ATM MACHINE

ATM or rather Automated Teller Machine is also called as ANY TIME MONEY by many. The ATM system is connected to banks.

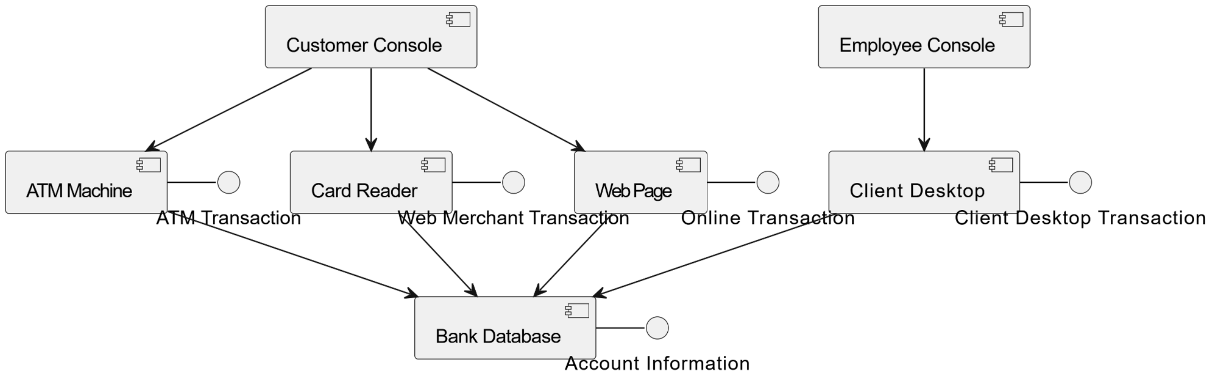
The ATM is given the utmost security in terms of technology because its a stand alone system and easily prone to malicious attacks.

1) Component Diagram:

Code:

```
@startuml
component "Customer Console"
component "Card Reader"
component "Bank Database"
component "ATM Machine"
component "Web Page"
component "Client Desktop"
component "Employee Console"

"Customer Console" --> "Card Reader"
"Customer Console" --> "ATM Machine"
"Customer Console" --> "Web Page"
"Card Reader" --> "Bank Database"
"ATM Machine" --> "Bank Database"
"Web Page" --> "Bank Database"
"Client Desktop" --> "Bank Database"
"Employee Console" --> "Client Desktop"

interface "ATM Transaction"
interface "Account Information"
interface "Web Merchant Transaction"
interface "Online Transaction"
interface "Client Desktop Transaction"

"ATM Machine" - [ATM Transaction]
"Bank Database" - [Account Information]
"Card Reader" - [Web Merchant Transaction]
"Web Page" - [Online Transaction]
"Client Desktop" - [Client Desktop Transaction]
@enduml
```

DIAGRAM:

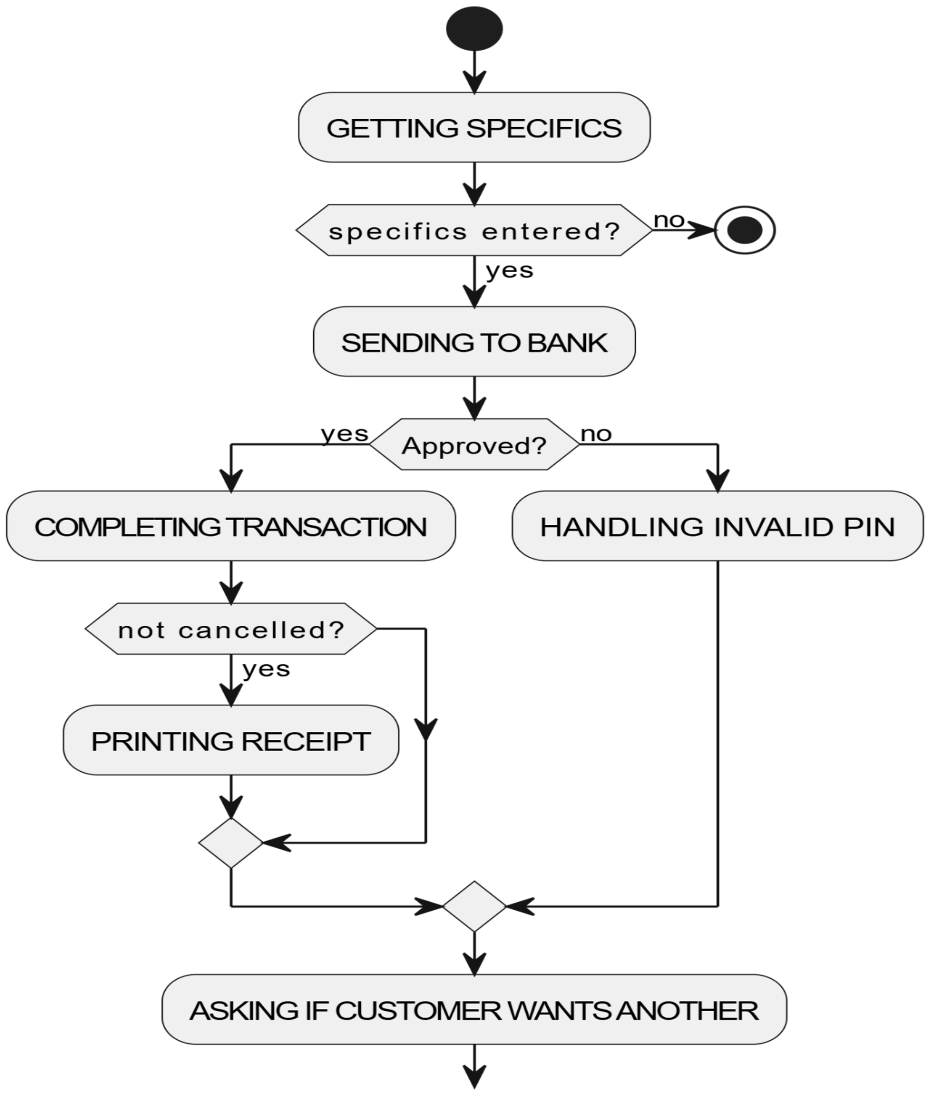


2)ACTIVITY DIAGRAM:

CODE:

```
@startuml
start
:GETTING SPECIFICS;
if (specifics entered?) then (yes)
  :SENDING TO BANK;
  if (Approved?) then (yes)
    :COMPLETING TRANSACTION;
    if (not cancelled?) then (yes)
      :PRINTING RECEIPT;
    endif
  else (no)
    :HANDLING INVALID PIN;
  endif
  :ASKING IF CUSTOMER WANTS ANOTHER;
else (no)
  stop
endif
@enduml
```

DIAGRAM



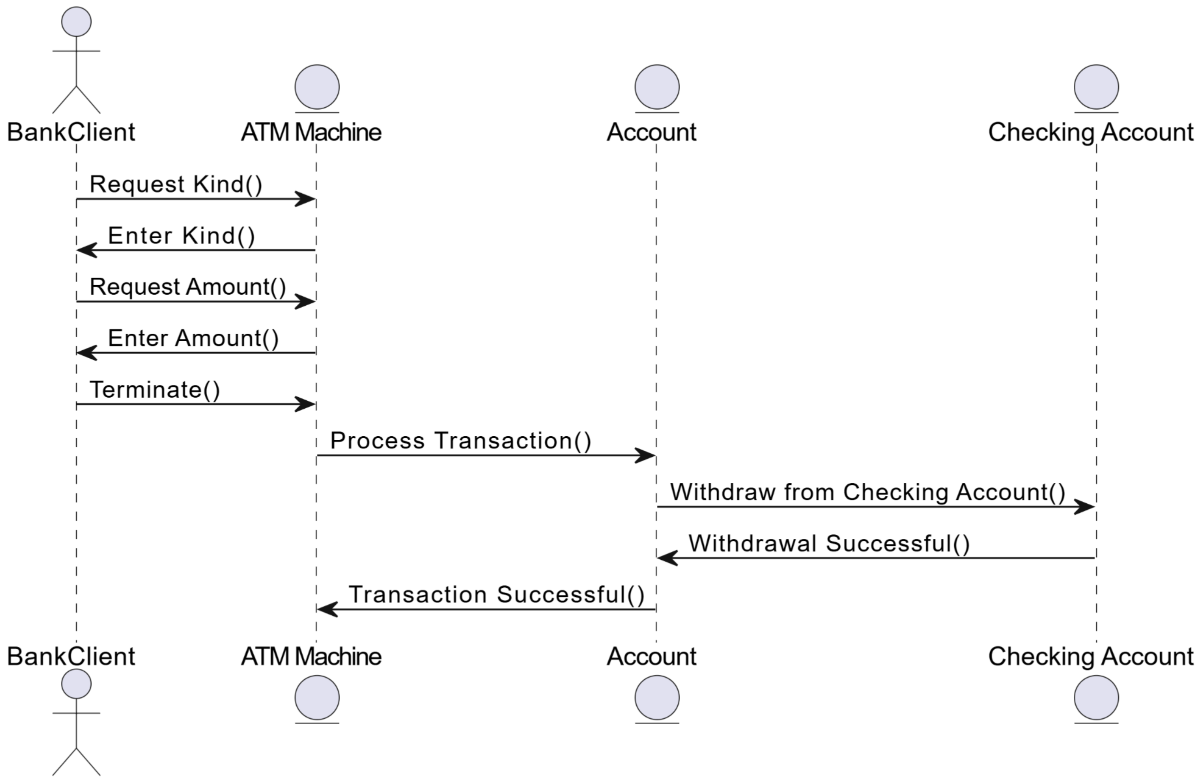
3) SEQUENCE DIAGRAM:

CODE:

```
@startuml
actor BankClient
entity "ATM Machine" as ATM
entity "Account" as Account
entity "Checking Account" as Checking_Account

BankClient -> ATM: Request Kind()
ATM -> BankClient: Enter Kind()
BankClient -> ATM: Request Amount()
ATM -> BankClient: Enter Amount()
```

```
BankClient -> ATM: Terminate()

ATM -> Account: Process Transaction()
Account -> Checking_Account: Withdraw from Checking Account()
Checking_Account -> Account: Withdrawal Successful()
Account -> ATM: Transaction Successful()
@enduml
```

DIAGRAM:



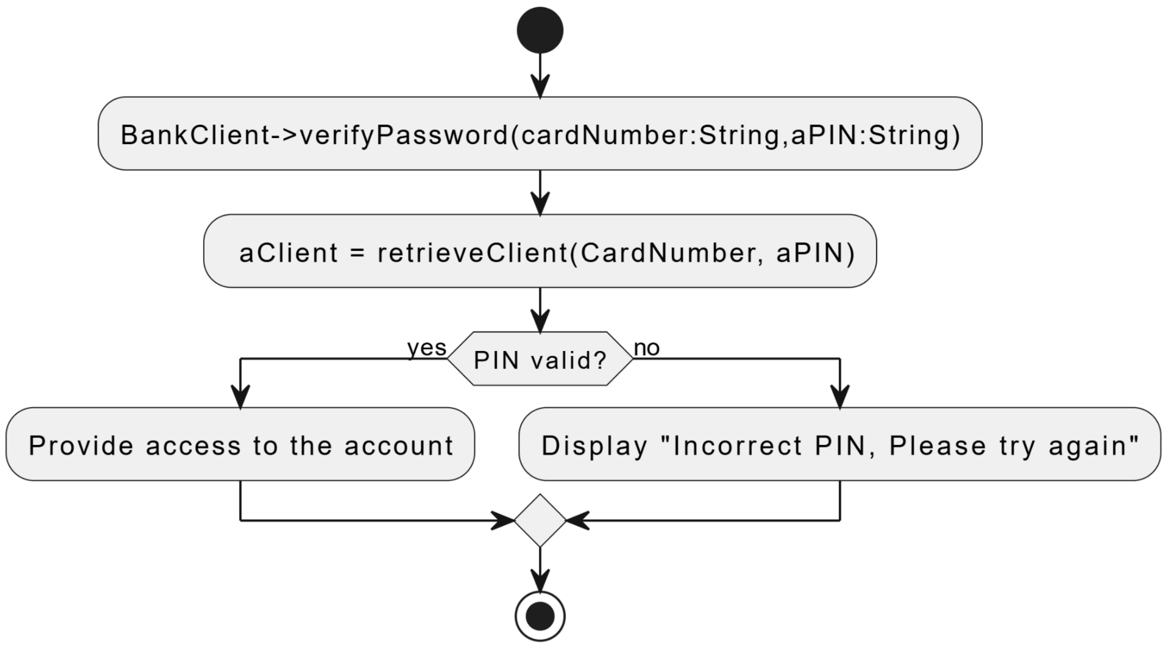
4) ACTIVITY DIAGRAM:

CODE:

```
@startuml
start
:BankClient->verifyPassword(cardNumber:String,aPIN:String);
: aClient = retrieveClient(CardNumber, aPIN);

if (PIN valid?) then (yes)
  :Provide access to the account;
else (no)
  :Display "Incorrect PIN, Please try again";
```

```
endif

stop
@enduml
```

DIAGRAM:

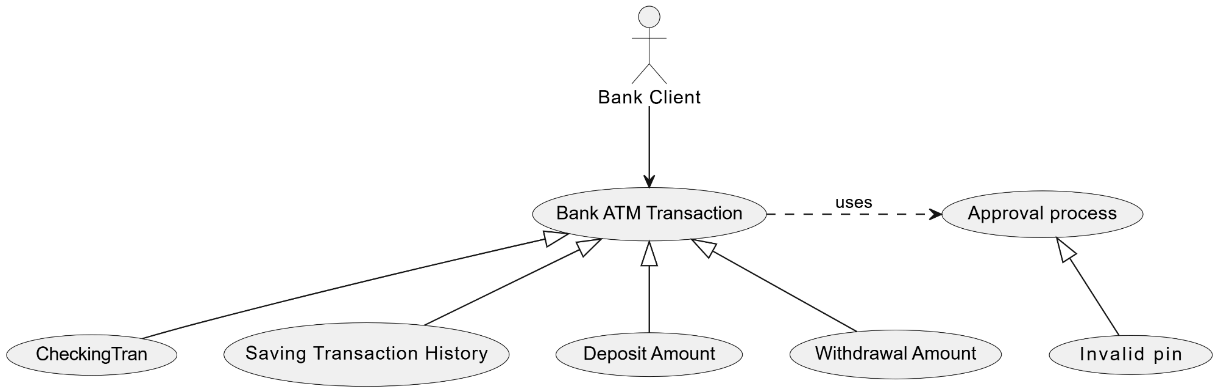

BankClient->verifyPassword(cardNumber:String,aPIN:String)
aClient = retrieveClient(CardNumber, aPIN)
PIN valid?
yes
no
Provide access to the account
Display "Incorrect PIN, Please try again"

USE CASE DIAGRAM:
CODE:

```
@startuml
actor "Bank Client"

usecase "Bank ATM Transaction"
usecase "CheckingTran" as UC1
usecase "Saving Transaction History" as UC2
usecase "Deposit Amount" as UC3
usecase "Withdrawal Amount" as UC4
usecase "Approval process" as UC5
usecase "Invalid pin" as UC6

"Bank Client" --> "Bank ATM Transaction"

"Bank ATM Transaction" <|-- UC1
"Bank ATM Transaction" <|-- UC2
```

```
"Bank ATM Transaction" <|-- UC3
"Bank ATM Transaction" <|-- UC4
"Bank ATM Transaction" .> UC5 : uses
UC5 <|-- UC6
@enduml
```

DIAGRAM:

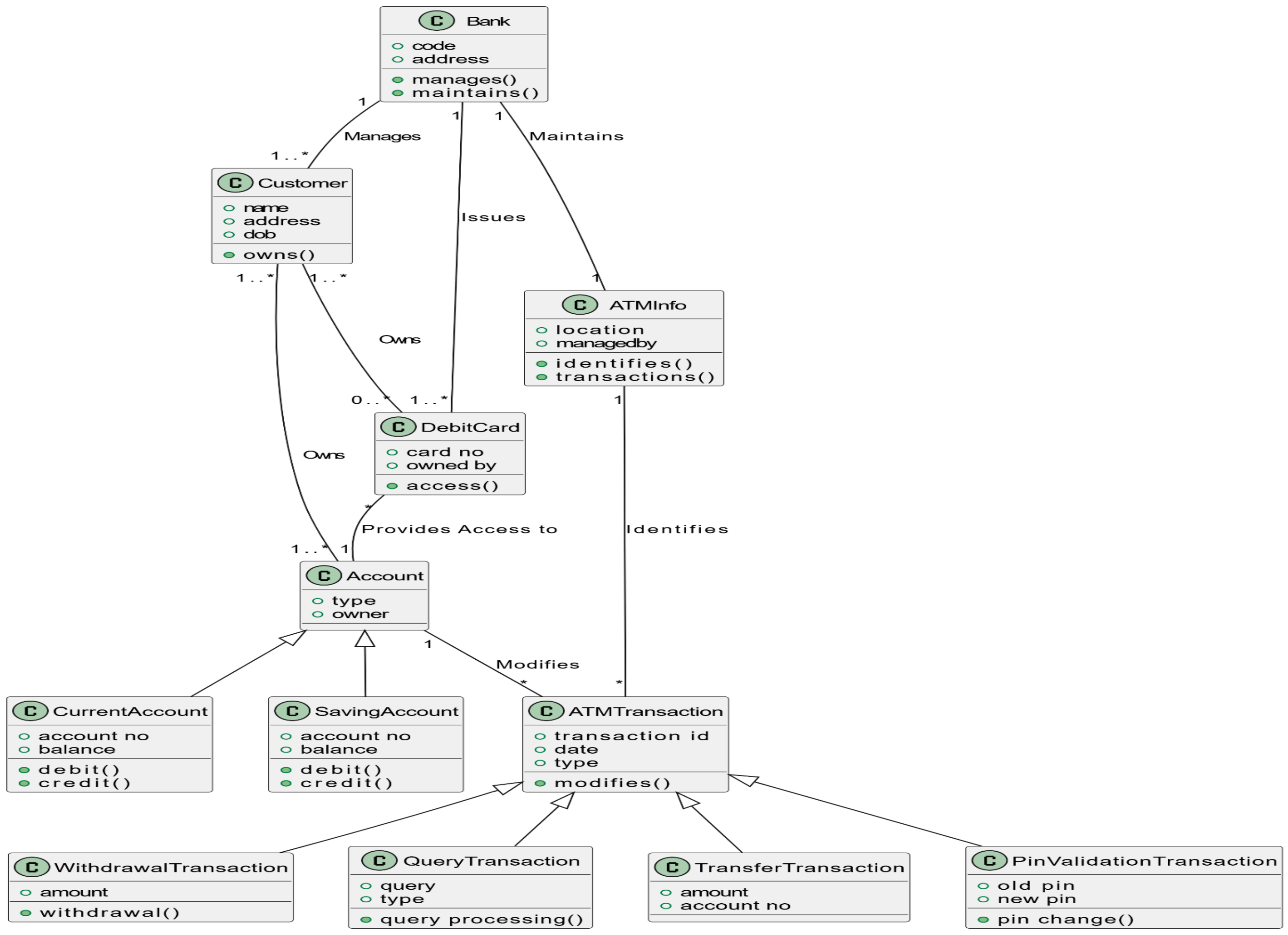


7)CLASS DIAGRAM:

CODE:

```
@startuml
class Bank {
   +code
   +address
   +manages()
   +maintains()
}

class Customer {
   +name
   +address
   +dob
   +owns()
}

class Account {
   +type
   +owner
}

class ATMInfo {
   +location
   +managedby
   +identifies()
   +transactions()
}

class DebitCard {
   +card no
   +owned by
   +access()
}

class ATMTransaction {
   +transaction id
   +date
   +type
   +modifies()
}

class CurrentAccount {
   +account no
   +balance
   +debit()
   +credit()
}

class SavingAccount {
   +account no
   +balance
   +debit()
   +credit()
}

class WithdrawalTransaction {
   +amount
   +withdrawal()
}

class QueryTransaction {
   +query
   +type
   +query processing()
}

class TransferTransaction {
   +amount
   +account no
}

class PinValidationTransaction {
   +old pin
   +new pin
   +pin change()
}

' Relationships
Bank "1" -- "1..*" Customer : "Manages"
Customer "1..*" -- "0..*" DebitCard : "Owns"
Customer "1..*" -- "1..*" Account : "Owns"
DebitCard "*" -- "1" Account : "Provides Access to"

Bank "1" -- "1..*" DebitCard : "Issues"

Bank "1" -- "1" ATMInfo : "Maintains"
ATMInfo "1" -- "*" ATMTransaction : "Identifies"
Account "1" -- "*" ATMTransaction : "Modifies"

Account <|-- CurrentAccount
Account <|-- SavingAccount

ATMTransaction <|-- WithdrawalTransaction
ATMTransaction <|-- QueryTransaction
ATMTransaction <|-- TransferTransaction
ATMTransaction <|-- PinValidationTransaction
@enduml
```

DIAGRAM:


Bank
code
address
manages()
maintains()
Customer
name
address
dob
owns()
ATMInfo
location
managedby
identifies()
transactions()
DebitCard
card no
owned by
access()
Account
type
owner
CurrentAccount
account no
balance
debit()
credit()
SavingAccount
account no
balance
debit()
credit()
ATMTransaction
transaction id
date
type
modifies()
WithdrawalTransaction
amount
withdrawal()
QueryTransaction
query
type
query processing()
TransferTransaction
amount
account no
PinValidationTransaction
old pin
new pin
pin change()
Manages
Issues
Maintains
Owns
Owns
Provides Access to
Identifies
Modifies
1
1..*
1
1
1
1..*
1..*
0..*
1..*
1
1..*
1
*
1
*
1
*