EX.NO:3
DATE

BANKING SYSTEMS

AIM.:

To develop a software for banking system by using software engineering methodology.

ABSTRACT:

All the banks are centralized with computers nowadays. All the accounts maintained in the bank and also the transaction effected, including banking transaction, are to be processed by the computers in the bank. An BANKING accepts relevant cash card, interacts with user, communicates with central system to carry out the transaction such as amount deposit,withdraw,balance enquiry and prints receipts. The system to be designed and implement must include appropriate record keeping and security provisions. The system must handle concurrent access to the same amount.

SYSTEM SPECIFICATION:

Software Requirements:

Operating system : Windows xP
Front end : Visual Basic 6.0
Back end : Microsoft Access

Hardware Requirements:

Processor : Intel Pentium @ 3.06GHz
RAM : 512MB DDR
Hard Disk : 80GB SATA
Monitor : 15''TFT
Keyboard : Multimedia Keyboard
Mouse : USB Optical type

PROGRAM ANALYSIS AND PROJECT PLANNING

PROBLEM STATEMENT:

This application BANKING System helps the bank client to access his/her account. The client has to first enter his/her name and account id number. Once when the details are confirmed the client can access the account, if the details are wrong then the client has to reenter the details. Next the system should ask for the type of transaction to perform, where the client has to choose the type of transaction they wish to carry on. And based on the transaction chosen the actions have to be carried out. And based on their transaction their account has to be updated.

SOFTWARE REQUIREMENT ANALYSIS

Modules.:

1. Card verification.
2. Selecting transaction.
3. Deposit.
4. Withdraw.

Card verification:

This module is an Approval process in which the client has to enter his/her name and account ID number, these details will be referred with the database and the service will be provided if the details are correct.

Selecting Transaction:

This module selects transaction in which the system displays the types of transaction (Deposit/Withdraw/Balance enquiry), the client has to choose any one, and the transaction will be carried on accordingly.

Deposit:

This module is Deposit saving module where the client has to enter the amount to be deposited. The account will be updated automatically and saved.

Withdraw:

This is a module where the client has to enter the amount to withdraw. After verifying that the funds are sufficient, the transaction is performed.

DATA MODELING

Data Dictionary:

Field Name	Description	Data Type	Field size	Default value	Validation
Name	Name of the client	String	20	NULL	(A-Z) or (a-z)
D.O.B	Date of birth	Date and time	10	NULL	(0-9)
Gender	Specify the gender	string	20	NULL	(A-Z) or (a-z)
Acc.no	Account Number of the Client	Integer	10	NULL	(0-9)
Balance	Amount In the client's Account	Integer	20	NULL	(0-9)

USE CASE DIAGRAM:

Use case diagram is a graph that describes the association between the actors and the use cases and a generalization among the use cases.

Actor:

An actor represent a set of roles that user of a use case play when interacting with the use cases. Actor identified here is the user and banker.

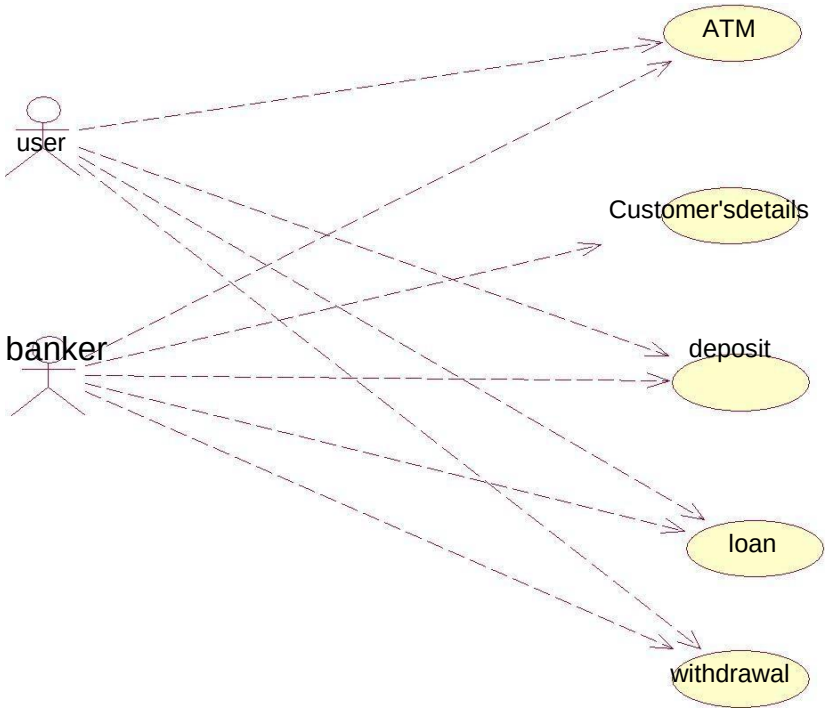
Use case:

A use case is a description of a set of sequence of actions that a system performs to yield result of value to an actor.

The Use Cases here it describes,

- Customers insert a card and verify it. Then he selects the type of transaction.
- BANKING performs the required transaction.

After transaction completes customer takes the card.



UML class diagram is the main static analysis diagram. This diagram shows the static structure of the model. This is a collection of static modeling elements, such as classes and their relationships, connected as a graph to each other and to their contents.

A class is drawn as a rectangle with three fields. The top field contains class name, the middle field contains attributes and the bottom field contains list of operations.

In our BANKING application, the classes are

1. New account
2. Deposit
3. Withdraw
4. Balance check
5. Transactions

Each class has its own attributes and properties

New account class:

The attributes involved are name,acc.no,dob,..
The operations involved are created and cancel.

Deposit class:

The attributes involved are account number and balance.
The operations involved are add and exit.

Withdraw class:

The attributes involved are account number and balance.
The operations involved are save and cancel.

Balance checking class:

This involves attributes like name, account number and balance amount.
The operation involved is getbalance.

Transactions:

This class involves attributes like name, account number, last transaction, date and amount in that account.
The operation involved is getdetails.

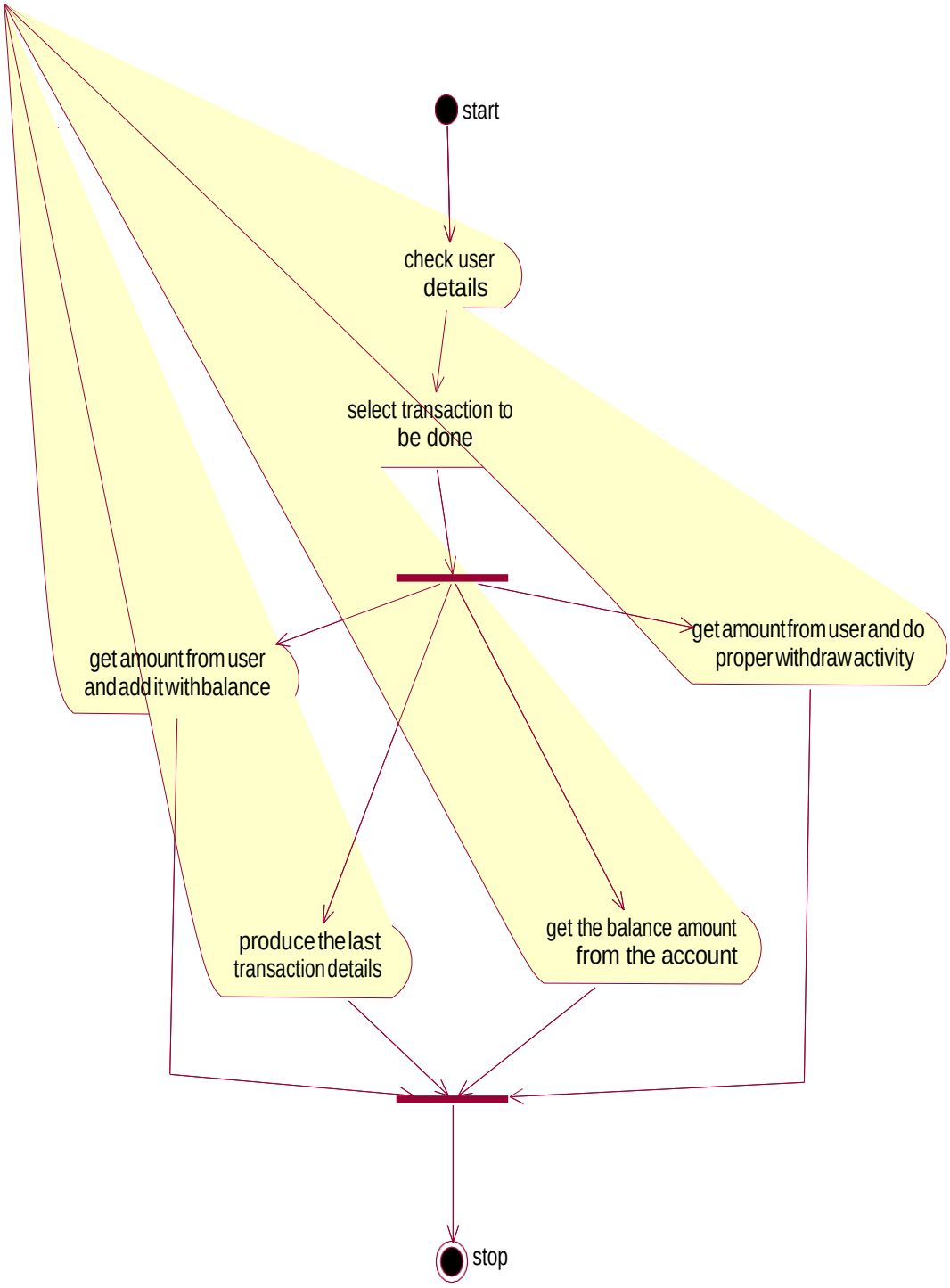
ACTIVITY DIAGRAM:

In this, the states are activities representing the performance of operations and the transactions are triggered by the completion of the operations.This diagram provides the view of flows and what is going on inside a usecase or among several classes.

The Activity diagram is used to describe the various activities taking place in an application. Here in our BANKING SYSTEM, we have various activities starting from login.

After login the client has to select the transactions to be done. If it is deposit, then add the given amount with the balance amount and update the database.

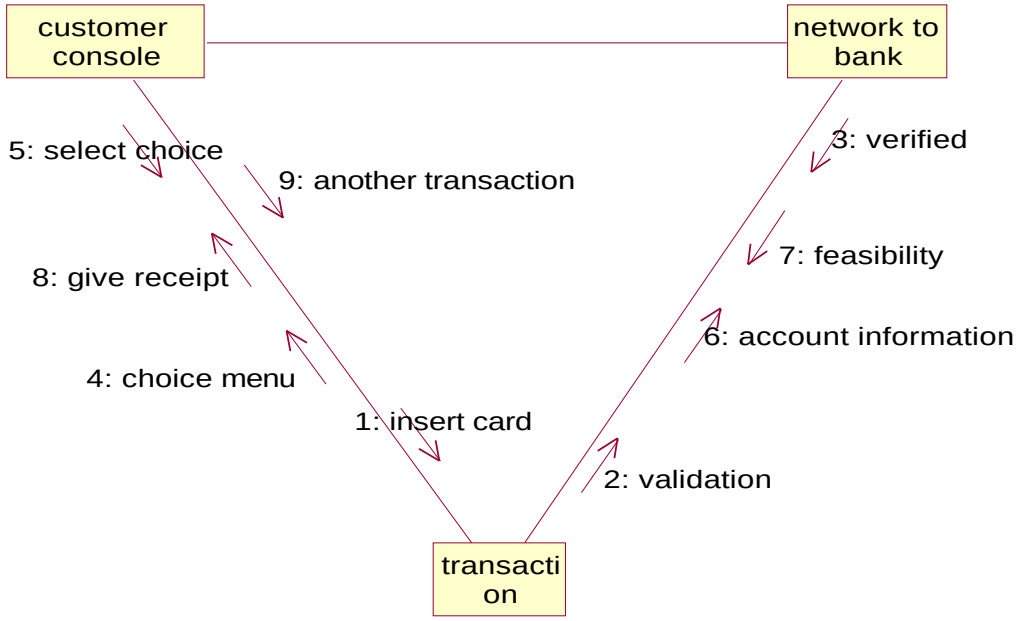
If it is withdraw, then decrement the given amount with the balance amount and update the database.


start
check user details
select transaction to be done
get amount from user and add it with balance
get amount from user and do proper withdraw activity
produce the last transaction details
get the balance amount from the account
stop

If it is balance checking, then get the balance amount from the database.

If it is ministatement generation, then obtain all the details from the database.

COLLABORATION DIAGRAM(ATM):



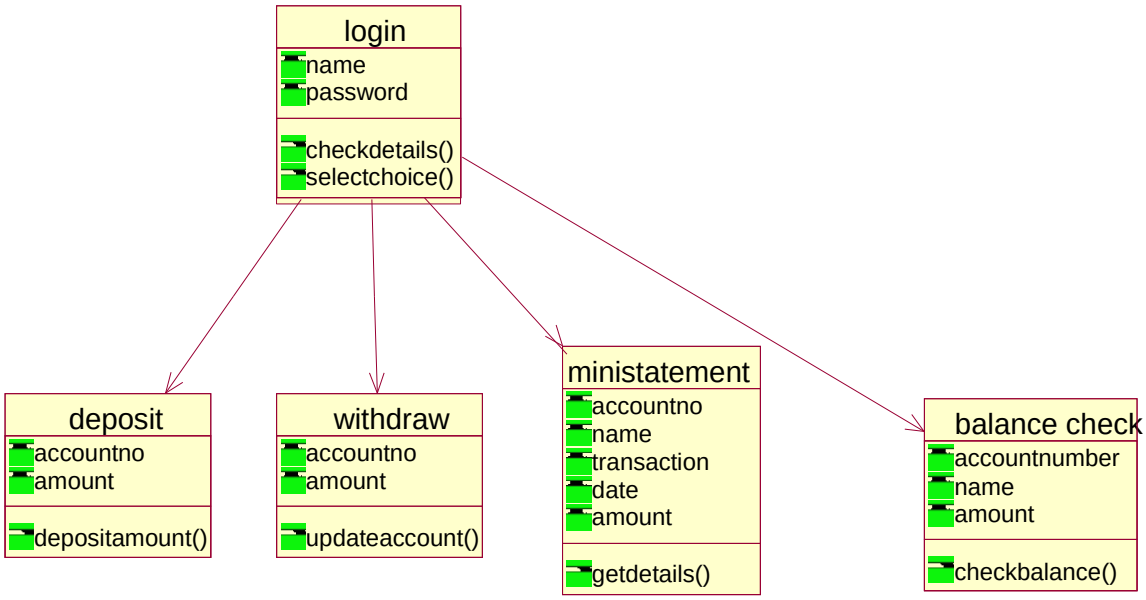
A collaboration diagram represents a collaboration, which is a set of objects related in a particular context, and interaction, which is a set of messages exchanged among the objects within he collaboration to achieve a desired outcome.

llaboration diagrams show exactly same information as the sequence diagram. However, collaboration diagram show this information in a different way and with different purpose.

In this collaboration diagram, the objects are represented as rectangle, the actors are stick figures. Whereas the sequence diagram illustrates the object and actor interaction overtime, the collaboration diagram shows the object and actor interaction without reference to time.

In our BANKING SYSTEM, each object interacts with each other or collaborates with each other; it gets represented by the solid line drawn between them.

CLASS DIAGRAM:



UML class diagram is the main static analysis diagram. This diagram shows the static structure of the model. This is a collection of static modeling elements, such as classes and their relationships, connected as a graph to each other and to their contents.

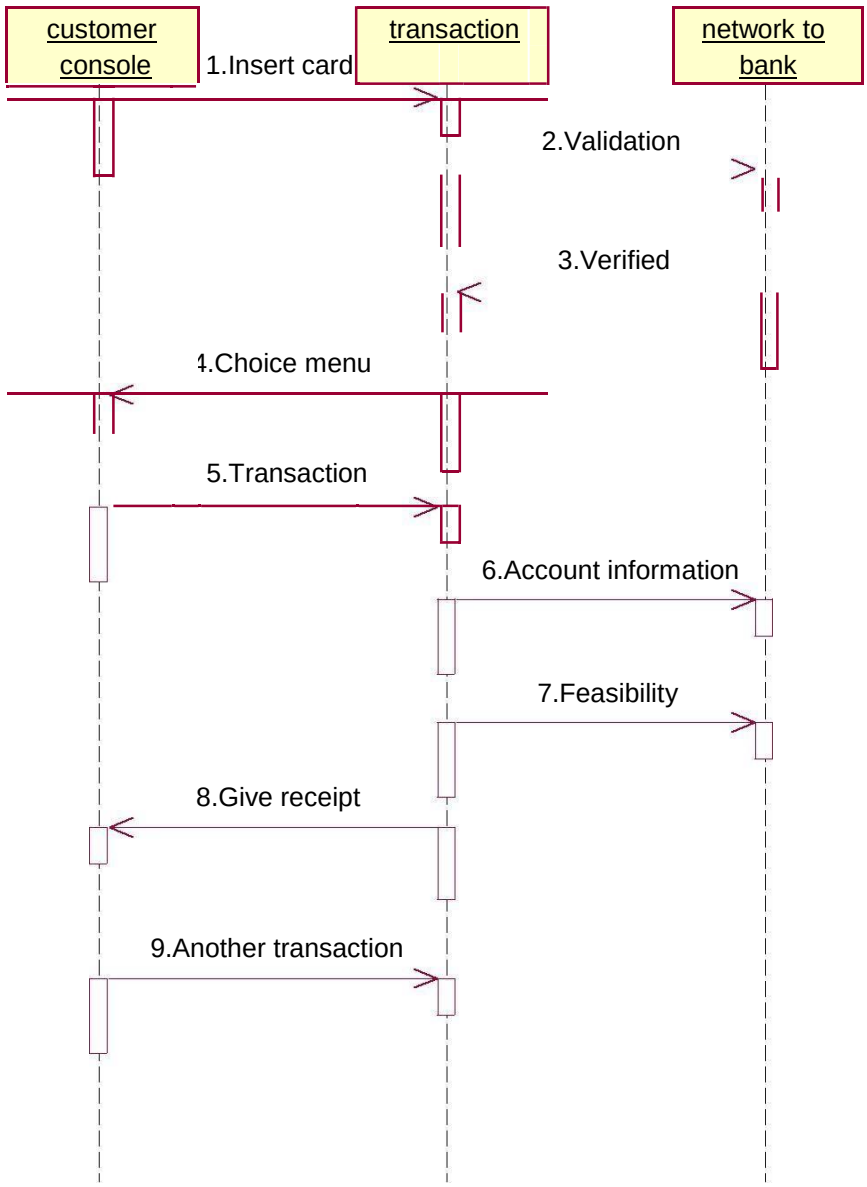
A class is drawn as a rectangle with three fields. The top field contains class name, the middle field contains attributes and the bottom field contains list of operations.

In our BANKING application, the classes are

1. New account
2. Deposit
3. Withdraw
4. Balance check
5. Transactions

Each class has its own attributes and properties

SEQUENCE DIAGRAM(ATM):



Sequence diagrams are easy and intuitive way of describing the behavior of a system by viewing the interaction between the system and its environment. A sequence diagram shows an interaction arranged in a time sequence.

The objects used in this sequence diagram are,
1.customer console
2. transaction
3. network to bank

The object client enters the login information. The object login access the details from the database and enters into the application. The object control obtains the amount for the operation either it is deposit or withdraw and update them according to the balance amount.

CODING:

Login Form:

```
Private Sub Command1_Click()
If Text1.Text = "rv" And Text2.Text = "mk" Then
   banking.Show
Else
MsgBox ("Invalid input")
End If
End Sub
```

Banking Form:

```
Private Sub Command1_Click()
newaccount.Show
End Sub
```

Account Form:

```
Private Sub crebtn_Click()
newaccountado.Recordset.Fields("Name") = txtname.Text
newaccountado.Recordset.Fields("Date of Birth") = txtdob.Text
newaccountado.Recordset.Fields("Gender") = txtgender.Text
newaccountado.Recordset.Fields("Address") = txtadd.Text
newaccountado.Recordset.Fields("Contact no") = txtphone.Text
newaccountado.Recordset.Update
MsgBox "User newaccount Successful"
withdraw.Show
End Sub

Private Sub Form_Load()
newaccountado.Recordset.AddNew
End Sub
```

Withdraw Form:

```
Private Sub Form_Load()
withdrawado.Recordset.AddNew
End Sub

Private Sub okbtn_Click()
withdrawado.Recordset.Fields("Accountno") = txtaccno.Text
withdrawado.Recordset.Fields("Withdraw amount") = txtwithdraw.Text
withdrawado.Recordset.Fields("Current Balance") = txtcurbal.Text
withdrawado.Recordset.Fields("Minimum Balance") = txtminbal.Text
withdrawado.Recordset.Fields("New Balance") = txtnewbal.Text
withdrawado.Recordset.Update
MsgBox "User withdraw Successful"
Deposit.Show
End Sub
```

Deposit Form:

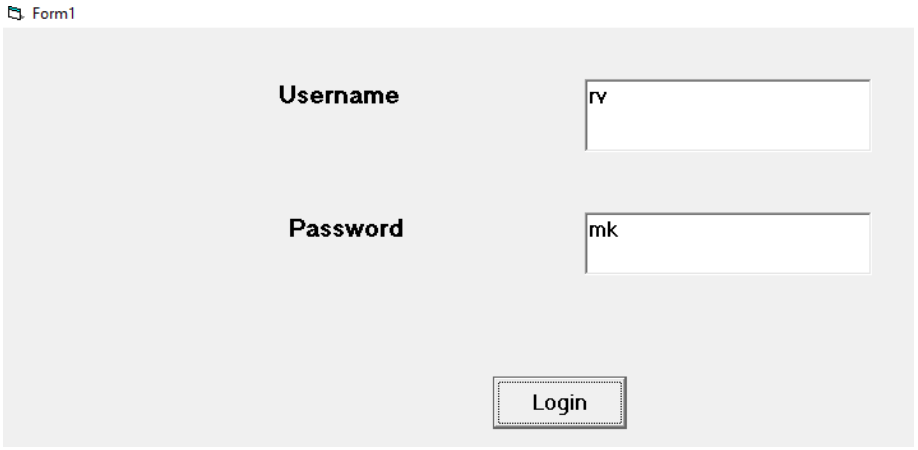
```
Private Sub addbtn_Click()
depositado.Recordset.Fields("Account no") = txtaccno.Text
depositado.Recordset.Fields("Current Balance") = txtcurbal.Text
depositado.Recordset.Fields("Deposit amount") = txtdepamt.Text
depositado.Recordset.Fields("New Balance") = txtnewbal.Text
depositado.Recordset.Update
MsgBox "User deposit Successful"
Transaction.Show
End Sub

Private Sub Form_Load()
depositado.Recordset.AddNew
End Sub
```

Transaction Form:

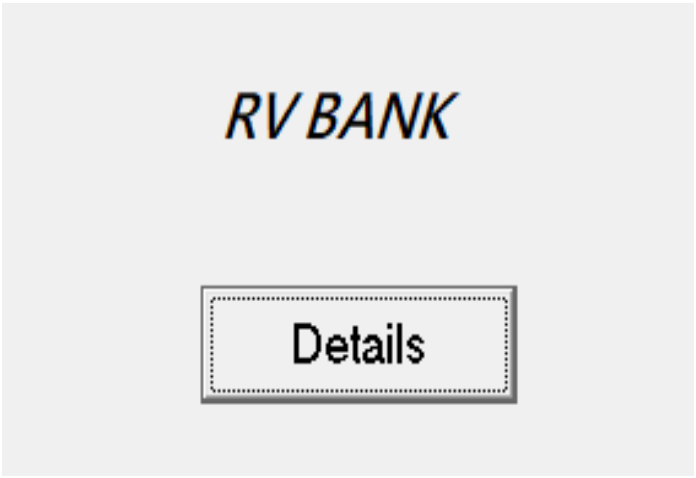
```
Private Sub Exitbtn_Click()
End
End Sub

Private Sub Form_Load()
transactionado.Recordset.AddNew
End Sub

Private Sub transactionbtn_Click()
transactionado.Recordset.Fields("Account no") = txtaccno.Text
transactionado.Recordset.Fields("Receiver Account no") = txtrecaccno.Text
```

```
transactionado.Recordset.Fields("Current Balance") = txtcurbal.Text
transactionado.Recordset.Fields("Minimum Balance") = txtminbal.Text
transactionado.Recordset.Fields("transaction Amount") = txttransamt.Text
transactionado.Recordset.Update
MsgBox "User transaction Successful"
End Sub
```

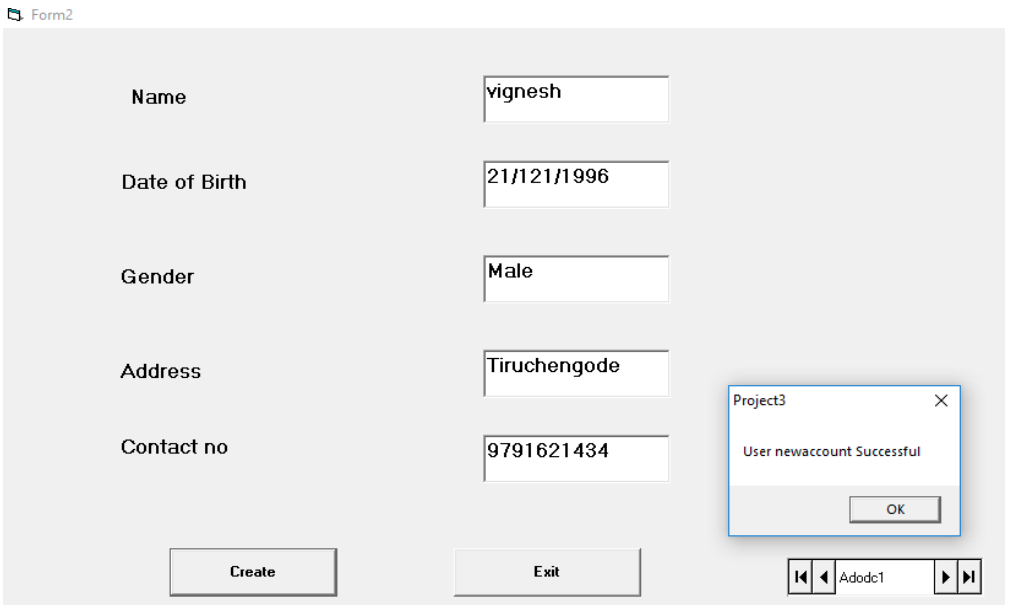
Output:
Form1:



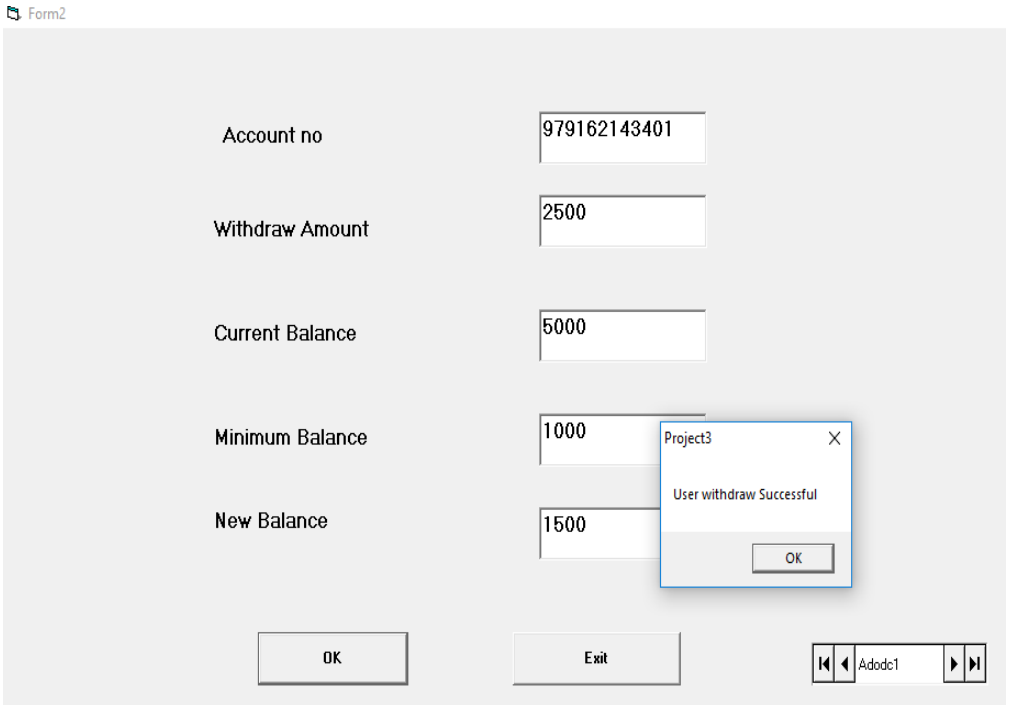
Form2:



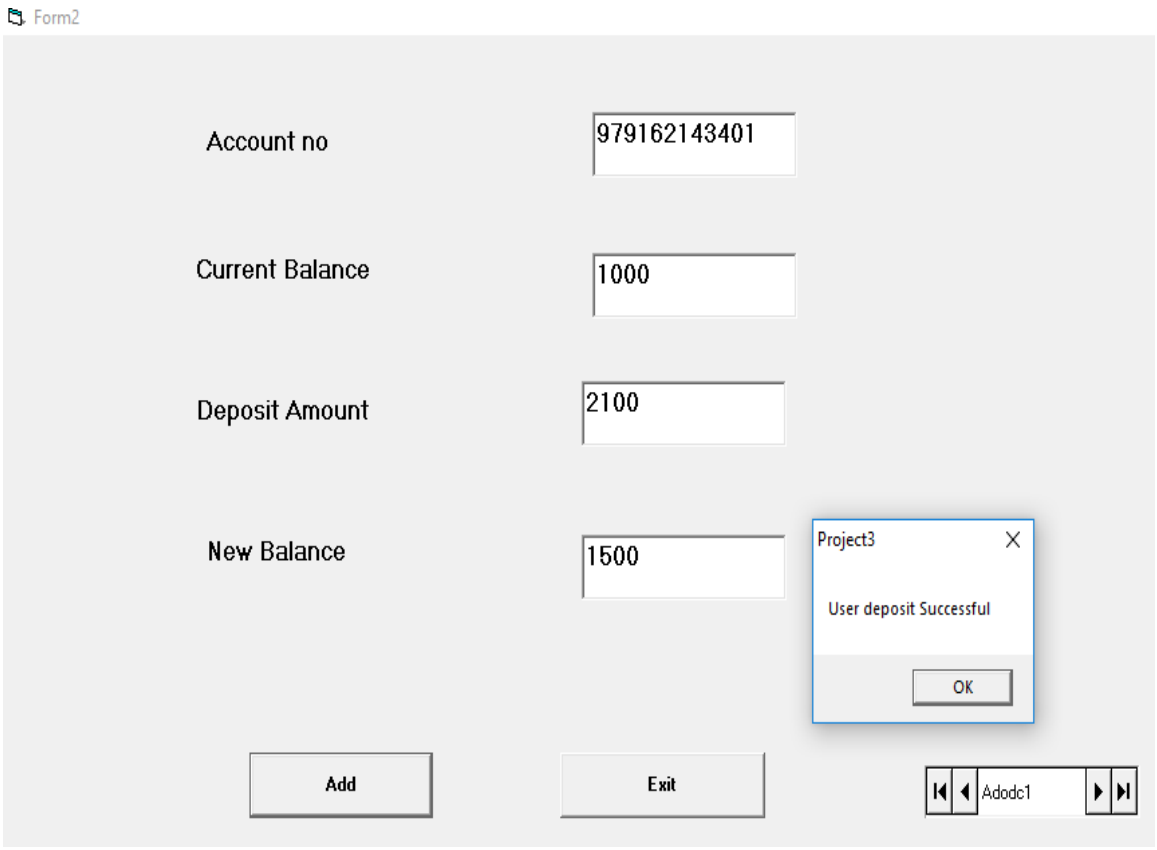
Form3:


Form2
Name
vignesh
Date of Birth
21/121/1996
Gender
Male
Address
Tiruchengode
Contact no
9791621434
Project3
User newaccount Successful
OK
Create
Exit
Adodc1

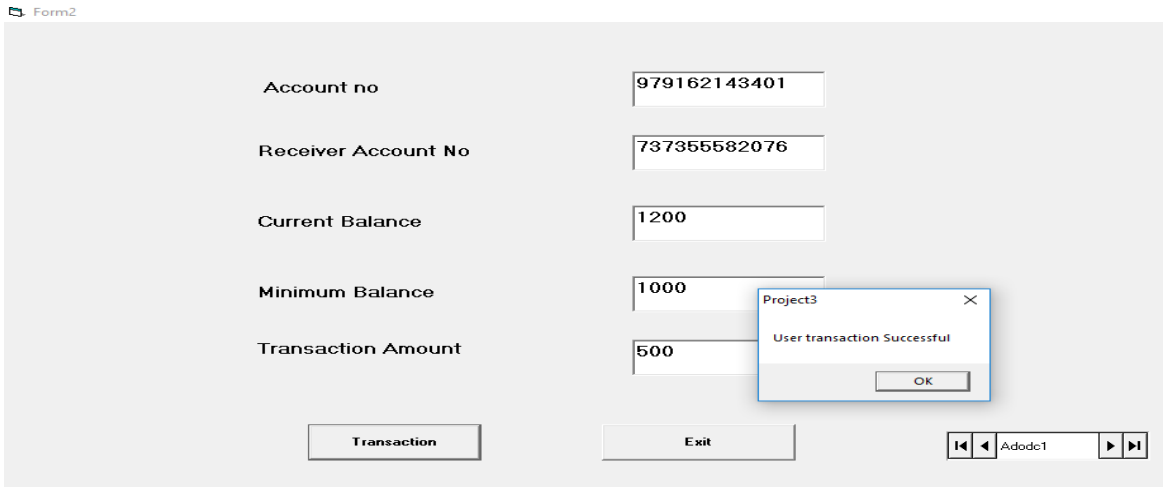
Form4:


Form2
Account no
979162143401
Withdraw Amount
2500
Current Balance
5000
Minimum Balance
1000
New Balance
1500
Project3
User withdraw Successful
OK
OK
Exit
Adodc1

Form5:

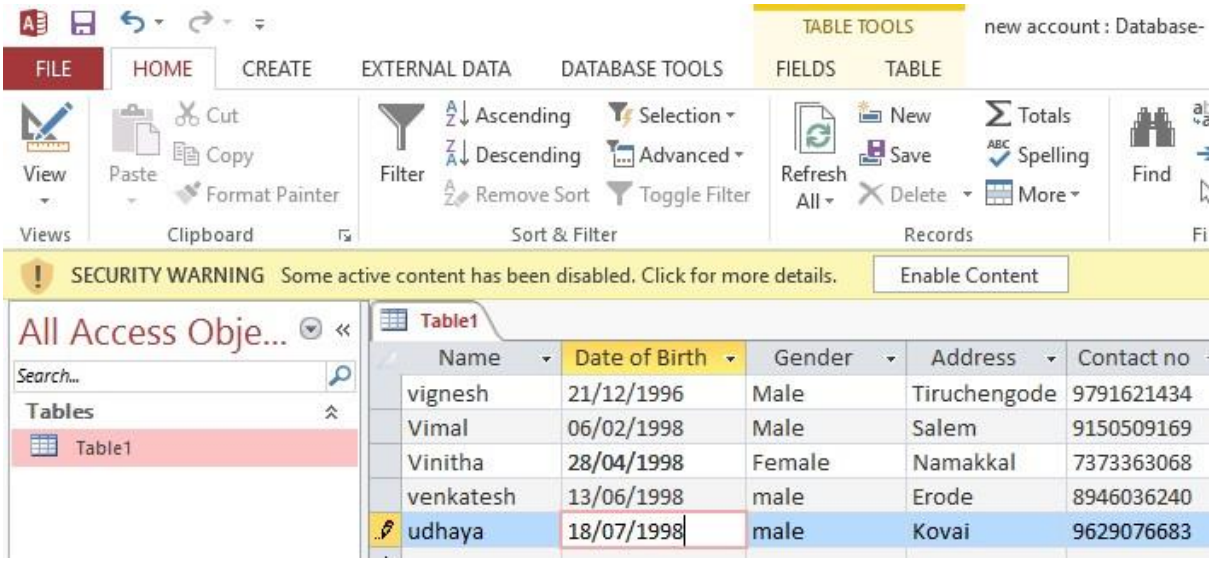

Form2
Account no
979162143401
Current Balance
1000
Deposit Amount
2100
New Balance
1500
Project3
User deposit Successful
OK
Add
Exit
Adodc1

Form6:

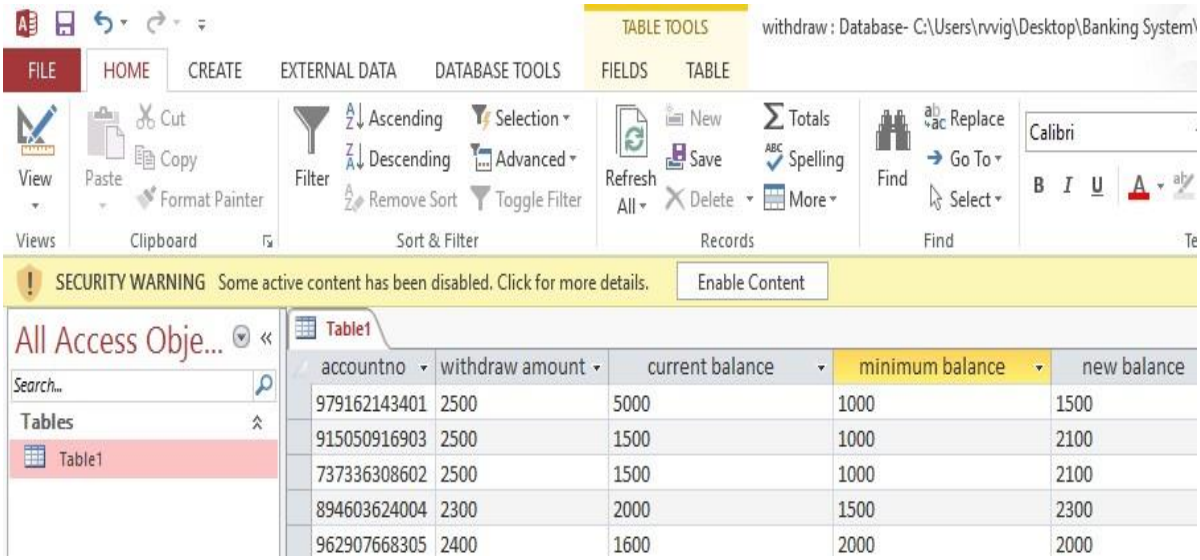

Form2
Account no
979162143401
Receiver Account No
737355582076
Current Balance
1200
Minimum Balance
1000
Transaction Amount
500
Project3
User transaction Successful
OK
Transaction
Exit
Adodc1

Database:

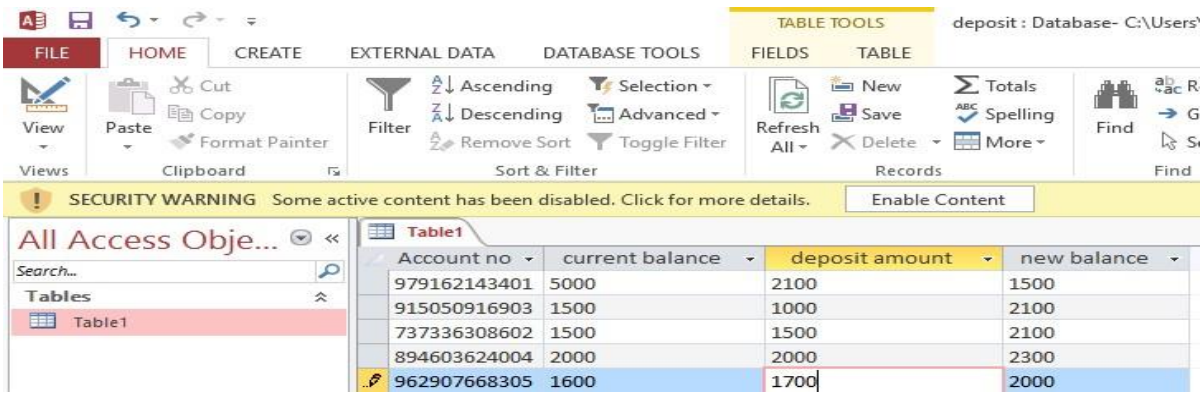
New Account Form



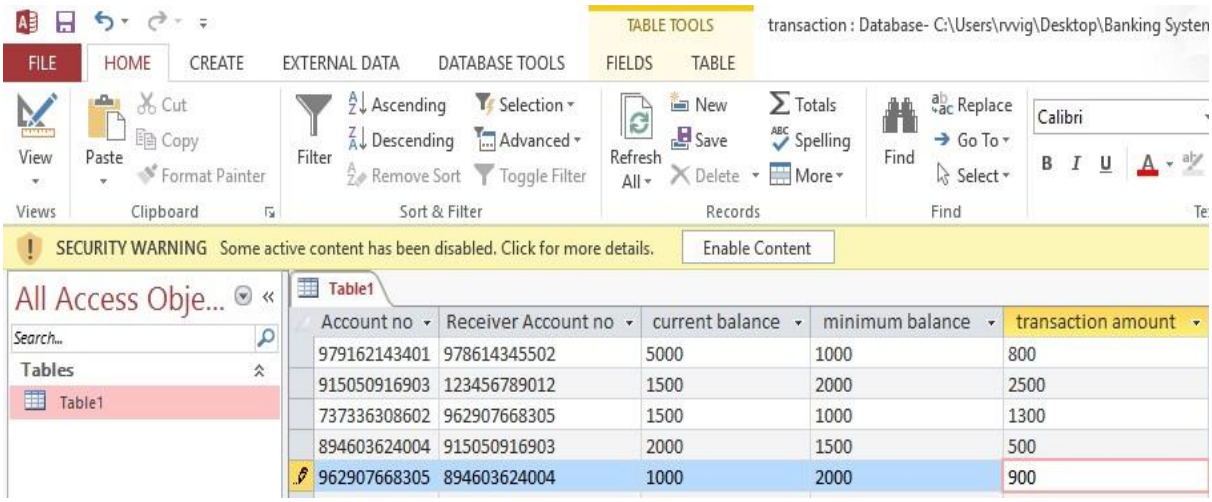
WithDraw Form



Deposit Form



Transaction Form



Result:

Thus the **BANKING SYSTEM** is developed with all necessary documents and UML diagrams using Software Engineering methodology.