# PROJECT REPORT

## 18CSC202J/ 18AIC203J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY

**(2018 Regulation)**

**II Year/ III Semester**

**Academic Year: 2022 -2023**

By


**V.SAI MANISH KUMAR(RA2111026010457)**

**P.PRANAV MURARI (RA2111026010454)**

Under the guidance of

**OM.PRAKESH**

**Assistant Professor**

**Department of Computational Intelligence**




**FACULTY OF ENGINEERING AND TECHNOLOGY**

**SCHOOL OF COMPUTING**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**Kattankulathur, Kancheepuram**


**NOVEMBER 2022**

## BONAFIDE

This is to certify that **18CSC202J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY project report** titled "**SMART FARM MONITORING SYSTEM"** is the bonafide work of **V.SAI MANISH KUMAR (RA2111026010457) AND P.PRANAV MURARI (RA2111026010454)**
who undertook the task of completing the project within the allotted time.


**Signature of the Guide**
**MR.OM PRAKASH**
Department of CINTEL,
SRM Institute of Science and Technology

**Signature of the II Year Academic Advisor**
**Professor and Head**
Department of CINTEL
SRM Institute of Science and Technology

## PROJECT ABSTRACT:

The ATM System is the project which is used to access their bank accounts in order to make cash withdrawals. Whenever the user need to make cash withdraws, they can enter their PIN number (personal identification number) and it will display the amount to be withdrawn in the form of 50's,100's and 500's. Once their withdrawn was successful, the amount will be debited in their account.

The ATM System is developed in VB.Net and back-end database as Ms-Access. VB.Net is the one of the powerful version of Framework and object oriented programming. Hence we use this software in our project.

The ATM will service one customer at a time. A customer will be required to enter ATM Card number, personal identification number (PIN) - both of which will be sent to the database for validation as part of each transaction. The customer will then be able to perform one or more transactions. Also customer must be able to make a balance inquiry of any account linked to the card.

The ATM will communicate each transaction to the database and obtain verification that it was allowed by the database. In the case of a cash withdrawal, a second message will be sent after the transaction has been physically completed (cash dispensed or envelope accepted). If the database determines that the customer's PIN is invalid, the customer will be required to re-enter the PIN before a transaction can proceed.

If a transaction fails for any reason other than an invalid PIN, the ATM will display an explanation of the problem, and will then ask the customer whether he/she wants to do another transaction.

The ATM will provide the customer with a printed receipt for each successful transaction, showing the date, time, machine location, type of transaction, account(s), amount, and ending and available balance(s) of the affected account ("to" account for transfers).

## 1. Introduction:

Automated Teller Machine enables the clients of a bank to have access to their account without going to the bank. This is achieved only by development the application using online concepts.

When the product is implemented, the user who uses this product will be able to see all the information and services provided by the ATM, when he enters the necessary option and arguments. The product also provides services like request for cheques, deposit cash and other advanced requirement of the user. The data is stored in the database and is retrieved whenever necessary. The implementation needs ATM machine hardware to operate or similar simulated conditions can also be used to successfully use the developed product.

To develop this ATM system the entire operation has been divided into the following

step:

1. verification process
2. language, service and account selection
3. Banking services
4. Transactions
5. Special services

The program is designed in such a way that the user has to card and pin number. Once verified, he is provided a menu and he/she had to enter the option provided in the menu. For example, when the user wants to view the list of payment history than he/she had to enter the option for payment history provided in the main menu. When the option is entered alone with the respective argument, then the payment history is displayed on the screen.

The user also must be given option to browse through the pages like previous page, next page, etc. The user may experience a delay in retrieving or viewing the data, when there are many users logged on to the same bank branch system.

## 1.2. Problem definition :

The system mainly used by the bank clients. When a client comes to ATM centre to update and delete their account. It reduces the time consumption and lot of paperwork. For any single operation it involves numerous references and updating also takes subsequent changes in other places.

## 1.3. Evidence of problem definition:

Now-a-days every one very busy in their work. So they feel that the job must be easier so the system is used to reduce their work which is done in the ATM system. Instead of keeping lots of paper into a record or file and it may be missed somewhere so, this system help to keep the record of the customer it also keeps the details of the customer. It is also easy to access.

## 1.4. Proposed solution:

The system customer transactions, satisfies the requirements of the existing system in full-fledged manner. Through this system, customer can make fast transactions and view the last transactions easily.

## 1.5. Scope:

It can be implemented in ATM machine by owner of bank or in charge of branch.

It is easy to learn the task.

## 1.6. Objectives:

Our main objective is to speed up the transactions done by customers. No manual transactions needed generally. The second objective is to save the time which is very important now-a-days.

It will include other objectives such as:

To render accurate services to customer.

The reduction of fraudulent activities

To achieve speedy processing of customer data

## 2. System Requirements:

### 2.2. Hardware Requirements:

Processor :- Intel Pentium 4 or Later or Compatible

Hard Disk:- 410GB or more

RAM: 1GB or more

Printer :- Any

Monitor:- SVGA Color Monitor (Touch Screen or Simple)

Pointing Device :- Touch Pad or Keys

### 2.3. Software Requirements:

Operating System :- Microsoft Windows XP or Later or Equivalen

Front End :- Visual Basic 6.0

Back End :- Oracle 8i

## 3. System Analysis :

### 3.2. Study of current/Existing system:

In the manual system, firstly the bank manager and its staff have to manage information regarding the accounts and transaction of all the customers manually. Doing this manual transaction was really tedious job. Secondly information regarding accounts and transactions of customers were to be maintained. This process is time consuming and it requires a great manual effort.

**Disadvantages:**

More time is consumed.

More hard work to maintain all records.

Bulk of paper is to be searched for a single search.

### 3.3. Feasibility study:

#### 3.3.1. Technical feasibility:

The system is being developed in Visual Basic 6.0. It provides comprehensive function to make it user friendly. The data entry and report generation is also made easy. Backup and restore of the database facility are also provided. It also provides easy retrieval of data. The machine configuration also supports this software.

#### 3.3.2. Social feasibility:

As this system is user friendly and flexible some problems will also be solved which employee may be facing when using existing system. So we can say that system is socially feasible.

#### 3.3.3.Economical feasibility:

The cost of converting from manual system to new automatic computerized system is not probably more. For construction of the new system, the rooms and its facilities are available so it does not require any extra resource, only the software requirement is there.

#### 3.3.4. Operation feasibility:

Since the system is being in user friendly way, the new customers within a few time can master it.

### 3.4. Design of new proposed system (UML):

This system provides paperless maintenance. Initially a cashier or an clerk can be appointed to do all the transaction and update and maintain records. In the new system the customer himself can do all the transaction and the computerized system automatically updates and maintains the records.

**Advantages:**

Less effort to complete transaction.

Less time required.

No need to maintain the bulk of papers.


**KEY FEATURES OF THE ATM MACHINE PROJECT:**

The program will do some tasks mentioned below to operate the ATM Machine:

-The C++ Program can Display the ATM Transaction.

- Pin verification system to login to ATM Machine.

-We can check balance from this ATM Machine project.

-We can Withdraw Cash from ATM Machine.

-User can Deposit Cash from ATM Machine.

**The procedures included in the process syntax structure:**

**ATM**

-Initially, we need to adjust or set the ATM pin along with the amount in random numbers.

-Now take ATM pin qs the input.

-If the provided input pin is identical to the adjusted pin, then we can furtherperform additional operations.

-For executing the operations like checking balance, depositing cash, withdrawing cash and exit, we will implement the switch statement.

-we use while loop to terminate or resume the atm procedure.

-The withdrawal process will be processed only if the entered pin number is correct and only if the bank account is with sufficient cash. Or else we can go for another transaction or check balance through the options.

-When we deposit money, automatically the new balance will be shown.

-From the third option user can check his/her balance when they perform any actions like deposition or withdrawal through ATM transaction.

*****C++ program to implement the ATM Management System*****

**INTPUT PROGRAM:-**

```
#include <iostream>
#include <stdlib.h>
#include <string.h>
using namespace std;
class Bank {

	// Private variables used inside class
private:
	string name;
	long long accnumber;
	char type[10];
	long long amount = 0;
	long long tot = 0;

	// Public variables
public:
	// Function to set the person's data
	void setvalue()
	{
		cout << "Enter name\n";
		cin.ignore();

		// To use space in string
		getline(cin, name);
```

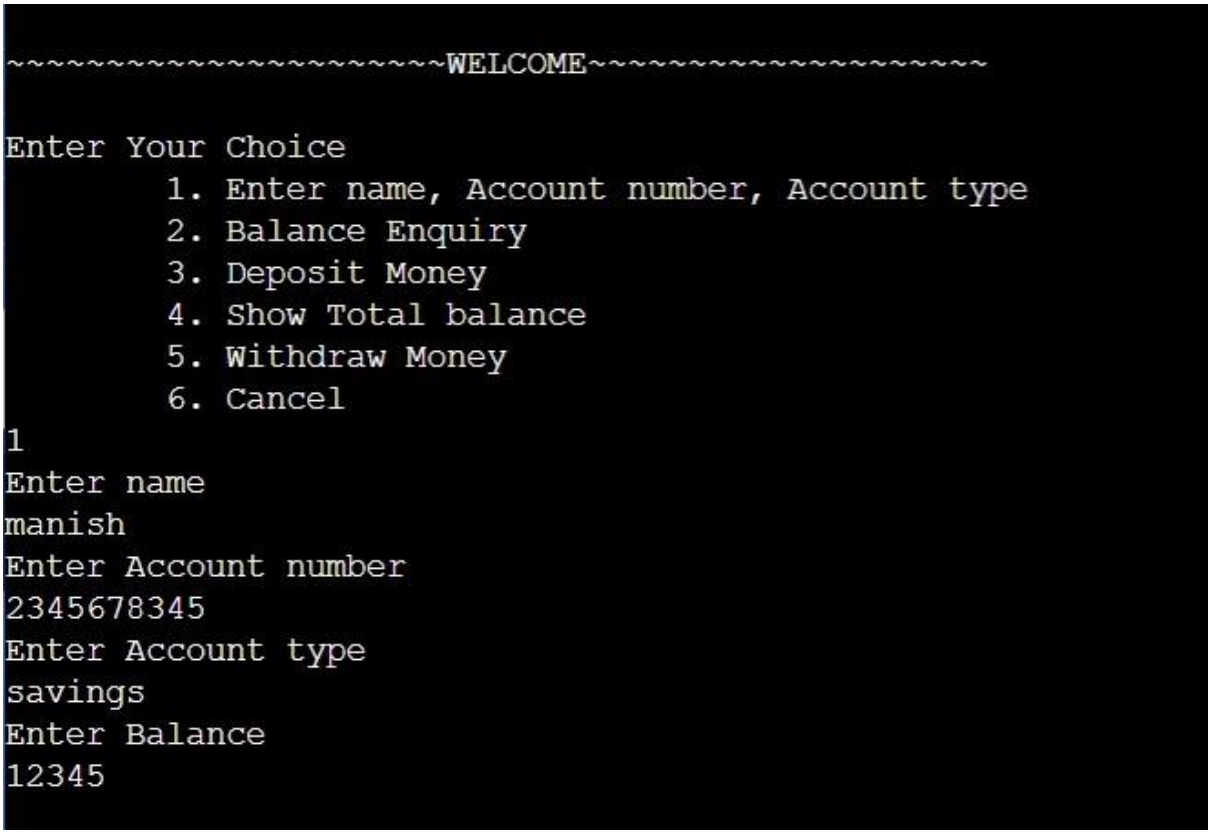
```
        cout << "Enter Account number\n";
        cin >> accnumber;
        cout << "Enter Account type\n";
        cin >> type;
        cout << "Enter Balance\n";
        cin >> tot;
}

// Function to display the required data
void showdata()
{
        cout << "Name:" << name << endl;
        cout << "Account No:" << accnumber << endl;
        cout << "Account type:" << type << endl;
        cout << "Balance:" << tot << endl;
}

// Function to deposit the amount in ATM
void deposit()
{
        cout << "\nEnter amount to be Deposited\n";
        cin >> amount;
}

// Function to show the balance amount
void showbal()
{
```

```
            tot = tot + amount;
            cout << "\nTotal balance is: " << tot;
        }

        // Function to withdraw the amount in ATM
        void withdrawl()
        {
            int a, avai_balance;
            cout << "Enter amount to withdraw\n";
            cin >> a;
            avai_balance = tot - a;
            cout << "Available Balance is" << avai_balance;
        }
};

// Driver Code
int main()
{
        // Object of class
        Bank b;

        int choice;

        // Infinite while loop to choose
        // options everytime
        while (1) {
            cout << "\n~~~~~~~~~~"
```

```
        << "~~~~~~~~~~"
        << "~~WELCOME~~~~~~~"
        << "~~~~~~~~~~"
        << "~~~\n\n";
cout << "Enter Your Choice\n";
cout << "\t1. Enter name, Account "
        << "number, Account type\n";
cout << "\t2. Balance Enquiry\n";
cout << "\t3. Deposit Money\n";
cout << "\t4. Show Total balance\n";
cout << "\t5. Withdraw Money\n";
cout << "\t6. Cancel\n";
cin >> choice;

// Choices to select from
switch (choice) {
case 1:
        b.setvalue();
        break;
case 2:
        b.showdata();
        break;
case 3:
        b.deposit();
        break;
case 4:
        b.showbal();
```

```
                break;
            case 5:
                b.withdrawl();
                break;
            case 6:
                exit(1);
                break;
            default:
                cout << "\nInvalid choice\n";
            }
        }
}
```

OUTPUT:-



```
Enter Your Choice
        1. Enter name, Account number, Account type
        2. Balance Enquiry
        3. Deposit Money
        4. Show Total balance
        5. Withdraw Money
        6. Cancel
2
Name:manish
Account No:2345678345
Account type:savings
Balance:12345
```

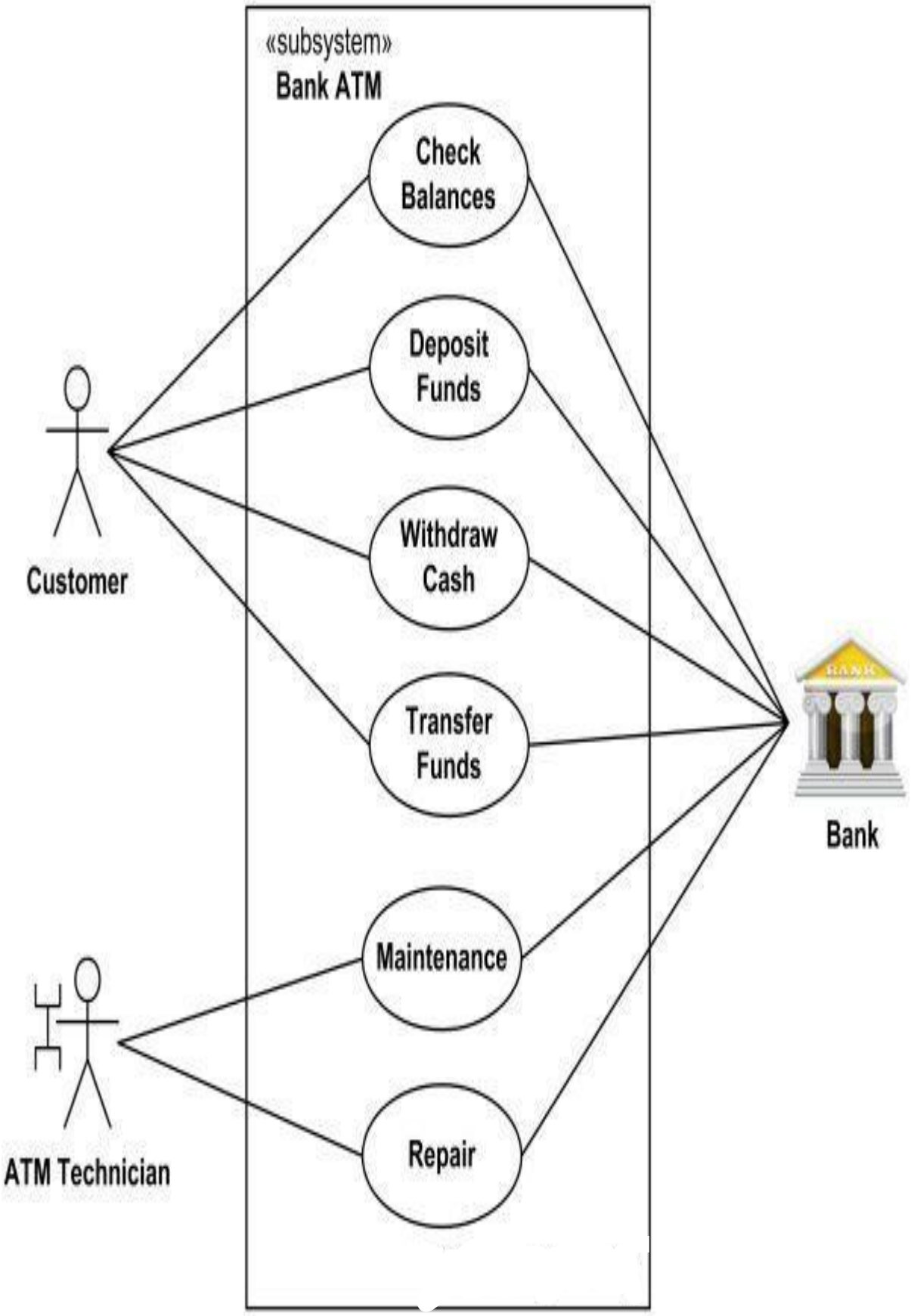
```
Enter Your Choice
        1. Enter name, Account number, Account type
        2. Balance Enquiry
        3. Deposit Money
        4. Show Total balance
        5. Withdraw Money
        6. Cancel
3

Enter amount to be Deposited
400
```

```
Enter Your Choice
        1. Enter name, Account number, Account type
        2. Balance Enquiry
        3. Deposit Money
        4. Show Total balance
        5. Withdraw Money
        6. Cancel
4

Total balance is: 12745
```

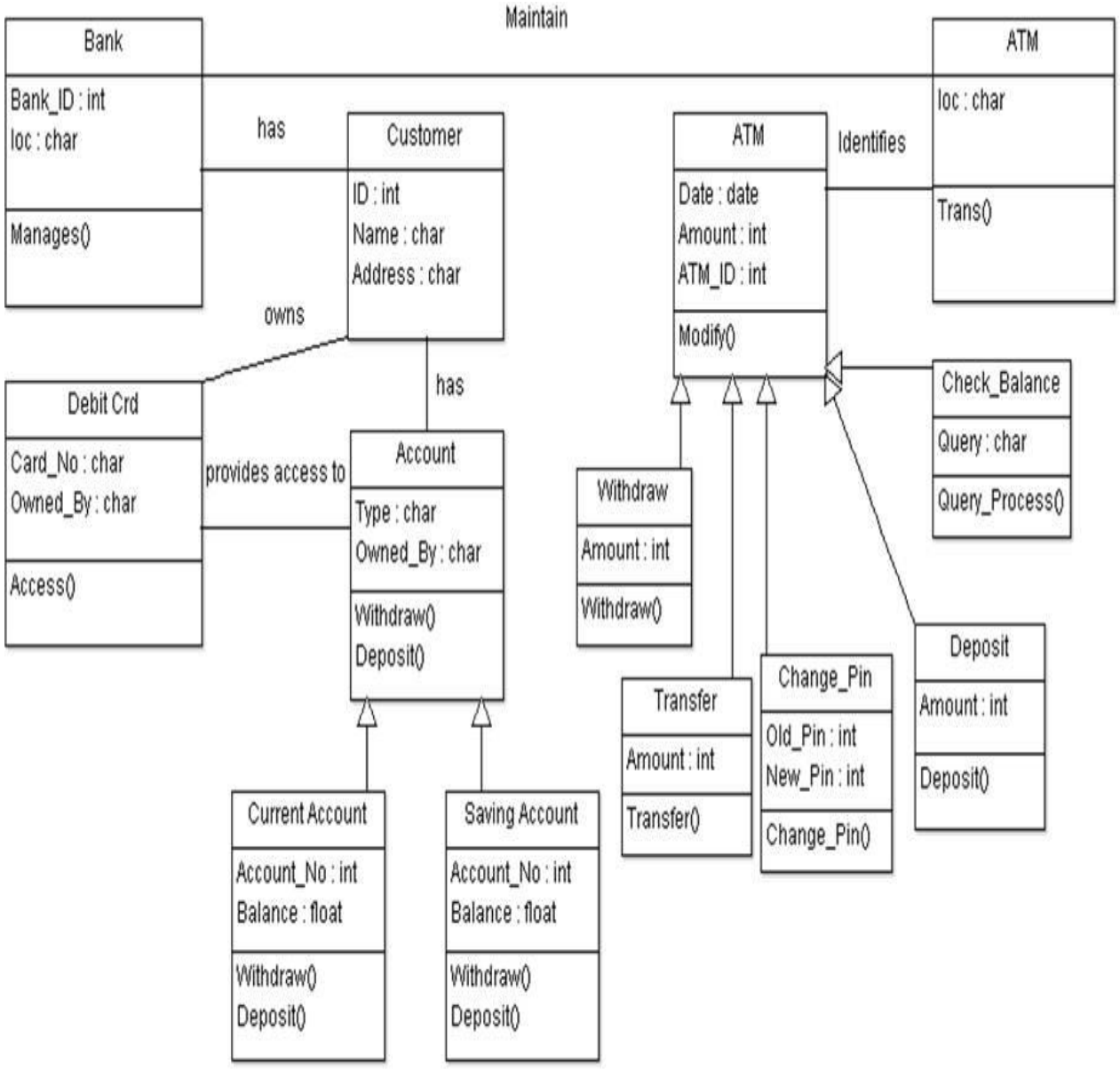
```
Enter Your Choice
        1. Enter name, Account number, Account type
        2. Balance Enquiry
        3. Deposit Money
        4. Show Total balance
        5. Withdraw Money
        6. Cancel
5
Enter amount to withdraw
12345
Available Balance is: 400
```

```
Enter Your Choice
        1. Enter name, Account number, Account type
        2. Balance Enquiry
        3. Deposit Money
        4. Show Total balance
        5. Withdraw Money
        6. Cancel
6


...Program finished with exit code 1
Press ENTER to exit console.
```

# USE CASE DIAGRAM


«subsystem»
Bank ATM
Check Balances
Deposit Funds
Withdraw Cash
Transfer Funds
Maintenance
Repair
Customer
ATM Technician
Bank

# CLASS DIAGRAM




dig.Class Diagram for ATM System

## Class Checklist

| Sl. No. | Parameters | Class Checklist(Y/N) | | | | | |
|---|---|---|---|---|---|---|---|
| | | Customer | Transaction | Personal Banker | Transaction history | Budget | Wallet |
| 1. | Put common terminology for names | Yes | Yes | Yes | Yes | Yes | Yes |
| 2. | Choose complete singular nouns over class names | Yes | Yes | Yes | Yes | Yes | Yes |
| 3. | Names operations with a strong verb | Yes | Yes | Yes | Yes | Yes | Yes |
| 4. | Names attributes with a domain-based noun | Yes | Yes | Yes | Yes | Yes | Yes |
| 5. | Never show classes with just two compartments | Yes | Yes | Yes | Yes | Yes | Yes |
| 6. | Label uncommon class compartments | Yes | Yes | Yes | Yes | Yes | Yes |
| 7. | Include an ellipsis (...) at the end of incomplete lists | Yes | Yes | Yes | Yes | Yes | Yes |
| 8. | List static operations/attributes before instance operations/attributes | Yes | Yes | Yes | Yes | Yes | Yes |
| 9. | List operations/attributes in decreasing visibility | Yes | Yes | Yes | Yes | Yes | Yes |
| 10. | For parameters that are objects only list their type | Yes | Yes | Yes | Yes | Yes | Yes |
| 11. | Develop consistent method signatures | Yes | Yes | Yes | Yes | Yes | Yes |
| 12. | Avoid stereotypes implied by language naming conventions | Yes | Yes | Yes | Yes | Yes | Yes |
| 13. | Indicate exceptions in an operation's property string. | Yes | Yes | Yes | Yes | Yes | Yes |

| Sl. No. | Parameters | Class Checklist(Y/N) | | | | | |
|---|---|---|---|---|---|---|---|
| | | Customer | Transaction | Personal Banker | Transaction history | Budget | Wallet |
| 1. | Interface definitions must reflect implementation language constraints | Yes | Yes | Yes | Yes | Yes | Yes |
| 2. | Name interfaces according to language naming conventions | Yes | Yes | Yes | Yes | Yes | Yes |
| 3. | Apply "Lollipop" notations to indicate that a class realizes an interface | Yes | Yes | Yes | Yes | Yes | Yes |
| 4. | Define interfaces separately from your classes | Yes | Yes | Yes | Yes | Yes | Yes |
| 5. | Do not model the operations and attributes of an interface in your classes | Yes | Yes | Yes | Yes | Yes | Yes |
| 6. | Consider an interface to be a contract | Yes | Yes | Yes | Yes | Yes | Yes |

| Sl. No. | Parameters | Class Checklist(Y/N) | | | | | |
|---|---|---|---|---|---|---|---|
| | | Customer | Transaction | Personal Banker | Transaction history | Budget | Wallet |
| 1. | Put in the sentence rule for inheritance | Yes | Yes | Yes | Yes | Yes | Yes |
| 2. | Put subclasses below super classes | Yes | Yes | Yes | Yes | Yes | Yes |
| 3. | Ensure that you are aware of data-based interface | Yes | Yes | Yes | Yes | Yes | Yes |
| 4. | A subclass must inherit everything | Yes | Yes | Yes | Yes | Yes | Yes |

# SEQUENCE DIAGRAM

**Explanation:**

The sequence diagram is used primarily to show the interactions between objects in the sequential order that those interactions occur. The main purpose of a sequence diagram is to define event sequences that result in some desired outcome.

The diagram conveys this information along the horizontal and vertical dimensions: the vertical dimension shows, top-down, the time sequence of messages/calls as they occur, and the horizontal dimension shows, left to right,the object instances that the messages are sent to.Lifelines When drawing a sequence diagram, lifeline notation elements are placed across the top of the diagram. Lifelines represent either roles or object instances that participate in the sequence being modeled.

Here we have used five lifelines namely user, login success, ATM management, order management, and bill management.Messages

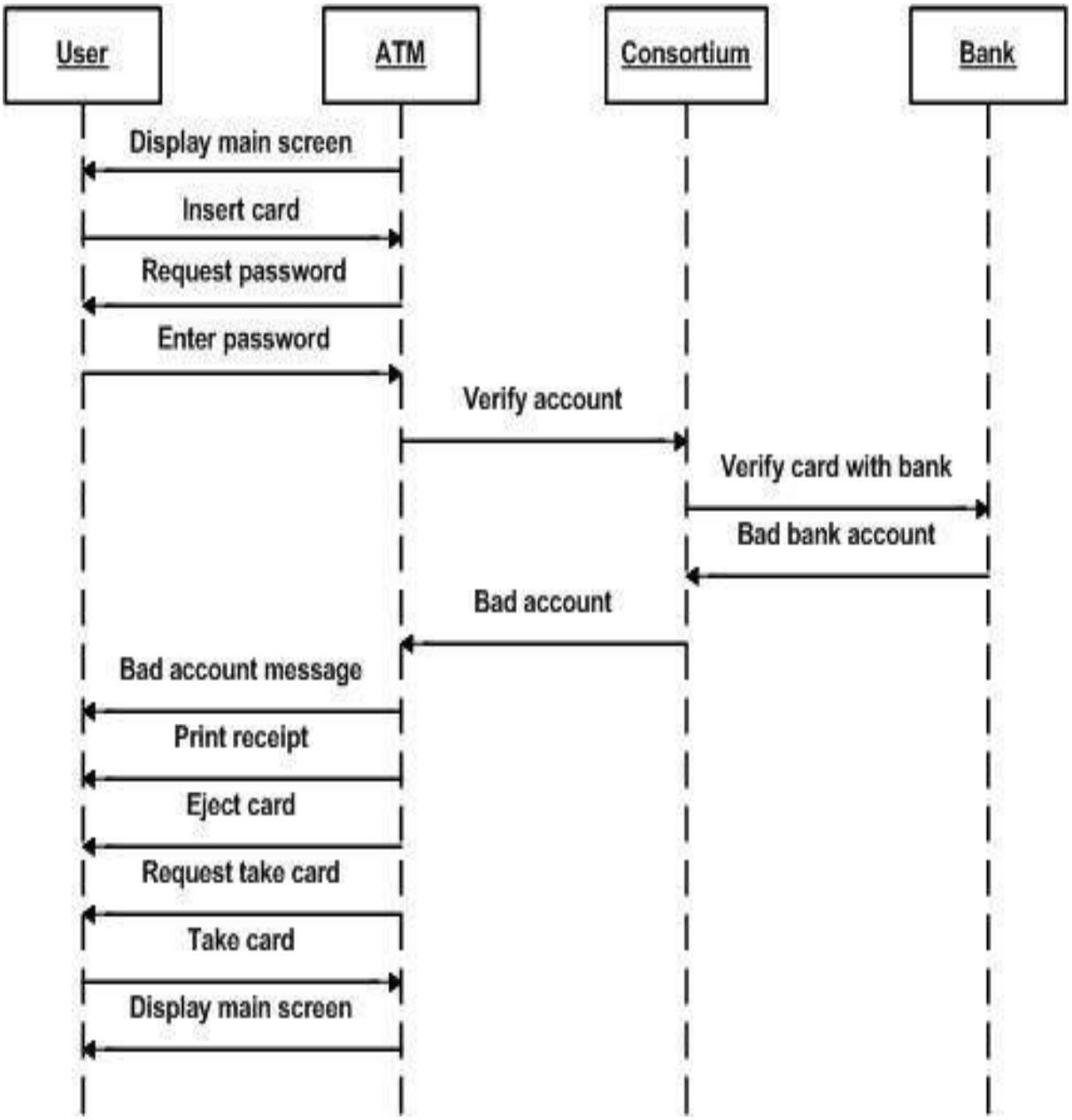
The first message of a sequence diagram always starts at the top and is typically located on the left side of the diagram for readability. There are many kinds ofmessages like synchronous, asynchronous, self, etc.

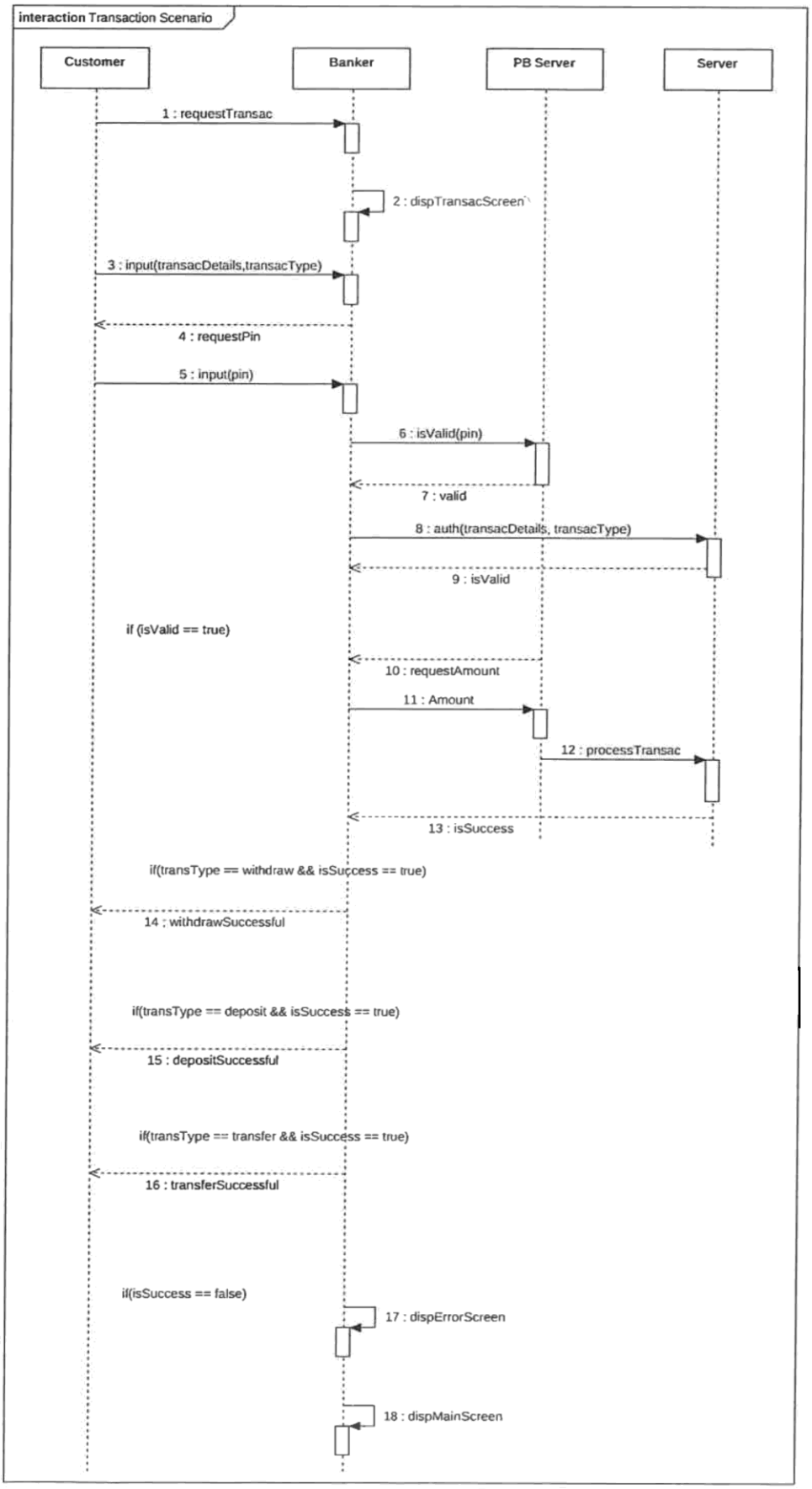
Sequence diagrams show an overview of the interactions between objects in a system over time. Also called event diagrams or event scenarios, sequence diagrams depict the sequential events that occur and the processes that are completed.

Sequence diagrams descend from top to bottom to show the sequence of interactions. The objects, or actors, are shown at the top of the diagram as named rectangles. Each object has a vertical lifeline descending from it to show the different processes that exist simultaneously in the system during the time period captured by the diagram.

The interactions between them are represented by horizontal arrows that can go from left to right and vice versa. These arrows actually depict messages in the system and the style of arrow tells you whether the messages are synchronous, asynchronous, or whether the message creates or deletes objects in the system. The arrows descend in the order in which the events they represent occur.

You can depict complex interactions between multiple objects in the sequence diagram by using sequence fragments. These act as frames for parts of an interaction. The fragment is marked by an operator in the top-left corner.


User
ATM
Consortium
Bank
Display main screen
Insert card
Request password
Enter password
Verify account
Verify card with bank
Bad bank account
Bad account
Bad account message
Print receipt
Eject card
Request take card
Take card
Display main screen



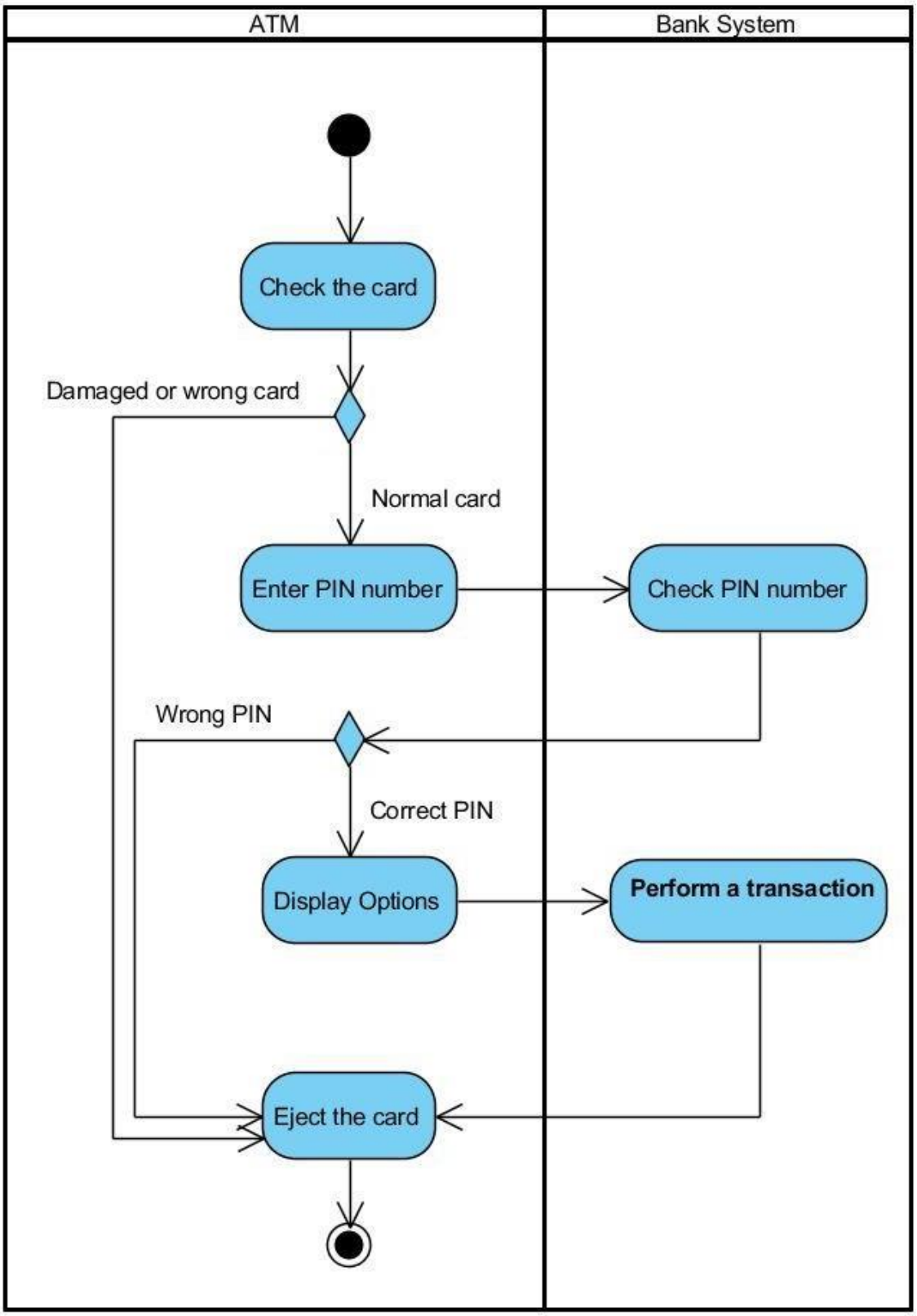

SEQUENCE DIAGRAM

TRANSACTION SCENARIO .

## 5.4. Sequence Checklist

| S. No. | Sequence Checklist | Check(Y/N) |
|---|---|---|
| 1 | Messages are from Left-To-Right | Yes |
| 2 | Actors named consistently with Use Case Diagram | Yes |
| 3 | Classes named consistently with class Diagram | Yes |
| 4 | Human and Organisation actors on left most side | Yes |
| 5 | Reactive system actors on right most side | Yes |
| 6 | Proactive system actors on left most side | Yes |
| 7 | Message names beside arrowhead justified | Yes |
| 8 | Do not return value when it is obvious | Yes |
| 9 | Use return value only when you need to refer it elsewhere | Yes |

# ACTIVITY DIAGRAM


ATM
Bank System
Check the card
Damaged or wrong card
Normal card
Enter PIN number
Check PIN number
Wrong PIN
Correct PIN
Display Options
Perform a transaction
Eject the card

**ACTIVITY DIAGRAM ON ATM:**

**Explanation:**

An activity diagram visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram. Activity diagram sare often used in business process modeling. They can also describe the steps in a use case diagram.

TOOLS:

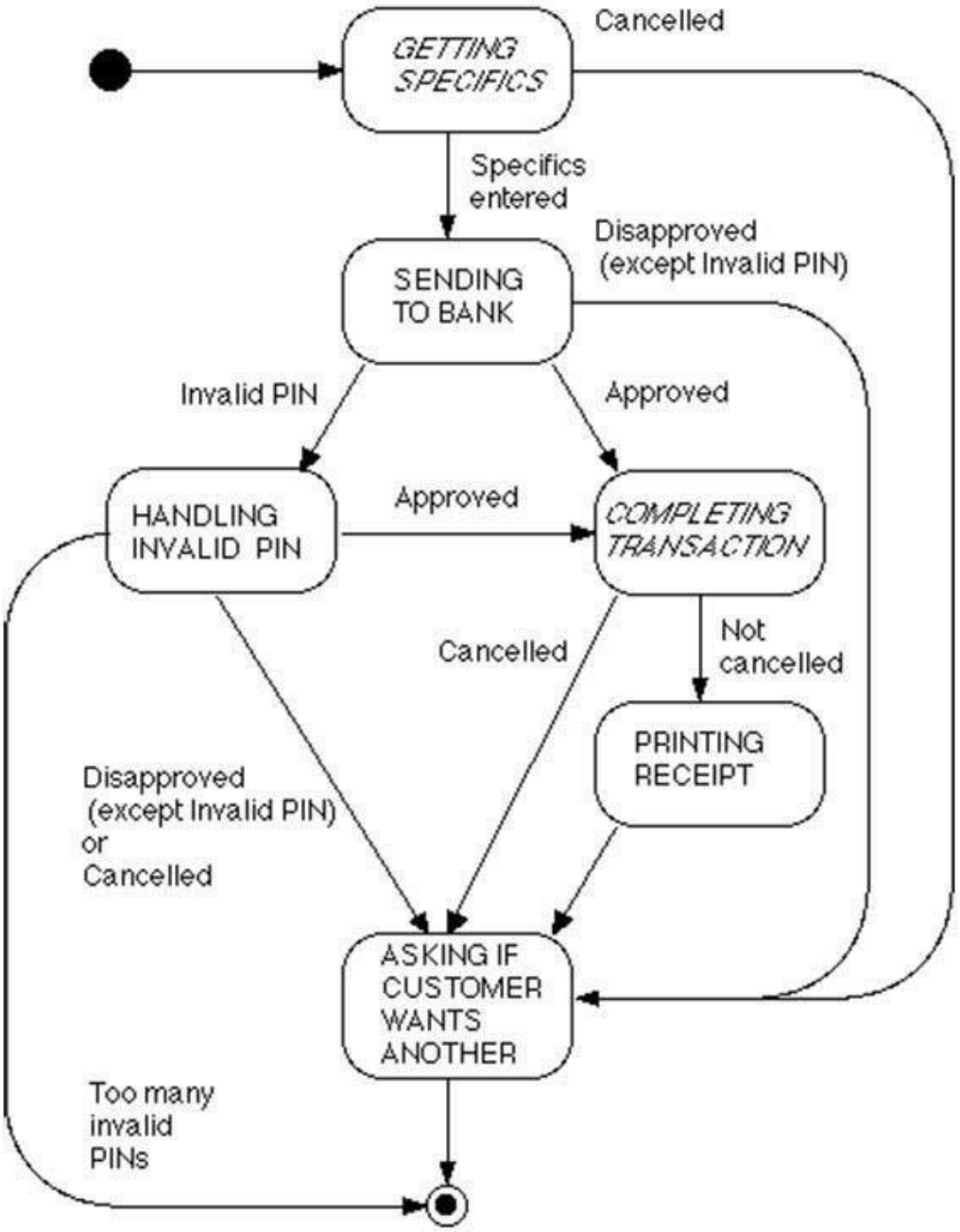
- INITIAL
- ACTION
- FINAL
- CONTROL FLOW
- MERGE
- DECISION
- FORK NODE
- JOIN NODE

Starting from the beginning we used actions which are connected with control flow and used a decision tool for taking decisions and a fork node to divide one action into two and join nodes to combine 2 actions from the fork node and finallywe ended the diagram with Final tool.

REPORT:

Activity diagrams describe the steps performed in a UML use case. Illustrate a business process or workflow between users and the system. Simplify and improve any process by clarifying complicated use cases. Model software architecture elements, such as method, function, and operation. We successfully performed and executed the ACTIVITY DIAGRAM for ATM.

# STATE CHART DIAGRAM


GETTING SPECIFICS
Cancelled
Specifics entered
SENDING TO BANK
Disapproved (except Invalid PIN)
Invalid PIN
Approved
HANDLING INVALID PIN
Approved
COMPLETING TRANSACTION
Cancelled
Not cancelled
PRINTING RECEIPT
Disapproved (except Invalid PIN) or Cancelled
ASKING IF CUSTOMER WANTS ANOTHER
Too many invalid PINs

EXPLANATION:

State diagrams enable you to describe the behavior of objectsduring their entire life span. In addition the different states andstate changes as well as events causing transactions can bedescribed .

It depicts the whole process of ATM. NOTATIONS USED :

- initial state
- state
- final state
- transition

REPORT:

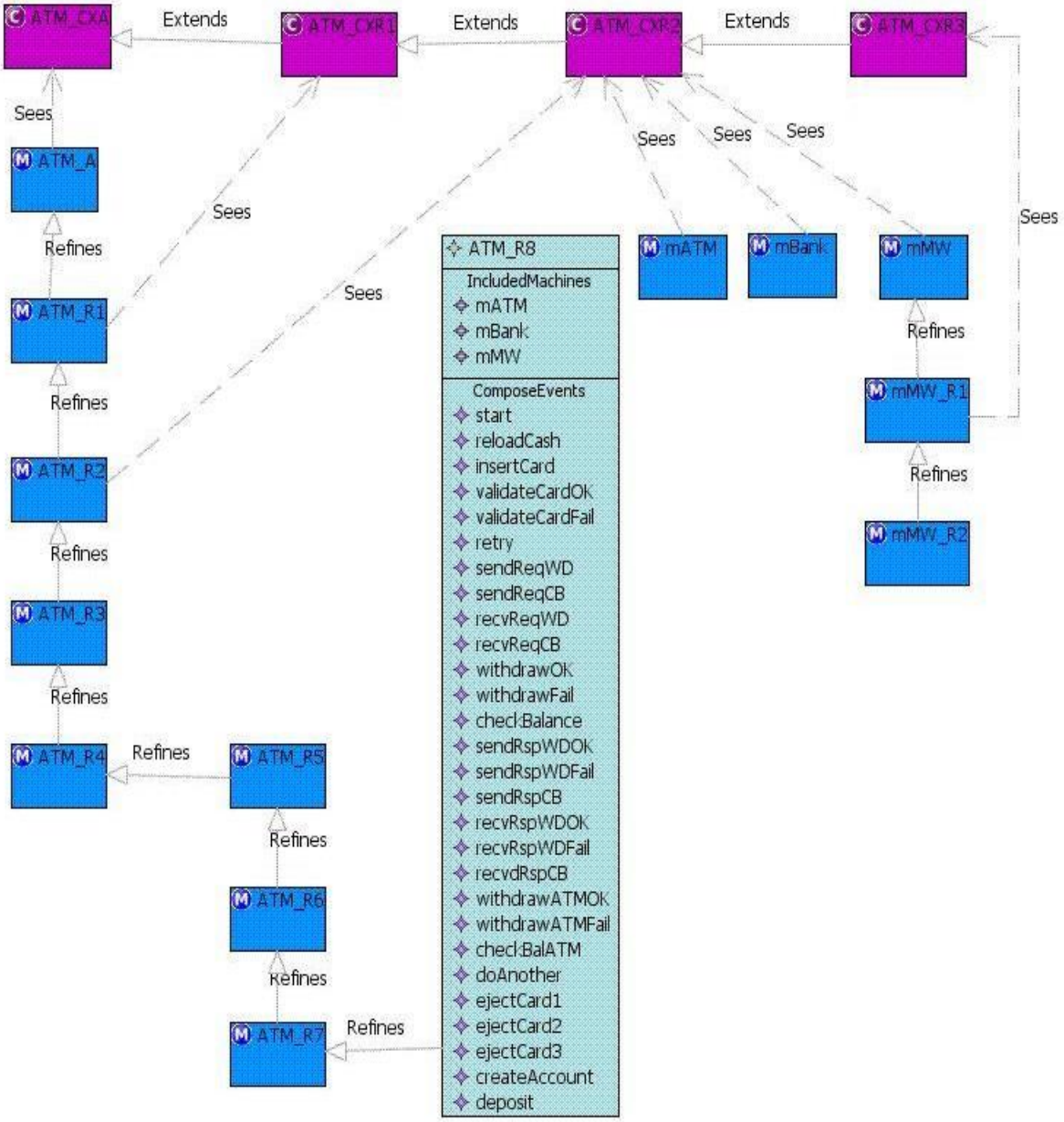
The above state chart diagram depicts the whole ATM process.

In general, illustrating use case scenarios in a business context describing how an object moves through various states within a lifetime Hence we successfully performed STATE CHART DIAGRAM ON ATM using star UML.

When the customer inserts the bank or credit card in the ATM's card reader, the entry action i.e readcard is performed by the ATM machine. If the card is not valid then the machine will perform exit action. After the card is being read successfully, the ATM machine will ask for Pin. Then the customer enters the pin and ATM machine then reads pin.

If the pin entered is not valid then machine will perform exit action. If the pin entered is valid, then the machine further process towards transaction. After successful transaction, machine undergoes the exit action i.e., ejectcard that discharges the customer's card.

# PACKAGE DIAGRAM


ATM_CXA
Extends
ATM_CXR1
Extends
ATM_CXR2
Extends
ATM_CXR3
Sees
ATM_A
Refines
ATM_R1
Refines
ATM_R2
Refines
ATM_R3
Refines
ATM_R4
Refines
ATM_R5
Refines
ATM_R6
Refines
ATM_R7
Refines
Sees
Sees
Sees
Sees
Sees
Sees
ATM_R8
IncludedMachines
mATM
mBank
mMW
ComposeEvents
start
reloadCash
insertCard
validateCardOK
validateCardFail
retry
sendReqWD
sendReqCB
recvReqWD
recvReqCB
withdrawOK
withdrawFail
checkBalance
sendRspWDOK
sendRspWDFail
sendRspCB
recvRspWDOK
recvRspWDFail
recvdRspCB
withdrawATMOK
withdrawATMFail
checkBalATM
doAnother
ejectCard1
ejectCard2
ejectCard3
createAccount
deposit
mATM
mBank
mMW
Refines
mMW_R1
Refines
mMW_R2

**Package Diagram ATM System :**

The package diagram shows how the various classes are grouped into packages. There are two "top-level" classes - ATMMain and ATMApplet - which allow the system to be run (respectively) as an application or as an Applet. (Only one of the two would be instantiated in any particular use of the system.)

Each of these classes, in turn, depends on the package atm which contains the class ATM that represents the system as a whole, and the class Session that represents one session. ATM depends on Session, and vice versa - since the ATM creates Sessions, and each Session, in turn, uses the ATM to interact with the customer.

The subpackage transaction contains the classes used to represent individual transactions that a customer initiates. The class Session depends on the transaction package because it creates individual transaction objects. These, in turn, again depend on the ATM to interact with the customer.

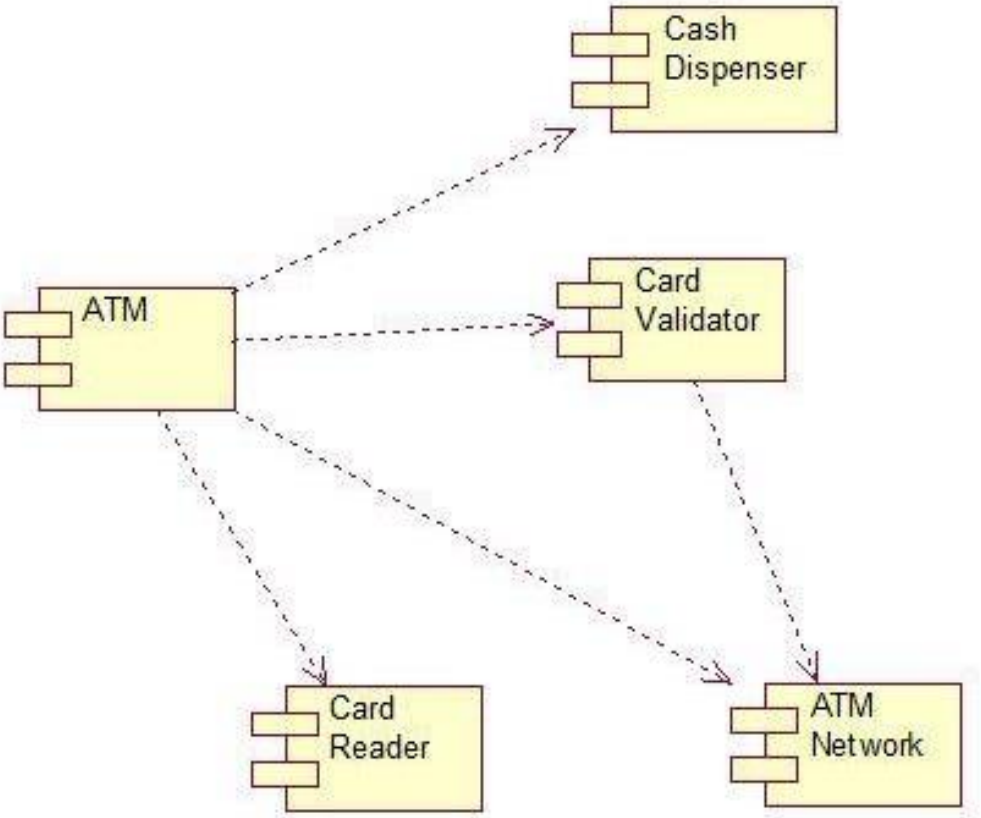
The subpackage physical contains the classes that represent the various physical components of the ATM. For the purposes of this simulation, these are simulated by a GUI. A real ATM would have quite different classes in this package - classes that actually control its physical components.

The class ATM makes use of these components, and Session and the various kinds of transaction gain access to them through ATM to actually perform the needed operations.

Finally, the package banking contains classes that represent the banking enterprise itself and the information communicated back and forth between the ATM and the bank - i.e. classes which might be the same in a totally different implementation of an ATM that interacts with the same bank.

This is, of course, a simulation. However, most of the code that is specific to the simulation resides in the package physical, plus the two top-level classes. Presumably, the other classes and packages might be similar in a real system.

# COMPONENT DIAGRAM


Cash
Dispenser
ATM
Card
Validator
Card
Reader
ATM
Network

## ➡COMPONENT DIAGRAM OF ATM:

The component diagram of ATM system is used to show how the system components work together to make the ATM operate correctly. A component diagram shows how the software's parts are organized and how they depend on each other. This diagram gives a high-level look at the parts of a system.

Components of an ATM component diagram can be part of software or hardware. They could be a database, a user interface, or something else that helps the ATM system work.

The ATM system is specialized software that aids in managing a bank account or holder's funds simply. Users can check account balances, make cash withdrawals or deposits, print a record of account activities or transactions, and even purchase stamps via the system.

Banks have complete control over their ATM networks with the help of the ATM system. With warnings through SMS, email, and mobile app, the unified 'one-look' dashboard provides a snapshot of the state of health of all ATMs and the gadgets inside them.

These statements were collected as the information concepts in developing the component diagram. The concept formulated will be applied to the component diagram illustration.

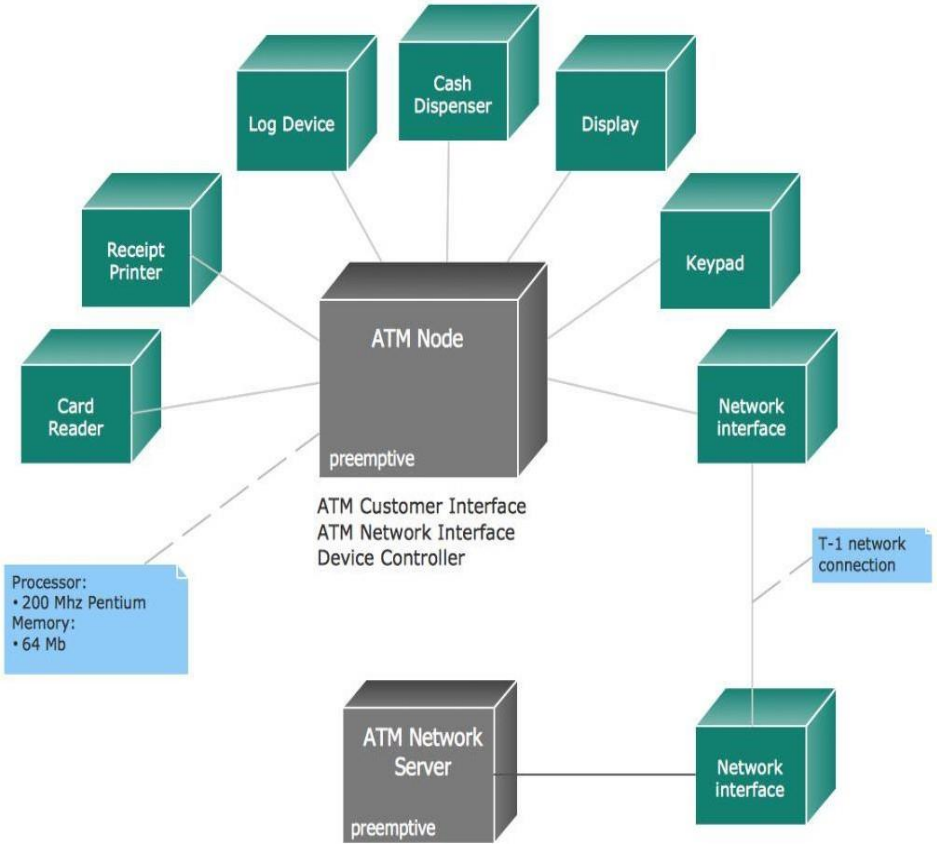
The ATM system is specialized software that aids in managing a bank account or holder's funds simply. Users can check account balances, make cash withdrawals or deposits, print a record of account activities or transactions, and even purchase stamps via the system.

Customers can utilize the ATM system to conduct self-service transactions such as deposits, cash withdrawals, bill payments, and account transfers. The bank where the account is stored, the ATM operator, or both, frequently levy cash withdrawal fees.

Banks have complete control over their ATM networks with the help of the ATM system. With warnings through SMS, email, and mobile app, the unified 'one-look' dashboard provides a snapshot of the state of health of all ATMs and the gadgets inside them.

These statements were collected as the information concepts in developing the component diagram. The concept formulated will be applied to the

# DEPLOYMENT DIAGRAM


Log Device
Cash Dispenser
Display
Receipt Printer
Keypad
ATM Node
preemptive
Card Reader
Network interface
ATM Customer Interface
ATM Network Interface
Device Controller
Processor:
• 200 Mhz Pentium
Memory:
• 64 Mb
T-1 network connection
ATM Network Server
preemptive
Network interface

## ➡DEPLOYMENT DIAGRAM ON ATM:

A deployment diagram in the Unified modeling language models the physical

deployment of artifacts on nodes. To describe website, for example, a deployment diagram would show what hardware components ("nodes") exist , what software components ("artifacts") run on each node and how the different pieces are connected The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have subnodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

There are two types of Nodes:

1. Device Node

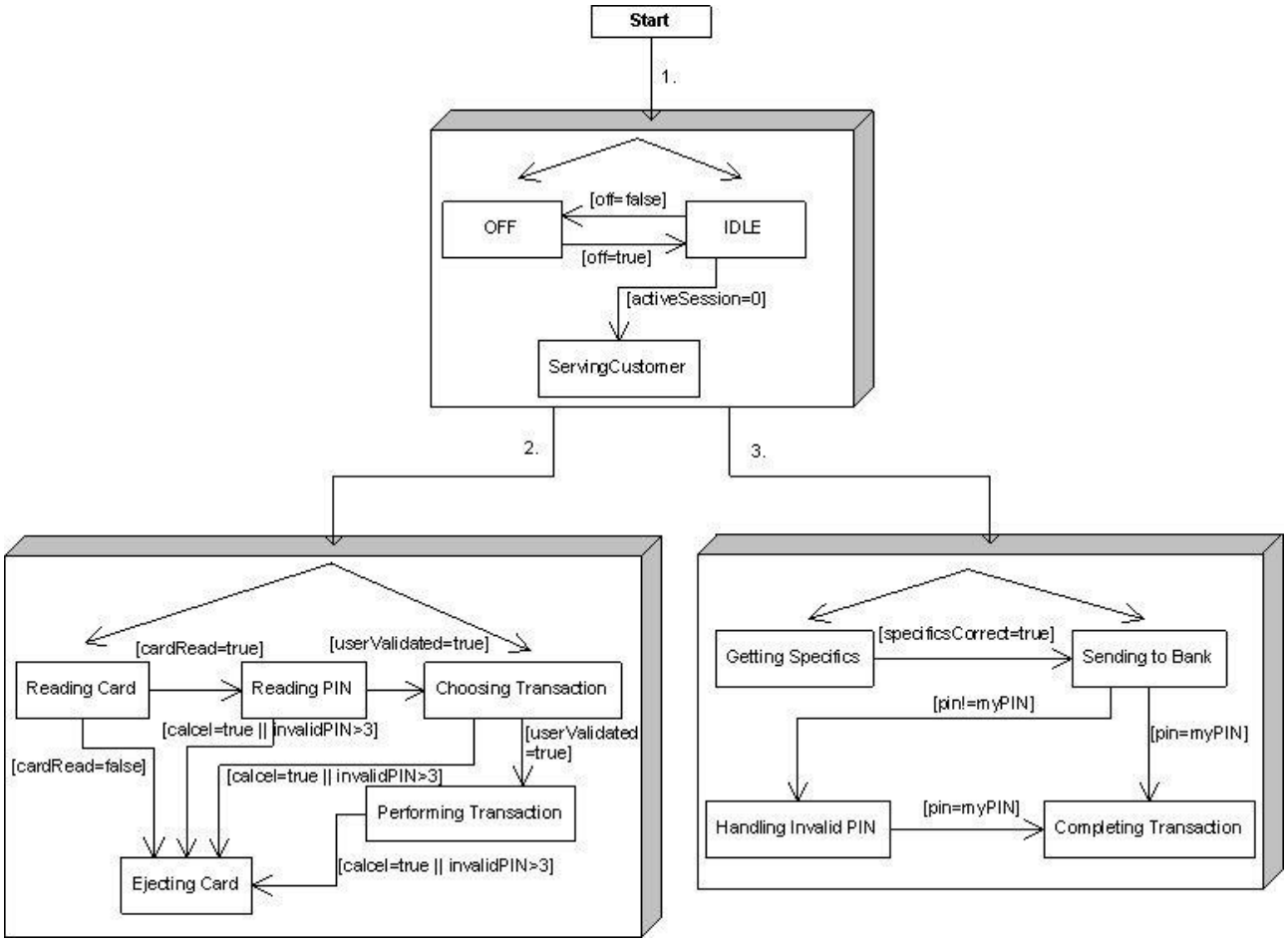
2. Execution Environment Node

Device nodes are physical computing resources with processing memory and services to execute software, such as typical computers or mobile phones. An execution environment node (EEN) is a software computing resource that runs within an outer node and which itself provides a service to host and execute another executable software elements.

Deployment diagrams are used to visualize the hardware processors/ nodes/ devices of a system, the links of communication between them and the placement of software files on that hardware.A deployment diagram is a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them.

Deployment diagrams are typically used to visualize the physical hardware and software of a system. Using it you can understand how the system will be physically deployed on the hardware. Deployment diagrams help model the hardware topology of a system compared to other UML diagramtypes which mostly outline the logical components of a system.

# COMMUNICATION DIAGRAM


Start
1.
OFF
IDLE
[off=false]
[off=true]
[activeSession=0]
ServingCustomer
2.
3.
Reading Card
Reading PIN
Choosing Transaction
[cardRead=true]
[userValidated=true]
[cardRead=false]
[calcel=true || invalidPIN>3]
[userValidated =true]
[calcel=true || invalidPIN>3]
Performing Transaction
[calcel=true || invalidPIN>3]
Ejecting Card
Getting Specifics
[specificsCorrect=true]
Sending to Bank
[pin!=myPIN]
[pin=myPIN]
Handling Invalid PIN
[pin=myPIN]
Completing Transaction

## ➡ COLLABORATION DIAGRAM ON BANKING:

COLLABORATION DIAGRAM named as COMMUNICATION DIAGRAM between two objects how messages can be communicated can be clearly shown on a communication or collaboration diagram Collaboration diagrams messages are always coming with a number In the communication diagram elements are organized with space They are very useful for visualizing relations between objects that collaborate with each other to perform a specific task This helps to determine the accuracy of a static model Messages that objects send to themselves are loops.

## ➡ APPLICATIONS OF COLLABORATION DIAGRAMS:

i) MODELING COLLABORATIONS

ii) Visualizing the complex logic behind an operation

iii) Exhibiting many alternative scenarios for the same use case

## ➡ Purpose of Communication Diagram

- Model message passing between objects or roles that deliver the functionalities of use cases and operations
- Model mechanisms within the architectural design of the system
- Capture interactions that show the passed messages between objects and roles within the collaboration scenario
- Model alternative scenarios within use cases or operations that involve the collaboration of different objects and interactions
- Support the identification of objects (hence classes), and their attributes (parameters of message) and operations (messages) that participate in use Cases

## ➡ Conclusion:-

Banking these days has become an essential part of our lives. Many individuals withdraw, deposit or transfer funds on a daily basis. To ease this process up, we have made the Personal Banker application which will make this tedious process a bit easier by directly linking your accounts and providing the customer with a one touch transaction option.