HMR INSTITUTE OF TECHNOLOGY & MANAGEMENT
Hamidpur, Delhi-110036

(An iso 9001:2008 certified, AICTE approved & GGSIP university affiliated institute)

SOFTWARE ENGINEERING LAB

Submitted To:

Ms Renu Chaudhary

Assistant Professor

(IT Department)

Submitted By:

Nitin Tyagi

(41113303118)

(Btech(IT))

HMR INSTITUTE OF TECHNOLOGY AND MANAGEMENT

HAMIDPUR, DELHI 110036

Affiliated to

GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY Sector - 16C
Dwarka, Delhi - 110075, India 2018-2022

HMR INSTITUTE OF TECHNOLOGY & MANAGEMENT
Hamidpur, Delhi-110036

(An iso 9001:2008 certified, AICTE approved & GGSIP university affiliated institute)

INDEX

EXPERIMENT – 1

AIM:-

To write the problem statement of ATM MANAGEMENT SYSTEM.

REQUIREMENTS:-

1. SOFTWARE REQUIREMENT – Microsoft Word
2. HARDWARE REQUIREMENT – Computer, Keyboard, Mouse, CPU

THEORY:-

The software proposes to manage and enable smooth functioning of an ATM which would include the standard ATM transaction type such as withdrawals, balance, enquiry, pin change, mini statement, cash deposit and funds transfer. It may also include some value added services like bill payment or mobile top-ups etc.

The ATM system will provide the following services to the user:
• Better ATM transaction operations.
• Minimum balance maintenance for account.
• Routine checkup of ATM.
• If under theft circumstances, we can block the ATM card from further transactions by reversing the pin.
• Frequent fraudulent transactions and fund transfers to particular account should be reported to bank manager.
• Reward points also given for value added services as per the bank criteria.
• Easier interactive system for usage.
• Ability to send request for cheque book.

CONCLUSION:-

Problem statement of ATM MANAGEMENT SYSTEM has been written successfully.

EXPERIMENT – 2

AIM:-

To do the requirement analysis and develop Software Requirement Specification Sheet for ATM MANAGEMENT SYSTEM.

REQUIREMENTS:-

1. SOFTWARE REQUIREMENT – Microsoft Word
2. HARDWARE REQUIREMENT – Computer, Keyboard, Mouse, CPU

THEORY:-

1. Introduction

The ATM management software is to be developed for Automated Teller Machines (ATM). An automated teller machine (ATM) is computerized telecommunications device that provides a financial institution's customers a secure method of performing financial transactions, in a public space without the need for a human bank teller. Through ATM, customers interact with a user-friendly interface that enables them to access their bank accounts and perform various transactions.

1.1 Purpose

The ATM management system (AMS) maintains the records of account holder transactions by interacting with the bank servers to keep them up to date. The ATM management system provides the user to perform bank related operations without going in the bank.

1.2 Scope

The proposed AMS must be able to perform the following functions:
1. Provide login facility to account holder in bank using ATM card and pin.
2. Provide cash dispensing facility to account holder on successful transaction.
3. Block card in case of continuous three wrong pin attempts.
4. Block card in case of reverse pin input for enhanced safety.
5. Provide facility to customers to check their account balance.

6. Provide facility to customers to change their own ATM card pin
7. Provide facility to customers to generate a mini account statement of last 10 transactions.
8. Provide facility to customers to transfer funds from their account to the beneficiary account.
9. Provide facility to customers to request cheque book from bank.
10. Provide following additional Value Added Services to the customers:
    - Check Reward Points
    - Recharge prepaid mobile number
    - Pay bill such as electricity, water etc.
11. Provide login facility to the maintainer.
12. Provide following services to the maintainer:
    - Start the ATM services
    - Stop the ATM services
    - Update the ATM software

1.3 Definitions, Acronyms, and Abbreviations

AMS: ATM Management System
ATM: Automated Teller Machine - An unattended electronic machine in a public place, connected to a data system and related equipment and activated by a bank customer to obtain cash withdrawals and other banking services.
RAM: Random Access Memory
ATM Card: Card encoded with details of customer's account details.
Customer/ Account Holder: Any person holding an account in a bank.
Bank: Financial Institution that provides various types of currency related facilities such as storage of currency, dispensing currency etc.
Maintainer: Any person that is designated to resolve the problems/issues related to smooth functioning of ATM.
Bank Server: A machine that holds details about the accounts associated with the bank.
Card Pin: A 4-digit combination that is related to ATM card needed to be entered in ATM to access ATM services.

1.4 References

- Object-Oriented Software Engineering by Yogesh Singh & Ruchika Malhotra, PHI Learning Pvt. Ltd., 2012
- IEEE Recommended Practice for Software Requirements Specifications – IEEE Std. 830-1998.
- IEEE Standard for Software Test Documentation – IEEE Std. 829-1998.

1.5 Overview

The rest of the SRS document describes various system requirements, interfaces, features and functionalities.

2. Overall Description

AMS facilitates the customer to perform various transactions in his/her account without going to bank. This software offers benefits such cash withdrawals, balance transfers, deposits, inquiries and other banking related operations for customers. It also allows the maintainer to fix the issues of ATM and update its software.
The software takes card number and pin as input and the bank account number of the customer is retrieved for further purposes. The outputs then comprise of an interactive display that lets the customer select the desirable function that he/she wants to perform.

2.1 Product Perspective

- The AMS shall be developed using client-server architecture and be compatible with Linux and UNIX based Operation System. The front-end of the system will be developed using PHP, HTML, CSS and the back-end will be developed using PostgreSQL 12.
- The ATM is a single functional unit consisting of various sub- components.
- AMS allows the user to access their bank accounts remotely through an ATM without any aid of human bank teller.
- AMS also allows to perform various other functions apart from just accessing his bank account such as mobile bill clearings etc.
- Some of its hardware components are cassettes, memory, drives, dispensers i.e. for receipts and cash, a card reader, printer, switches, a console, a telephone dialer port, a networking port and disks.
- The ATM communicates with the bank's central server through a dial-up communication link.

2.1.1 System Interfaces

None

2.1.2 User Interfaces

The AMS will have the following user-friendly and menu-driven interfaces:
a. **Login**: to allow the entry of only authorized users through valid card and pin.
b. **Cash Withdrawal**: to allow account holder to get cash from ATM.
c. **Fund Transfer**: to allow account holder to transfer fund from his/her account to another account holder's account.
d. **Check Balance**: to allow account holder to see the details of his/her current account balance.
e. **Change Pin**: to allow account holder to change his/her card current pin.
f. **Request Cheque Book**: to allow account holder to send request for a cheque book to the bank.
g. **View Mini Statement**: to see summary of past transactions.
h. **VAD Services**: to provide additional facilities to account holder that are not provided by traditional financial institutes in their premises.
i. **Start ATM Services**: to start services at any ATM where the services are currently stopped.
j. **Stop ATM Services**: to stop ongoing services at any ATM.
k. **Update ATM Software**: to update the current software system of ATM with new software version.

2.1.3 Hardware Interfaces

a. Screen Resolution of at least 800 x 600 or above.
b. Support for touch screen.
c. Support for card reader.
d. ATM systems will be in the networked environment as it is a multi-user system.

2.1.4 Software Interfaces

a. Linux/ Unix Operating System
b. PHP for designing front-end
c. PostgreSQL 12 for back-end

2.1.5 Communication Interfaces

Communication is via local area network (LAN).

2.1.6 Memory Constraints

At least 256 MB RAM and 100 MB space hard disk will be required to run software.

2.1.7 Operations

None

2.1.8 Site Adaptation Requirements

The terminal at ATM will have support for the hardware and software interfaces specified in sections 2.1.3 and 2.1.4 respectively.

2.2 Product Functions

The major functions that AMS performs are described as follows:

- **Account Maintenance**: The various functions that a user can perform with his account are as follows:
    a) Withdrawal/Deposit: The software allows the user to select the kind of operation to be performed i.e. whether he/she wants to withdraw or deposit the money.
    b) Fund Transfer: Fund transfer shall be facilitated between two accounts linked to the bank.
    c) Balance Enquiry: Balance enquiry for any account linked to the card shall be facilitated.
- **Billing**: Any transaction shall be recorded in the form of a receipt and the same would be dispensed to the customer. The billing procedures are handled by the billing module that enable user to choose whether he/she wants the printed statement of the transaction or just the updating in his/her account.
- **Cancelling**: The customer shall abort a transaction with the press of a Cancel key. For example on entering a wrong depositing amount. In addition the user can also cancel the entire session by pressing the abort key and can start a fresh session all over again.
- **Mobile Recharge**: The machine also allows the user to recharge his/her mobile number there only, if the name of his/her operator is mentioned there in the list. The machine displays the list of the companies supported by that bank to the user.
- **Bill Payments**: The machine also allows the user to clear off his pending bills there only, if the name of his/her operator is mentioned there in the list. The machine displays the list of the companies supported by that bank to the user.

2.3 User Characteristics

- **Qualification**: At least matriculation and comfortable with English.
- **Experience**: Should be well versed/informed about inserting the card in the card reader.
- Technical **Knowledge**: Elementary knowledge of keypad and touch screen.

2.4 Constraints

- The software does not create bank account.
- User will not be allowed to update their account number.
- ATM must service at most one person at a time.
- The number of invalid pin entries attempted must not exceed three. After three unsuccessful login attempts, the card is seized/blocked and need to be unlocked by the bank
- The simultaneous access to an account through both, the ATM and the bank is not supported.
- The minimum amount of money a user can withdraw is ₹ 100/- and the maximum amount of money a user can withdraw in a session is ₹ 10,000/- and the maximum amount he/she can withdraw in a day is ₹ 25,000/-
- Before the transaction is carried out, a check is performed by the machine to ensure that a minimum amount of ₹ 2000/- is left in the user's account after the withdrawal failing which the withdrawal is denied.
- A user can select only that cellular operator for mobile bill clearings that is supported by the bank.
- There shall be a printer installed with the machine to provide the user with the printed statement of the transaction.

2.5 Assumptions and Dependencies

- One major dependency that the project might face is the changes that need to be incorporated with the changes in the bank policies regarding different services. As the policies changes the system needs to be updated with the same immediately. A delay in doing the same will result to tremendous loss to the bank. So this should be changed as and when required by the developer.
- Another constraint relating to the operating environment is that we are specific to PostgreSQL database.
- The project could be largely affected if some amount is withdrawn from the user's account from the bank at the same time when someone is accessing that account through the ATM machine. Such a condition shall be taken care of.
- At this stage no quantitative measures are imposed on the software in terms of speed and memory although it is implied that all functions will be optimized with respect to speed and memory.
- The account and card are added by the bank to the bank server.

3. Specific Requirements

This section contains the software requirements in detail along with various forms to be developed.

3.1 External Interface Requirements

3.1.1 User Interfaces

The following user interfaces (or forms) will be provided by the system.

i. Login Screen

This will be the first form, which will be displayed. It will allow the user to access the different forms based on his/her account.

Various fields available on this form will be:

- *Card No*. : 12 digit numeric values that is displayed by card reader after reading the card. Special characters and spaces are not allowed.
- *Pin*: A 4 digit numeric value associated with the card inserted in the card reader. Special characters and spaces are not allowed.

ii. Customer Home

The system provides the customer with variety of options to choose for further operations on ATM.

iii. Cash Withdrawal

This form facilitates the customer to enter amount to money that he/she needs to get from ATM system.

The field available on this form will be:

- *Cash Amount*: A numeric value (multiple of 100) ranging from minimum of ₹ 100 to maximum of ₹ 10000. Special characters and spaces are not allowed.

iv. Fund Transfer

This form facilitates the customer to transfer fund from his/her account to another account holder's account.

Various fields available on this form will be:

- *Account Number*: A 12 digit number representing the account number of beneficiary. Special characters and spaces are not allowed.
- *Amount*: A numeric value ranging with minimum of ₹ 100. Special characters and spaces are not allowed.

v. Check Balance

The system facilitates the customer to check his/her account balance.

vi. Change Pin

This form facilitates the customer to change his/her ATM card pin.

Various fields available on this form will be:

- *Old Pin:* A 4 digit numeric value associated with the card inserted in the card reader. Special characters and spaces are not allowed.
- *New Pin:* A 4 digit numeric value that customer wants to use as pin with the card inserted in the card reader. Special characters and spaces are not allowed.

vii. Request Cheque Book

This form facilitates the customer to send request for a new cheque book to the bank.

The field available on this form will be:

- *Account Number*: A 12 digit number representing the account number of customer. Special characters and spaces are not allowed.

viii. View Mini Statement

The system facilitates the customer to see summary of past transactions.

ix. Value Added Services

The system provides the customer with various additional services apart from basic banking services.

a. *Check Reward Points*

The system facilitates the customer to check his/her reward points earned using ATM card services.

b. *Mobile Recharge*

This form facilitates the customer to recharge his/her mobile number using his/her account balance.

Various fields available on this form are:

- *Choose Operator:* A limited set of mobile service operators

supported by ATM provided to customer to choose his/her mobile number operator.

- *Mobile Number:* A 10 digit numeric value representing the mobile number that is needed to be recharged. Special characters and spaces are not allowed.
- *Recharge Amount:* A numeric value ranging from ₹ 10 to ₹ 2000 and representing the amount to add to mobile number account. Special characters and spaces are not allowed.

c. *Bill Payment*

This form facilitates the customer to pay outstanding amount of his/her bills such as electricity, water and PNG using his/her account balance.

Various fields available on this form are:

- Choose Bill Type: A limited set of type of bill services supported by ATM provided to customer to choose his/her bill service type.

- CA Number: A 10 digit alphanumeric value representing the customer bill service account number for which bill payment is needed. Special characters and spaces are not allowed.
- Bill Amount: A numeric value starting from ₹ 10 and representing the amount of bill. Special characters and spaces are not allowed.

x. Maintainer Home:

The system provides the maintainer with variety of options to choose for further operations on ATM network.

xi. Start ATM Services

The system facilitates the customer to start services at any ATM where the services are currently stopped.

The field available on this form is:

- ATM ID: A 10 digit alphanumeric values to uniquely identify an ATM. Special characters are allowed but spaces are not allowed.

xii. Update ATM Software

The system facilitates the customer to update the current software system of ATM with new software version.

The field available on this form is:

- ATM ID: A 10 digit alphanumeric values to uniquely identify an ATM. Special characters are allowed but spaces are not allowed.

xiii. Stop ATM Services

The system facilitates the customer to stop ongoing services at any ATM.

The field available on this form is:

- ATM ID: A 10 digit alphanumeric values to uniquely identify an ATM. Special characters are allowed but spaces are not allowed.

3.1.2 Hardware Interfaces

There are various hardware components with which the machine is required to interact. Various hardware interface requirements that need to be fulfilled for successful functioning of the software are as follows:

- The ATM power supply shall have a 10/220 V AC manual switch.
- The ATM card should have the following physical dimensions:

a) Width – 85.47mm-85.72mm
b) Height – 53.92mm-54.03mm
c) Thickness – 0.76mm+0.08mm

- The card reader shall be a magnetic stripe reader
- The card reader shall have Smart card option.

- The slot for a card in the card reader may include an extra indentation for the embossed area of the card. In effect it acts as a polarization key and may be used to aid the correct insertion orientation of the card. This is an additional characteristic to the magnetic field sensor which operates off the magnetic stripe and is used to open a mechanical gate on devices such as ATMs.
- There shall be a 40 column dot matrix receipt printer.
- There shall be a 40 column dot matrix statement printer.
- The receipt dispenser shall be a maximum of 4" width and 0.5" thickness.
- The statement dispenser shall be a maximum of 5" width and 0.5" thickness.
- The envelope depository shall be a maximum of 4.5" width, 10" length and 0.5" thickness.
- Screen resolution of at least 800 x 600-required for proper and complete viewing of screens. Higher resolution would not be a problem.

3.1.3 Software Interfaces

In order to perform various different functions, this software needs to interact with various other software. So there are certain software interface requirements that need to be fulfilled which are listed as follows:

- The transaction management software used to manage the transaction and keep track of resources shall be BMS version 2.0.
- The card management software used to verify pin no and login shall be CMS version 3.0.
- The database used to keep record of user accounts shall be PostgreSQL 12.0.

3.1.4 Communication Interface Requirements

None

3.2 Other Non Functional Requirements

3.2.1 Performance Requirements

- The ATM shall provide customers a 24 hour service.
- The card verification time must not exceed 0.8 sec. under normal server workload and 1 sec. under peak server workload.
- The pin number verification time must not exceed 0.3 sec. under normal server workload and 0.5 sec. under peak server workload.

- Account balance display time must not exceed 2 sec. under normal server workload and 3 sec. under peak server workload.
- Account balance transfer time must not exceed 3 sec. under normal server workload and 4 sec. under peak server workload.
- Cash withdrawal transaction time must not exceed 4 sec. under normal server workload and 5 sec. under peak server workload.
- Receipt printing time after must not exceed 3 sec. under normal server and peak server workload.
- Touch screen and button response time must not exceed 5000ms.

3.2.2 Design Constraints

None

3.2.3 Software System Attributes

Usability

- The application will be user-friendly and easy to operate and the functions will be easily understandable.

Reliability

- The application will have non-volatile memory system and have a high degree of fault tolerance.

Availability

- The system will have a backup power supply in case of power failures.
- Any abnormal operations shall result in shutting down of the system. After abnormal shutdown of the ATM, the system shall have to be manually restarted by a maintenance personnel.
- There should be no inconsistency introduced in the account in case of abnormal system shutdown.

Security

- The application shall be password protected. Users will have to enter correct pin for their card to access the application.
- User should be provided with only three attempts to enter pin after which card will be blocked for further use.
- There shall be a security camera installed near the ATM.
- There shall be a secured cash vault with a combination locking system.
- The ATM cabinet cover shall be manufactured using Fibre glass for security purposes.

Maintainability

- The application will be designed in a maintainable manner. It will be easy to incorporate new requirements in the individual modules.
- The system should have the mechanism of self-monitoring periodically in order to detect any fault.
- The system should inform the main branch automatically as soon as it detects any error. The kind of fault and the problem being encountered should also be mentioned by the system automatically.

CONCLUSION:-

Requirement analysis of ATM MANAGEMENT SYSTEM has been done and its Software Requirement Specification Sheet has been written successfully.

EXPERIMENT – 3

AIM:-

To perform the function oriented diagram: Data Flow Diagram(DFD) and Structured chart.

REQUIREMENTS:-

1. SOFTWARE REQUIREMENT – Microsoft Word
2. HARDWARE REQUIREMENT – Computer, Keyboard, Mouse, CPU

THEORY:-

1. **Data Flow Diagram(DFD)**

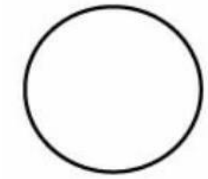
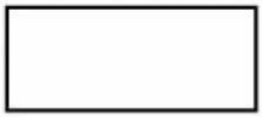
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both. It shows how data enters and leaves the system, what changes the information, and where data is stored.

The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart.

1.1 The following observations about DFDs are essential:

- All names should be unique. This makes it easier to refer to elements in the DFD.
- Remember that DFD is not a flow chart. Arrows is a flow chart that represents the order of events; arrows in DFD represents flowing data. A DFD does not involve any order of events.
- Suppress logical decisions. If we ever have the urge to draw a diamond-shaped box in a DFD, suppress that urge! A diamond-shaped box is used in flow charts to represents decision points with multiple exists paths of which the only one is taken. This implies an ordering of events, which makes no sense in a DFD.
- Do not become bogged down with details. Defer error conditions and error handling until the end of the analysis.

1.2 Standard symbols for DFDs are derived from the electric circuit diagram analysis and are shown in fig:

Symbols	Name	Function
→	**Data flow**	Used to connect processes to each other, to sources or sinks,to arrow head indicates direction of data flow.
	**Process**	Performs some transformation of input data to yield output data.
	**Source or Sink (external entity)**	A source of system inputs or sink of system outputs.
	**Data Store**	A repository of data , the arrow heads indicate the net input and net output to store.

- **Circle:** A circle (bubble) shows a process that transforms data inputs into data outputs.
- **Data Flow:** A curved line shows the flow of data into or out of a process or data store.
- **Data Store:** A set of parallel lines shows a place for the collection of data items. A data store indicates that the data is stored which can be used at a later stage or by the other processes in a different order. The data store can have an element or group of elements.
- **Source or Sink:** Source or Sink is an external entity and acts as a source of system inputs or sink of system outputs.

1.3 Levels in Data Flow Diagrams (DFD)

The DFD may be used to perform a system or software at any level of abstraction. Infact, DFDs may be partitioned into levels that represent increasing information flow and functional detail. Levels in DFD are numbered 0, 1, 2 or beyond. Here, we will see primarily three levels in the data flow diagram, which are: 0-level DFD, 1-level DFD, and 2-level DFD.

1.3.1 0-level DFD

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. Then the system is decomposed and described as a DFD with multiple bubbles. Parts of the system represented by each of these bubbles are then decomposed and documented as more and more detailed DFDs. This process may be repeated at as many levels as necessary until the program at hand is well understood. It is essential to preserve the number of inputs and outputs between levels, this concept is called leveling by DeMacro. Thus, if bubble "A" has two inputs x1 and x2 and one output y, then the expanded DFD, that represents "A" should have exactly two external inputs and one external output as shown in fig:

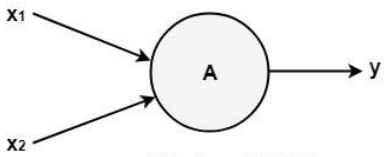


Fig: Level-0 DFD.

The Level-0 DFD, also called context diagram of the result management system is shown in fig. As the bubbles are decomposed into less and less abstract bubbles, the corresponding data flow may also be needed to be decomposed.

1.3.2. 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 as shown in fig:

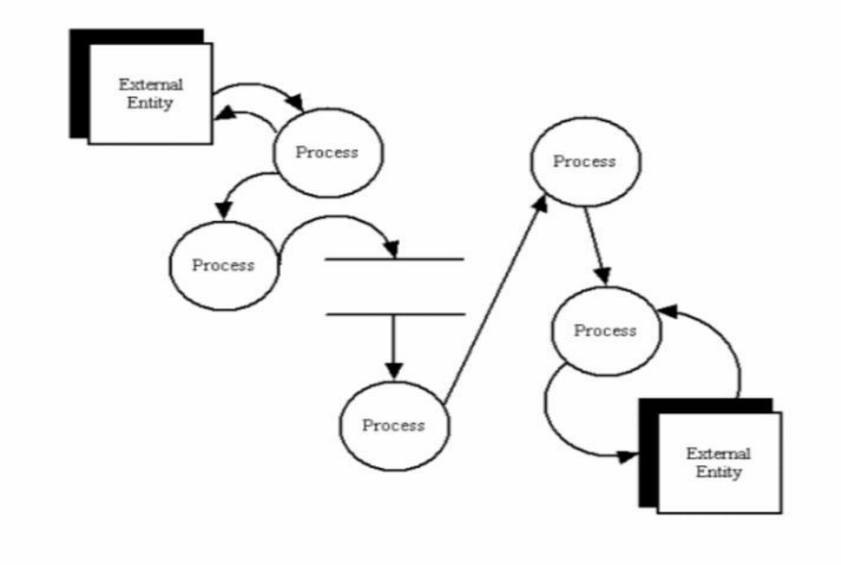
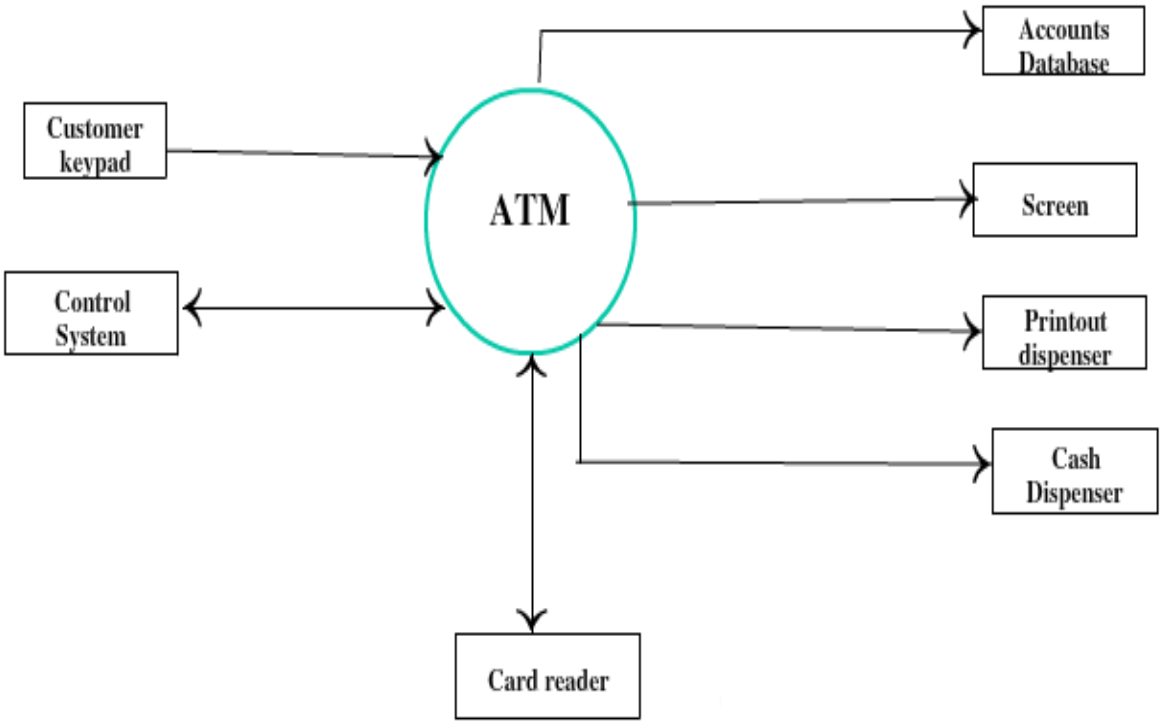


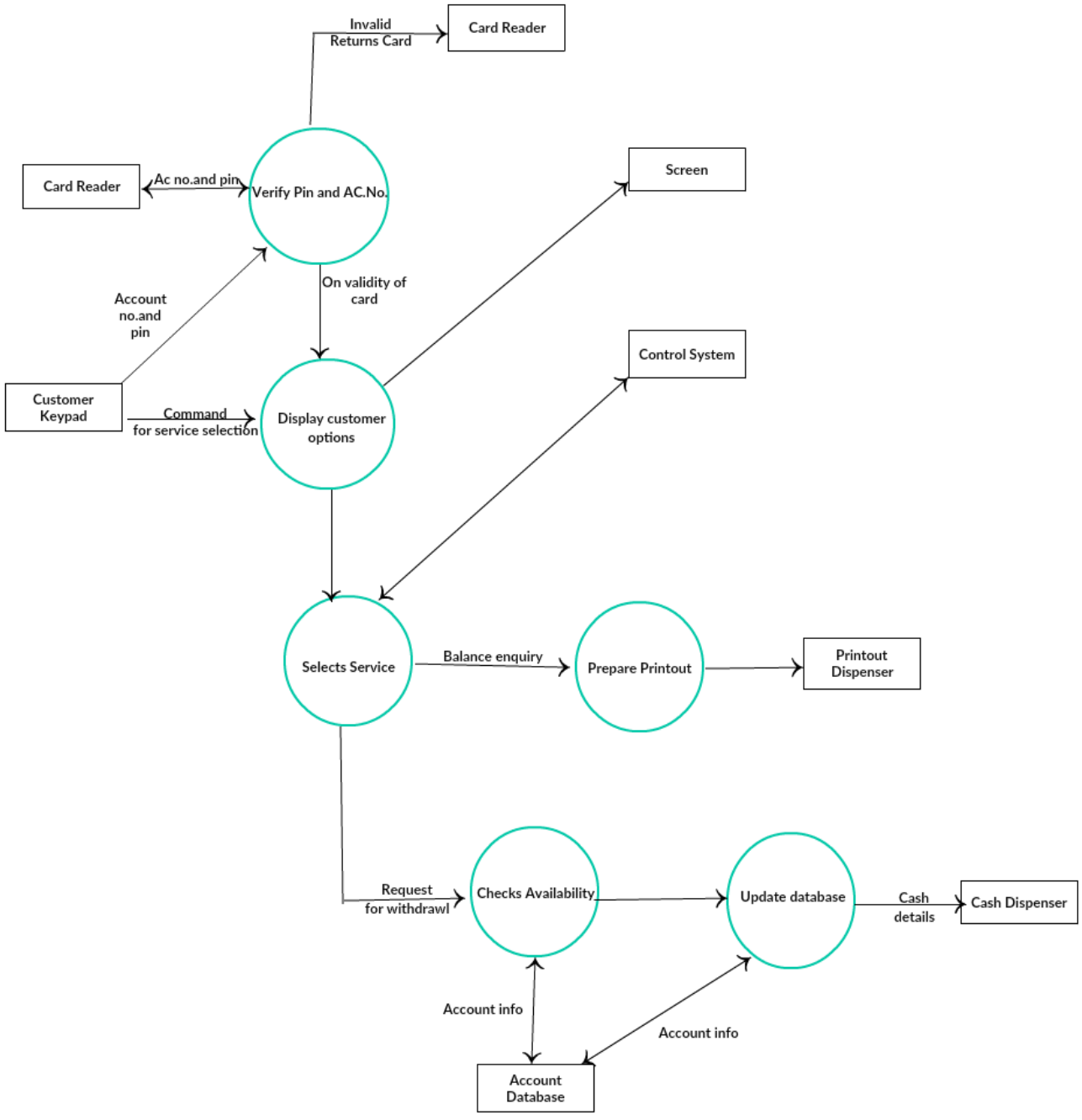
Fig: Level-1 DFD

1.4 DFD OF ATM management system

1.4.1 Level-0 DFD of ATM management system



1.4.2 Level-1 DFD of ATM management system



2. Structured Chart

Structure Chart represents hierarchical structure of modules. It breaks down the entire system into lowest functional modules, describing functions and sub-functions of each module of a

system to a greater detail. Structure Chart partitions the system into black boxes (functionality of the system is known to the users but inner details are unknown). Inputs are given to the black boxes and appropriate outputs are generated.

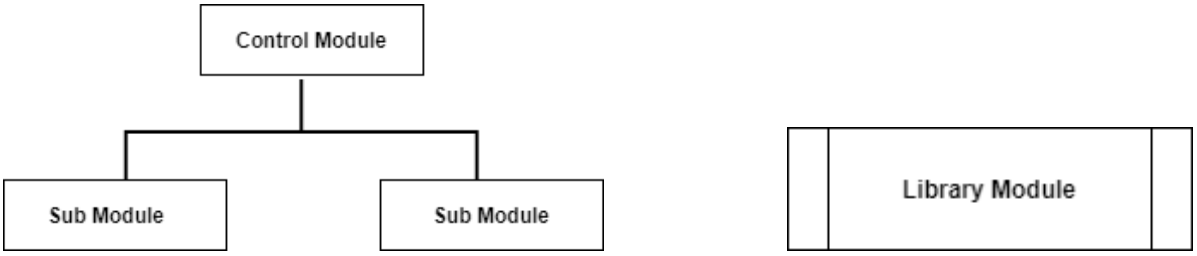
Modules at top level called modules at low level. Components are read from top to bottom and left to right. When a module calls another, it views the called module as black box, passing required parameters and receiving results.

2.1 Symbols used in construction of structured chart

1. Module

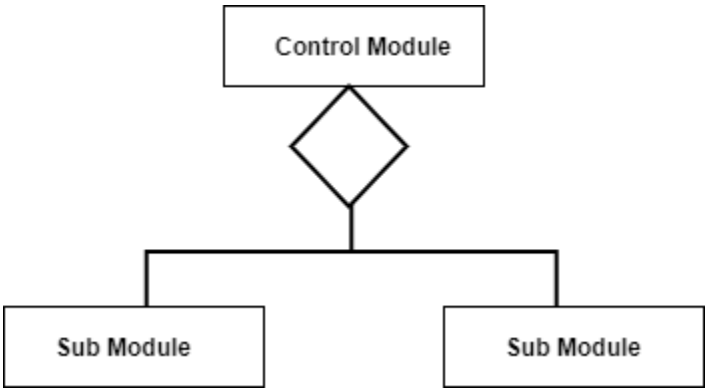
It represents the process or task of the system. It is of three types as shown in below figure.

- **Control Module :** A control module branches to more than one sub module.
- **Sub Module :** Sub Module is a module which is the part (Child) of another module.
- **Library Module :** Library Module are reusable and invokable from any module.



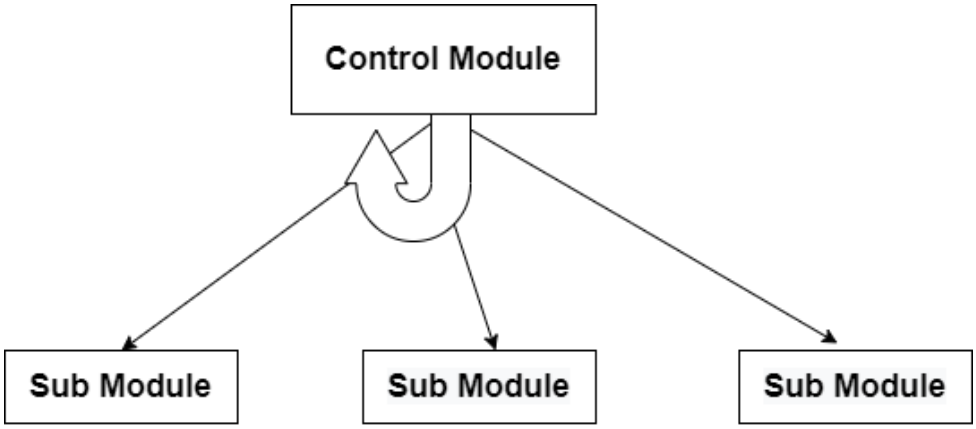
2. Conditional Call

It represents that the control module can select any of the sub modules on the basis of some condition as shown in below figure.



3. Loop (Repetitive call of module)

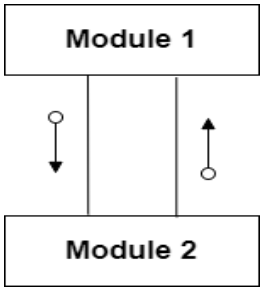
It represents the repetitive execution of a module by the sub module.
A curved arrow represents a loop in the module.



All the sub modules covered by the loop repeat execution of the module.

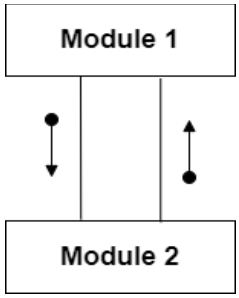
4. Data Flow

It represents the flow of data between the modules. It is represented by directed arrow with an empty circle at the end.



5. **Control Flow**

It represents the flow of control between the modules. It is represented by a directed arrow with a filled circle at the end.



6. **Physical Storage**

Physical Storage is where all the information is to be stored.

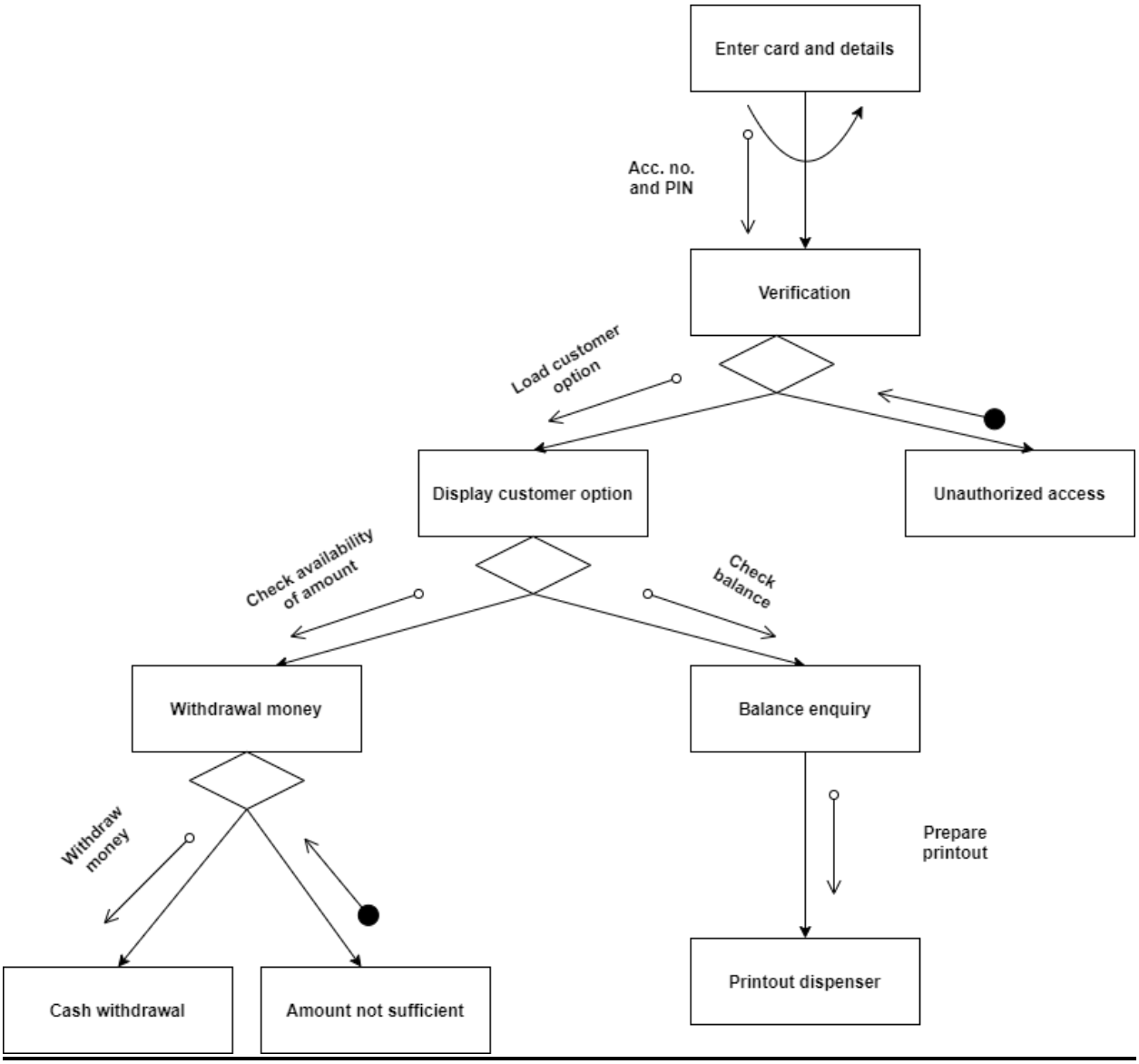
2.2 Types of Structure Chart:

- **Transform Centered Structured:**
  These types of structure charts are designed for the systems that receive an input which is transformed by a sequence of operations being carried out by one module.
- **Transaction Centered Structure:**
  This structure describes a system that processes a number of different types of transaction.

2.3 Steps in drawing a structure chart

- Review the DFDs and object models
- Identify modules and relationships
- Add couples, loops, and conditions
- Analyze the structure chart, the DFDs, and the data dictionary

2.4 Structure Chart of ATM management system



CONCLUSION:-

- Level – 0 and Level – 1 Data Flow Diagram (DFD) for ATM MANAGEMENT SYSTEM have been drawn successfully.
- Structure Chart for ATM MANAGEMENT SYSTEM have been drawn successfully.

EXPERIMENT – 4

AIM:-

To perform the user's view analysis for the suggested system: Use case diagram.

REQUIREMENTS:-

1. SOFTWARE REQUIREMENT – Microsoft Word, Umbrello
2. HARDWARE REQUIREMENT – Computer, Keyboard, Mouse, CPU

THEORY:-

According to the UML specification a use case diagram is "a diagram that shows the relationships among actors and use cases within a system." Use case diagrams are often used to:

- Provide an overview of all or part of the usage requirements for a system or organization in the form of an essential model or a business model
- Communicate the scope of a development project
- Model your analysis of your usage requirements in the form of a system use case model

Use case models should be developed from the point of view of your project stakeholders and not from the (often technical) point of view of developers. There are guidelines for:

- Use Cases
- Actors
- Relationships
- System Boundary Boxes

1. Use Cases

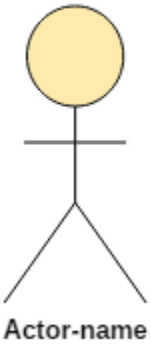
A use case describes a sequence of actions that provide a measurable value to an actor. A use case is drawn as a horizontal ellipse on a UML use case diagram.

1. Use Case Names Begin With a Strong Verb
2. Name Use Cases Using Domain Terminology
3. Place Your Primary Use Cases In The Top-Left Corner Of The Diagram
4. Imply Timing Considerations By Stacking Use Cases.

2. Actors

An actor is a person, organization, or external system that plays a role in one or more interactions with your system (actors are typically drawn as stick figures on UML Use Case diagrams).

1. Place Your Primary Actor(S) In The Top-Left Corner Of The Diagram
2. Draw Actors To The Outside Of A Use Case Diagram
3. Name Actors With Singular, Business-Relevant Nouns
4. Associate Each Actor With One Or More Use Cases
5. Actors Model Roles, Not Positions
6. Use <<system>> to Indicate System Actors
7. Actors Don't Interact With One Another
8. Introduce an Actor Called "Time" to Initiate Scheduled Events



3. Relationships

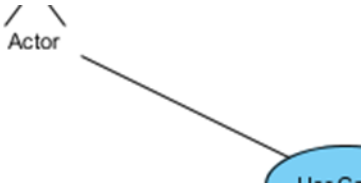
There are several types of relationships that may appear on a use case diagram:

- An association between an actor and a use case
- An association between two use cases
- A generalization between two actors

- A generalization between two use cases

Associations are depicted as lines connecting two modeling elements with an optional open-headed arrowhead on one end of the line indicating the direction of the initial invocation of the relationship. Generalizations are depicted as a close-headed arrow with the arrow pointing towards the more general modeling element.

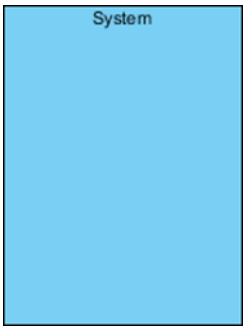
1. Indicate An Association Between An Actor And A Use Case If The Actor Appears Within The Use Case Logic
2. Avoid Arrowheads On Actor-Use Case Relationships
3. Apply <<include>> When You Know Exactly When To Invoke The Use Case
4. Apply <<extend>> When A Use Case May Be Invoked Across Several Use Case Steps
5. Introduce <<extend>> associations sparingly
6. Generalize Use Cases When a Single Condition Results In Significantly New Business Logic
7. Do Not Apply <<uses>>, <<includes>>, or <<extends>>
8. Avoid More Than Two Levels Of Use Case Associations
9. Place An Included Use Case To The Right Of The Invoking Use Case
10. Place An Extending Use Case Below The Parent Use Case
11. Apply the “Is Like” Rule to Use Case Generalization
12. Place an Inheriting Use Case Below The Base Use Case
13. Apply the “Is Like” Rule to Actor Inheritance
14. Place an Inheriting Actor Below the Parent Actor



4. System Boundary Boxes

The rectangle around the use cases is called the system boundary box and as the name suggests it indicates the scope of your system – the use cases inside the rectangle represent the functionality that you intend to implement.

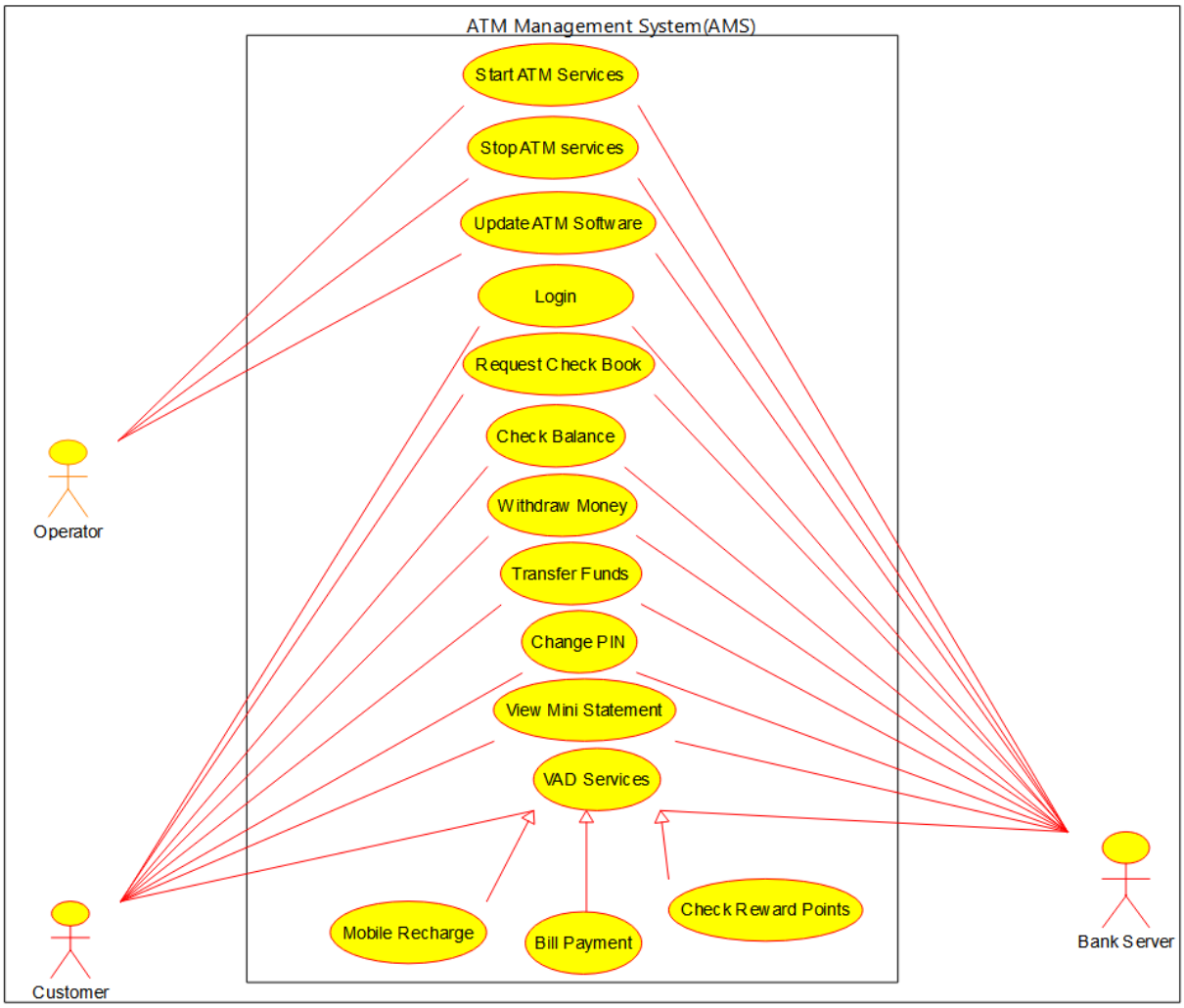
1. Indicate Release Scope with a System Boundary Box.
2. Avoid Meaningless System Boundary Boxes.



CREATING USE CASE DIAGRAM

We start by identifying as many actors as possible. You should ask how the actors interact with the system to identify an initial set of use cases. Then, on the diagram, you connect the actors with the use cases with which they are involved. If actor supplies information, initiates the use case, or receives any information as a result of the use case, then there should be an association between them.

ATM MANAGEMENT SYSTEM USE CASE DIAGRAM



Conclusion:-

Use case diagram for ATM MANAGEMENT SYSTEM has been drawn successfully.

EXPERIMENT – 5

AIM:-

To draw the structural view diagram for the system: Class diagram, object diagram .

REQUIREMENTS:-

1. SOFTWARE REQUIREMENT – Microsoft Word, Umbrello
2. HARDWARE REQUIREMENT – Computer, Keyboard, Mouse, CPU

THEORY:-

Class Diagram

A description of a group of objects all with similar roles in the system, which consists of:

1. **Structural features** (attributes) define what objects of the class "know"
- Represent the state of an object of the class
- Are descriptions of the structural or static features of a class

2. **Behavioral features** (operations) define what objects of the class "can do"
- Define the way in which objects may interact
- Operations are descriptions of behavioral or dynamic features of a class

Class Notation

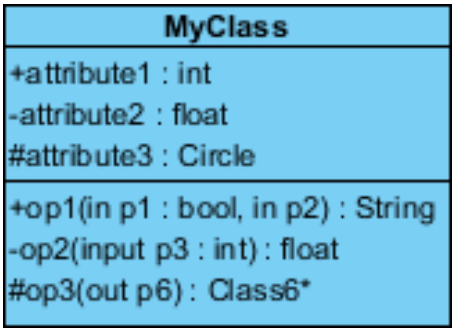
A class notation consists of three parts:

1. **Class Name**
- The name of the class appears in the first partition.

2. **Class Attribute**
- Attributes are shown in the second partition
- The attribute type is shown after the colon.
- Attributes map onto member variables (data members) in code.

3. **Class Operations** (Methods)

- Operations are shown in the third partition. They are services the class provides.
- The return type of a method is shown after the colon at the end of the method signature.
- The return type of method parameters is shown after the colon following the parameter name.
- Operations map onto class methods in code



The graphical representation of the class - MyClass as shown above:

- MyClass has 3 attributes and 3 operations
- Parameter p3 of op2 is of type int
- op2 returns a float
- op3 returns a pointer (denoted by a *) to Class6

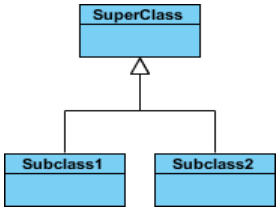
Class Relationships

A class may be involved in one or more relationships with other classes. A relationship can be one of the following types: (Refer to the figure on the right for the graphical representation of relationships).

Relationship Type	Graphical Representation

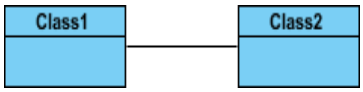
Inheritance (or Generalization):

- Represents an "is-a" relationship.
- An abstract class name is shown in italics.
- SubClass1 and SubClass2 are specializations of Super Class.
- A solid line with a hollow arrowhead that point from the child to the parent class



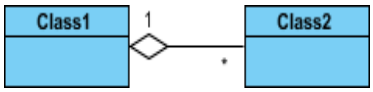
Simple Association:

- A structural link between two peer classes.
- There is an association between Class1 and Class2
- A solid line connecting two classes



Aggregation:

A special type of association. It represents a "part of" relationship.



- Class2 is part of Class1.
- Many instances (denoted by the *) of Class2 can be associated with Class1.
- Objects of Class1 and Class2 have separate lifetimes.
- A solid line with an unfilled diamond at the association end connected to the class of composite

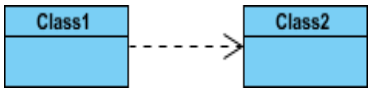
Composition:

A special type of aggregation where parts are destroyed when the whole is destroyed.

- Objects of Class2 live and die with Class1.
- Class2 cannot stand by itself.
- A solid line with a filled diamond at the association connected to the class of composite

Dependency:

- Exists between two classes if the changes to the definition of one may cause changes to the other (but not the other way around).
- Class1 depends on Class2
- A dashed line with an open arrow

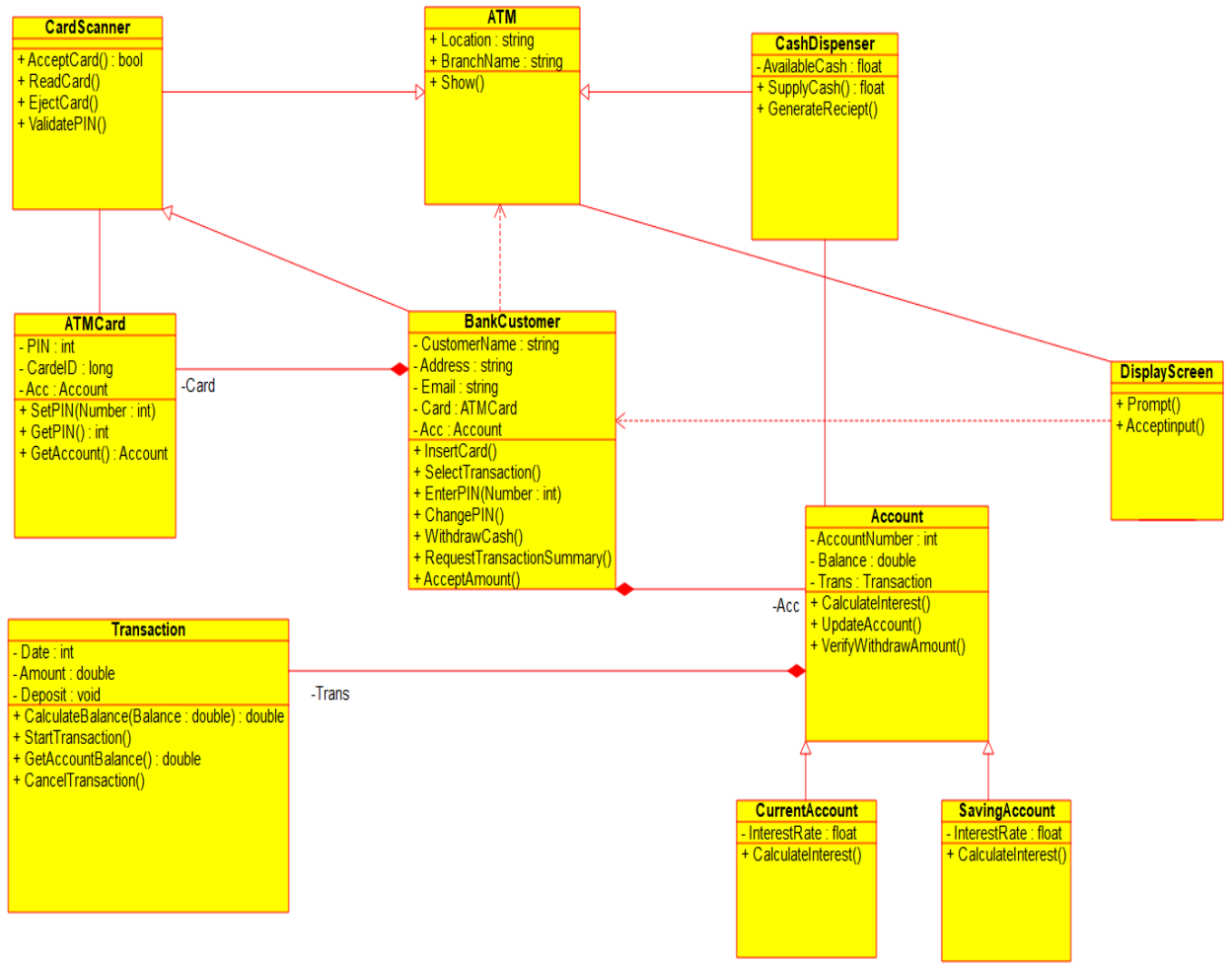


Visibility of Class attributes and Operations

In object-oriented design, there is a notation of visibility for attributes and operations. UML identifies four types of visibility: **public**, **protected**, **private**, and **package**. The +, -, # and ~ symbols before an attribute and operation name in a class denote the visibility of the attribute and operation.

- + denotes public attributes or operations
- - denotes private attributes or operations
- # denotes protected attributes or operations
- ~ denotes package attributes or operations

Class Diagram of ATM Management System



Object Diagram

Object is an instance of a class in a particular moment in runtime that can have its own state and data values. Likewise a static UML object diagram is an instance of a class diagram; it shows a snapshot of the detailed state of a system at a point in time, thus an object diagram encompasses objects and their relationships which may be considered a special case of a class diagram or a communication diagram.

Purpose of Object Diagram

The use of object diagrams is fairly limited, mainly to show examples of data structures.

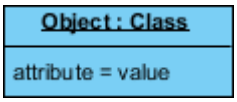
- During the analysis phase of a project, you might create a class diagram to describe the structure of a system and then create a set of object diagrams as test cases to verify the accuracy and completeness of the class diagram.
- Before you create a class diagram, you might create an object diagram to discover facts about specific model elements and their links, or to illustrate specific examples of the classifiers that are required.

Basic Object Diagram Symbols and Notations

Object Names:

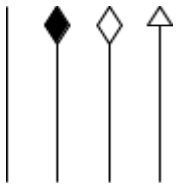
- Every object is actually symbolized like a rectangle, that offers the name from the object and its class underlined as well as divided with a colon.

Object Attributes:



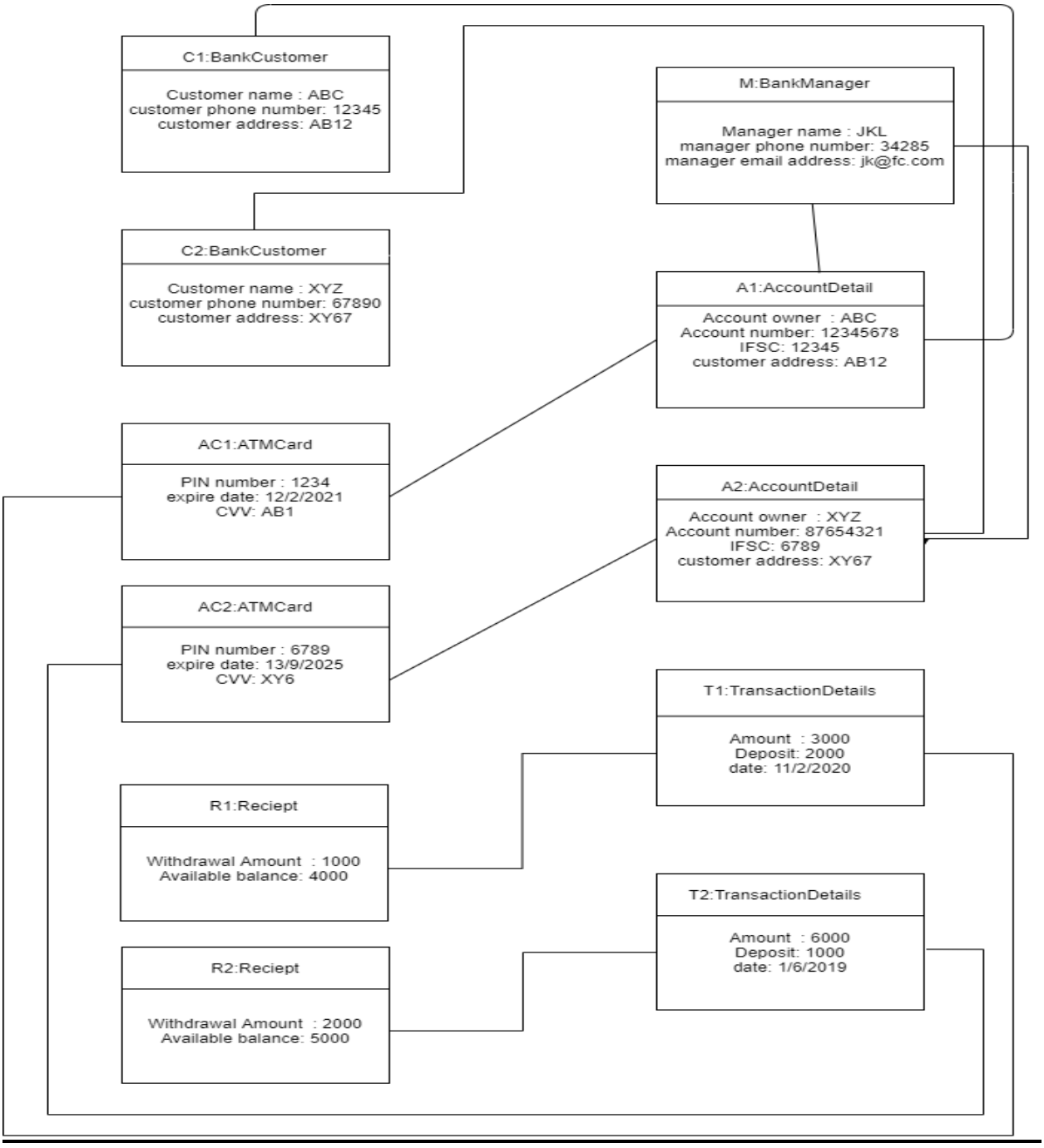
- Similar to classes, you are able to list object attributes inside a separate compartment. However, unlike classes, object attributes should have values assigned for them.

Links:



- Links tend to be instances associated with associations. You can draw a link while using the lines utilized in class diagrams.

Object Diagram of ATM Management System:



CONCLUSION:

Class diagram and object diagram for ATM MANAGEMENT SYSTEM has been drawn successfully.

EXPERIMENT – 6

AIM:-

To draw the behavioral view diagram : State-chart diagram, Activity diagram

REQUIREMENTS:-

1. SOFTWARE REQUIREMENT – Microsoft Word, Umbrello
2. HARDWARE REQUIREMENT – Computer, Keyboard, Mouse, CPU

THEORY:-

A. State Chart diagram

A state diagram is used to represent the condition of the system or part of the system at finite instances of time. It's a behavioral diagram and it represents the behavior using finite state transitions. State diagrams are also referred to as State machines and State-chart Diagrams. These terms are often used interchangeably. So simply, a state diagram is used to model the dynamic behavior of a class in response to time and changing external stimuli. We can say that each and every class has a state but we don't model every class using State diagrams. We prefer to model the states with three or more states.

Uses of statechart diagram –

- We use it to state the events responsible for change in state (we do not show what processes cause those events).
- We use it to model the dynamic behavior of the system .
- To understand the reaction of objects/classes to internal or external stimuli.

Basic components of a statechart diagram –

1. **Initial state –** We use a black filled circle to represent the initial state of a System or a class.

initial state notation

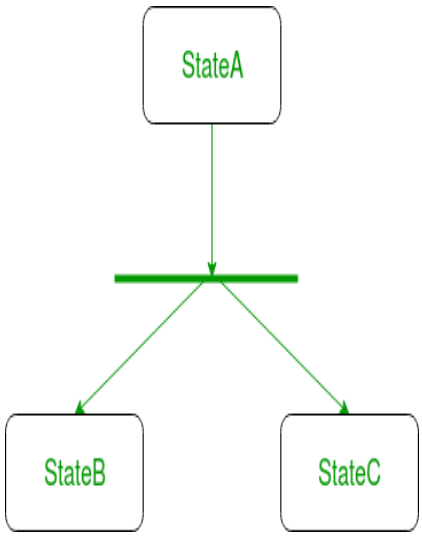
2. **Transition –** We use a solid arrow to represent the transition or change of control from one state to another. The arrow is labelled with the event which causes the change in state.

Transition notation

3. **State –** We use a rounded rectangle to represent a state. A state represents the conditions or circumstances of an object of a class at an instant of time.

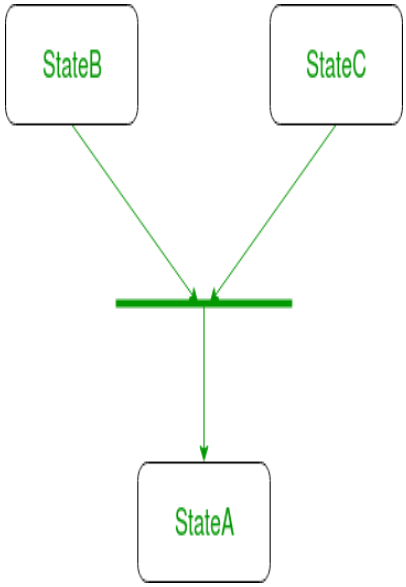
state notation

4. **Fork –** We use a rounded solid rectangular bar to represent a Fork notation with incoming arrow from the parent state and outgoing arrows towards the newly created states. We use the fork notation to represent a state splitting into two or more concurrent states.



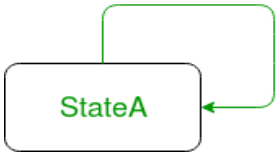
fork notation

5. **Join –** We use a rounded solid rectangular bar to represent a Join notation with incoming arrows from the joining states and outgoing arrow towards the common goal state. We use the join notation when two or more states concurrently converge into one on the occurrence of an event or events.



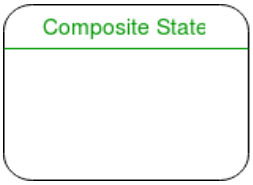
join notation

6. **Self transition –** We use a solid arrow pointing back to the state itself torepresent a self transition. There might be scenarios when the state of the object does not change upon the occurrence of an event. We use self transitions to represent such cases.



self transition notation

7. **Composite state –** We use a rounded rectangle to represent a composite state also.We represent a state with internal activities using a composite state.



a state with internal activities

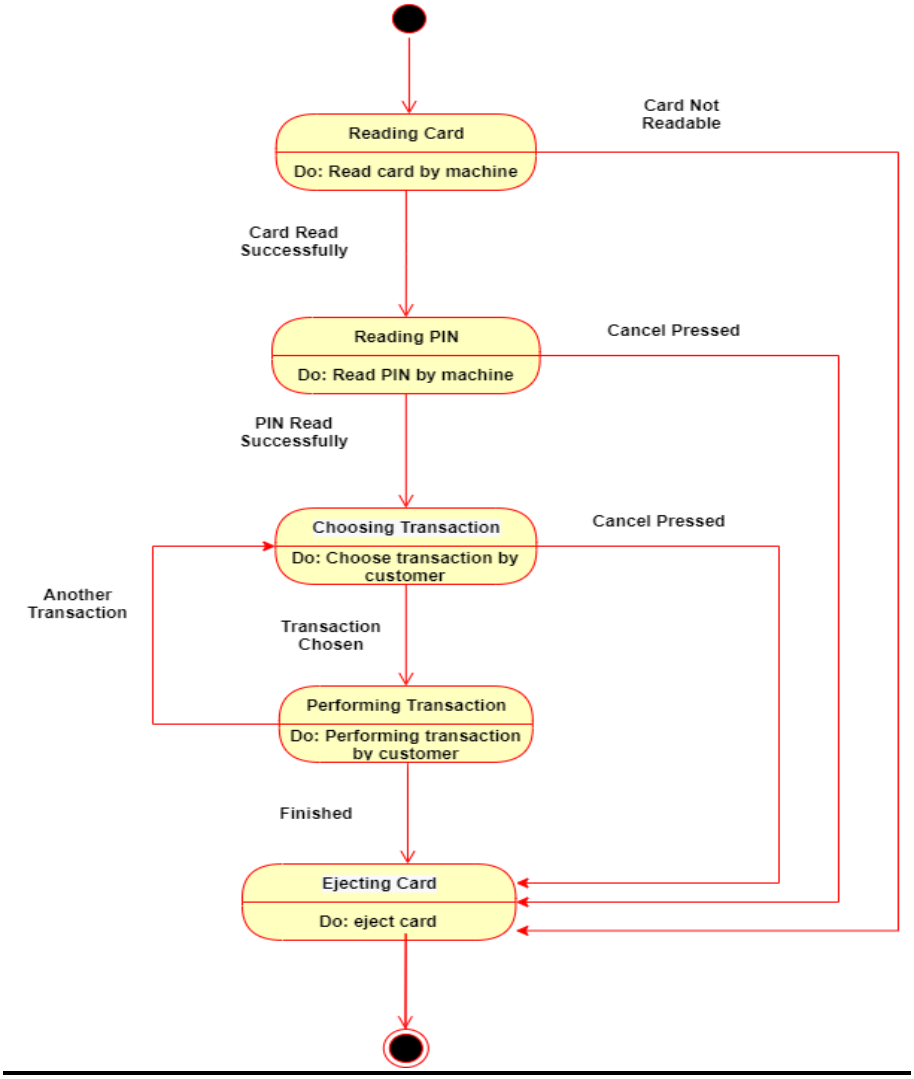
8. **Final state –** We use a filled circle within a circle notation to represent the final state in a state machine diagram.

final state notation

Steps to draw a state diagram –

1. Identify the initial state and the final terminating states.
2. Identify the possible states in which the object can exist (boundary values corresponding to different attributes guide us in identifying different states).
3. Label the events which trigger these transitions.

State Chart diagram of ATM Management System :-



B. Activity Diagram

We use **Activity Diagrams** to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case. We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram. An activity diagram focuses on the condition of flow and the sequence in which it happens. We describe or depict what causes a particular event using an activity diagram.
UML models basically three types of diagrams, namely, structure diagrams, interaction diagrams, and behavior diagrams. An activity diagram is a **behavioral diagram** i.e. it depicts the behavior of a system.

An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed. We can depict both sequential processing and concurrent processing of activities using an activity diagram. They are used in business and process modelling where their primary use is to depict the dynamic aspects of a system.

Activity Diagram Notations –

1. **Initial State –** The starting state before an activity takes place is depicted using the initial state.

notation for initial state or start state

A process can have only one initial state unless we are depicting nested activities. We use a black filled circle to depict the initial state of a system. For objects, this is the state when they are instantiated. The Initial State from the UML Activity Diagram marks the entry point and the initial Activity State.

2. **Action or Activity State –** An activity represents execution of an action on objects or by objects. We represent an activity using a rectangle with rounded corners. Basically any action or event that takes place is represented using an activity.

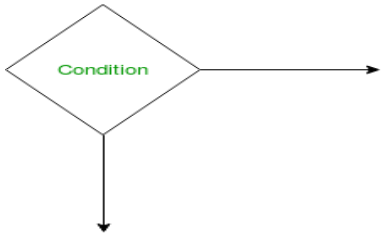
notation for an activity state

3. **Action Flow or Control flows –** Action flows or Control flows are also referred to as paths and edges. They are used to show the transition from one activity state to another.

notation for control Flow

An activity state can have multiple incoming and outgoing action flows. We use a line with an arrow head to depict a Control Flow. If there is a constraint to be adhered to while making the transition it is mentioned on the arrow.

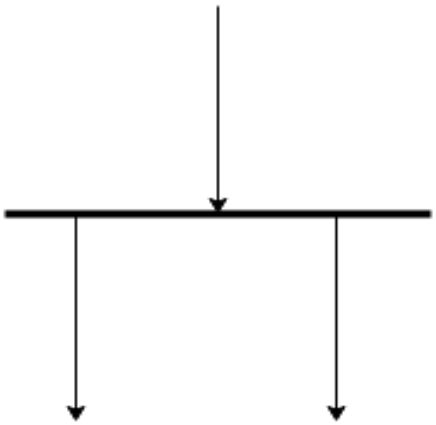
4. **Decision node and Branching –** When we need to make a decision before deciding the flow of control, we use the decision node.



notation for decision node

The outgoing arrows from the decision node can be labelled with conditions or guard expressions.It always includes two or more output arrows.

5. **Fork –** Fork nodes are used to support concurrent activities.

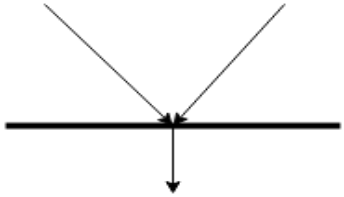


fork notation

When we use a fork node when both the activities get executed concurrently i.e. no decision is made before splitting the activity into two parts. Both parts need to be executed in case of a fork statement.
We use a rounded solid rectangular bar to represent a Fork notation with incoming arrow

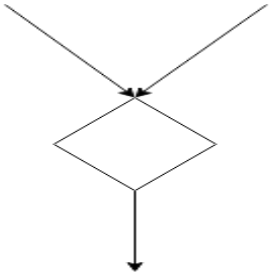
from the parent activity state and outgoing arrows towards the newly created activities.

6. **Join –** Join nodes are used to support concurrent activities converging into one. For join notations we have two or more incoming edges and one outgoing edge.



join notation

7. **Merge or Merge Event –** Scenarios arise when activities which are not being executed concurrently have to be merged. We use the merge notation for such scenarios. We can merge two or more activities into one if the control proceeds onto the next activity irrespective of the path chosen.



merge notation

8. **Final State or End State –** The state which the system reaches when a particular process or activity ends is known as a Final State or End State. We use a filled circle within a circle notation to represent the final state in a state machine diagram. A system or a process can have multiple final states.

Figure – notation for final state

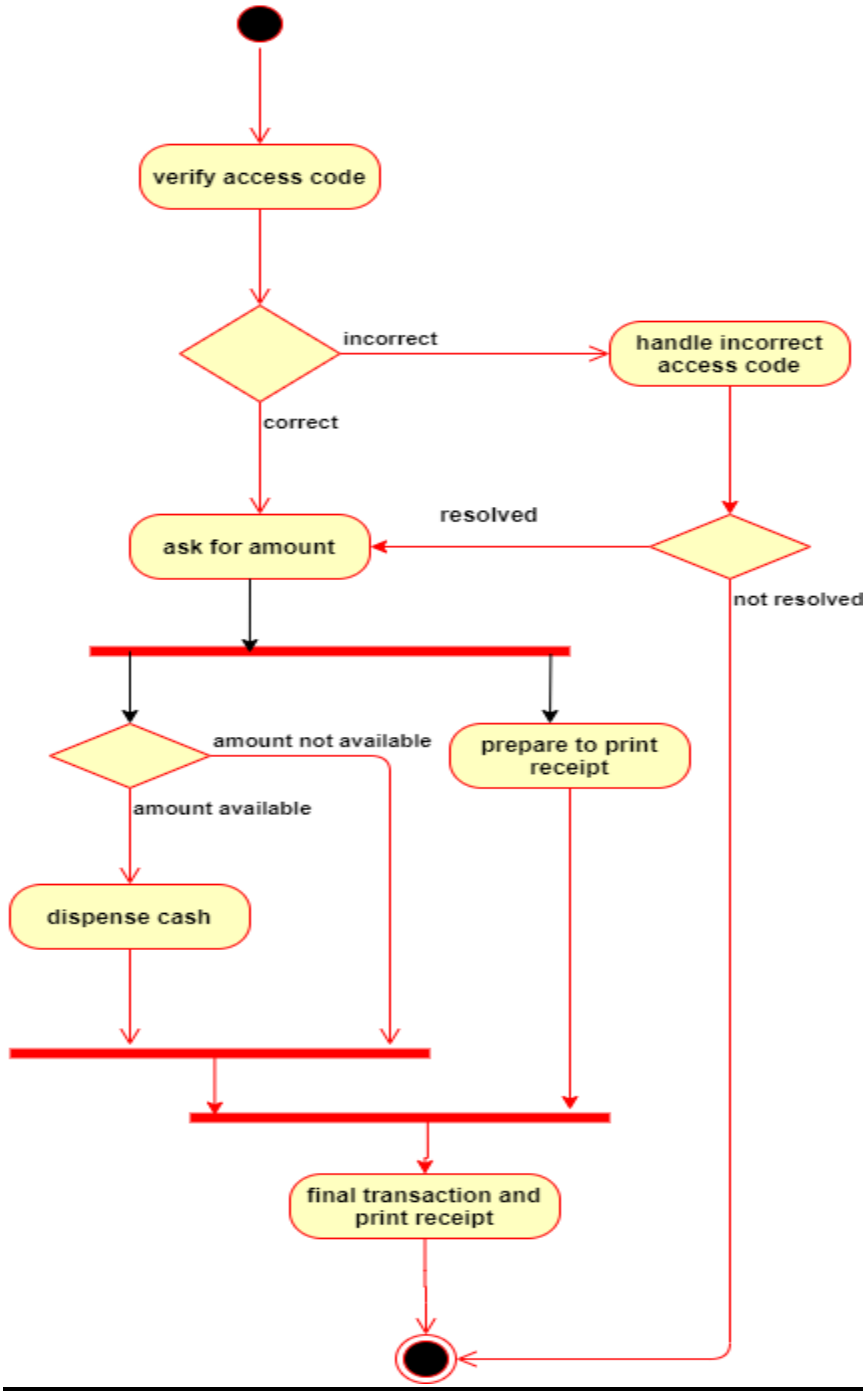
Steps to Draw an activity diagram –

1. Identify the initial state and the final states.
2. Identify the intermediate activities needed to reach the final state from he initial state.
3. Identify the conditions or constraints which cause the system to change control flow.
4. Draw the diagram with appropriate notations.

Uses of an Activity Diagram –

- Dynamic modelling of the system or a process.
- Illustrate the various steps involved in a UML use case.
- Model software elements like methods,operations and functions.
- We can use Activity diagrams to depict concurrent activities easily.
- Show the constraints, conditions and logic behind algorithms.

Activity Diagram for ATM Management System:-



CONCLUSION:

State Chart diagram and Activity diagram for ATM MANAGEMENT SYSTEM has been drawn successfully.

EXPERIMENT – 7

AIM:-

To perform the behavioral view diagram for the suggested system : Sequence diagram, Collaboration diagram

REQUIREMENTS:-

1. SOFTWARE REQUIREMENT – Microsoft Word, Umbrello
2. HARDWARE REQUIREMENT – Computer, Keyboard, Mouse, CPU

THEORY:-

A. Sequence Diagrams

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

Sequence Diagram Notations –

1. **Actors –** An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram.

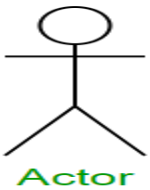


Figure – notation symbol for actor
We use actors to depict various roles including human users and other external subjects. We represent an actor in a UML diagram using a stick person notation. We can have multiple actors in a sequence diagram.

2. **Lifelines –** A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram. The standard in UML for naming a lifeline follows the following format – Instance Name : Class Name

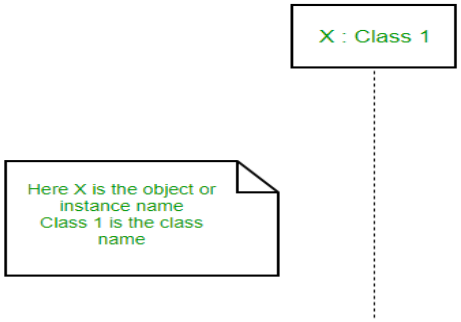


Figure – lifeline

We display a lifeline in a rectangle called head with its name and type. The head is located on top of a vertical dashed line (referred to as the stem) as shown above. If we want to model an unnamed instance, we follow the same pattern except now the portion of lifeline's name is left blank.

3. **Messages –** Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline. We represent messages using arrows. Lifelines and messages form the core of a sequence diagram. Messages can be broadly classified into the following **categories** :
    - **Synchronous messages –** A synchronous message waits for a reply before the interaction can move forward. The sender waits until the receiver has completed the processing of the message. The caller continues only when it knows that the receiver has processed the previous message i.e. it receives a reply message. A large number of calls in object oriented programming are synchronous. We use a solid arrow head to represent a synchronous message.

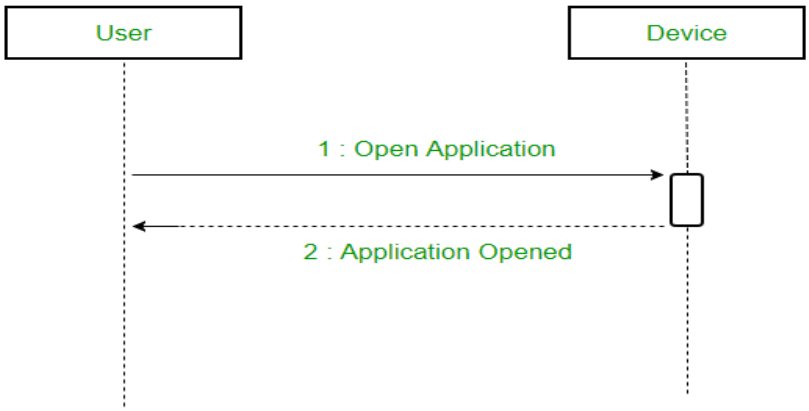


Figure – a sequence diagram using a synchronous message

- **Asynchronous Messages –** An asynchronous message does not wait for a reply from the receiver. The interaction moves forward irrespective of the receiver processing the previous message or not. We use a lined arrow head to represent an asynchronous message.

- **Create message –** We use a Create message to instantiate a new object in the sequence diagram. There are situations when a particular message call requires the creation of an object. It is represented with a dotted arrow and create word labelled on it to specify that it is the create Message symbol.
  For example – The creation of a new order on a e-commerce website would require a new object of Order class to be created.

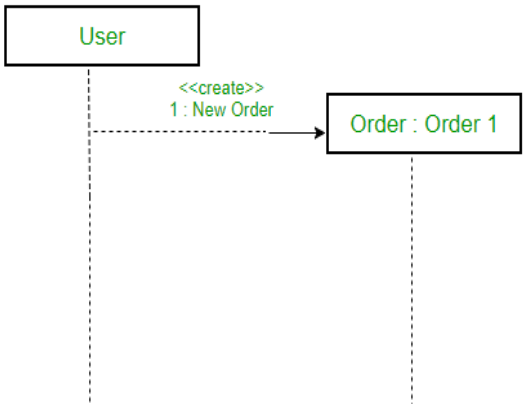


Figure – a situation where create message is used

- **Delete Message –** We use a Delete Message to delete an object. When an object is deallocated memory or is destroyed within the system we use the Delete Message symbol. It destroys the occurrence of the object in the system.It is represented by an arrow terminating with a x. For example – In the scenario below when the order is received by the user, the object of order class can be destroyed.

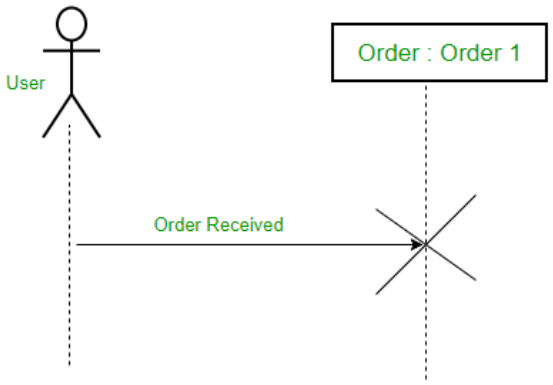


Figure – a scenario where delete message is used

- **Self Message –** Certain scenarios might arise where the object needs to send a message to itself. Such messages are called Self Messages and are represented with a U shaped arrow.

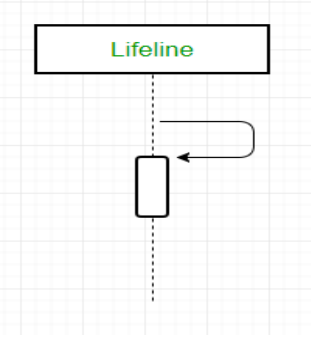


Figure – self message

- **Reply Message –** Reply messages are used to show the message being sent from the receiver to the sender. We represent a return/reply message using an open arrowhead with a dotted line. The interaction moves forward only when a reply message is sent by the receiver.

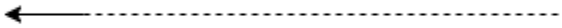


Figure – reply message

- **Found Message –** A Found message is used to represent a scenario where an unknown source sends the message. It is represented using an arrow directed towards a lifeline from an end point. For example: Consider the scenario of a hardware failure.

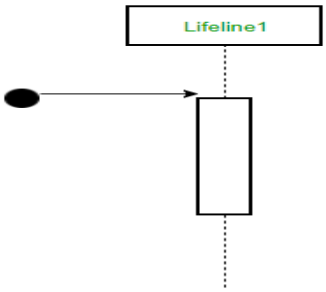


Figure – found message

- **Lost Message –** A Lost message is used to represent a scenario where the recipient is not known to the system. It is represented using an

arrow directed towards an end point from a lifeline. For example:
Consider a scenario where a warning is generated.

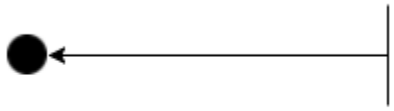


Figure – lost message

4. **Guards –** To model conditions we use guards in UML. They are used when we need to restrict the flow of messages on the pretext of a condition being met. Guards play an important role in letting software developers know the constraints attached to a system or a particular process.
   For example: In order to be able to withdraw cash, having a balance greater than zero is a condition that must be met as shown below.

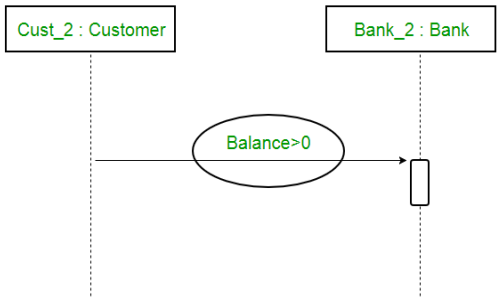


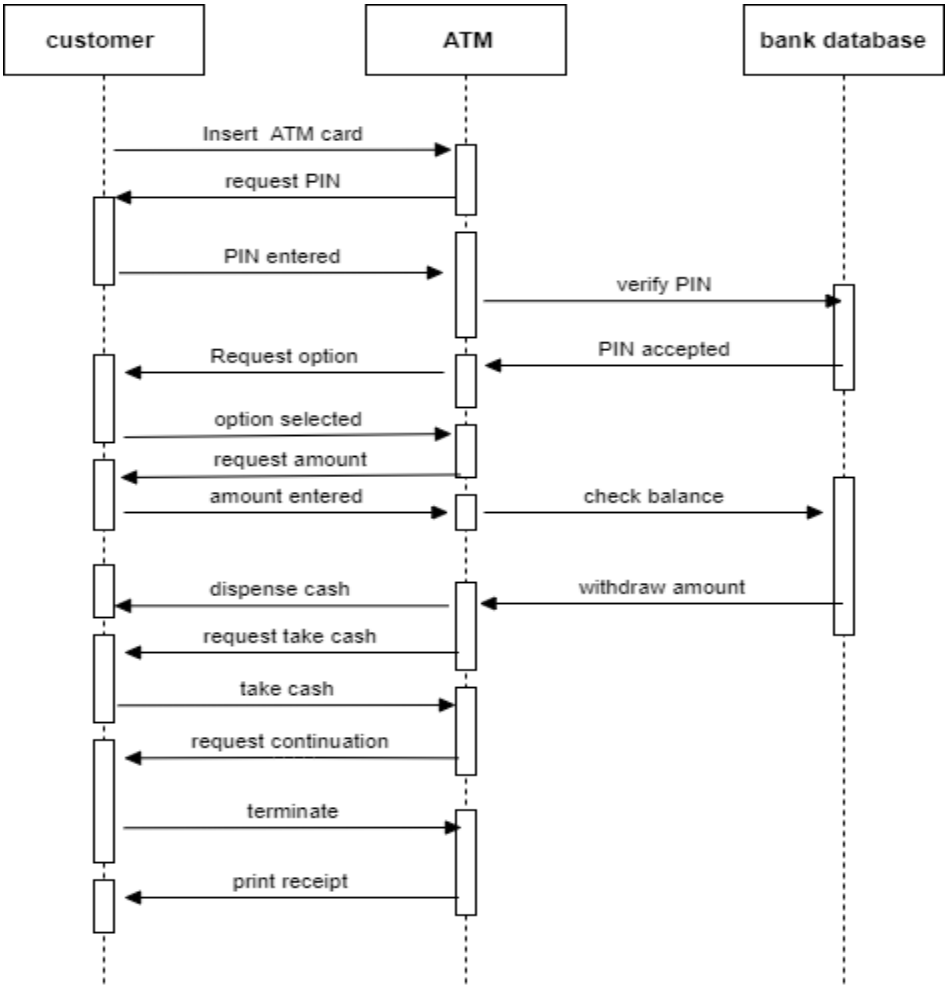
Figure – sequence diagram using a guard

Uses of sequence diagrams –

- Used to model and visualise the logic behind a sophisticated function, operation or procedure.
- They are also used to show details of UML use case diagrams.
- Used to understand the detailed functionality of current or future systems.

- Visualise how messages and tasks move between objects or components in a system.

Sequence Diagrams of ATM Management System:-



B. Collaboration Diagram

Collaboration Diagram represents the interaction of the objects to perform the behavior of a particular use case or a part of use case. The designers use the Sequence diagram and Collaboration Diagrams to define and clarify the roles of the objects that perform a particular flow of events of a use case.

• The collaboration diagram is used to show the relationship between the objects in a system.

• Both the sequence and the collaboration diagrams represent the same information but differently.
• Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming.
• An object consists of several features.
• Multiple objects present in the system are connected to each other.
• The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system.

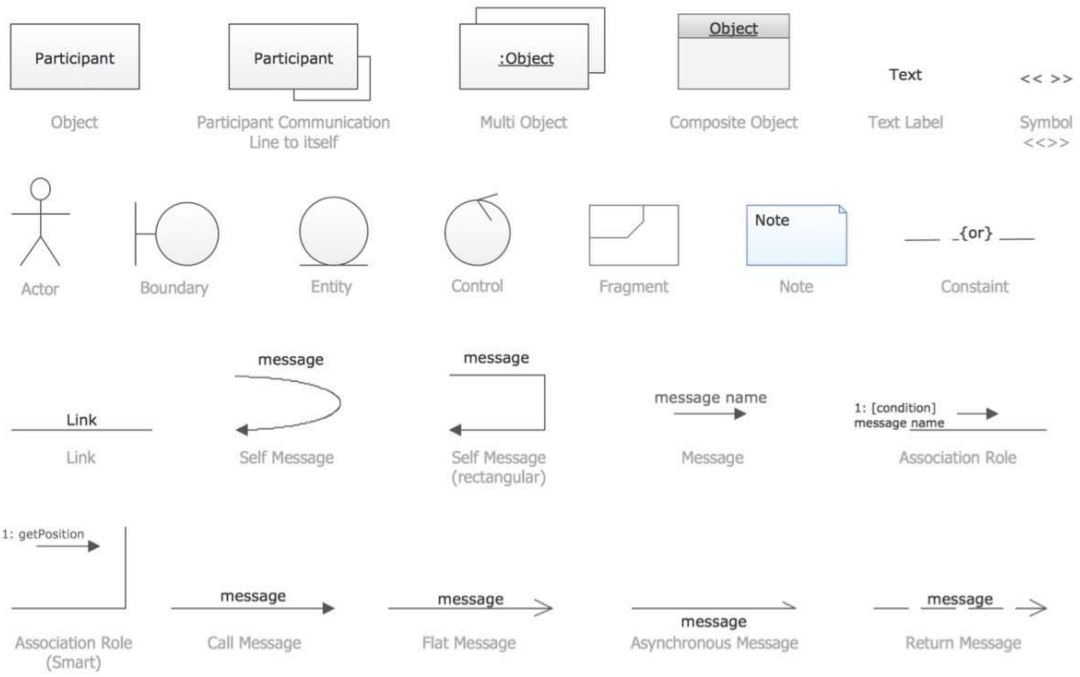
Notations of a Collaboration Diagram

- **Objects:** The representation of an object is done by an object symbol with its name and class underlined, separated by a colon.
  In the collaboration diagram, objects are utilized in the following ways:
  – The object is represented by specifying their name and class.
  – It is not mandatory for every class to appear.
  – A class may constitute more than one object.
  – In the collaboration diagram, firstly, the object is created, and then its class is specified.
  – To differentiate one object from another object, it is necessary to name them.

- **Actors:** In the collaboration diagram, the actor plays the main role as it invokes the interaction. Each actor has its respective role and name. In this, one actor initiates the use case.

- **Links:** The link is an instance of association, which associates the objects and actors. It portrays a relationship between the objects through which the messages are sent. It is represented by a solid line. The link helps an object to connect with or navigate to another object, such that the message flows are attached to links.

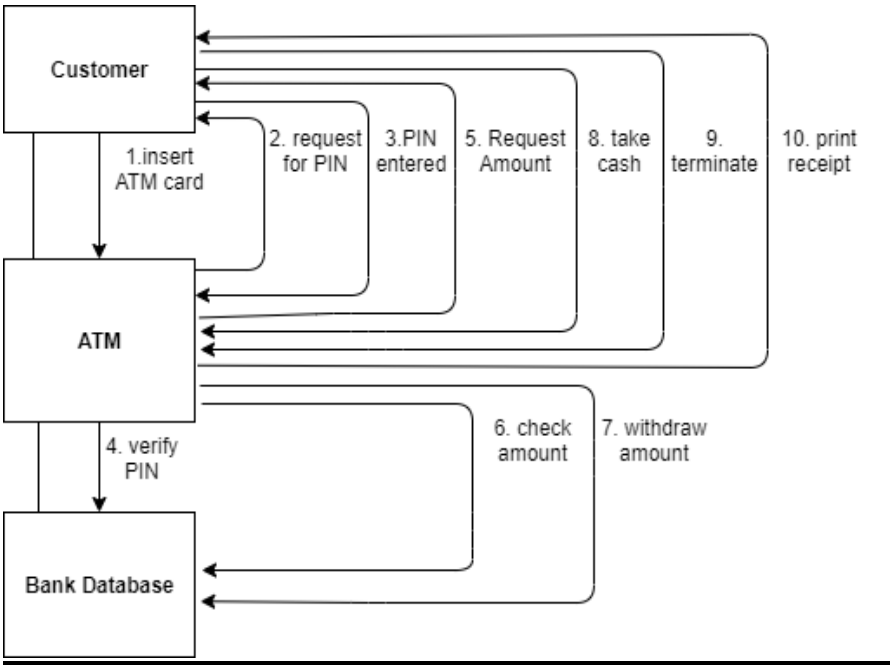
- **Messages:** It is a communication between objects which carries information and includes a sequence number, so that the activity may take place. It is represented by a labeled arrow, which is placed near a link. The messages are sent from the sender to the receiver, and the direction must be navigable in that particular direction. The receiver must understand the message.



When to use a Collaboration Diagram

The collaborations are used when it is essential to depict the relationship between the object. Both the sequence and collaboration diagrams represent the same information, but the way of portraying it quite different. The collaboration diagrams are best suited for analyzing use cases.

Collaboration Diagram of ATM Management System:-



CONCLUSION:

Sequence diagram and Collaboration diagram for ATM MANAGEMENT SYSTEM has been drawn successfully.

EXPERIMENT – 8

AIM:-

To perform the implementation view diagram : Component diagram for the System

REQUIREMENTS:-

1. SOFTWARE REQUIREMENT – Microsoft Word, Umbrello
2. HARDWARE REQUIREMENT – Computer, Keyboard, Mouse, CPU

THEORY:-

Component Diagram

Component diagrams are used to visualize the organization of system components and the dependency relationships between them. They provide a high-level view of the components within a system.
The components can be a software component such as a database or user interface; or a hardware component such as a circuit, microchip or device; or a business unit such as supplier, payroll or shipping.
Component diagrams

- Are used in Component-Based-Development to describe systems with Service-Oriented-Architecture
- Show the structure of the code itself
- Can be used to focus on the relationship between components while hiding specification detail
- Help communicate and explain the functions of the system being built to stakeholders

Component Diagram Symbols

We have explained below the common component diagram notations that are used to draw a component diagram.

- **Component**

There are three ways the component symbol can be used.

1) Rectangle with the component stereotype (the text <<component>>). The component stereotype is usually used above the component name to avoid confusing the shape with a class icon.

2) Rectangle with the component icon in the top right corner and the name of the component.

3) Rectangle with the component icon and the component stereotype.

- **Provided Interface and the Required Interface**

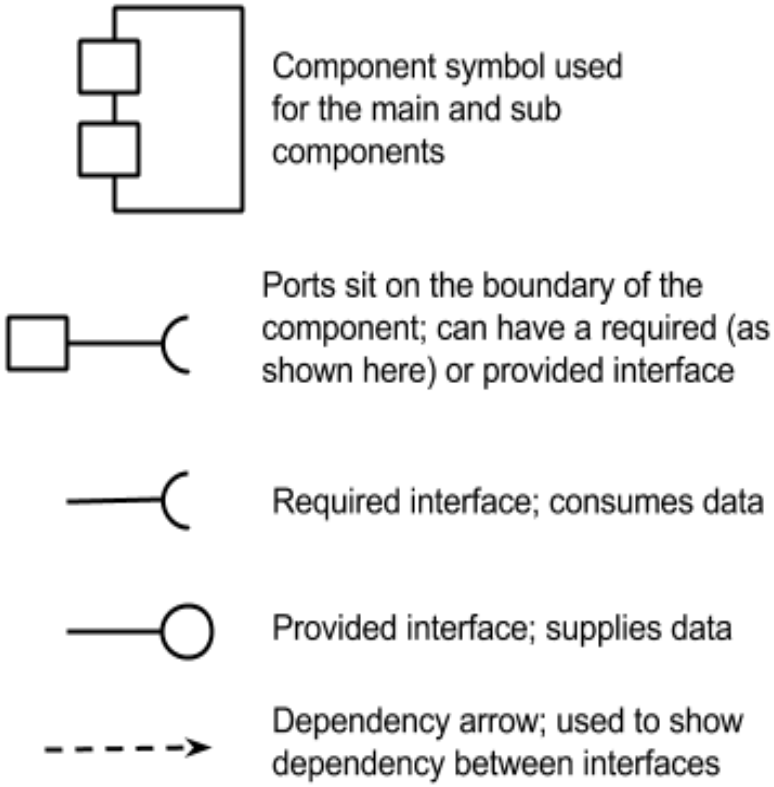
Interfaces in component diagrams show how components are wired together and interact with each other. The assembly connector allows linking the component's required interface (represented with a semi-circle and a solid line) with the provided interface (represented with a circle and solid line) of another component. This shows that one component is providing the service that the other is requiring.

- **Port**

Port (represented by the small square at the end of a required interface or provided interface) is used when the component delegates the interfaces to an internal class.

- **Dependencies**

Although you can show more detail about the relationship between two components using the ball-and-socket notation (provided interface and required interface), you can just as well use a dependency arrow to show the relationship between two components.



How to Draw a Component Diagram

You can use a component diagram when you want to represent your system as components and want to show their interrelationships through interfaces. It helps you get an idea of the implementation of the system. Following are the steps you can follow when drawing a component diagram.

Step 1: Figure out the purpose of the diagram and identify the artifacts such as the files, documents etc. in your system or application that you need to represent in your diagram.

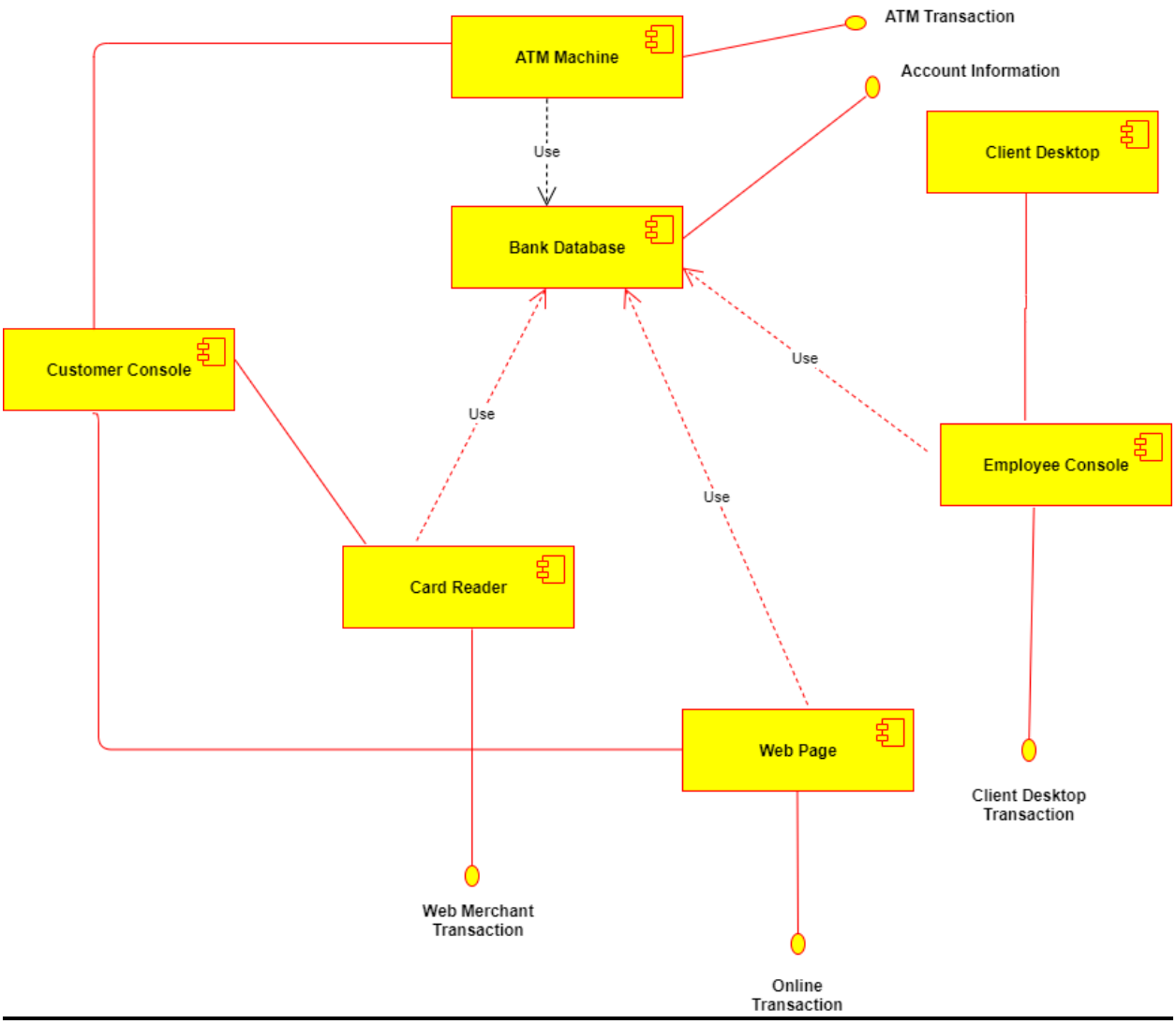
Step 2: As you figure out the relationships between the elements you identified earlier, create a mental layout of your component diagram

Step 3: As you draw the diagram, add components first, grouping them within other components as you see fit

Step 4: Next step is to add other elements such as interfaces, classes, objects, dependencies etc. to your component diagram and complete it.

Step 5: You can attach notes on different parts of your component diagram to clarify certain details to others.

Component Diagram for ATM Management System



CONCLUSION:

Component diagram for ATM MANAGEMENT SYSTEM has been drawn successfully.