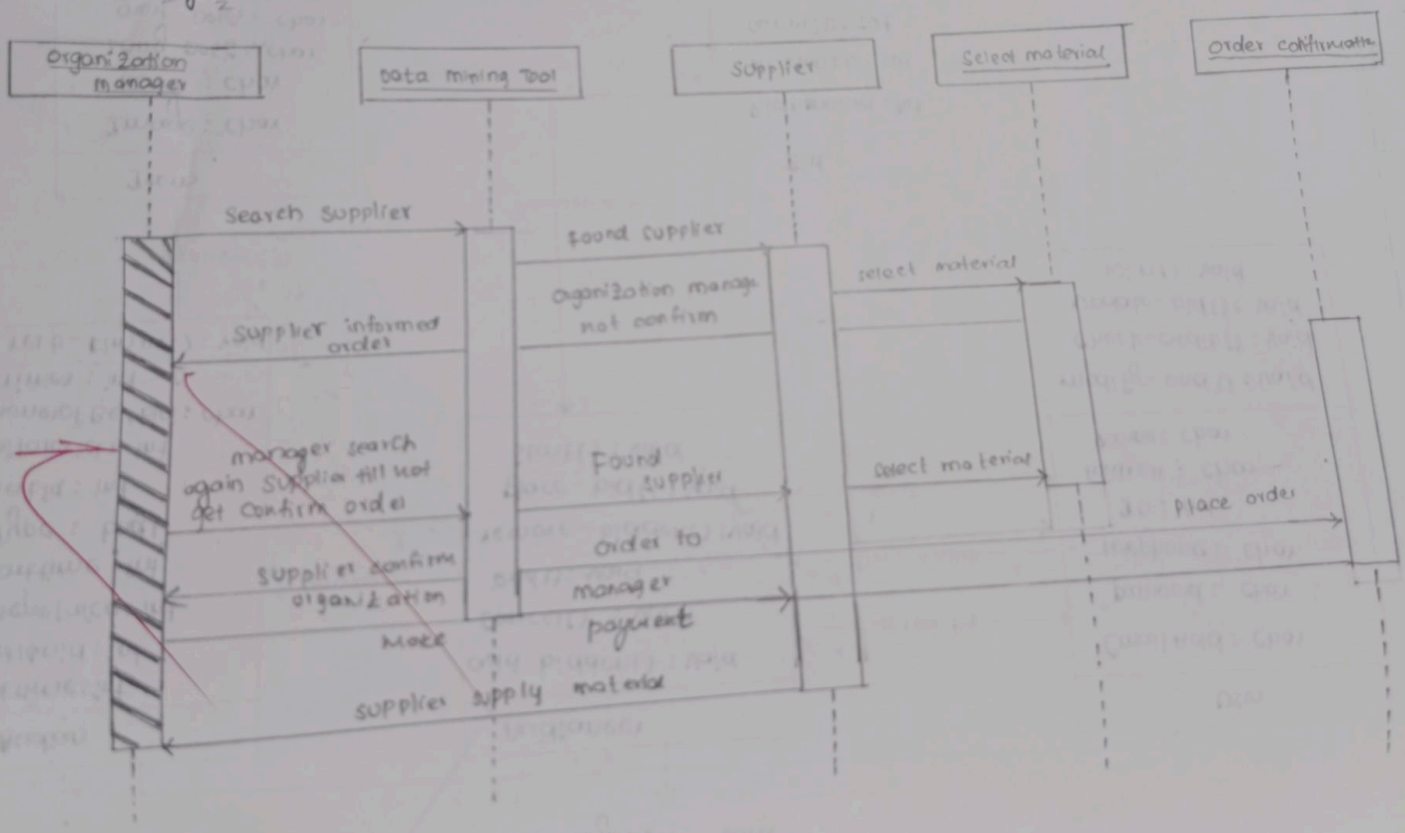
Activity Diagram:



Class Diagram :

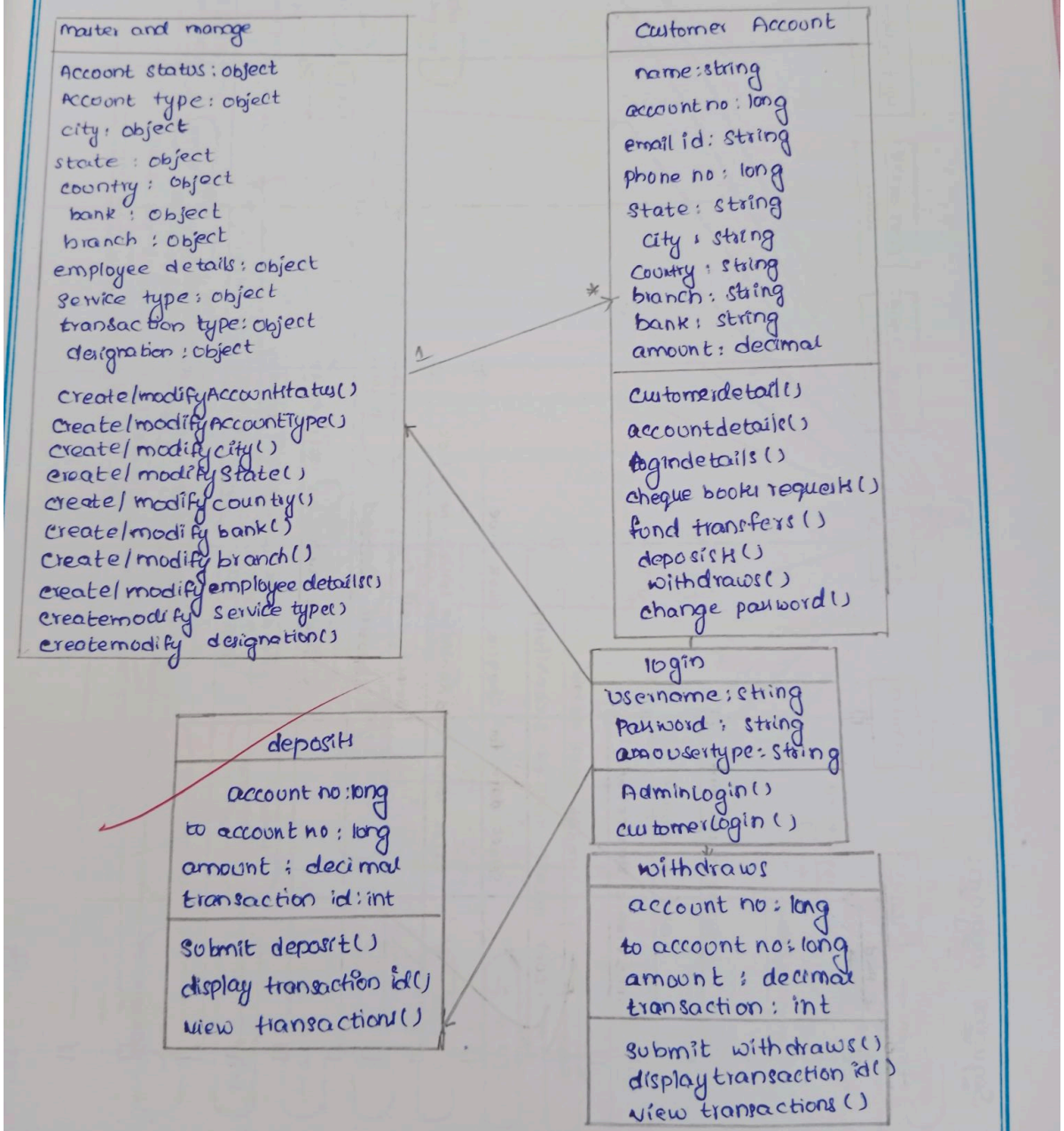


Sequence Diagram:

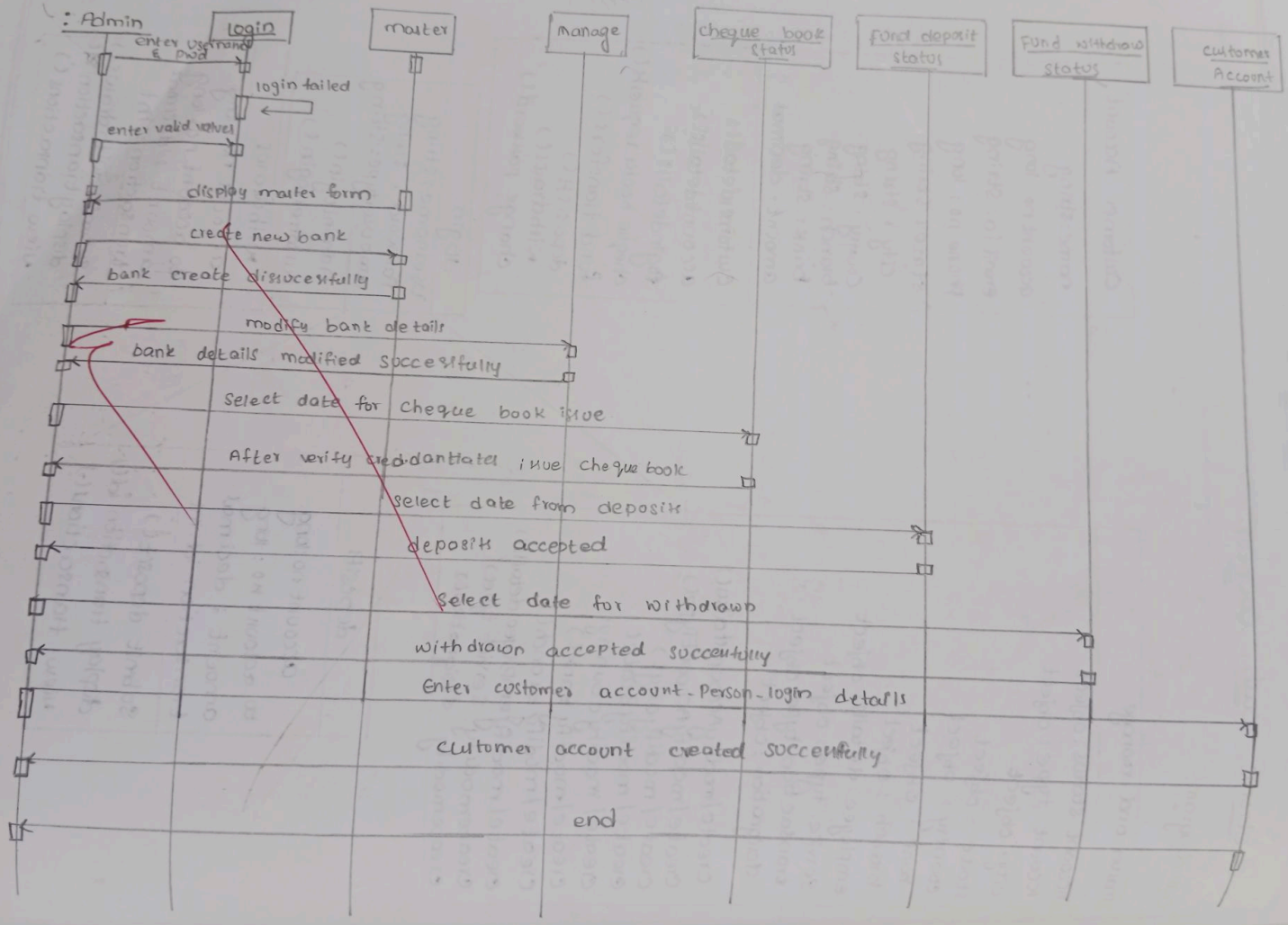


Electronic cash Counter.

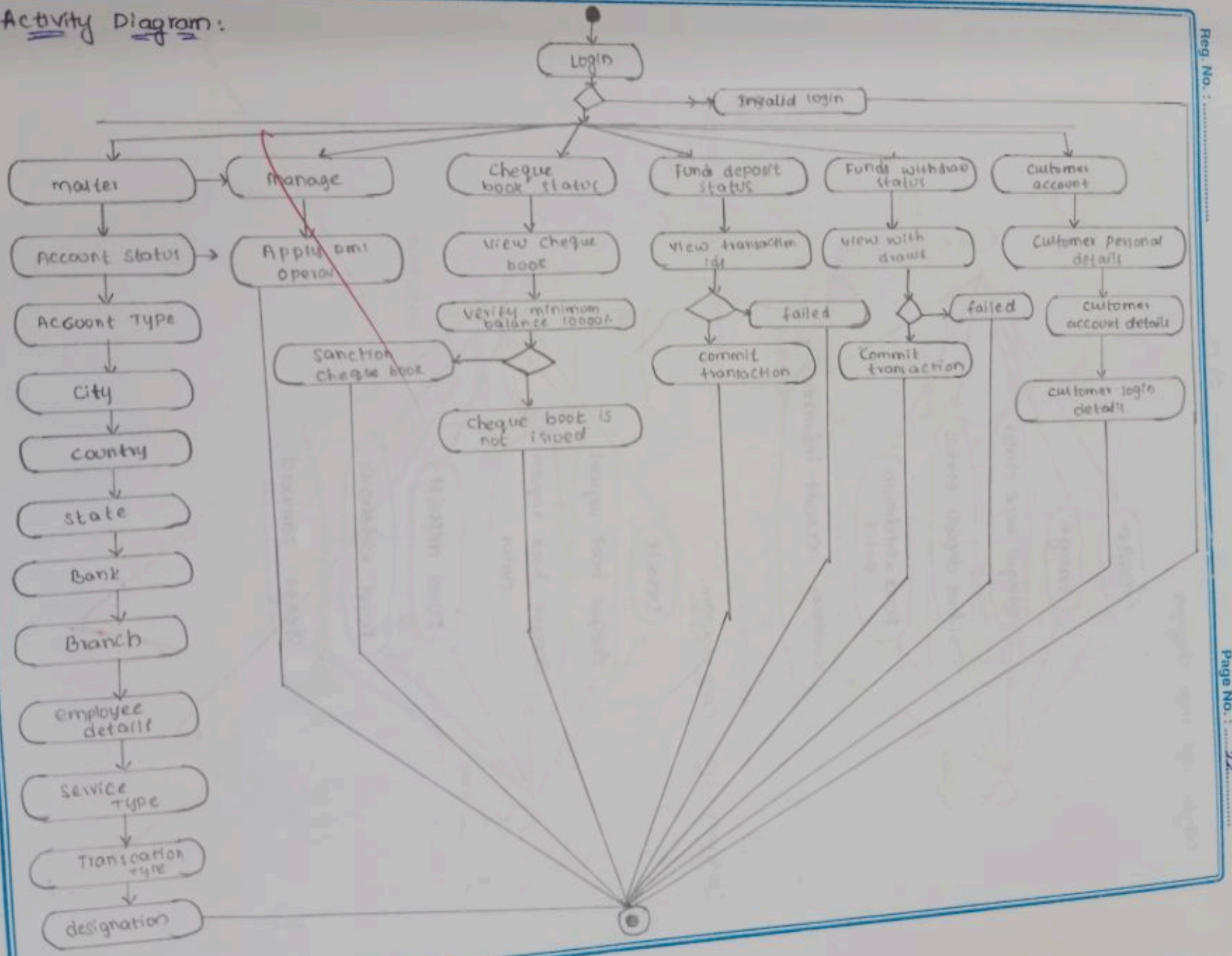
Class diagram:



Sequence diagram:



Activity Diagram:

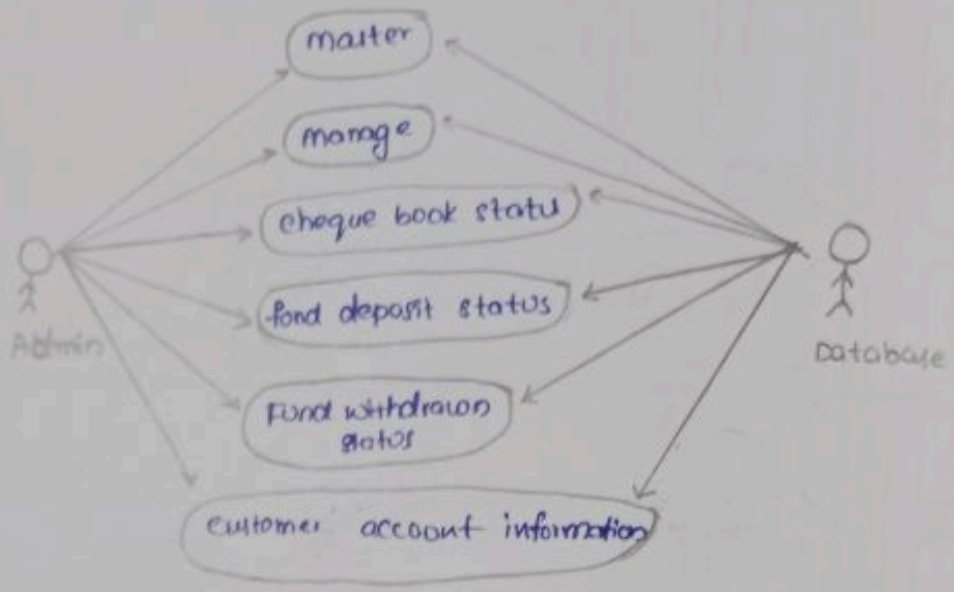


Reg. No. : .....

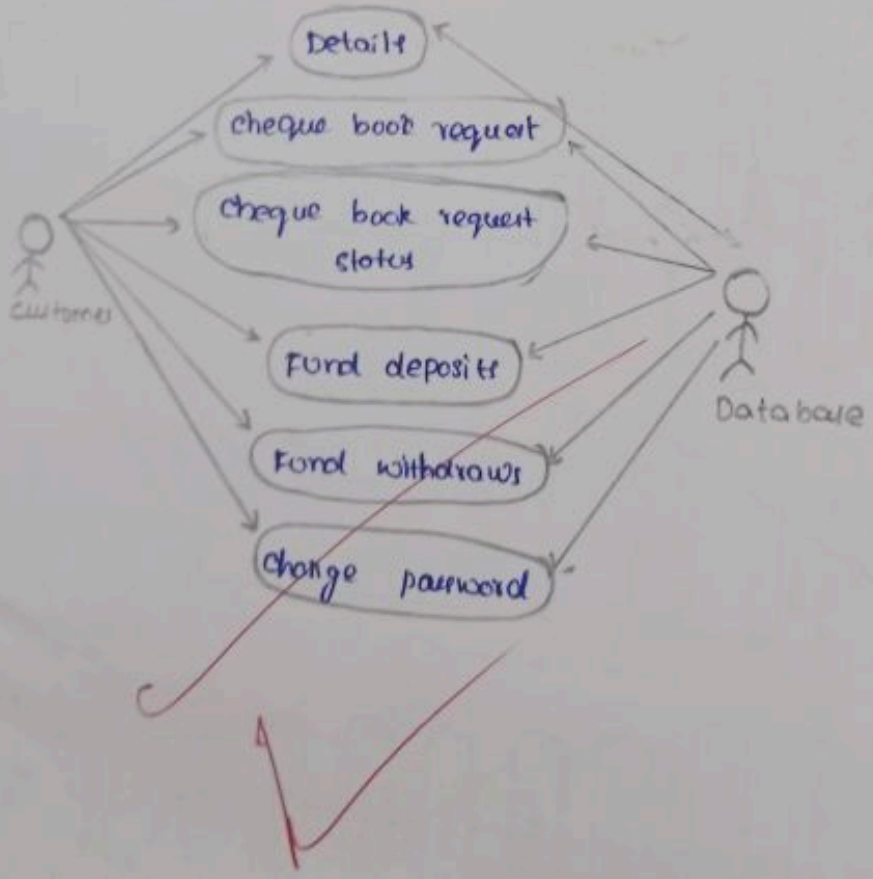
Page No. : 22

INSTITUTE OF TECHNOLOGY

Admin use case diagram

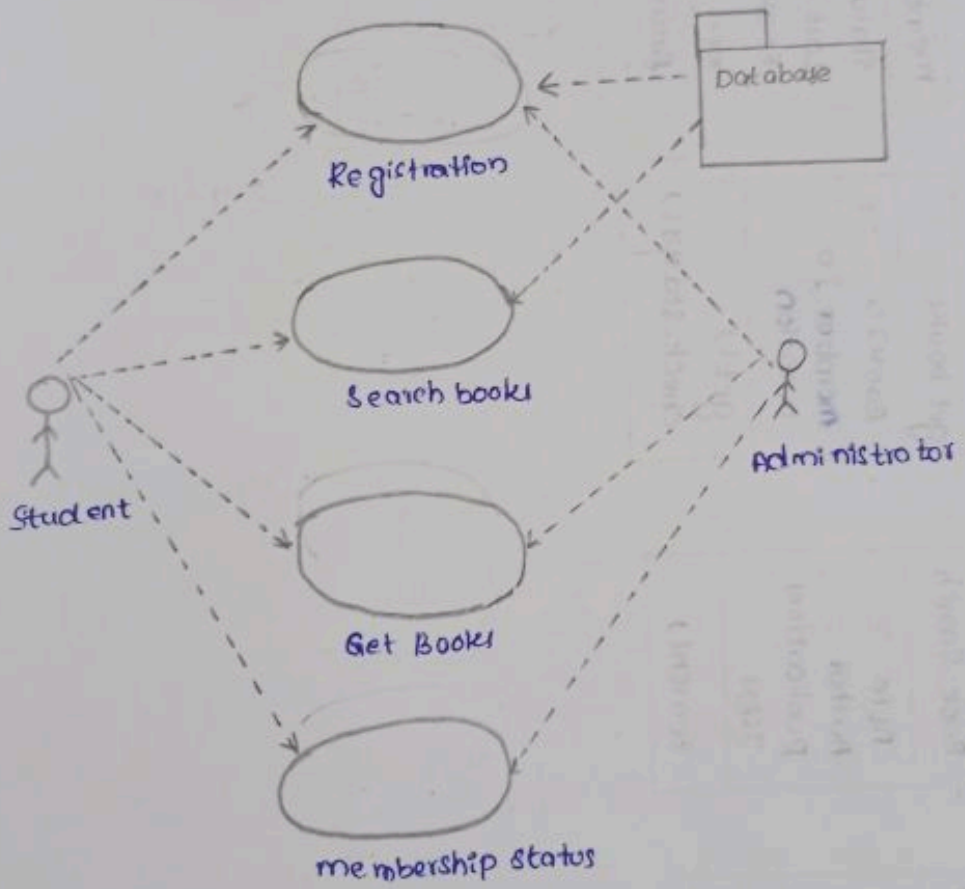


Customer use case diagram:

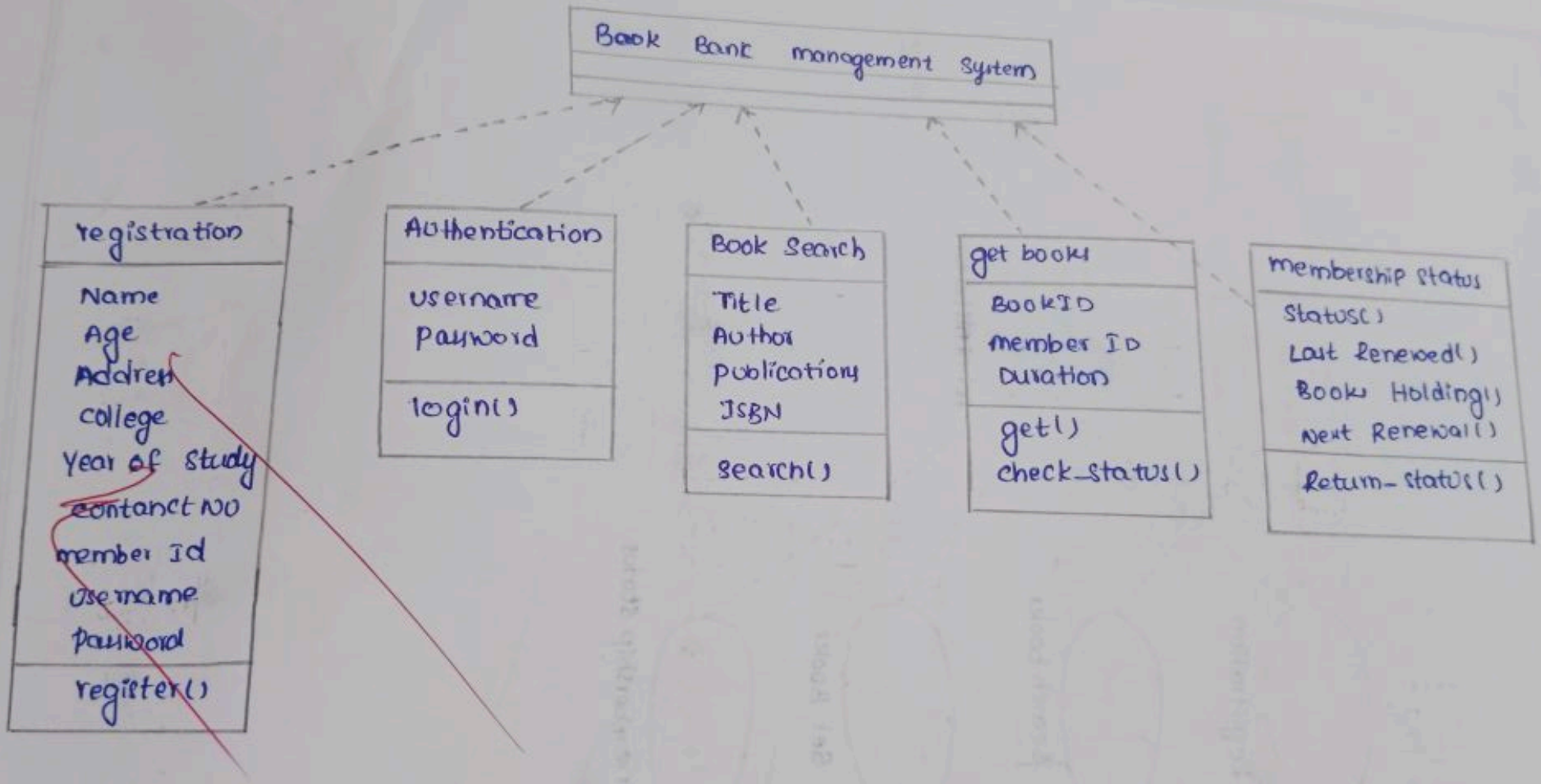


Book Bank

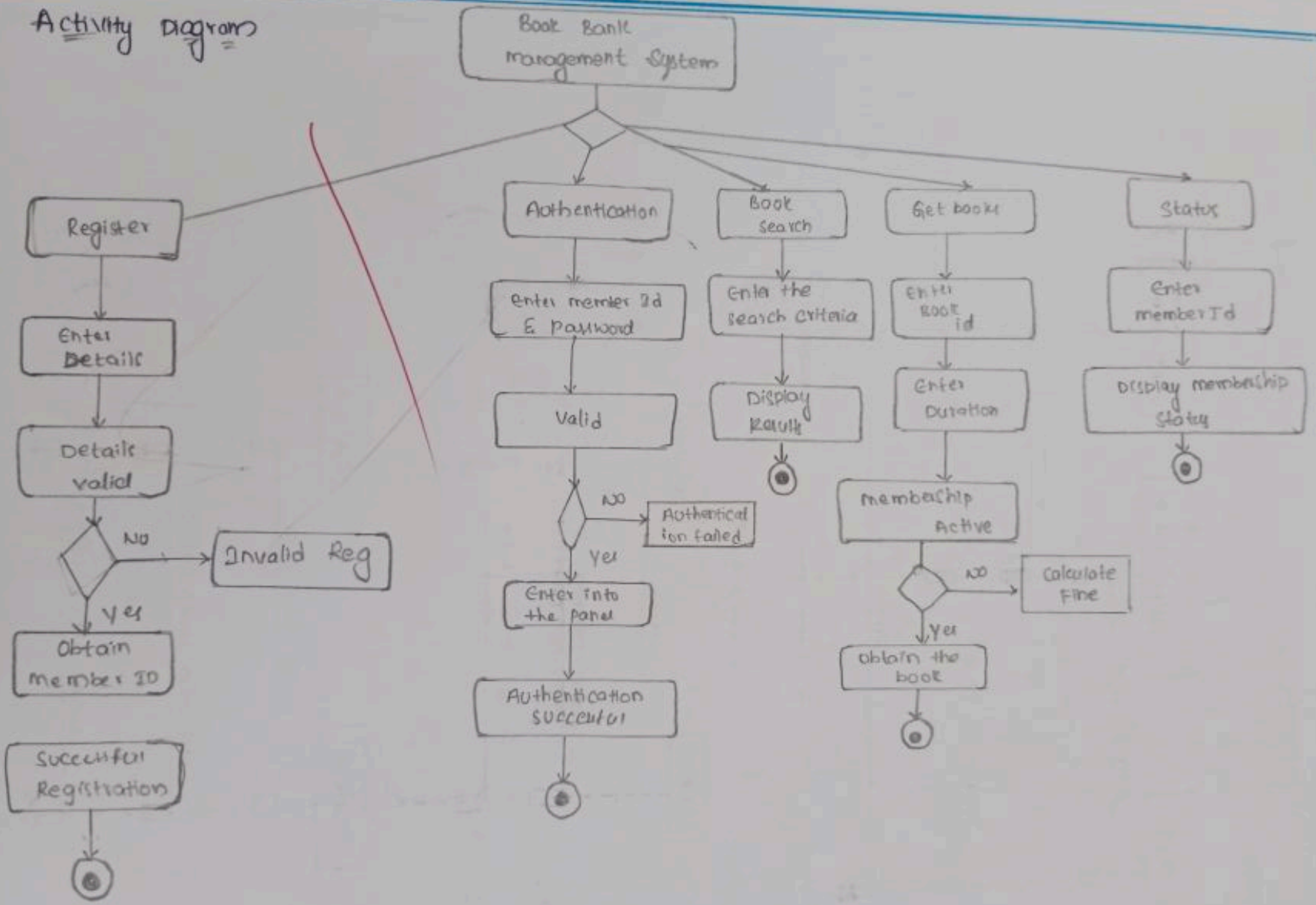
Use case diagram:



Class Diagram:



Activity Diagram



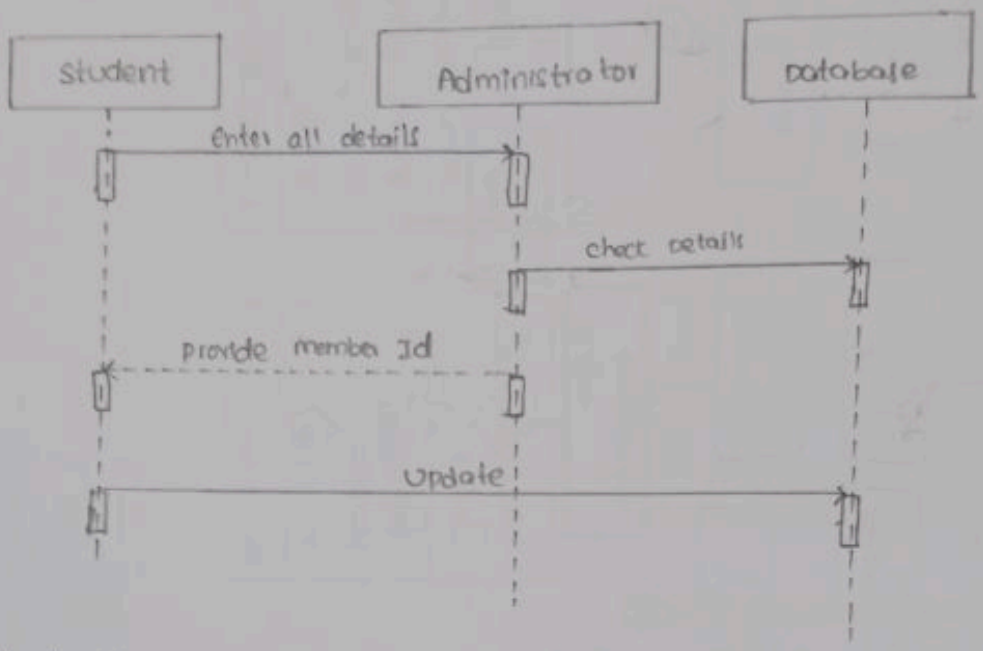
Reg. No. :

INSTITUTE OF TECHNOLOGY

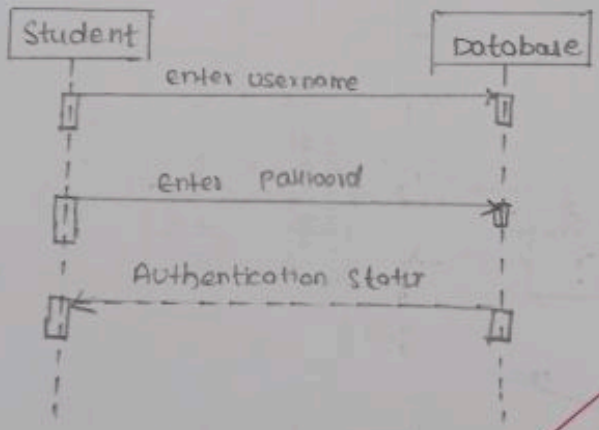
Page No. : 24

Sequence Diagram:

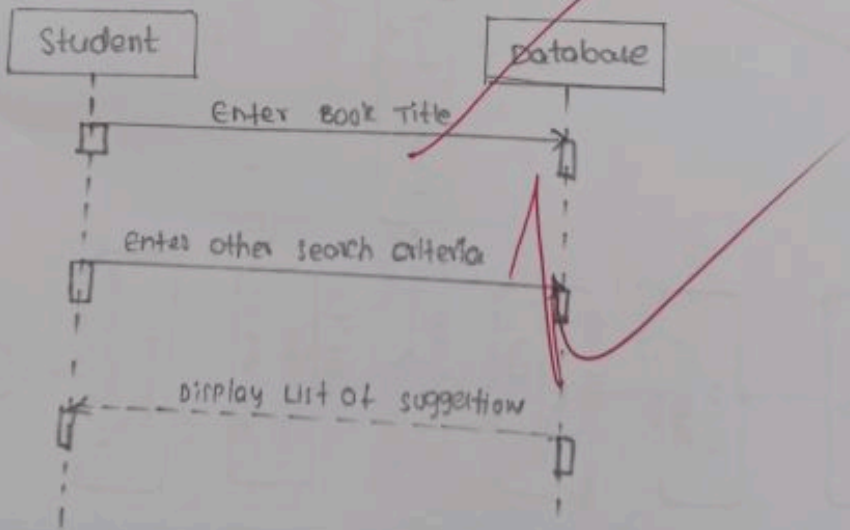
Registration:



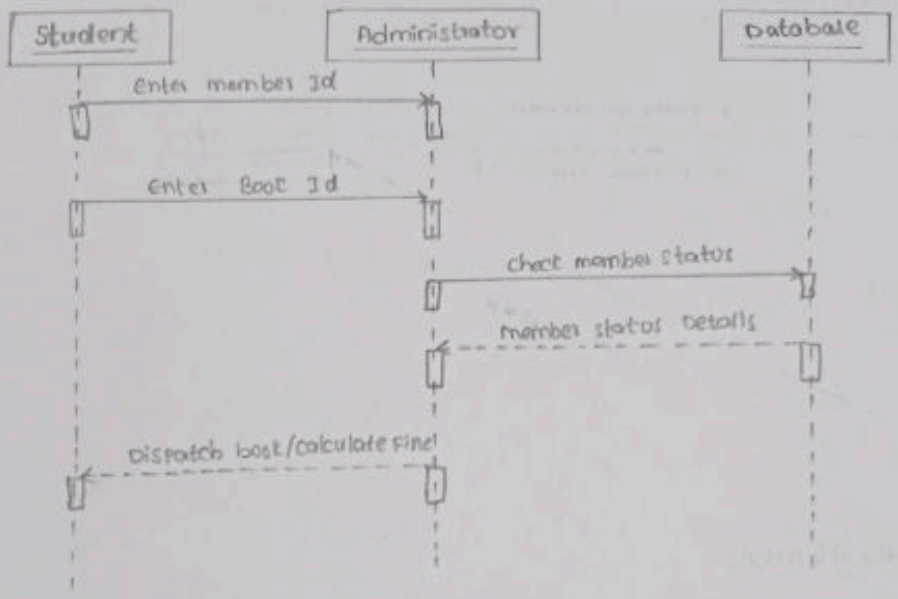
Authentication:



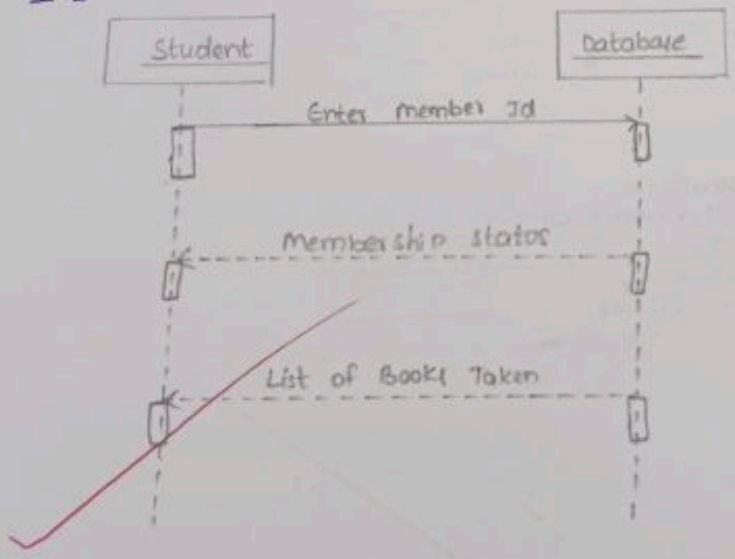
Search:



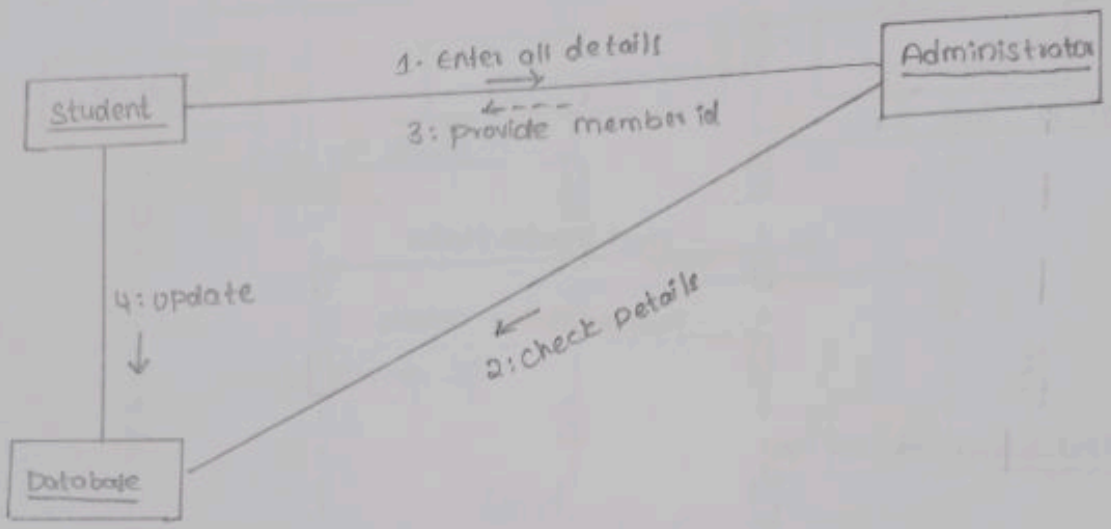
Get book



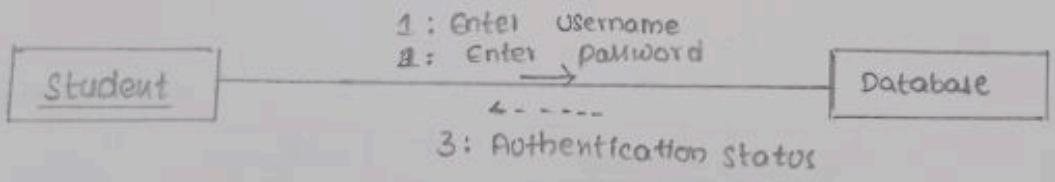
Status:



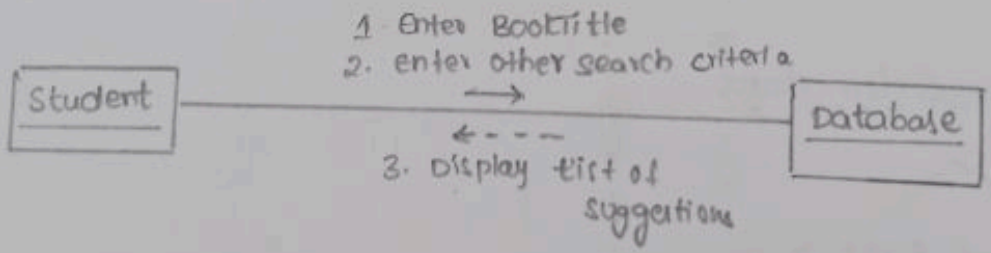
Collaboration Registration:



Collaboration Authentication:

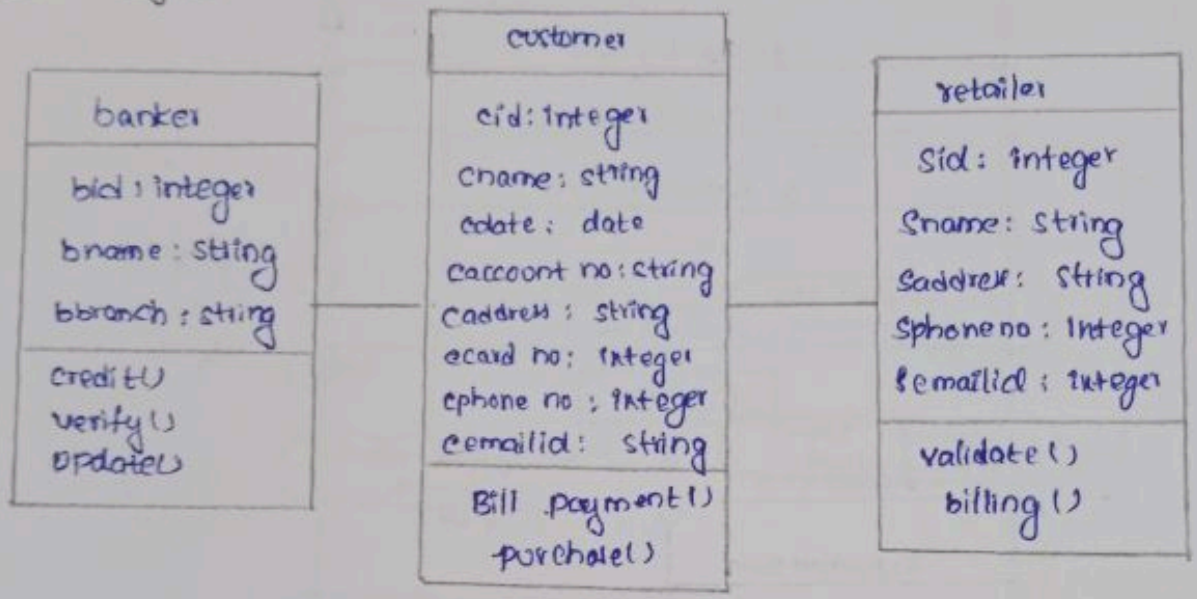


Search:

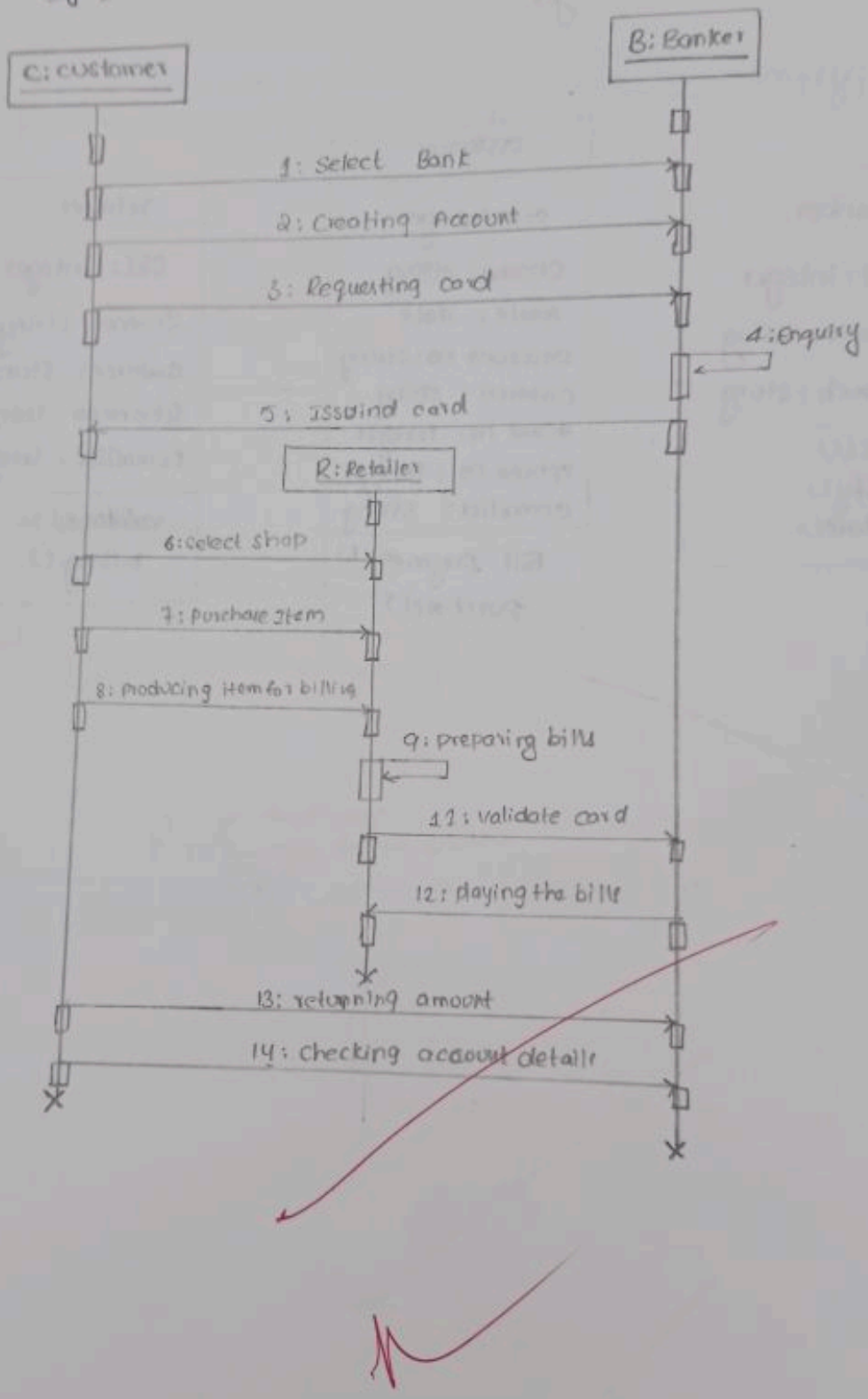


Credit Card processing:

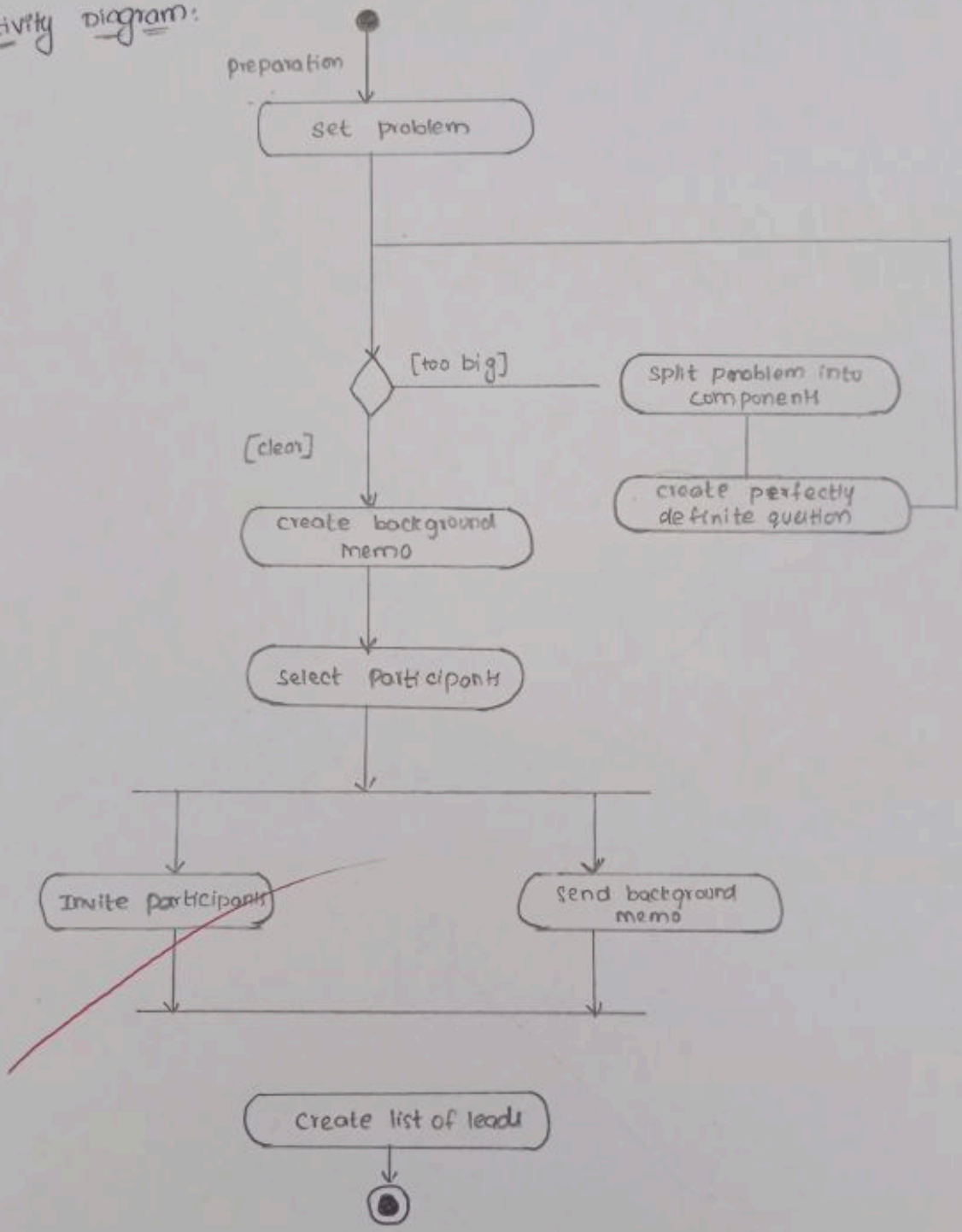
Class Diagram:



Sequence Diagram:



Activity Diagram:



Use case Diagram :