* Week-7: Develop class diagram and Object diagram using Rational Rose.

* Library Management System:

A library management system is designed to manage all the functions of a library. It helps automates all your library activities. It can be used to maintain library records such as number of books, issues, etc....

Class Diagram: These are generally used for conceptual modelling of static view of a software application and for translating models into programming code in a detailed manner.

Classes of LMS:

LMS class: Manages all operations of LMS. Central part of Organization.

User class: Manages all operations of user.

Librarian class: Manages all operations of Library database.

Book Class: Manages all operations of books.

Account class: Manages all operations of Accounts.

Staff class: Manages all operations of the staff.

Attributes of LMS:

LMS: userType, userName, password.

User: name, Id.

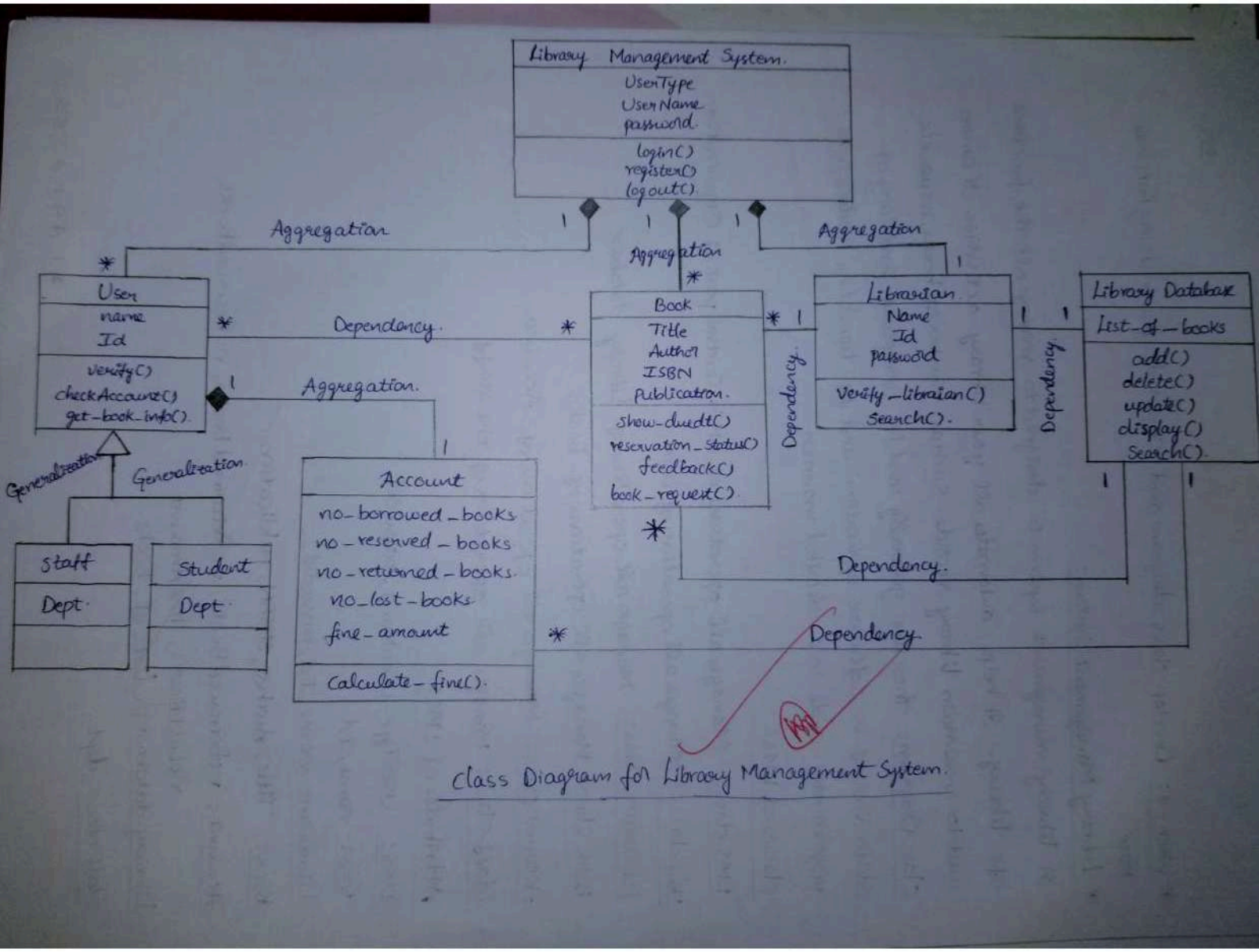
Librarian: name, Id, password,

Book: Title, Author, ISBN, publication.

Account: noBorrowedBooks, noReturnedBooks, noReservedBooks, noLostBooks, fineAmount.

Library database: list_of_books.

Staff class: dept.



Class Diagram for Library Management System.

Methods of LMS:

LMS: login(), register(), logout().

User: verify(), checkAccount(), get-book-info().

Librarian: verify-librarian(), search().

Book: show-duedt(), reservation-status(), feedback(), book-request().

Librarian database: Add(), Delete(), Update(), Display(), Search().

Concepts Used:

Aggregation: It simply shows a relationship where one thing can exist independently at other thing. It will be represented by an edge with a diamond end pointing towards superclass.

Generalization: It is a relationship which implements the concept of object oriented called Inheritance. It can be utilized in class, component, deployment and usecase diagrams to specify that the child inherits actions, characteristics and relationships from its parent.

Dependency: A dependency relationship is the kind of relationship in which a client is dependent on a supplier.

Multiplicity: Number of eleements of a class is associated with another class. These relations can be one to one, many to many, many to one, one to many.

To denote one element, we use 1.

For zero elements, we use 0.

For many elements, we use *.

* ATM Application:

Automated Teller Machine (ATM), also known as ABM is a banking system which allows users to have access to financial transactions. These can be done in public space without any need for clerk (or) cashier (or) bank teller.

Classes:

Bank: Manages all bank transactions and information.

ATMinfo: Manages all ATM transactions for every bank.

Customer: Behaves as the user for banks.

Debit card: Handles all debit and account details.

Account: Handles all account types and numbers for customers.

ATM Transaction: Holds all transactional information.

Current Account: Handles currently using account details.

Withdrawl: Handles amount withdrawn by user/customer.

PIN validation: Changing PIN's and validating current transaction.

Attributes:

Bank: Code, address.

ATMInfo: location, managed By.

DebitCard: Cardno, owned By.

Customer: name, address, dob.

Account: type, owner.

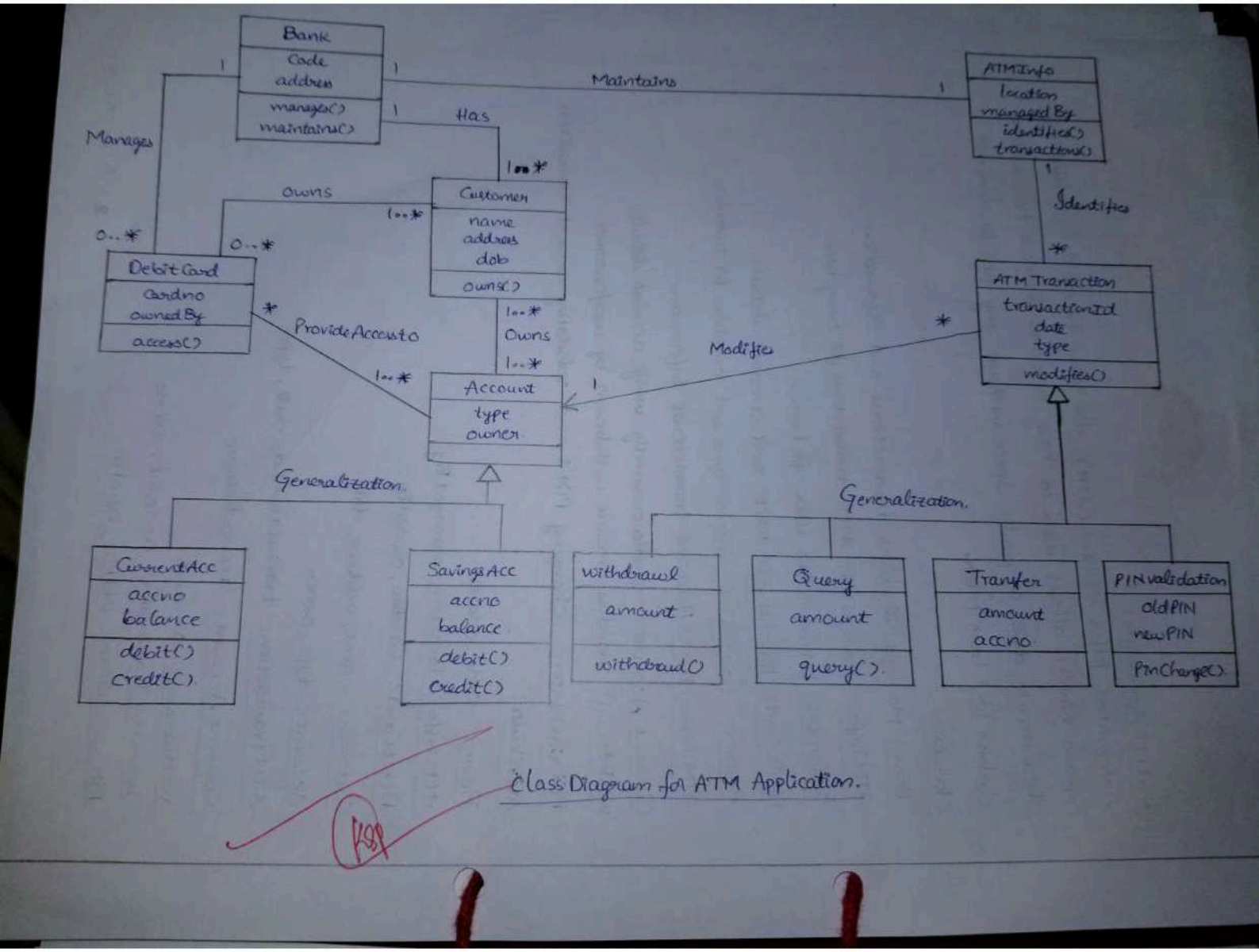
ATMTransaction: transactionId, date, type

Current Account: accno, balance.

Withdrawl: amount.

Transfer Transaction: amount, accno.

PIN validation: oldPin, newPin.



Class Diagram for ATM Application.

Methods:

Bank: manages(), modifies()

ATMInfo: identifies(), transactions().

Customer: owns().

DebitCard: access().

ATMTransaction: modifies().

Savings Acc: debit(), credit().

withdrawl: withdrawl().

PINValidation: pinChange().

Concepts Used:

Dependency: It is a type of relationship in which a client is dependent on a Supplier.

Generalization: It is a type of relationship which implements the concept of object oriented called inheritance. It can be utilized in class, Component, deployment and usecase diagrams to specify that the child inherits actions, characteristics from its parent.

Association: It is a semantic relationship between classes that show how one instance is connected (a) merged with others in system. Since it connects the object of one class to object of another class, it is categorized as a Structural Relationship.

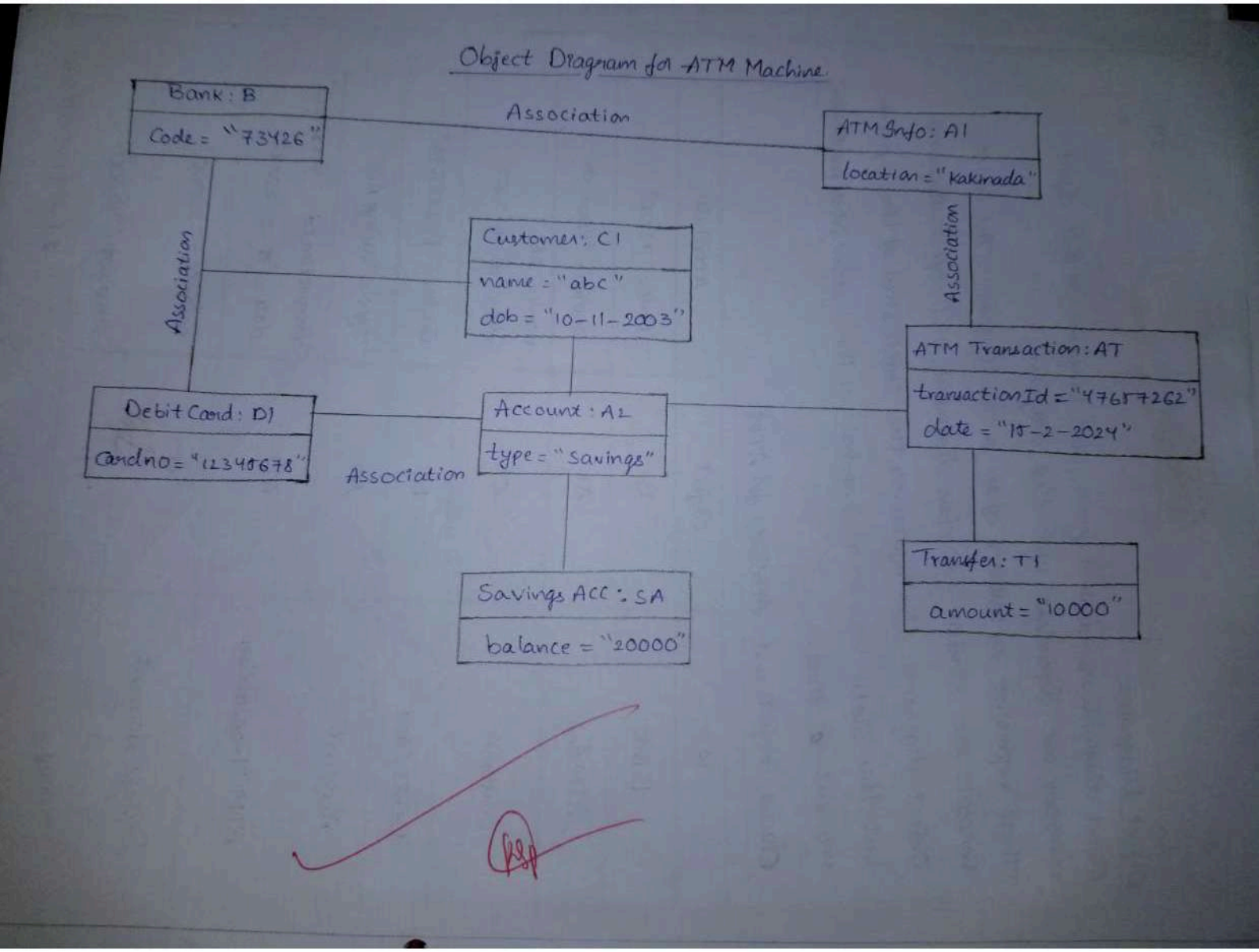
Object Diagram:

Object diagrams are derived from class Diagram so that object diagrams are dependent upon class diagram.

These represent an instance of an a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represents the static view of the system but this static view in a snapshot of the system at a particular moment of time.

Classes, Objects and Attributes for ATM:

class	Object	Attributes.
Bank	B	Code = "73426"
ATM Info	A1	location = "Kakinada
Customer	C1	name = "abc" dob = "10-11-2003"
Debit Card	D1	Cardno = "12345678"
Account	A2	type = "savings Acc"
ATM Transaction	AT1	transactionId = "47657262" date = "15-2-2024"
Savings Account	SA	balance = "20000"
Transfer	T1.	Amount = "10000".


Object Diagram for ATM Machine
Bank : B
Code = "73426"
Association
ATM Info: A1
location = "Kakinada"
Association
Association
Customer: C1
name = "abc"
dob = "10-11-2003"
ATM Transaction: AT
transactionId = "47657262"
date = "15-2-2024"
Debit Card: D1
CardNO = "12340678"
Association
Account : A1
type = "savings"
Transfer: T1
amount = "10000"
Savings ACC : SA
balance = "20000"

Class, Object and Attributes for LMS:

Class	Object	Attribute
Library Management System.	LMS	Username = "abc" password = "123"
User	U	name = "abc" id = "321"
Book	B	title = "story"
Librarian	L1	name = "xyz" id = "7272"
Library Database.	LDB.	list_of_books = "100"
Account	A1	no_of_borrow = "4" fine_amount = "40".
Staff	S1	dept = "cse"
Student	S2	class = "cse".

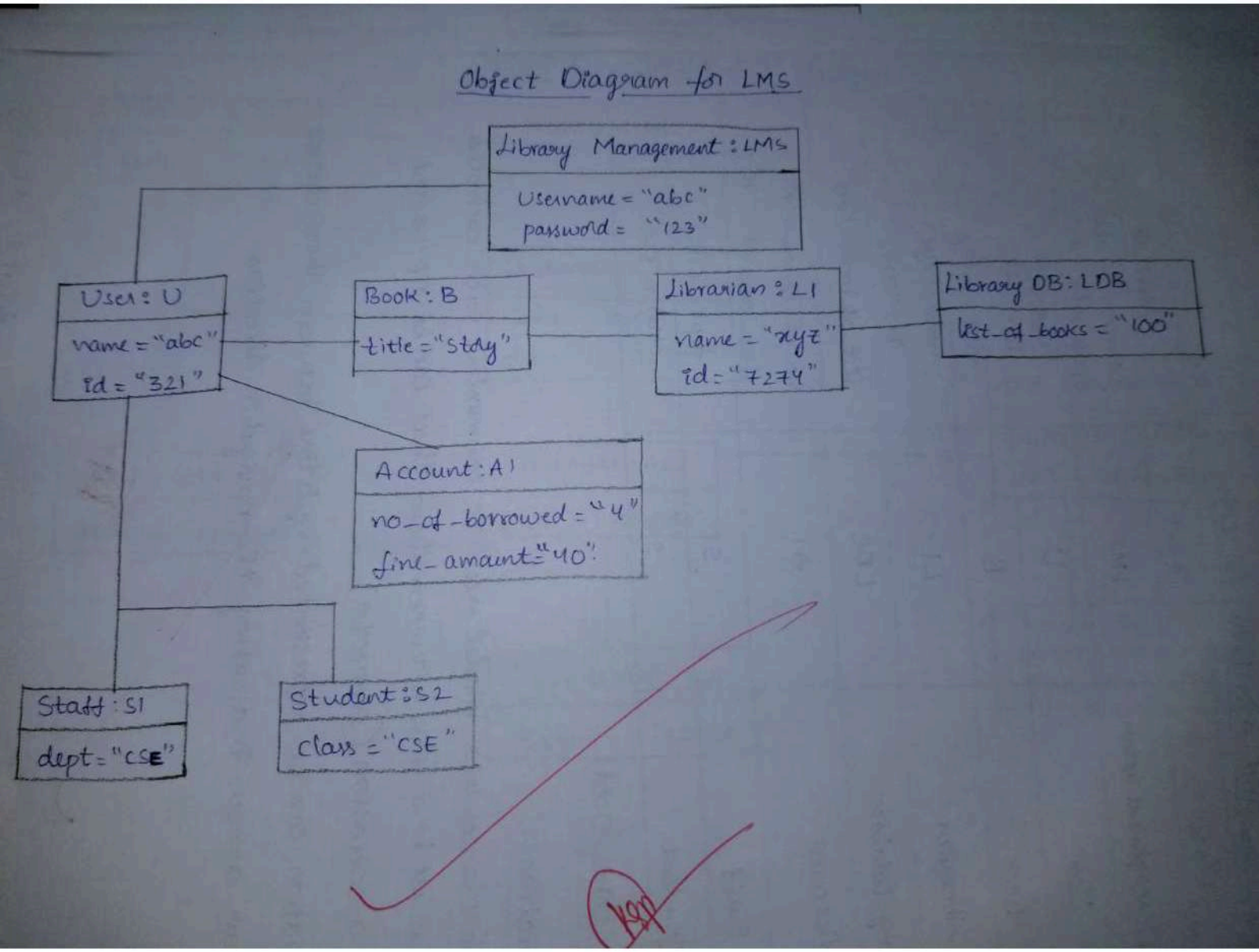
Relationships Used:

Association:

If two classes in a model need to communicate with each other, there must be a link between them, that can be represented by an association (connected).

Association can be represented by a line between these classes with an arrow indicating the navigation direction.

Object Diagram for LMS



Expt. No. : 8.
Date :

* Week-8:

Aim: Develop Usecase diagrams and elaborate usecase descriptions and scenarios.

Description: A Usecase diagram is a vital tool in system design, it provides a visual representation of how users interact with a system. It serves as a blue print for understanding the functional requirements of a system for a user's perspective, aiding in the communication of development process.

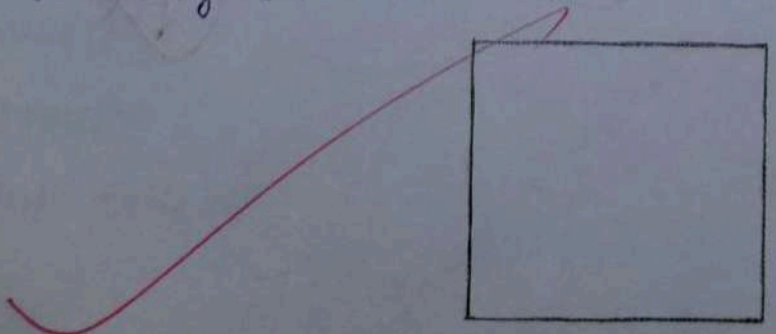
Actors: Actors are external entities that interact with the system. These can include users, other systems or hardware devices.

Actor.

Usecases: Usecases are like scenes in the play. They represent specific things your system can do.

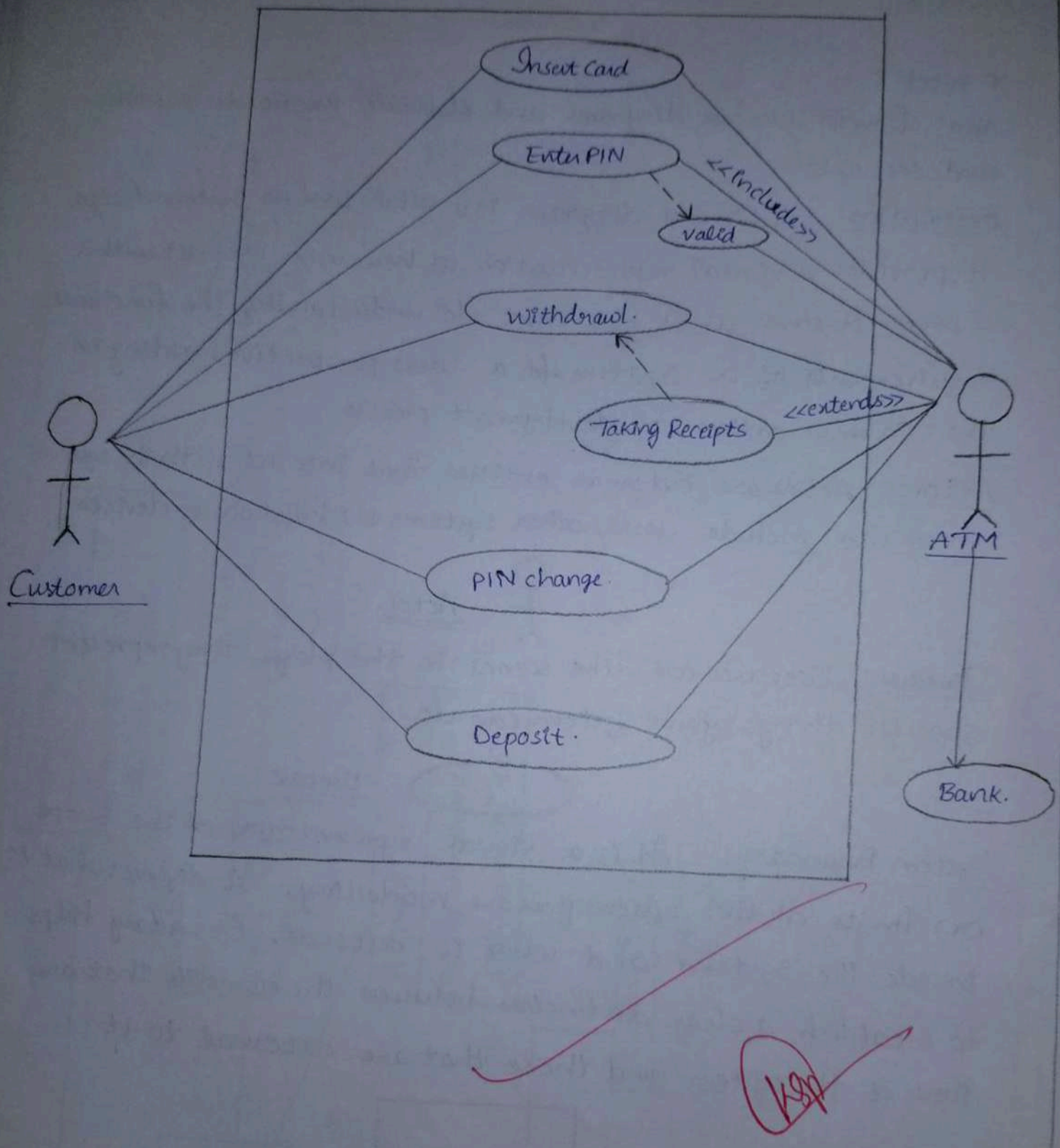
Usecase

System Boundary: It is a visual representation of the scope or limits of the system you are modelling. It defines what is inside the system and what is outside. Boundary helps to establish a clear distinction between the elements that are part of the system and those that are external to it.



System Boundary

Use Case Diagram for ATM



Actors in ATM:

Customer
Actor.

Use cases in ATM:

Insert Card
Enter PIN ---<<includes>>---> Valid.
Withdrawl ---<<extends>>---> Taking Receipt.
PIN change
Deposit.

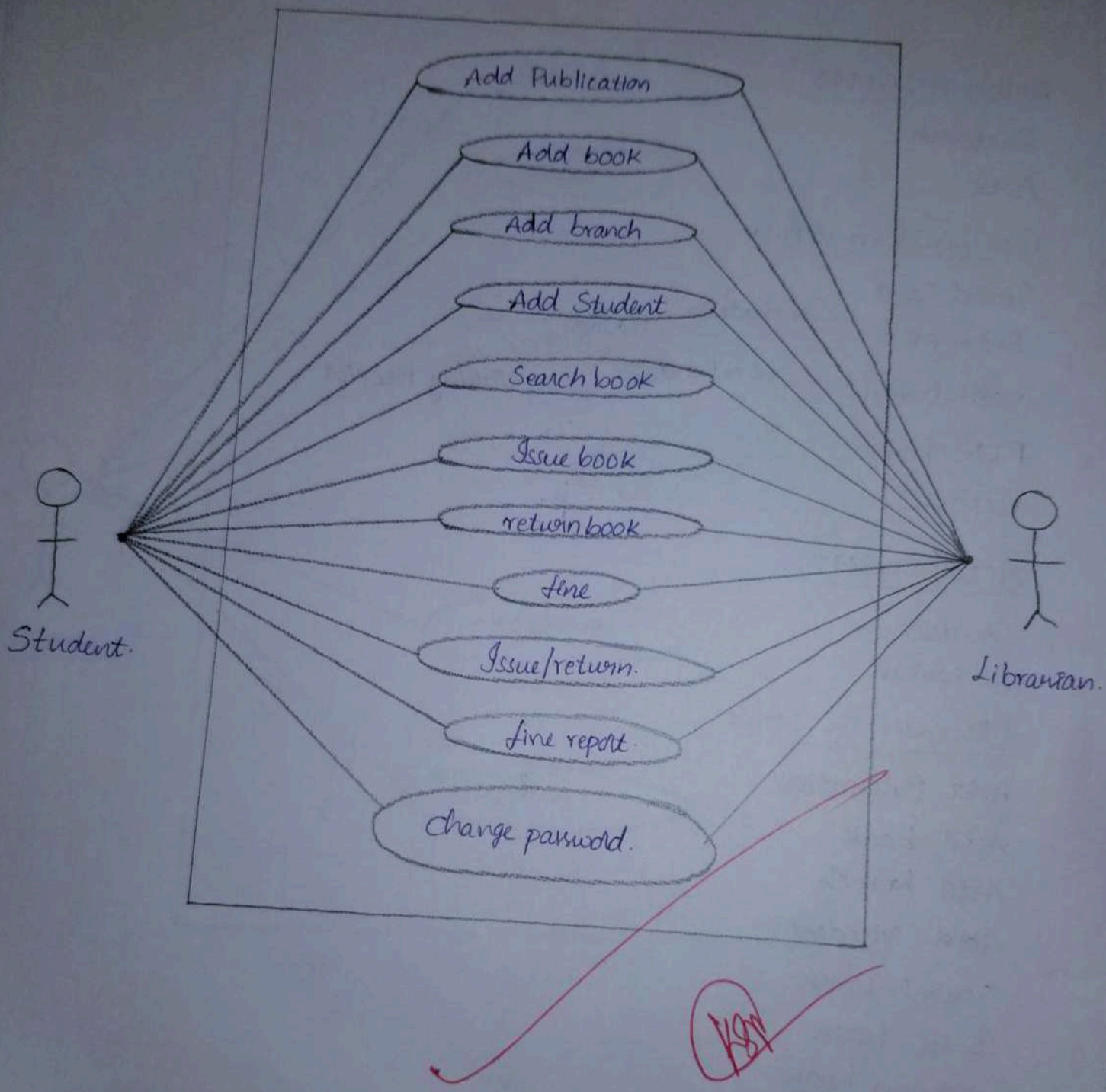
Actors in LMS:

Student
Librarian.

Usecases in LMS:

Add Publication
Add book
Add branch
Add student
Search book
Issue book
return book
fine
Issue/return.
fine report
change password.

Use Case Diagram for LMS

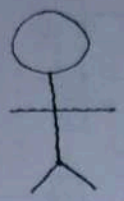


* Week-9:

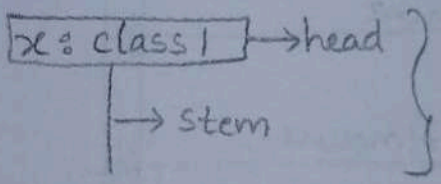
Aim: Develop detailed sequence diagrams / communication for each usecase showing interaction among all three-layer objects.

Description:

Actor: An actor is UML represents a role of a person who is interacting with the system of with the objects. It is always an external entity in the scope of Software application.

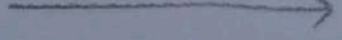


Lifelines: Lifeline is a named element which represents an individual participation in a sequence diagram. It basically represents the behaviour of each instance.



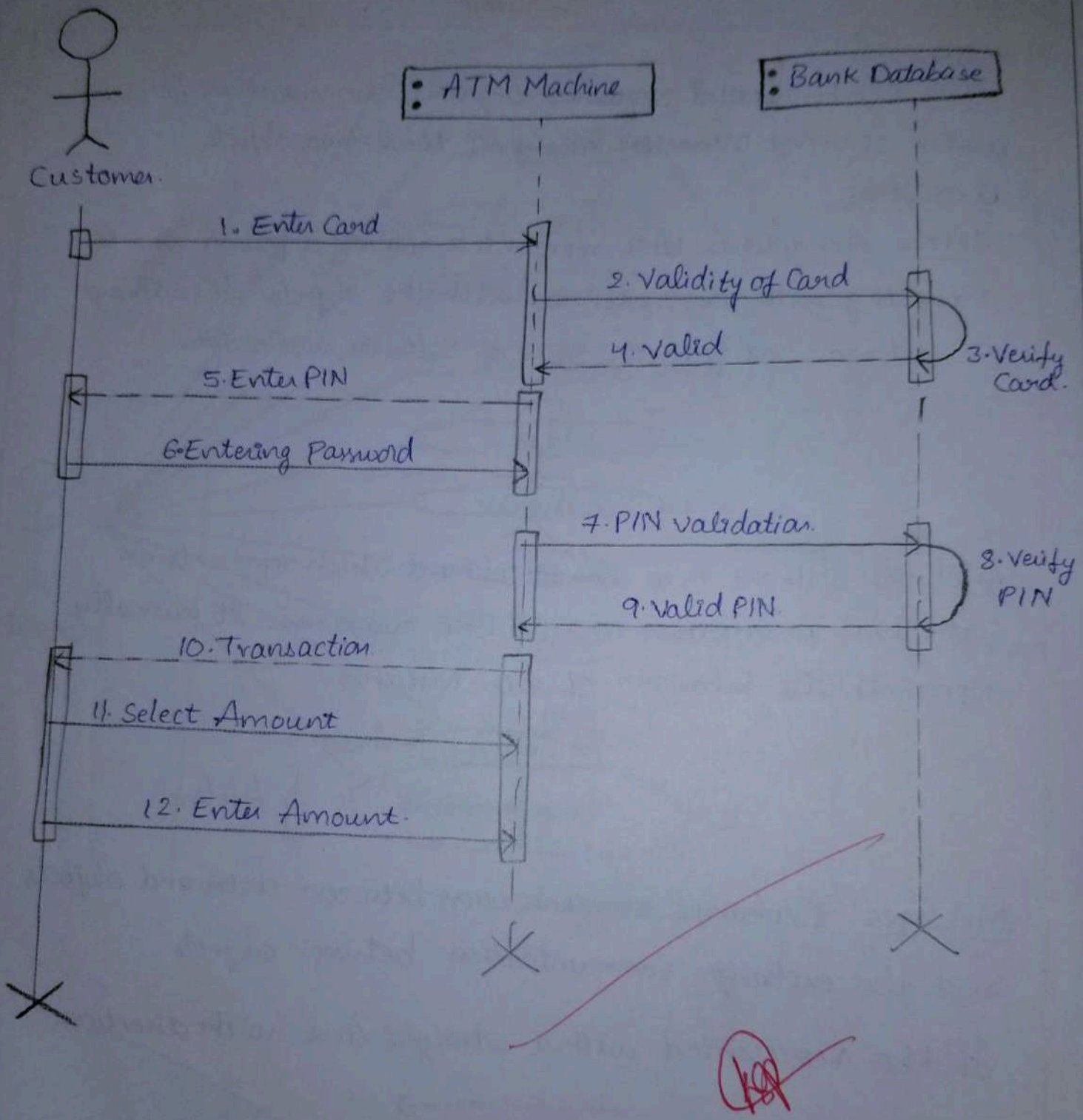
Message: Exchanges communication between actor and objects and also exchanges communication between objects.

It'll be represented with a straight line with direction.



Dead Object: In sequential diagram if the interaction is completed, then that object will be represented as an dead state, which we call as Dead Object. It'll be represented with "X".

Sequence Diagram for ATM.



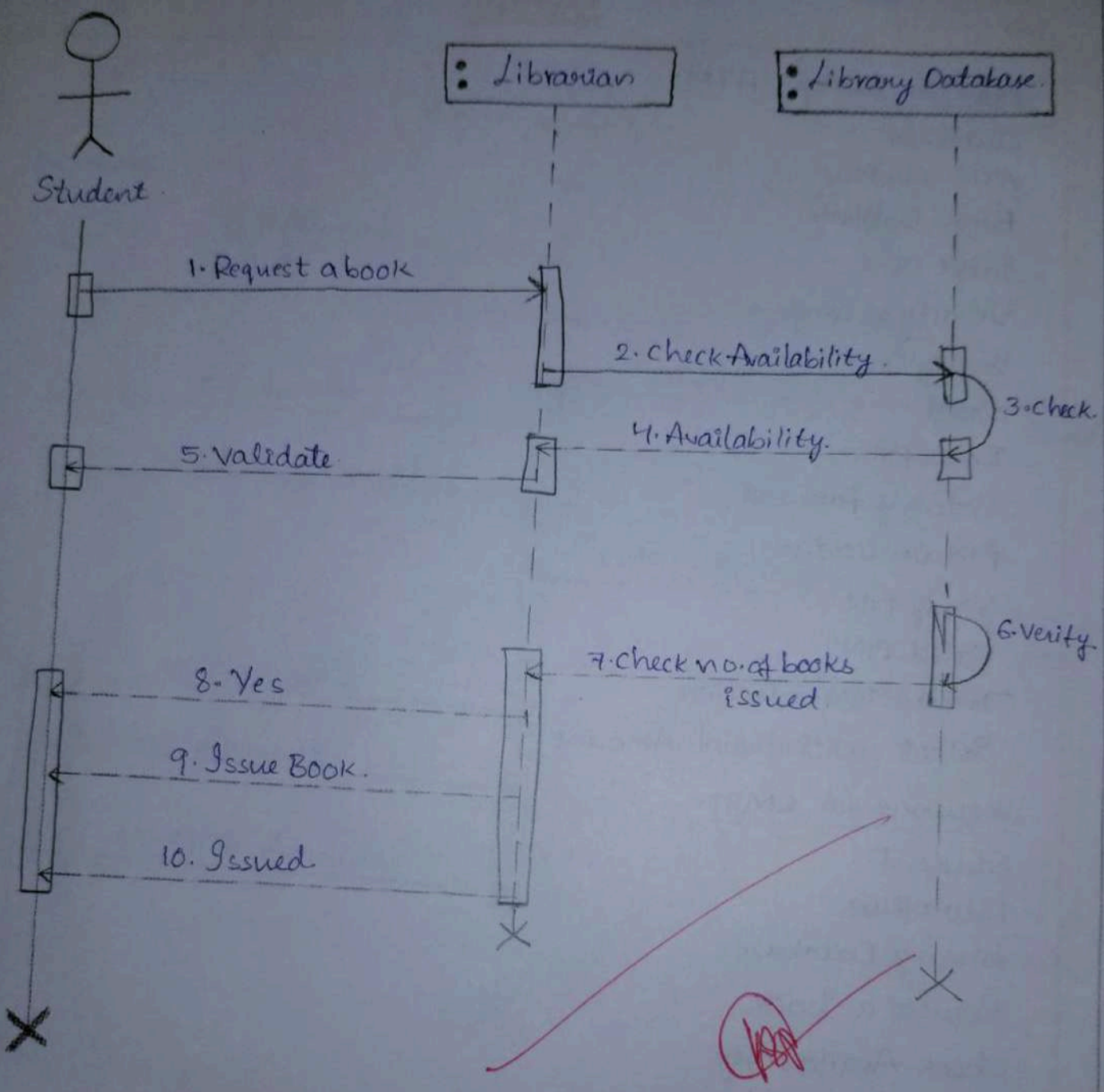
Sequences used for ATM:

Customer
ATM Machine.
Bank Database.
Insert Card
Validity of Card
Verify Card
Valid
Enter PIN
Entering Password
PIN verification
Verify PIN
Valid PIN
Transaction selection
Select Withdrawl Amount.

Sequences for LMS:

Student
Librarian
Library Database
Request a book.
Check Availability
Check Available
Validate
Verify
Check number of books issued
Yes
Issue book
Issued.

Sequence Diagram for LMS.



* Week-10:

Aim: Develop sample diagrams for State chart Diagrams.

Description: State chart Diagrams defines the State of a component and these state changes are dynamic in nature. It's specified purpose is to define the State changes triggered by events. Events are internal and external factors influencing the System.

Symbols:

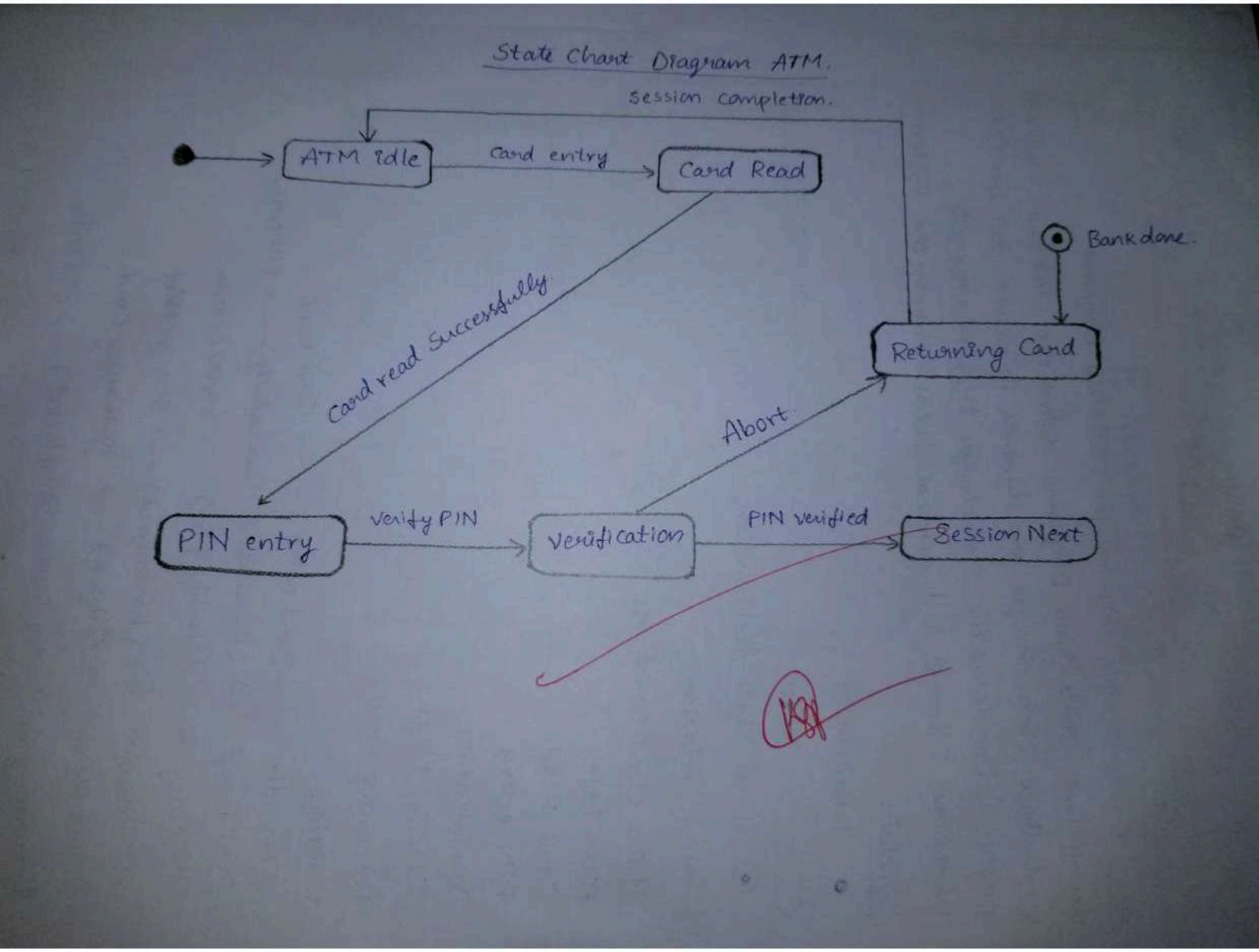
- Intialization.
- States
- final State / termination.
- ⟶ Relation

States Used in ATM:

ATM idle
Card Read
PIN entry
Verification
Session Next.
Returning Card.

Relations:

ATM idle ⟶ (Card entry) ⟶ Card Read
Card Read ⟶ (Card read Succesfully) ⟶ PIN entry.
PIN entry ⟶ (Verify pin) ⟶ Verification.
Verification ⟶ (Pin verified) ⟶ Session Next.
Verification ⟶ (Abort) ⟶ Returning Card
Returning Card ⟶ (Session Completed) ⟶ ATM idle.


State Chart Diagram ATM.
Session completion.
ATM Idle
Card entry
Card Read
Bank done.
Card read Successfully
Returning Card
Abort
PIN entry
Verify PIN
Verification
PIN verified
Session Next

States Used in LMS:

Student / Faculty Login
Search Book
Request Book
Receive Book
Return Book and pay fine (if any)
Profile update and sign out.

Relations Used:

Student/Faculty Login → (user id & password) → Search book.

Search book → (found book) → request book.

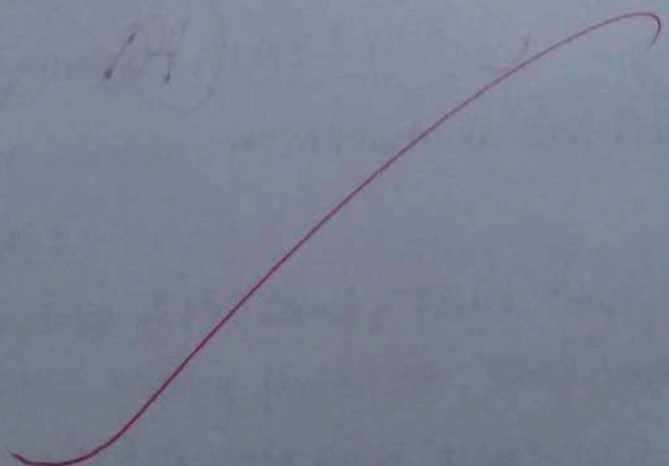
request book → (request librarian for book) → receive book.

receive book → (return back book) → return book & pay fine (if any)

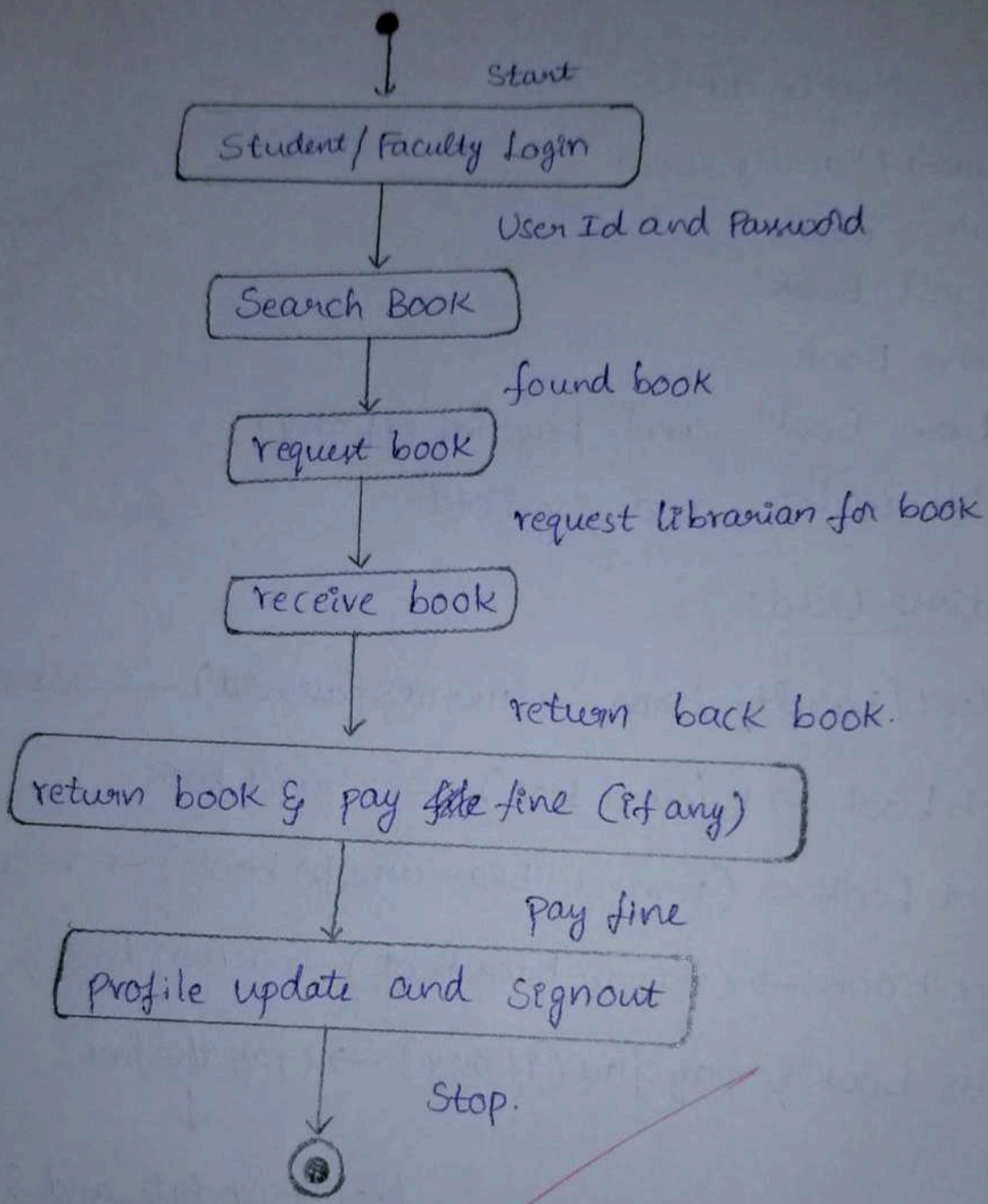
return book & pay fine (if any) → (pay the fine)

↓

Profile Update and SignOut.



State Chart Diagram for LMS.



* Week-11:

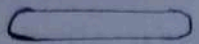
Aim: Develop detailed design using Activity Diagrams

Description:

Activity diagrams describe the workflow behaviour of a system. Activity diagrams are used in process modelling and analysis of during requirements engineering.

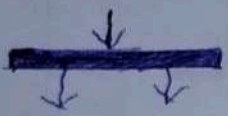
Notations of Activity Diagram:

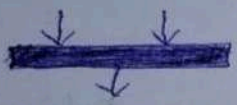
Start / Initiate 

Activity 

Control Flow 

Decision Box  Condition.

Fork 

Join 

Terminate / End 

A typical business process which synchronizes several external incoming events can be represented by Activity Diagram.

Activity Diagram for ATM:

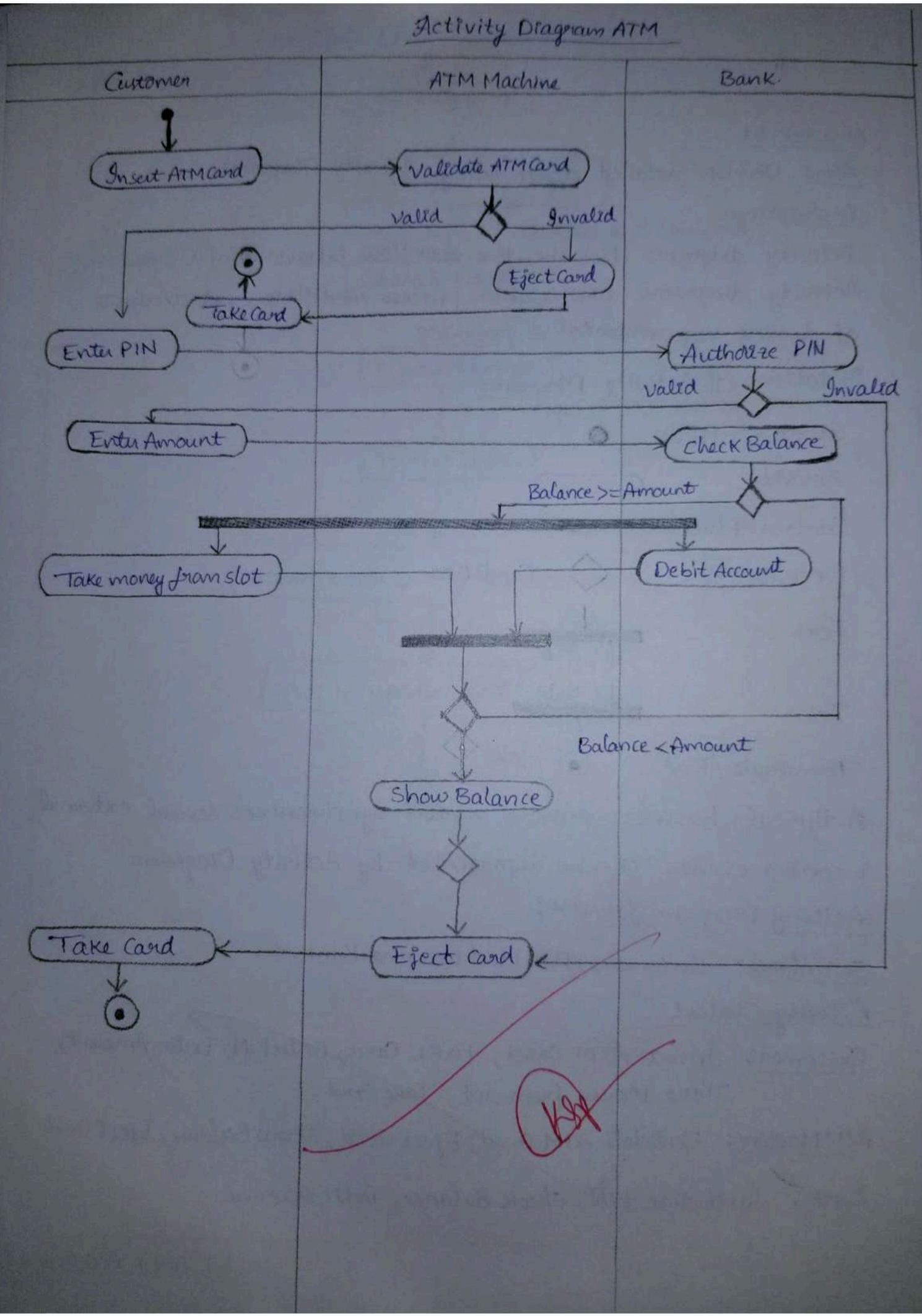
Swimlanes: Customer, ATM Machine and Bank.

Activity States:

Customer: Insert ATM Card, Take Card, Enter PIN, Enter Amount, Take Money from slot, Take Card.

ATM Machine: Validate ATM Card, Eject Card, Show Balance, Eject Card.

Bank: Authorize PIN, Check Balance, Debit Account

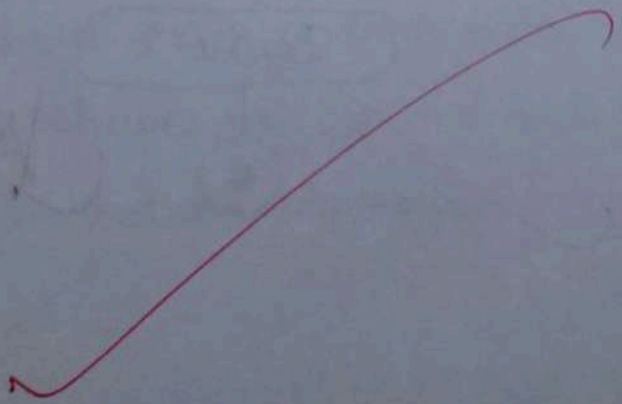

Activity Diagram ATM
Customer
ATM Machine
Bank
Insert ATM Card
Validate ATM Card
Valid
Invalid
Eject Card
Take Card
Enter PIN
Authorize PIN
Valid
Invalid
Enter Amount
Check Balance
Balance >= Amount
Take money from slot
Debit Account
Balance < Amount
Show Balance
Take Card
Eject card

Activity Diagram for LMS:

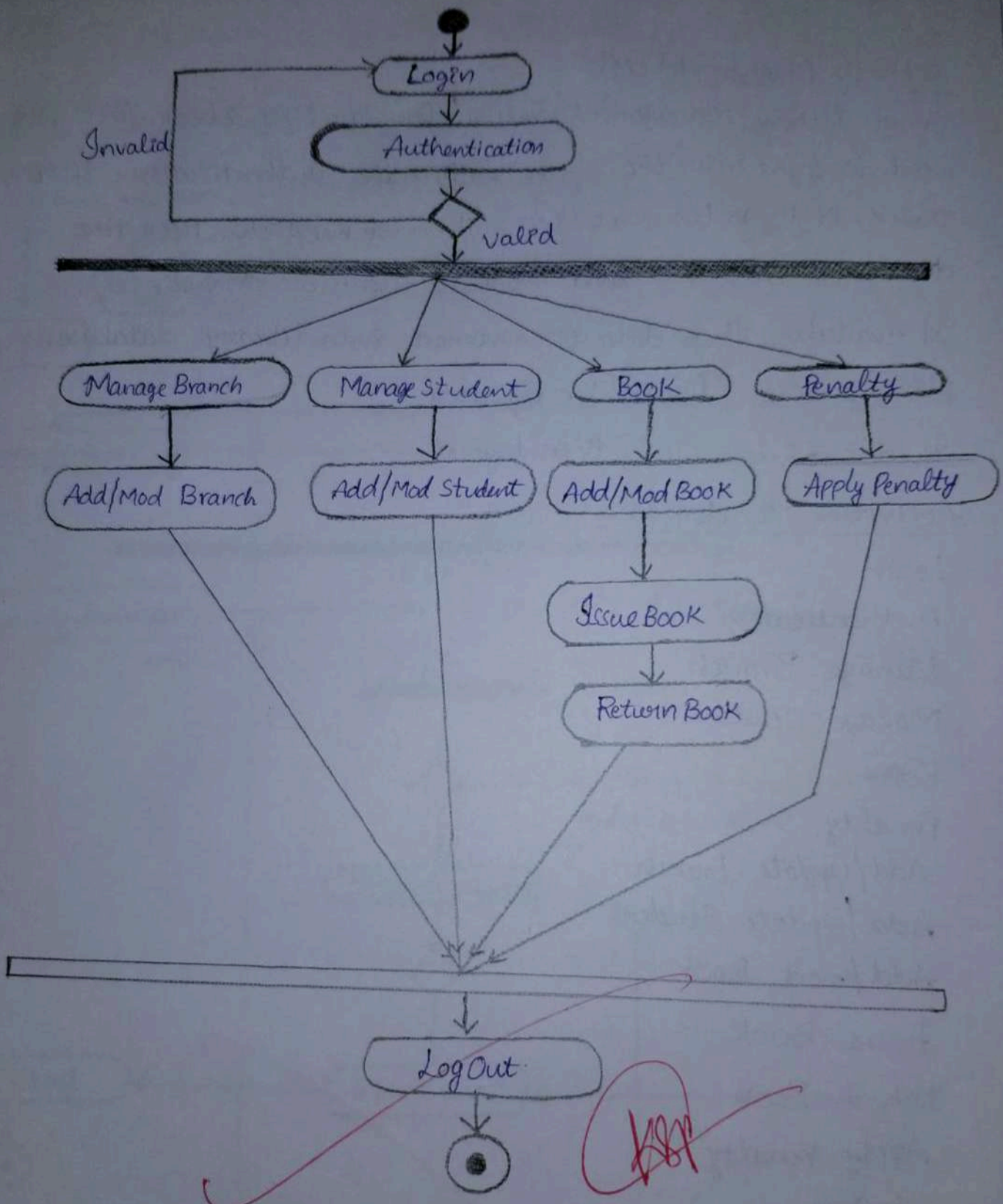
In an Library Management System, for searching a book first you need to logen into the system. check the authentication. If it is valed, it'll go to next step. Then we need to check the details of Student and book is available (or) not.
If available, then data is entered into library database
Else, process should be repeated.
If it is successful, then logout.

Activities Mentioned:

Login
Authentication
Manage Branch
Manage Student
Book
Penalty
Add/Update Branch
Add/Update Student
Add/mod Book
Issue Book
Return Book
Apply Penalty
Log Out.



Activity Diagram for LMS



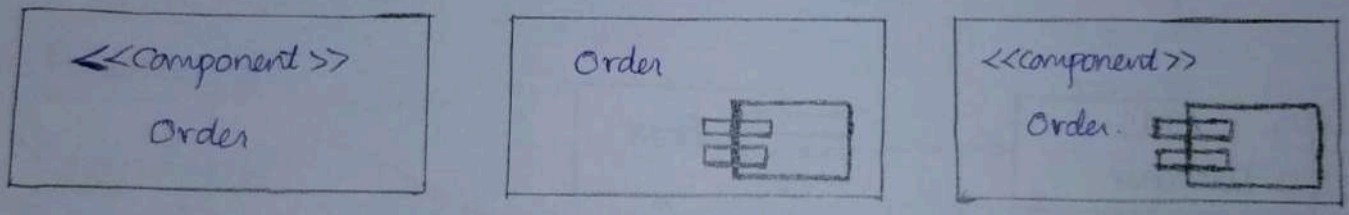
*Week-12:

Aim: Develop sample diagrams for other UML diagrams – Component diagram and deployment diagram

Description:

Component: A component represents the modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment. In UML, a component is drawn as a rectangle with optional component stacked vertically. A high level, abstracted view of a component in UML can be modelled as:

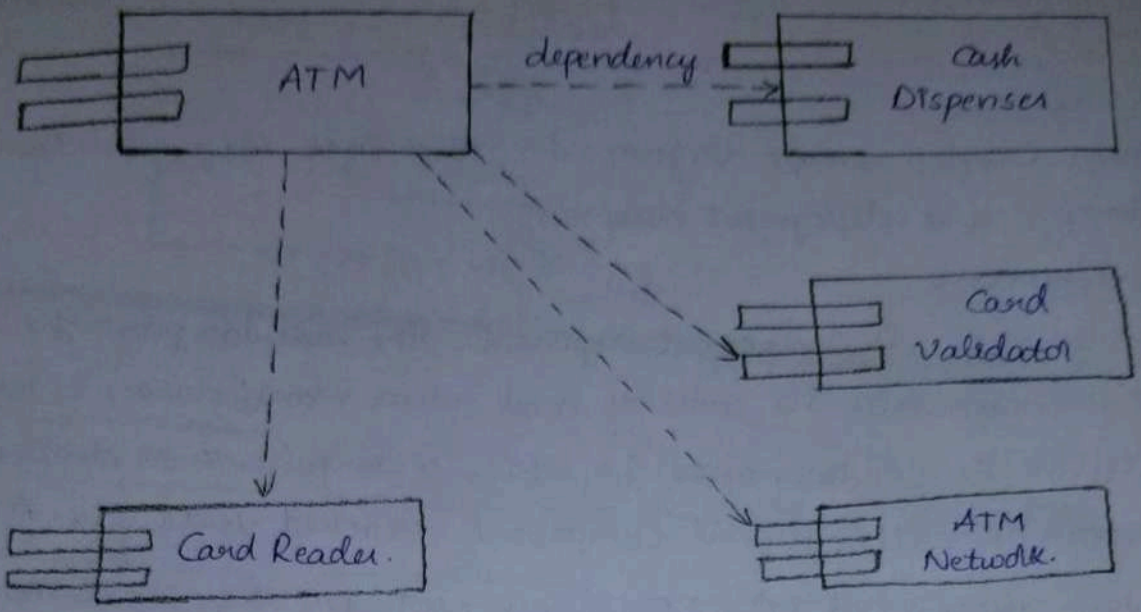
1) Rectangle with component's name.
2) Rectangle with component icon.
3) Rectangle with stereotype text and/or icon.



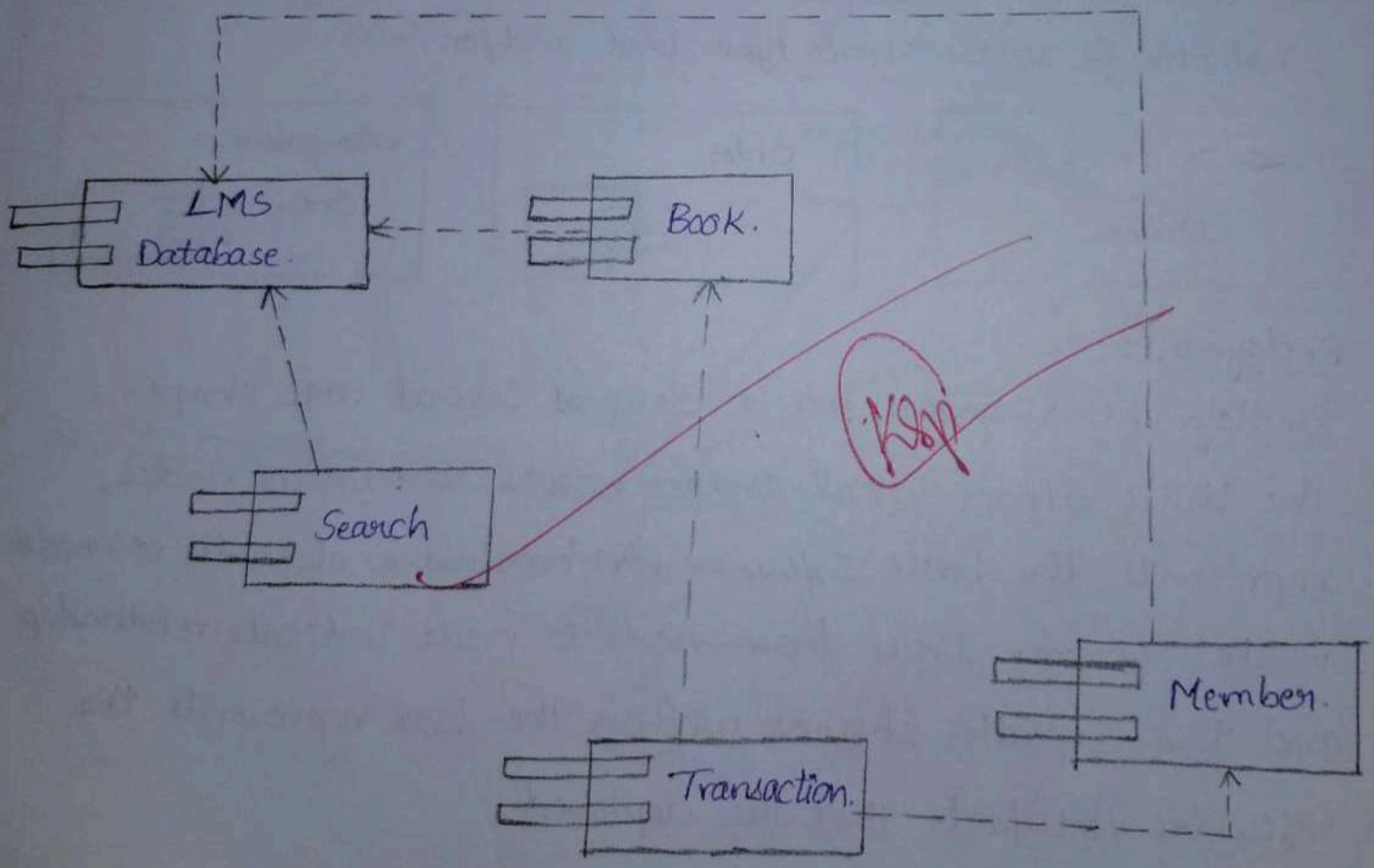
Deployment:

Deployment diagrams are made up of several UML shapes. The three dimensional shapes boxes known as nodes, represents the basic software (or) hardware elements (or) nodes in the system. Lines from node to node indicate relationship. and the smaller shapes within the box represents the Software Artifacts that are deployed.

Component Diagram for ATM



Component Diagram for LMS.



Components used in ATM:

ATM
Cash Dispenser
Card Validator
ATM Network
Card Reader

Components Used in LMS:

LMS Database
Book
Search
Transaction
Member.

Relationships:

Dependency:

Dependency depicts how various things within a system are dependent on each other. In UML, a dependency relationship is the kind of relationship in which a client (one element) is dependent on Supplier (another element).

It is used in class diagrams, Component diagrams, deployment diagrams and use-case diagrams.

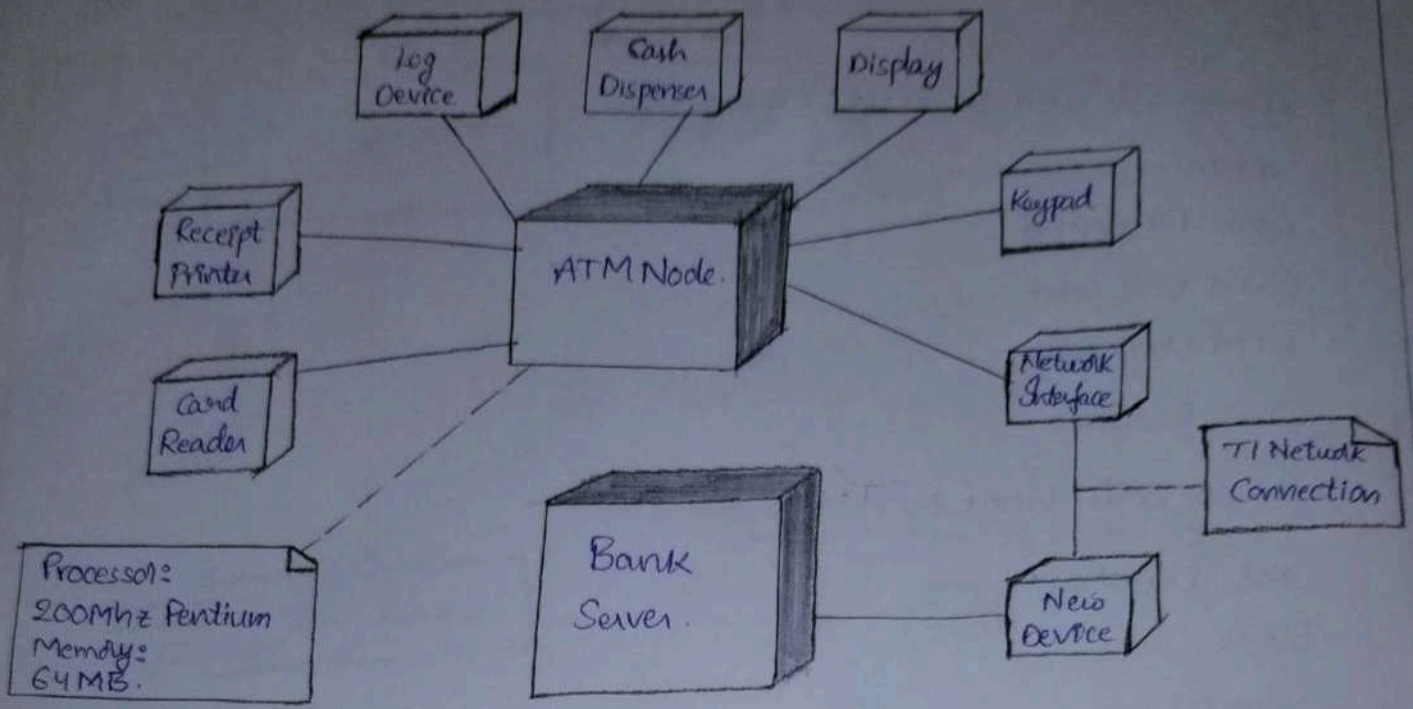
Deployment Nodes in ATM:

ATM Node
Bank Server
Card Reader
Receipt Printer
Log Device
Cash Dispenser
Display
Keypad
Network Interface
New Device.

Deployment Nodes in LMS:

Client Tier
Web Browser.
Web Tier
User Interface.
Application Tier
Business Logic.
Data Access.
Data Tier.
Stored Procedures.

Deployment Diagram for ATM



Deployment Diagram for LMS