EX. NO.: **CASE TOOLS**
DATE:

Introduction:

CASE tools known as Computer-aided software engineering tools is a kind of component-based development which allows its users to rapidly develop information systems. The main goal of case technology is the automation of the entire information systems development life cycle process using a set of integrated software tools, such as modeling, methodology and automatic code generation. Component based manufacturing has several advantages over custom development. The main advantages are the availability of high quality, defect free products at low cost and at a faster time. The prefabricated components are customized as per the requirements of the customers. The components used are pre-built, ready-tested and add value and differentiation by rapid customization to the targeted customers. However the products we get from case tools are only a skeleton of the final product required and a lot of programming must be done by hand to get a fully finished, good product.

Characteristics of CASE:

Some of the characteristics of case tools that make it better than customized development are;

- It is a graphic oriented tool.
- It supports decomposition of process.

Some typical CASE tools are:

- Unified Modeling Language
- Data modeling tools, and
- Source code generation tools

Introduction to UML (Unified Modeling Language):

The UML is a language for specifying, constructing, visualizing, and documenting the software system and its components. The UML is a graphical language with sets of rules and semantics. The rules and semantics of a model are expressed in English in a form known as OCL (Object Constraint Language). OCL uses simple logic for specifying the properties of a system. The UML is not intended to be a visual programming language. However it has a much closer mapping to object-oriented programming languages, so that the best of both can be obtained. The UML is much simpler than other methods preceding it. UML is appropriate for modeling systems, ranging from enterprise information system to distributed web based application and even to real time embedded system. It is a very expensive language addressing all views needed to develop and then to display system even though understand to use. Learning to apply UML effectively starts forming a conceptual mode of languages which requires learning.

Three major language elements:

- UML basic building blocks
- Rules that dictate how this building blocks put together

- ❖ Some common mechanism that apply throughout the language

The primary goals in the design of UML are:

1. Provides users ready to use, expressive visual modeling language as well so they can develop and exchange meaningful models.
2. Provide extensibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development processes.
4. Provide formal basis for understanding the modeling language.
5. Encourage the growth of the OO tools market.
6. Support higher-level development concepts.
7. Integrate best practices and methodologies.

Every complex system is best approached through a small set of nearly independent views of a model. Every model can be expressed at different levels of fidelity. The best models are connected to reality. The UML defines nine graphical diagrams:

1. Class diagram
2. Use-case diagram
3. Behavior diagram
   - 3.1. Interaction diagram
     - 3.1.1. sequence diagram
     - 3.1.2. collaboration diagram
   - 3.2. state chart diagram
   - 3.3. activity diagram
4. Implementation diagram
   - 4.1 component diagram
   - 4.2 deployment diagram

1. UML class diagram:

The UML class diagram is also known as object modeling. It is a static analysis diagram. These diagrams show the static structure of the model. A class diagram is a connection of static model elements, such as classes and their relationships, connected as a graph to each other and to their contents.

2. Use-case diagram:

The functionality of a system can be described in a number of different use-cases, each of which represents a specific flow of events in a system. It is a graph of actors, a set of use-cases enclosed in a boundary, communication, associations between the actors and the use-cases, and generalization among the use-cases.

3. Behavior diagram:

It is a dynamic model unlike all the others mentioned before. The objects of an object oriented system are not static and are not easily understood by static diagrams. The behavior of the class's instance (an object) is represented in this diagram. Every

use-case of the system has an associated behavior diagram that indicates the behavior of the object. In conjunction with the use-case diagram we may provide a script or interaction diagram to show a time line of events. It consists of sequence and collaboration diagrams.

4. Interaction diagram

It is the combination of sequence and collaboration diagram. It is used to depict the flow of events in the system over a timeline. The interaction diagram is a dynamic model which shows how the system behaves during dynamic execution.

5. State chart diagram:

It consists of state, events and activities. State diagrams are a familiar technique to describe the behavior of a system. They describe all of the possible states that a particular object can get into and how the object's state changes as a result of events that reach the object. In most OO techniques, state diagrams are drawn for a single class to show the lifetime behavior of a single object.

6. Activity diagram:

It shows organization and their dependence among the set of components. These diagrams are particularly useful in connection with workflow and in describing behavior that has a lot of parallel processing. An activity is a state of doing something: either a real-world process, or the execution of a software routine.

7. Implementation diagram:

It shows the implementation phase of the systems development, such as the source code structure and the run-time implementation structure. These are relatively simple high level diagrams compared to the others seen so far. They are of two sub-diagrams, the component diagram and the deployment diagram.

8. Component diagram:

These are organizational parts of a UML model. These are boxes to which a model can be decomposed. They show the structure of the code itself. They model the physical components such as source code, user interface in a design. It is similar to the concept of packages.

9. Deployment diagram:

The deployment diagram shows the structure of the runtime system. It shows the configuration of runtime processing elements and the software components that live in them. They are usually used in conjunction with deployment diagrams to show how physical modules of code are distributed on the system.

Notation elements:

These are explanatory parts of UML model. They are boxes which may apply to describe and remark about any element in the model. They provide the information for understanding the necessary details of the diagrams.

Relations in the UML:

These are four kinds of relationships used in an UML diagram, they are:

- Dependency
- Association
- Generalization
- Realization

Dependency:

It is a semantic relationship between two things in which a change one thing affects the semantics of other things. Graphically a dependency is represented by a non-continuous line.

Association:

It is a structural relationship that describes asset of links. A link is being connected among objects. Graphically association is represented as a solid line possibly including label.

Generalization:

It is a specialized relationship in which the specialized elements are substitutable for object of the generalized element. Graphically it is a solid line with hollow arrow head parent.

Realization:

It is a semantic relation between classifiers. Graphically it is represented as a cross between generalization and dependency relationship.

Where UML can be used:

UML is not limited to modeling software. In fact it is expressive to model non-software such as to show in structure and behavior of health case system and to design the hardware of the system.

Conceptual model be UML:

UML you need to form the conceptual model of UML. This requires three major elements:

- UML basic building blocks.
- Rules that dictate how this building blocks are put together.
- Some common mechanism that apply throughout the language.

Once you have grasped these ideas, you may be able to read. UML create some basic ones. As you gain more experience in applying conceptual model using more advanced features of this language.

Building blocks of the UML:

The vocabulary of UML encompasses these kinds of building blocks.

Use CASE definition:

Description:

A use case is a set of scenarios tied together by a common user goal. A use case is a behavioral diagram that shows a set of use case actions and their relationships.

Purpose:

The purpose of use case is login and exchange messages between sender and receiver (Email client).

Main flow:

First, the sender gives his id and enters his login. Now, he enters the message to the receiver id.

Alternate flow:

If the username and id by the sender or receiver is not valid, the administrator will not allow entering and “Invalid password” message is displayed.

Pre-condition:

A person has to register himself to obtain a login ID.

Post-condition:

The user is not allowed to enter if the password or user name is not valid.

Class diagram:

Description:

- A class diagram describes the type of objects in system and various kinds of relationships that exists among them.
- Class diagrams and collaboration diagrams are alternate representations of object models.

During analysis, we use class diagram to show roles and responsibilities of entities that provide email client system behaviors design. We use to capture the structure of classes that form the email client system architecture.

The classes used in system are:

1. user
2. login
3. s
4. ds

A class diagram is represented as:

<<Class name>>
<<Attribute 1>>
<<Attribute n>>
<<Operation ()>>

Relationship used:

A change in one element affects the other

Generalization:

It is a kind of relationship

State chart:

Description:

- The state chart diagram made the dynamic behavior of individual classes.
- State chart shows the sequences of states that an object goes through events and state transitions.
- A state chart contains one state ‘start’ and multiple ‘end’ states.

The important objectives are:

Decision:

It represents a specific location state chart diagram where the work flow may branch based upon guard conditions.

Synchronization:

It gives a simultaneous workflow in a state chart diagram. They visually define forks and joints representing parallel workflow.

Forks and joins:

- A fork construct is used to model a single flow of control.
- Every work must be followed by a corresponding join.
- Joints have two or more flow that unit into a single flow.

State:

A state is a condition or situation during a life of an object in which it satisfies condition or waits for some events.
Transition:

It is a relationship between two activities and between states and activities.

Start state:

A start state shows the beginning of a workflow or beginning of a state machine on a state chart diagram.

End state:

It is a final or terminal state.

Activity diagram

Description:

Activity diagram provides a way to model the workflow of a development process. We can also model this code specific information such as class operation using activity diagram. Activity diagrams can model different types of diagrams. There are various tools involved in the activity diagram.

Activity:

An activity represents the performance of a task on duty. It may also represent the execution of a statement in a procedure.

Decision:

A decision represents a condition on situation during the life of an object, which it satisfies some condition or waits for an event.

Start state:

It represents the condition explicitly the beginning of a workflow on an activity.

Object flow:

An object on an activity diagram represents the relationship between activity and object that creates or uses it.

Synchronization:

It enables us to see a simultaneous workflow in an activity.

End state:

An end state represents a final or terminal state on an activity diagram or state chart diagram.

Sequence diagram:

Description:

A sequence diagram is a graphical view of scenario that shows object interaction in a time based sequence what happens first what happens next. Sequence diagrams are closely related to collaboration diagram.

The main difference between sequence and collaboration diagram is that sequence diagram show time based interaction while collaboration diagram shows objects associated with each other.

The sequence diagram for the e-mail client system consists of the following objectives:

Object:

An object has state, behavior and identity. An object is not based is referred to as an instance.

The various objects in e-mail client system are:

- User
- Website
- Login
- Groups

Message icon:

A message icon represents the communication between objects indicating that an action will follow. The message icon is the horizontal solid arrow connecting lifelines together.

Collaboration diagram:

Description:

Collaboration diagram and sequence diagrams are alternate representations of an interaction. A collaboration diagram is an interaction diagram that shows the order of messages that implement an operation or a transaction. Collaboration diagram is an interaction diagram that shows the order of messages that implement an operation or a transaction. Collaboration diagram shows object s, their links and their messages. They can also contain simple class instances and class utility instances.

During, analysis indicates the semantics of the primary and secondary interactions. Design, shows the semantics of mechanisms in the logical design of system.

Toggling between the sequence and collaboration diagrams

When we work in either a sequence or collaboration diagram, it is possible to view the corresponding diagram by pressing F5 key.

EX. NO.: **ATM SYSTEM**
DATE:

Aim:

To create a system to perform Bank ATM transaction

Problem statement:

- This system is build for the bank client and the manager.
- The bank client must be able to deposit and withdraw amount from his/her accounts using the ATM machine. Each transaction must be recorded and the client must be able to review all transactions performed in his/her account. Recorded transactions must include the date, time, transaction type, amount and account balance after the transaction.
- The bank manager must be able to view the ATM machine status that is the total balance of the ATM machine, today's withdrawal, today's balance and the limitations of the machine.
- The bank client is provided by login verification. If it is valid he/she will access their account otherwise an appropriate message is displayed to the client.

Software requirements:

Microsoft visual basic 6.0 is used as front-end for our project and ms-access is used as back-end to store the data.

USE-CASE diagram:

The ATM transaction use cases in our system are:

1. Login
2. Withdraw
3. Mini statement
4. ATM machine status
5. Deposit

Actors involved:

1. User
2. Bank manager

USE-CASE name: Login

The user enters a user name and password. If it is valid, the user's account becomes available. If it is invalid, an appropriate message is displayed to the user.

USE-CASE name: Withdraw

The user tries to withdraw an amount from his or her checking account. The amount is less than or equal to the checking account's balance, the transaction is performed and the available information is displayed. The system creates a record of the transaction and the display confirmation message is displayed to the client.

USE-CASE name: Mini statement

The bank user requests a history of transactions for a checking account. The system displays the transaction history for the checking account. The transaction history consists of amount, date, transaction type and balance of the particular account.

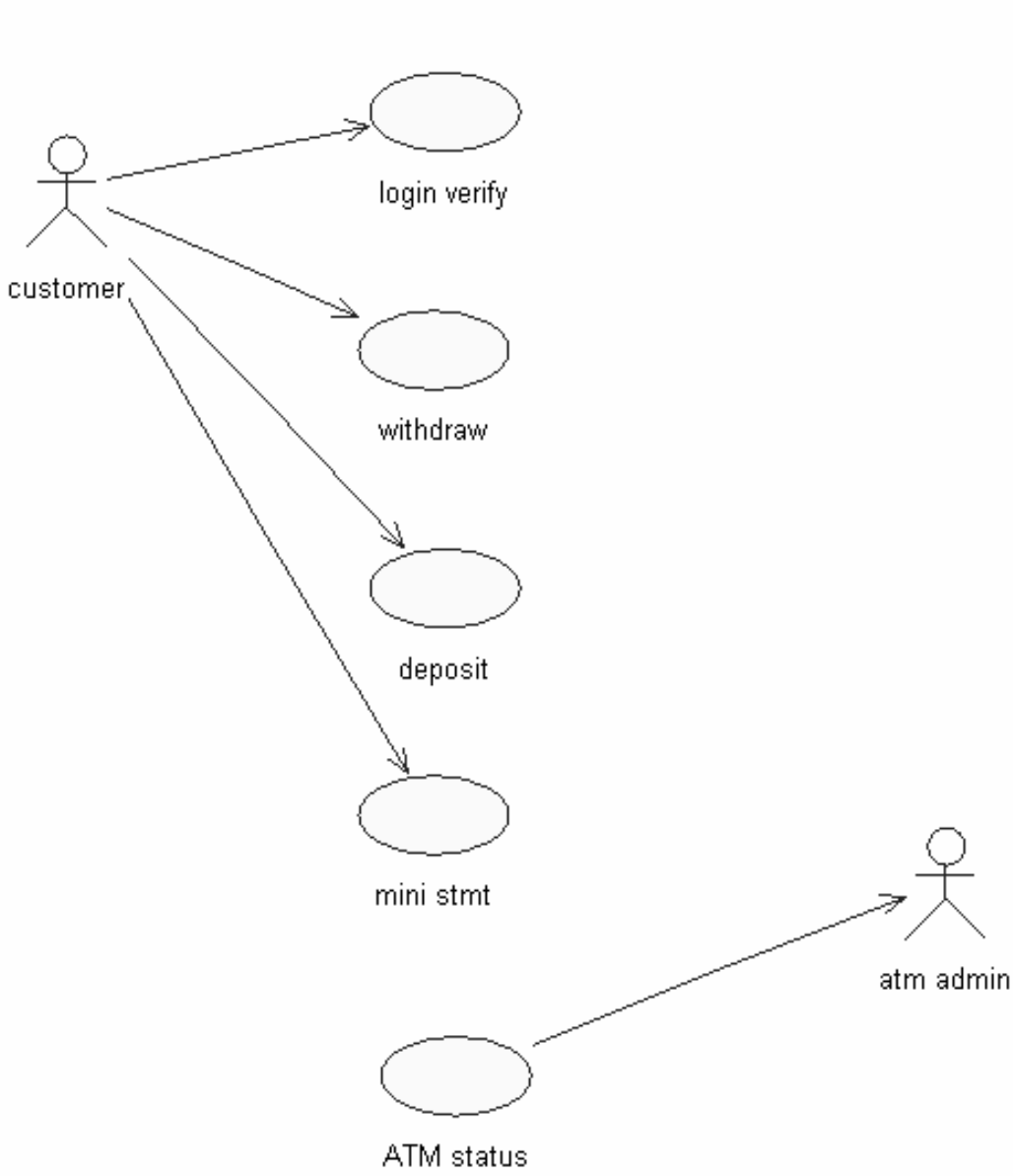
USE-CASE name: ATM machine status

The bank manager enters a username and password. If it is valid, the bank manager accesses the machine status. If it is invalid, an appropriate message is displayed to the user.

USE-CASE name: Deposit

The bank user requests the system to deposit money to an account. The user accesses the account for which a deposit is going to be made and enters the amount. The system creates a record of the transaction and an appropriate confirmation message (display confirmation) is displayed to the client. The transaction must include the date, type, amount and account balance after the transaction.

Use-case diagram for ATM system



Class diagram

The class diagram, also referred to as object modeling is the main static analysis diagram. The main task of object modeling is to graphically show what each object will do in the problem domain. The problem domain describes the structure and the relationships among objects.

The ATM system class diagram consists of four classes:

1. User class
2. ATM machine status
3. Account
4. Transaction

1) User class:

It consists of four attributes and two operations. The attributes are user name, password, address and DOB. The operations of this class are read (), display () and write ().

2) ATM machine status:

The attributes of this class are ATM balance, today's withdrawal, today's balance, and limitations. The operations are login verification (), ATM status () and display confirmation ().

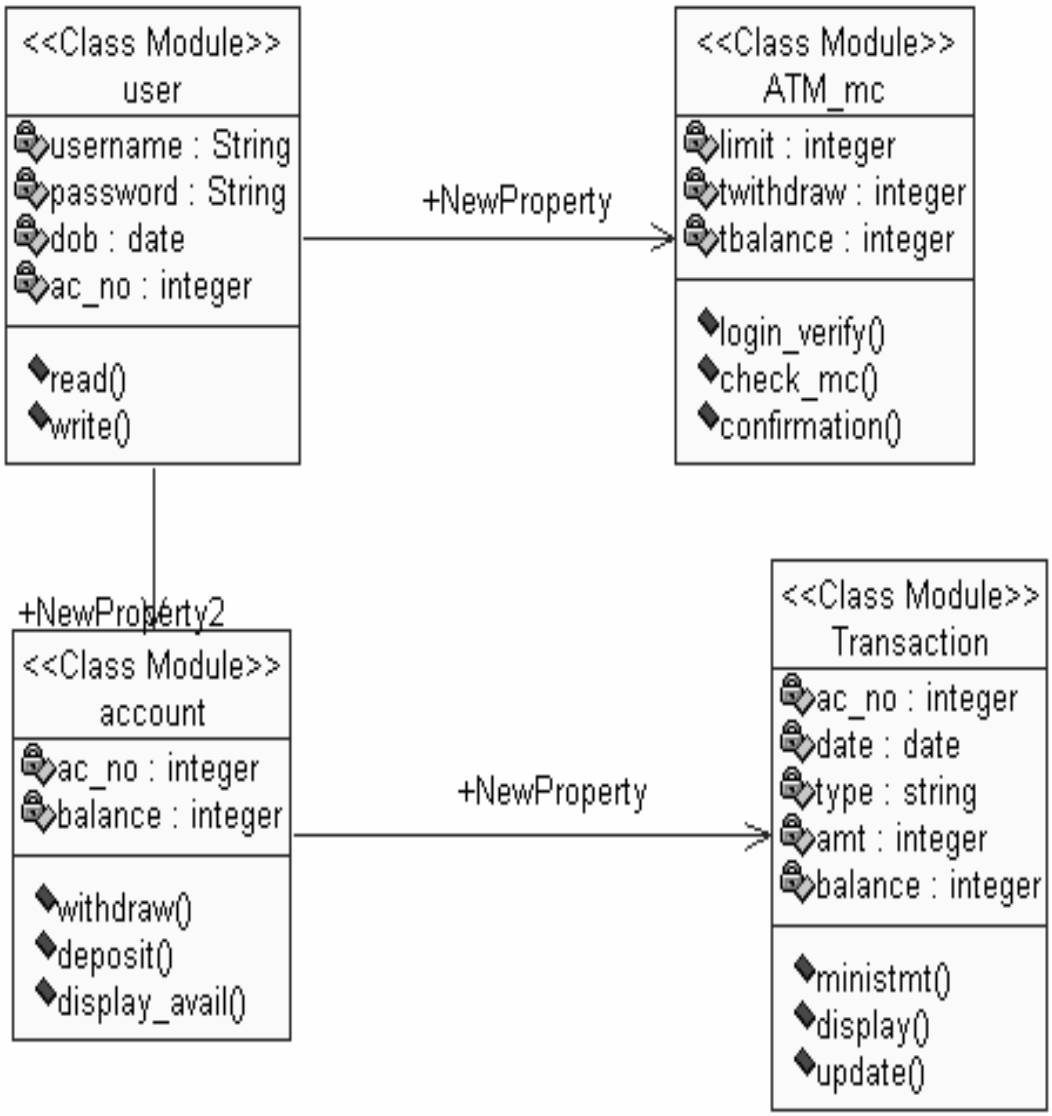
3) Account:

The attributes are account no. and balance and the operations are withdraw (), deposit () and display availability ().

4) Transaction:

The attributes of this class are account no, transaction type, data, amount, balance and the operations are mini statement () and create transaction ().

Class diagram for ATM system



Sequence diagram:

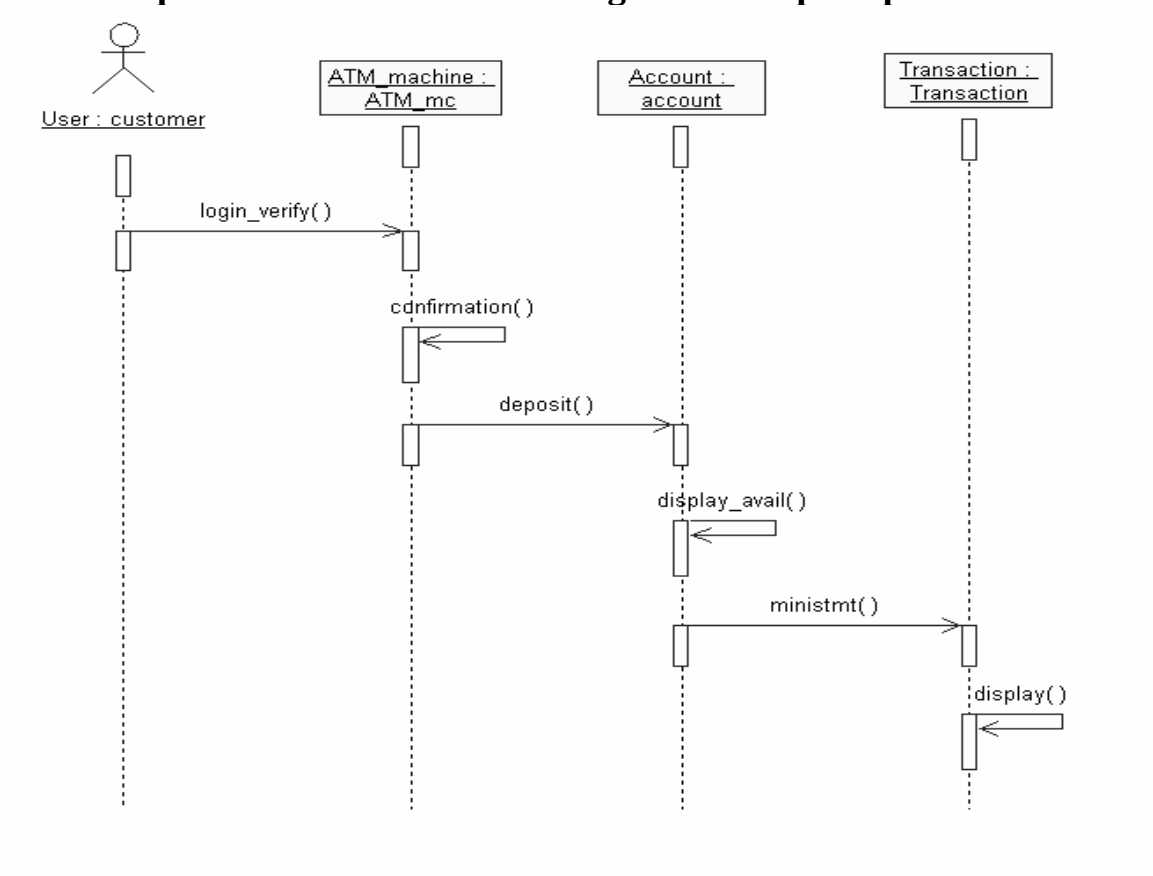
A sequence diagram represents the sequence and interactions of a given USE-CASE or scenario. Sequence diagrams can capture most of the information about the system. Most object to object interactions and operations are considered events and events include signals, inputs, decisions, interrupts, transitions and actions to or from users or external devices.

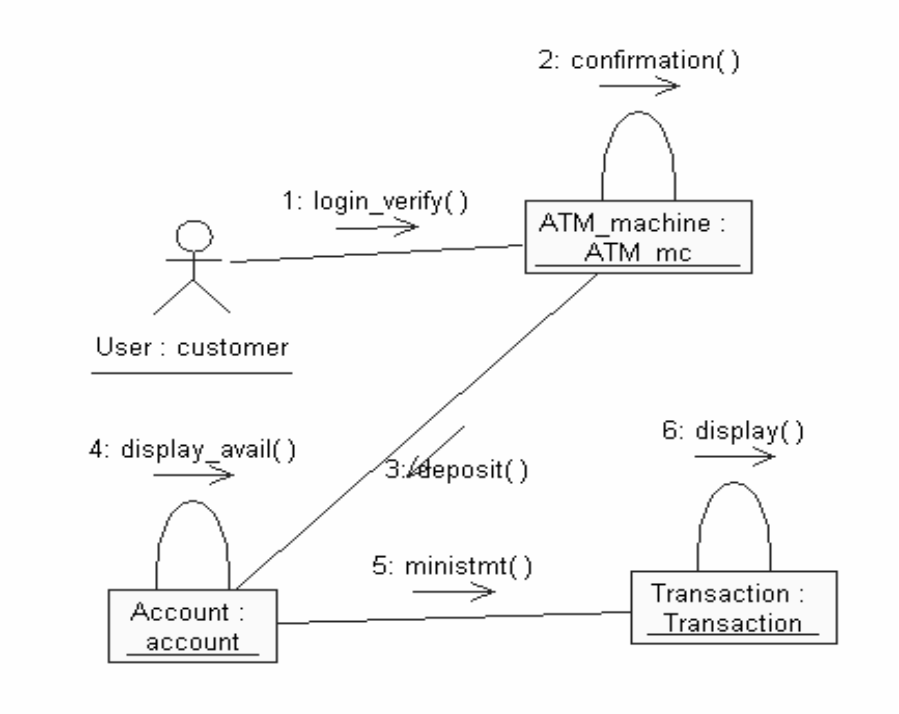
An event also is considered to be any action by an object that sends information. The event line represents a message sent from one object to another, in which the “form” object is requesting an operation be performed by the “to” object. The “to” object performs the operation using a method that the class contains.

It is also represented by the order in which things occur and how the objects in the system send message to one another.

The sequence diagram for each USE-CASE that exists when a user withdraws, deposits, needs information about ATM machine status and account are drowned.

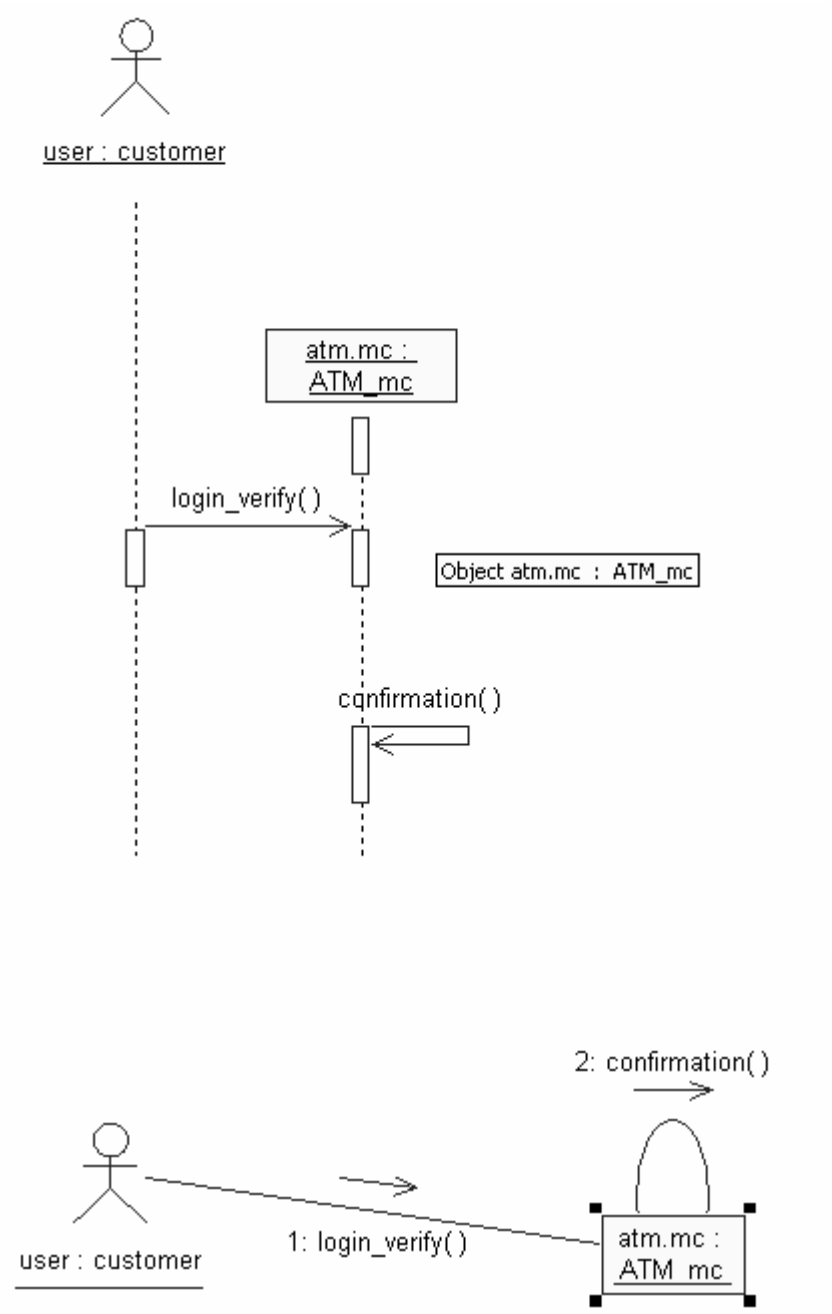
Sequence and collaboration diagram for deposit process





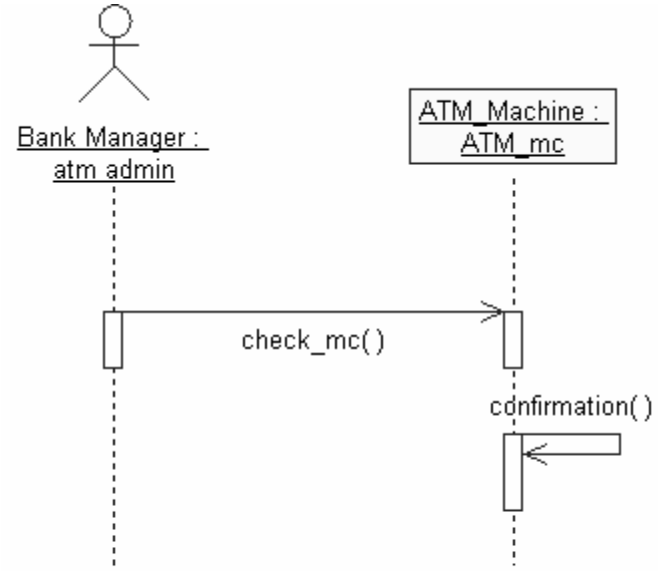
The diagrams show the entire deposit process in an ATM system. The user has to login to the ATM machine and deposit the amount of money as required by the user. The user may wish to get a ministatement and a screen about the details of the transaction.

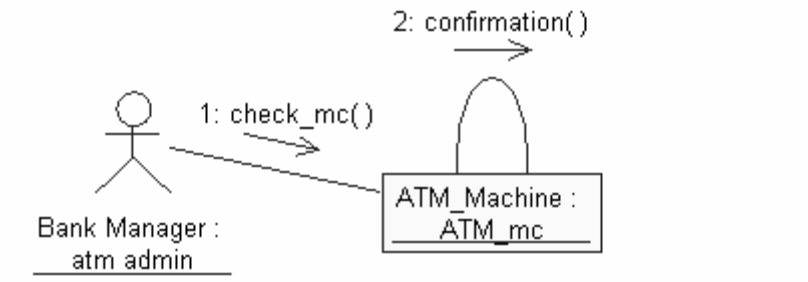
Sequence and collaboration diagram for login



The diagrams show the process of login by the user to the ATM system. The user has to enter his details. The details entered are verified by the system and the user is approved if the details match, otherwise an appropriate error message is displayed.

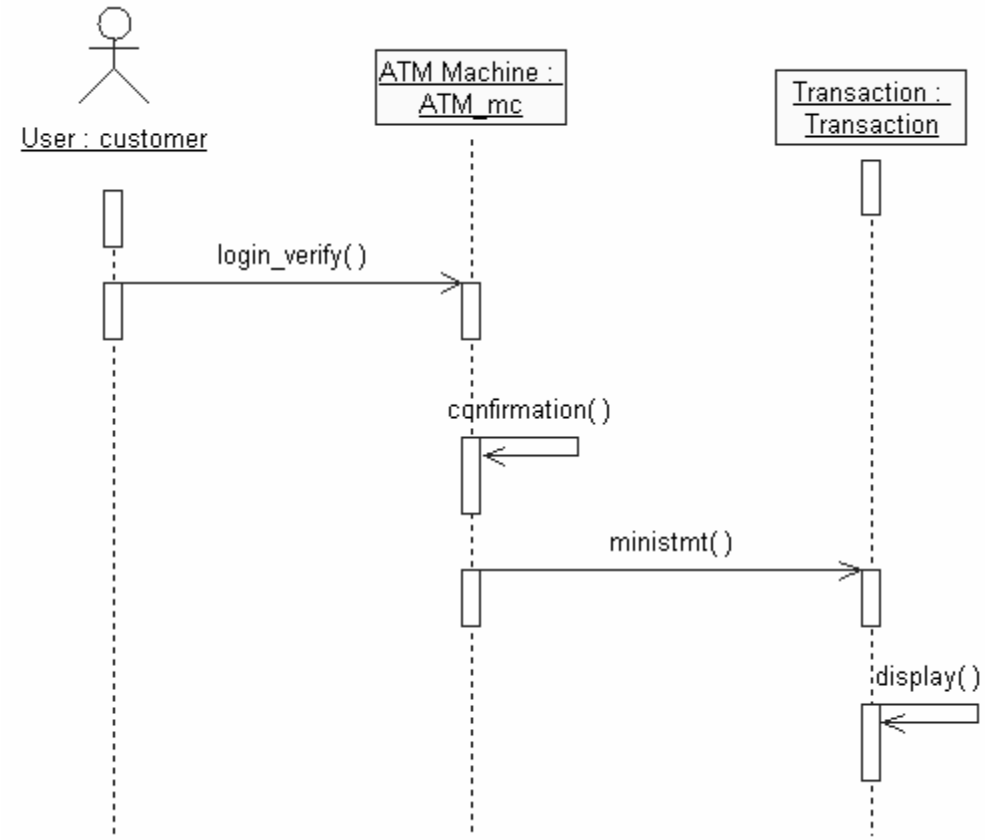
Sequence and collaboration diagram for checking machine status

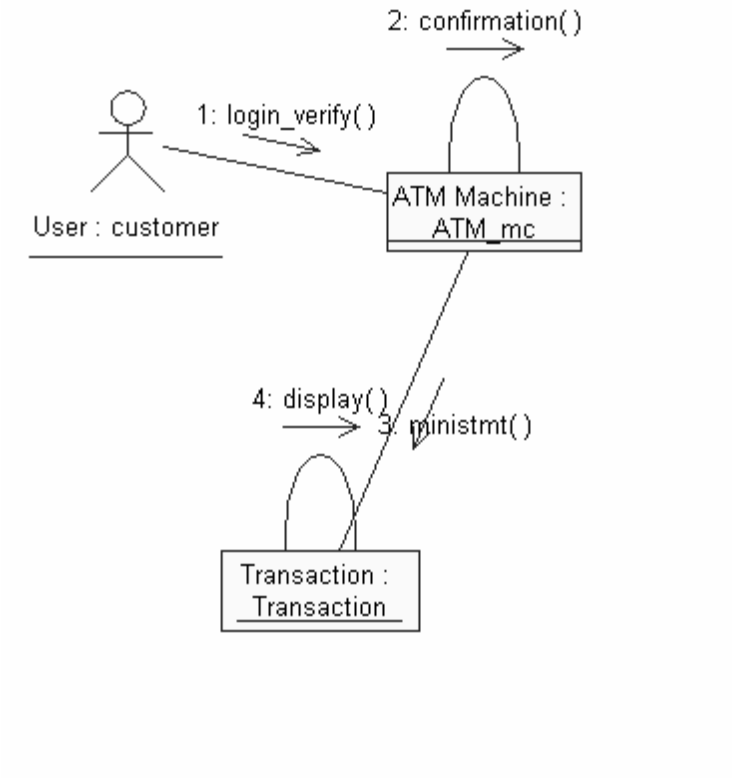




The Administrator of the ATM system has to maintain the details about the ATM, He has to check if there is enough money in the ATM and if the ATM is functional without major errors. For this, he may check the ATM machine status occasionally. The process is shown in the above diagrams.

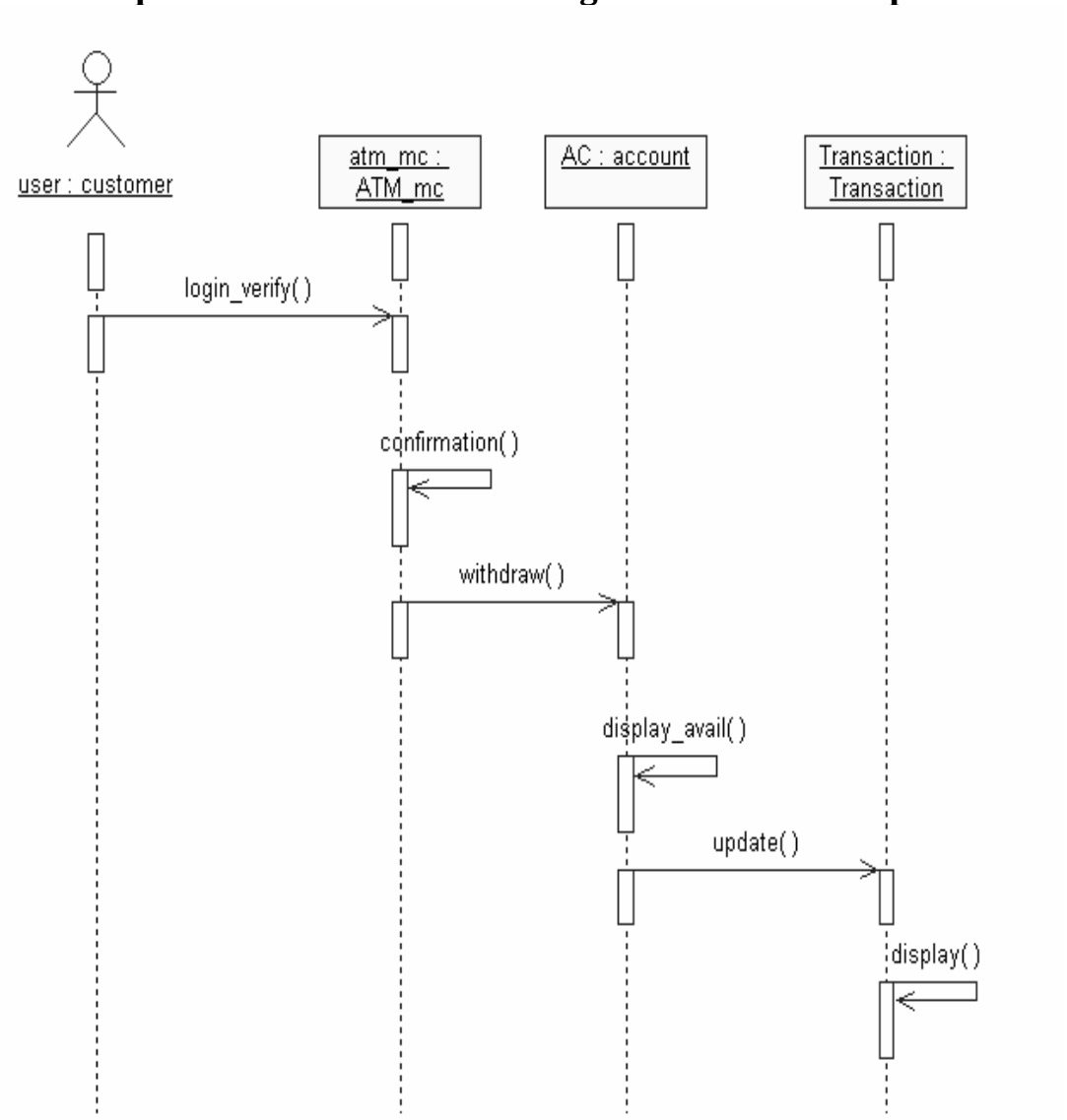
Sequence and collaboration diagram for printing ministatement

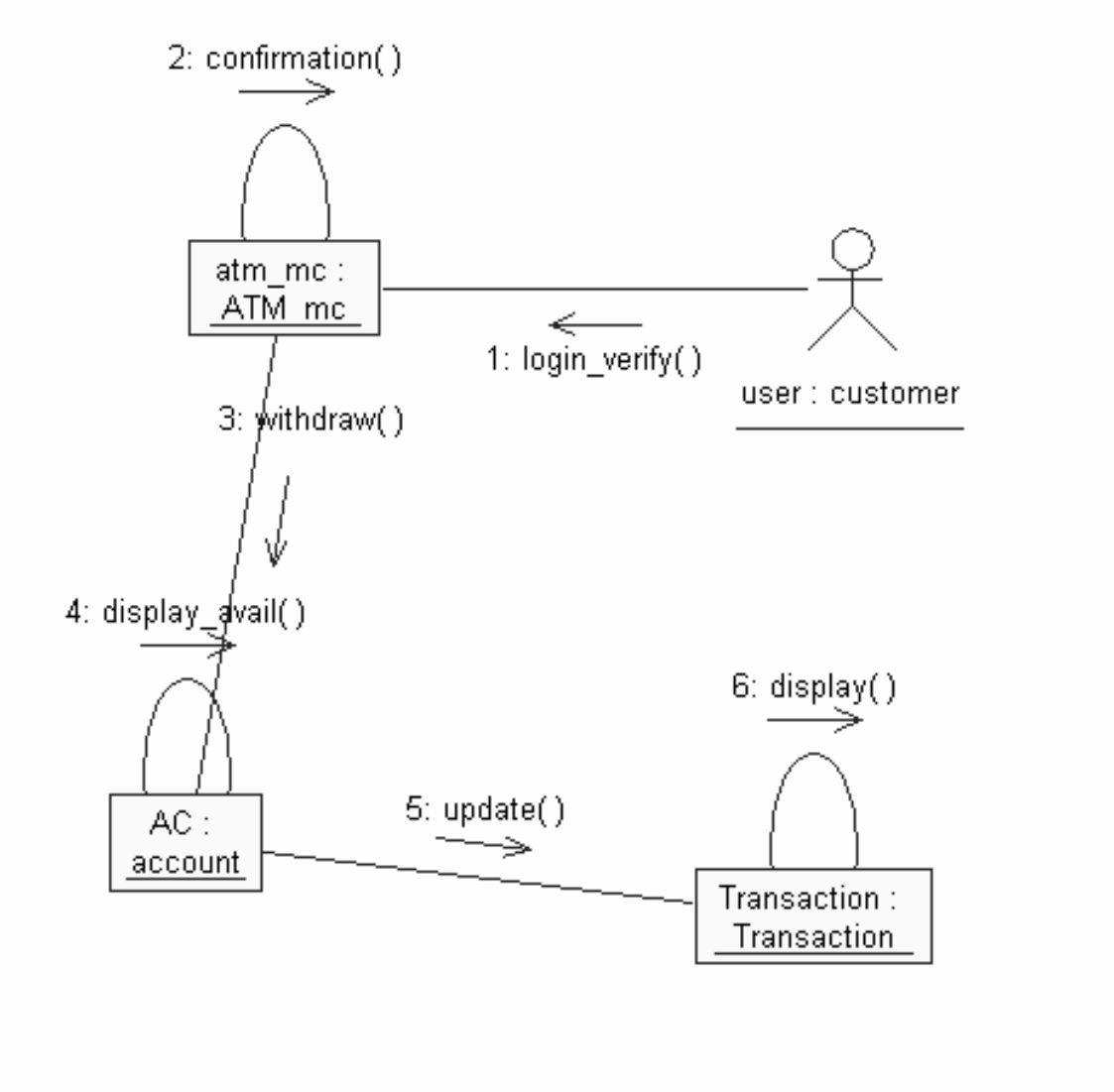




After a transaction is carried out successfully, the user must get a ministatement to tell him his account's details such as balance and transaction number. This process is depicted in the above diagrams.

Sequence and collaboration diagram for withdraw process





The user can make withdraw money from his account. The process is depicted in the diagrams above. The user has to login to the system using his username and password, which are verified by the system. After successful verification, the user can choose the amount of money he wants to withdraw from his account. The amount specified by the user is checked by the system to make sure there is enough balance in his account to carry out the transaction. After the transaction is carried out the resulting amount is displayed and the details are updated to the database.

TABLES:

ACCOUNT TABLE

act_table	
acctno	**bal**
111	1380
888	8000

TRANSACTION TABLE

tran_table				
acctno	**t-date**	**t-type**	**t-amt**	**t-bal**
111	9/5/2009	wd	50	950
111	9/5/2009	wd	40	910
111	9/5/2009	wd	100	1010
111	9/5/2009	wd	30	980
111	9/5/2009	deposit	400	1380

USER TABLE

user_table		
Username	**password**	**acctno**
Aaa	bbb	111
Ram	666	888

CODING:

ACCOUNT CLASS:

```
Option Explicit

Private acctno As String

Private bal As Integer

Public NewProperty As atm_mc

Public NewProperty2 As New transaction

Dim db As Database
Dim rs As Recordset
```

```
Public Sub withdraw(amount As Integer)
Set db = OpenDatabase("C:\Jaga\atm.mdb")
Set rs = db.OpenRecordset("act_table")


rs.MoveFirst
Do While Not rs.EOF
If rs(0).Value = Trim(Form1.Label1.Caption) Then
   If rs(1).Value > amount Then
            rs.Edit
            rs(0).Value = Trim(Form1.Label1.Caption)
            rs(1).Value = rs(1).Value - amount
            rs.Update
            MsgBox rs(1).Value

            Call NewProperty2.update_transaction(rs(0).Value, "wd", amount,
rs(1).Value)
            Call display_availability(1)


      Exit Do
   Else
      Call display_availability(0)
   End If
End If
rs.MoveNext
Loop

rs.Close
db.Close

End Sub

Public Sub deposit(amount As Integer)
Set db = OpenDatabase("C:\Jaga\atm.mdb")
Set rs = db.OpenRecordset("act_table")

rs.MoveFirst
Do While Not rs.EOF
If rs(0).Value = Trim(Form1.Label1.Caption) Then

      rs.Edit
      rs(1).Value = rs(1).Value + amount
      rs.Update
      Call NewProperty2.update_transaction(rs(0).Value, "deposit", amount,
rs(1).Value)
      Exit Do
```

```
End If
rs.MoveNext
Loop
rs.Close
db.Close

End Sub

Public Sub display_availability(flag As Integer)
  If flag = 1 Then
    MsgBox "success"
  Else
    MsgBox "not available"
  End If


End Sub
```

ATM CLASS:

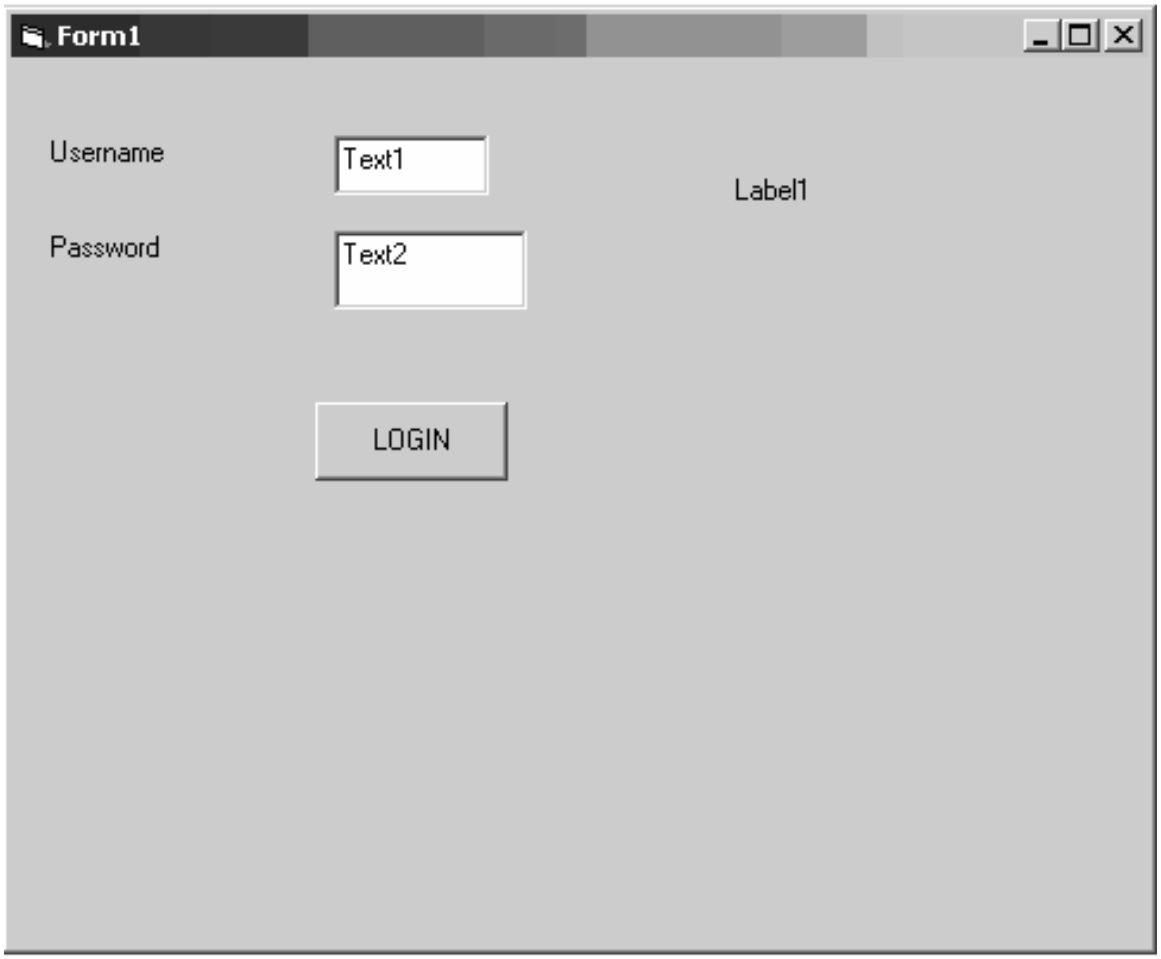
```
Option Explicit

Private t_bal As Integer

Private t_with As Integer

Public NewProperty As account
Dim db As Database
Dim rs As Recordset


Public Sub login_verify(username As String, password As String)
MsgBox "inside"
Set db = OpenDatabase("C:\Jaga\atm.mdb")
Set rs = db.OpenRecordset("user_table")
rs.MoveFirst
Do While Not rs.EOF
If rs(0).Value = username Then
    If rs(1).Value = password Then
       MsgBox rs(0).Value
       Form1.Label1.Caption = rs(2).Value
       Form1.Hide
       Form2.Show
       Exit Do
       End If
End If
rs.MoveNext
Loop
End Sub
```

```
Public Sub check_mc_status()

End Sub

Public Sub display_confirmation(flag As Integer)

End Sub
```

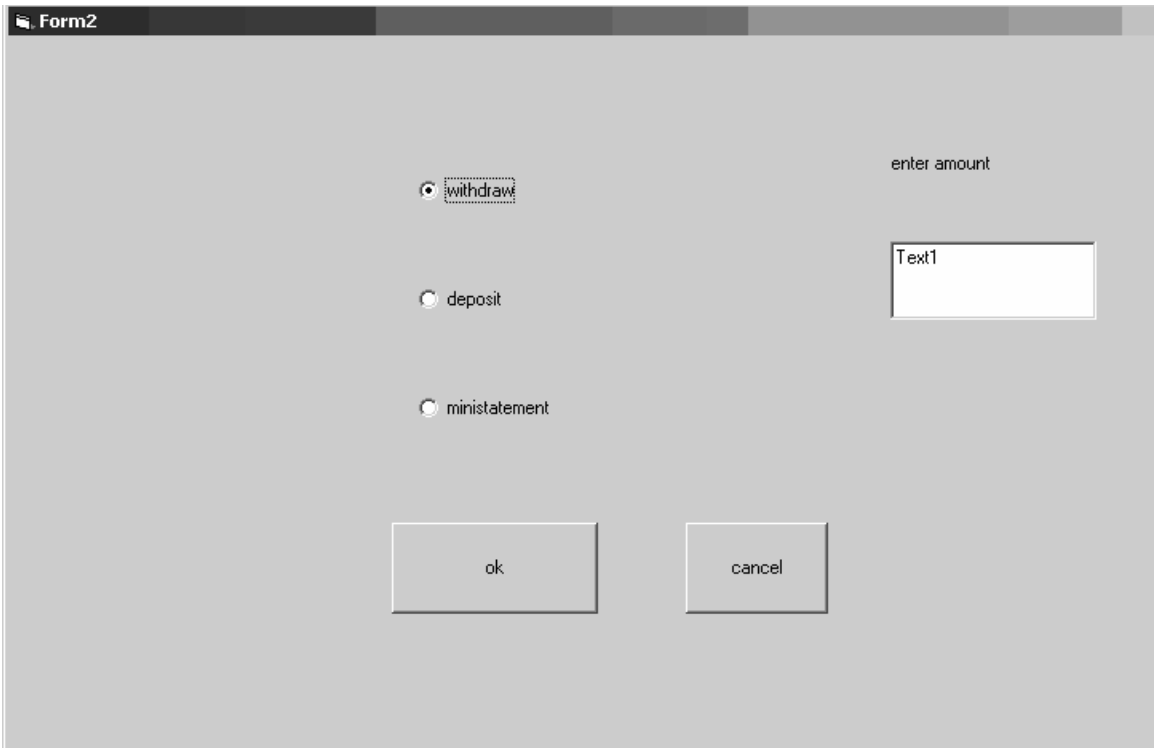
TRANSACTION CLASS:

```
Option Explicit

Private t_date As Variant

Private t_type As Variant

Private t_amt As Variant

Private t_bal As Variant

Public NewProperty As account
Dim db As Database
Dim rs As Recordset

Public Sub ministatement(actno As String)

End Sub

Public Sub update_transaction(actno As String, ttype As String, amount As Integer,
bal As Integer)
Set db = OpenDatabase("C:\Jaga\atm.mdb")
Set rs = db.OpenRecordset("tran_table")
'adding withdraw detail in  transaction table
        rs.AddNew
        'MsgBox Date

        rs(0).Value = actno
        rs(1).Value = Date
        If ttype = "wd" Then

        rs(2).Value = "wd"
        Else
        rs(2).Value = "deposit"
        End If

        rs(3).Value = amount
        rs(4).Value = bal
        rs.Update
        'end of updating transaction table
        rs.Close
```

```
        db.Close
    End Sub
Public Sub display()
End Sub
```




```
Private Sub logincmd_Click()

Dim log1 As atm_mc
Set log1 = New atm_mc
MsgBox "hi"
Call log1.login_verify(Trim(Text1.Text), Trim(Text2.Text))


End Sub
```




```
Private Sub Command1_Click()
Dim act As account
Dim tn As transaction
Set act = New account
Set tn = New transaction
If Option1.Value = True Then
MsgBox "withdraw"
Call act.withdraw(Trim(Text1.Text))
Else
If Option2.Value = True Then
MsgBox "deposit"
Call act.deposit(Trim(Text1.Text))
Else
If Option3.Value = True Then
MsgBox "mini"
Call tn.ministatement(Form1.Label1.Caption)
End If
End If
End If
End Sub
```

RESULT:

Thus the system for ATM is created and executed. The output is verified.

EX. NO.: **STUDENT MARK ANALYSIS SYSTEM**
DATE:

AIM:

To create a system to analyze the students marks stored in the database.

PROBLEM STATEMENT:

The purpose of this system is to analyze and perform operation on data stored in the database and to provide authentication to avoid unauthorized access by using MS-Access as back end and VB to use as the front end. The entire system is divided into various modules and it has its own objects and classes.

Reports are provided on their basis:

- Student marks
- Average
- Percentage

The main advantage in our system is that data entry becomes very easy and other manipulation, updating can be done easily.

Form description:

Form1: Login Form

Here the user name and password is checked and access is granted if the verified data is correct.

Form2: Student Profile Form

Here the detail of each student is entered and it is stored in the database. We can also search the details of a particular student and also make changes to the available data.

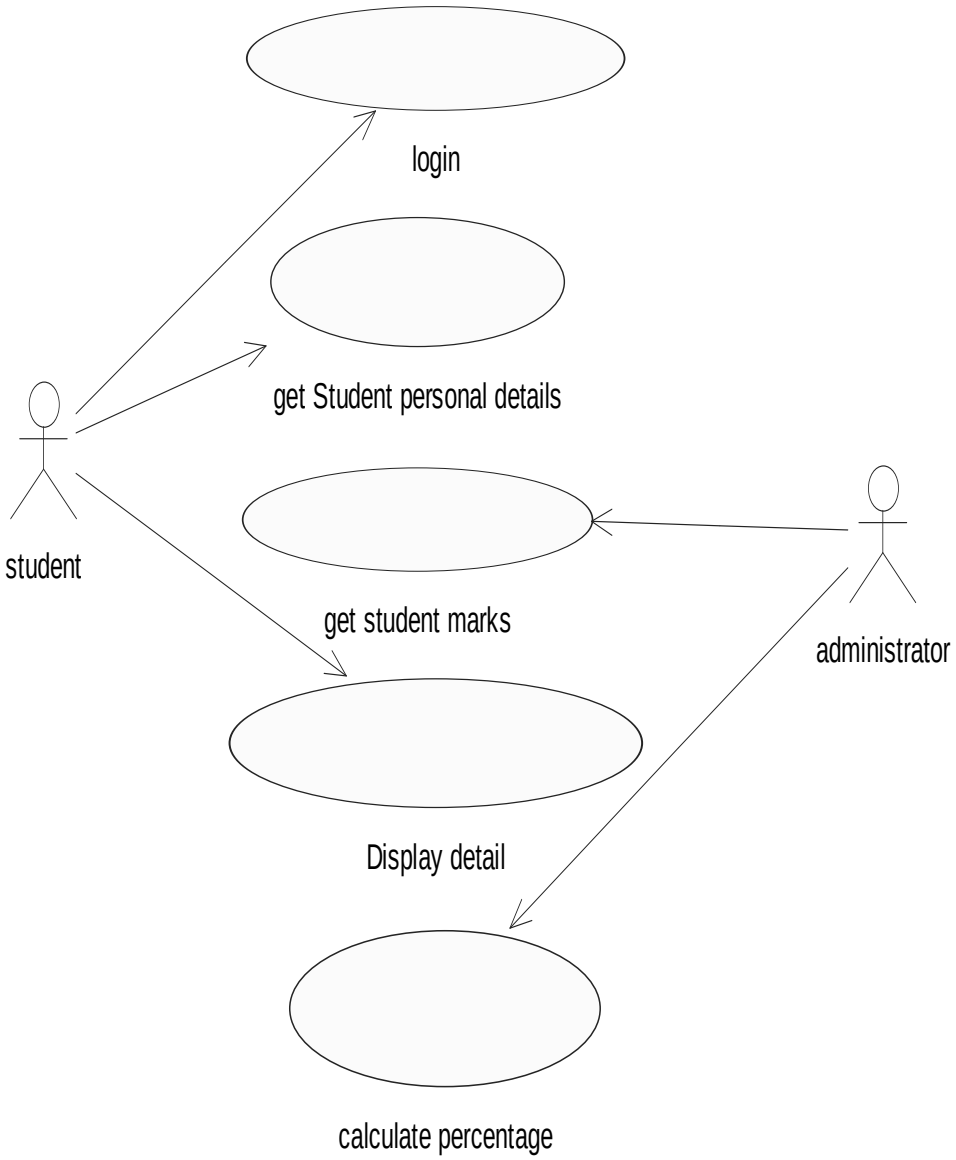
Form 3: Academic Details

Here the student name, register no, department and all the subject marks are entered and stored in the database. Finally, we are calculating the total, average and percentage for each student.

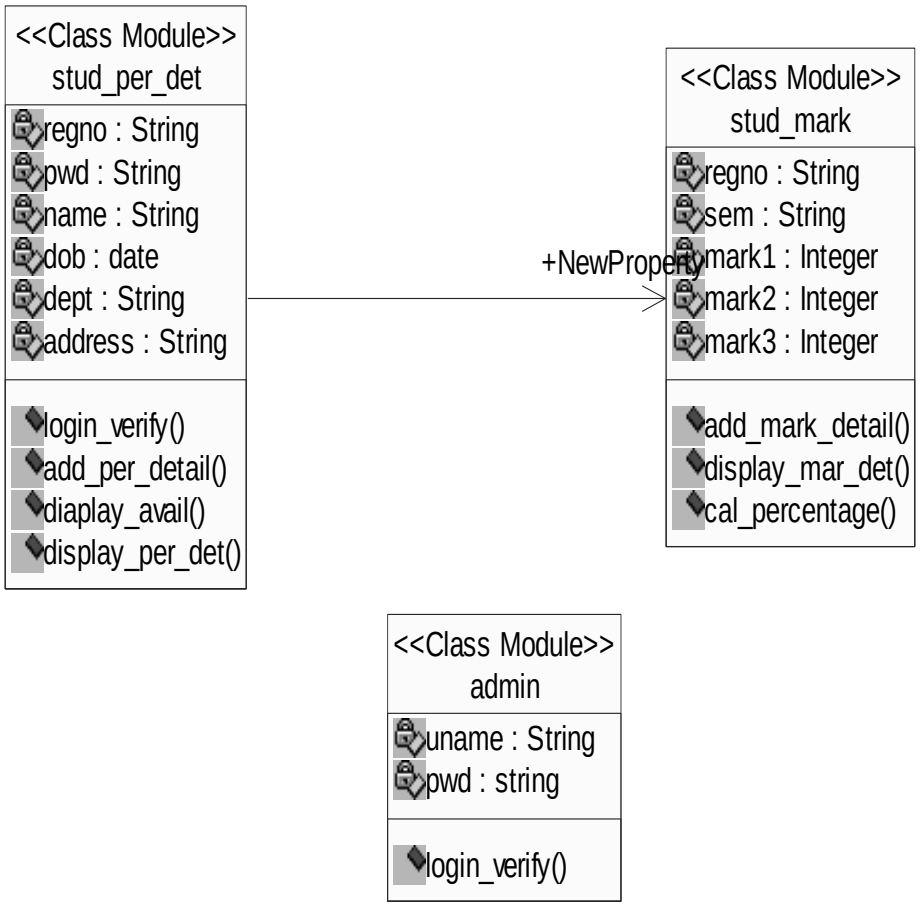
Software requirements:

Microsoft Visual Basic 6.0 is used as front- end for our project and MS-Access is used as back-end to store the data.

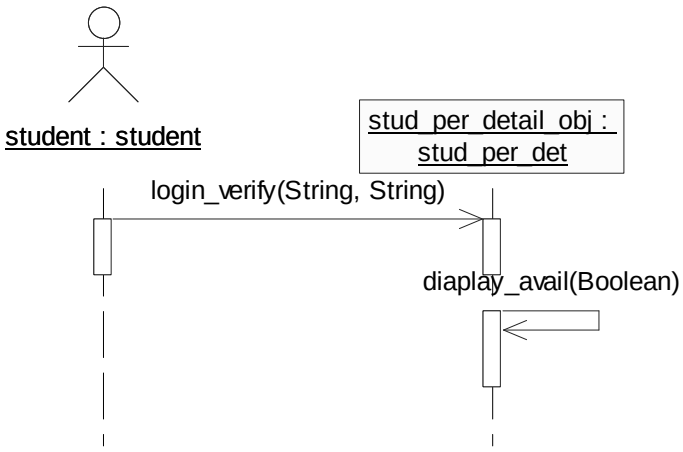
USE CASE DIAGRAM:



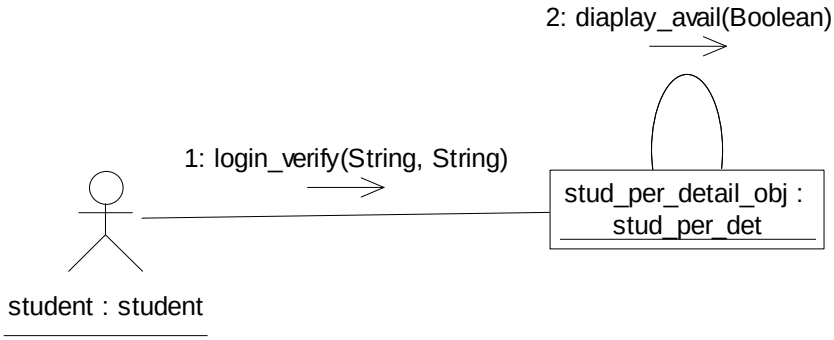
CLASS DIAGRAM:



LOGIN SEQUENCE DIAGRAM

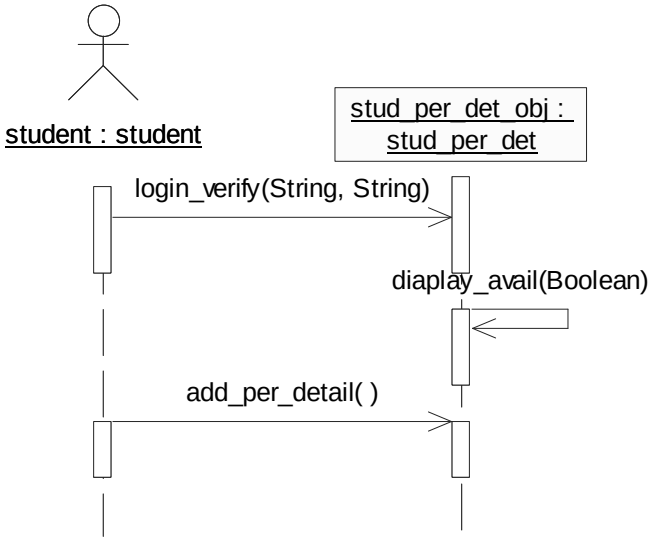


STUDENT MARK ANALYSIS: LOGIN COLLABORATION DIAGRAM

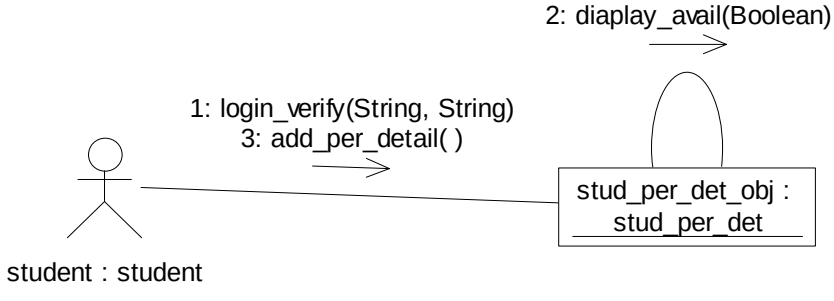


Student gets access to Student personal detail to provide information.

GET STUDENTS PERSONAL DETAILS :SEQUENCE DIAGRAM

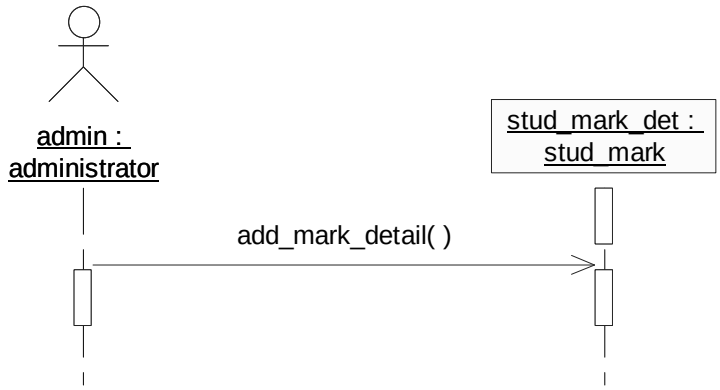


STUDENT MARK ANALYSIS: GET STUDENT PERSONAL DATA COLLABORATION DIAGRAM

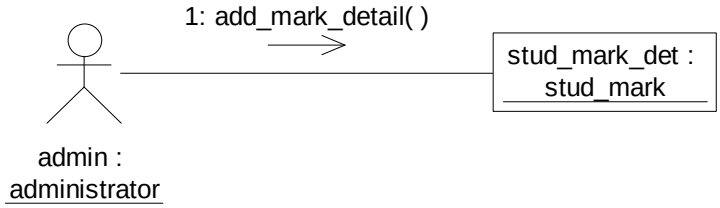


Student login is verified and their personal details are obtained.

GET STUDENTS MARK DETAILS: SEQUENCE DIAGRAM

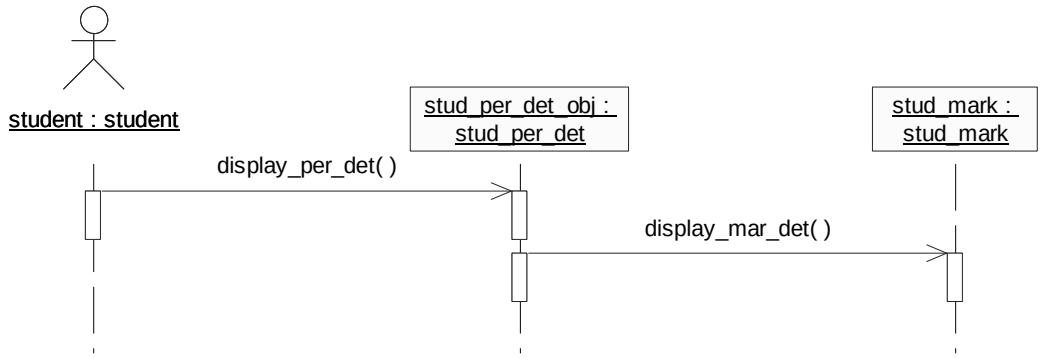


STUDENT MARK ANALYSIS: GET STUDENT MARK DETAILS COLLABORATION DIAGRAM

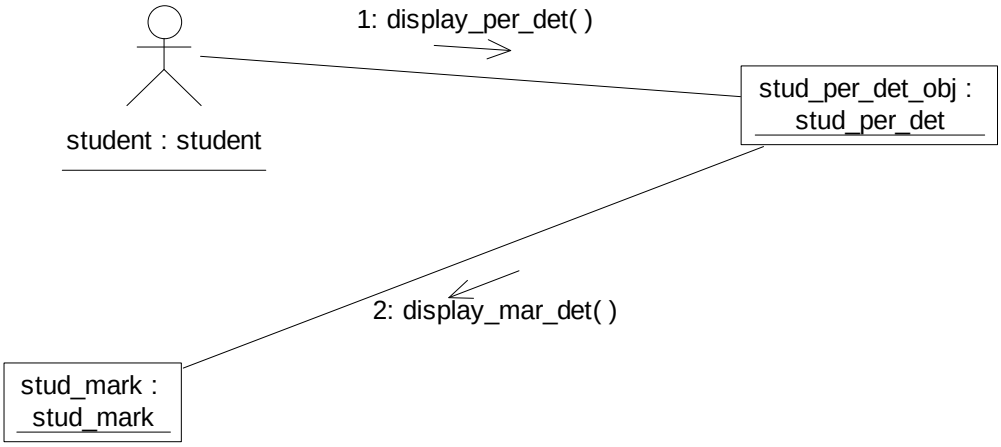


Administrator logins to add the mark details of students.

DISPLAY :SEQUENCE DIAGRAM

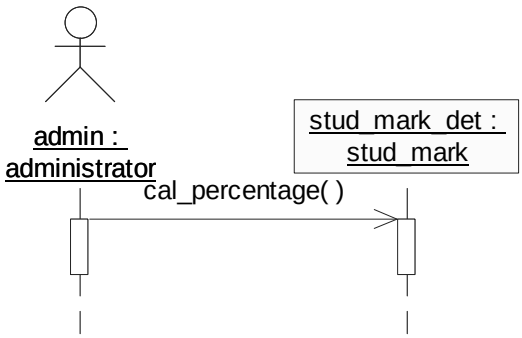


STUDENT MARK ANALYSIS: DISPLAY COLLABORATION DIAGRAM

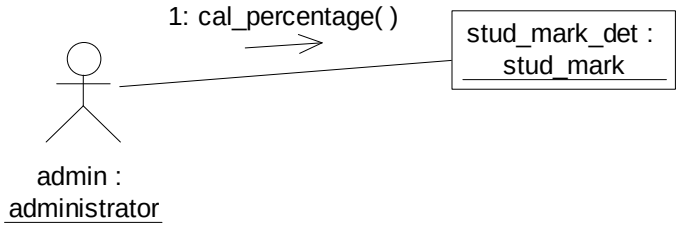


Students get access to display their personal details and mark details.

CALCULATE PERCENTAGE: SEQUENCE DIAGRAM



STUDENT MARK ANALYSIS: PERCENTAGE CALCULATION COLLABORATION DIAGRAM



Administrator calculates the percentage of student marks.

Table requirement:

There are two tables required. They are:

Student profile table:

In this table, we get the details of the students like name, father's name, date of birth, age, regulation, hobbies, e-mail id, address, blood group and contact number. It also has fields to search a particular student details, academic details and to modify the existing data.

Academic table:

In this table, we get the name, register no, department, marks for each subject, total and average. It also has fields to search , add, update and to calculate the percentage.

DATABASE TABLES:

Admin	
ad_name	**ad_pwd**
adm	Adm

stud_mark					
a_regno	**s_sem**	**s_mark1**	**s_mark2**	**s_mark3**	**s_per**
222	1	80	90	90	87

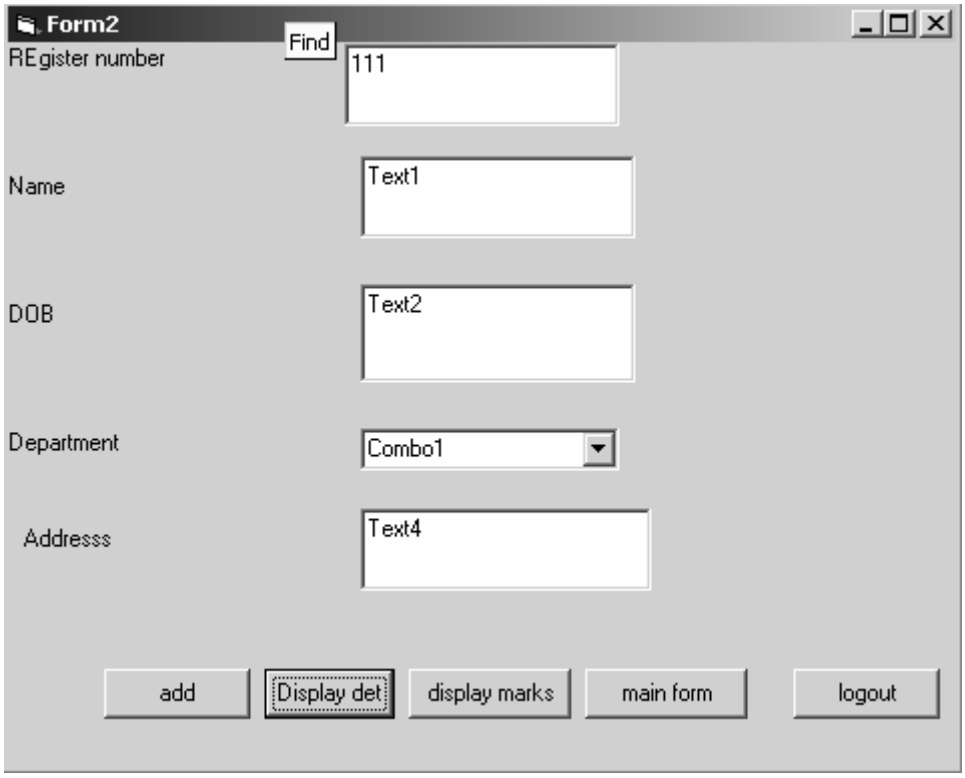
stud_per					
s_regno	**s_pwd**	**s_name**	**s_dob**	**s_dept**	**s_address**
111	aaa	arunraj	2/2/1997	CSE	Text4
222	bbb	abilasha	10/13/1987	ECE	kkdi
333	ccc	uma	12/10/1980	EEE	kkd

FORM 1 DESIGN:

FORM 1 CODING:

```
Private Sub Command1_Click()
Dim s As stud_per_det
Dim ad As admin

If Option1.Value = True Then
   Set s = New stud_per_det
   Call s.login_verify(Trim(Text1.Text), Trim(Text2.Text))
Else
   If Option2.Value = True Then
   Set ad = New admin
   Call ad.login_verify(Trim(Text1.Text), Trim(Text2.Text))
   End If
End If

End Sub
```

FORM2 DESIGN:

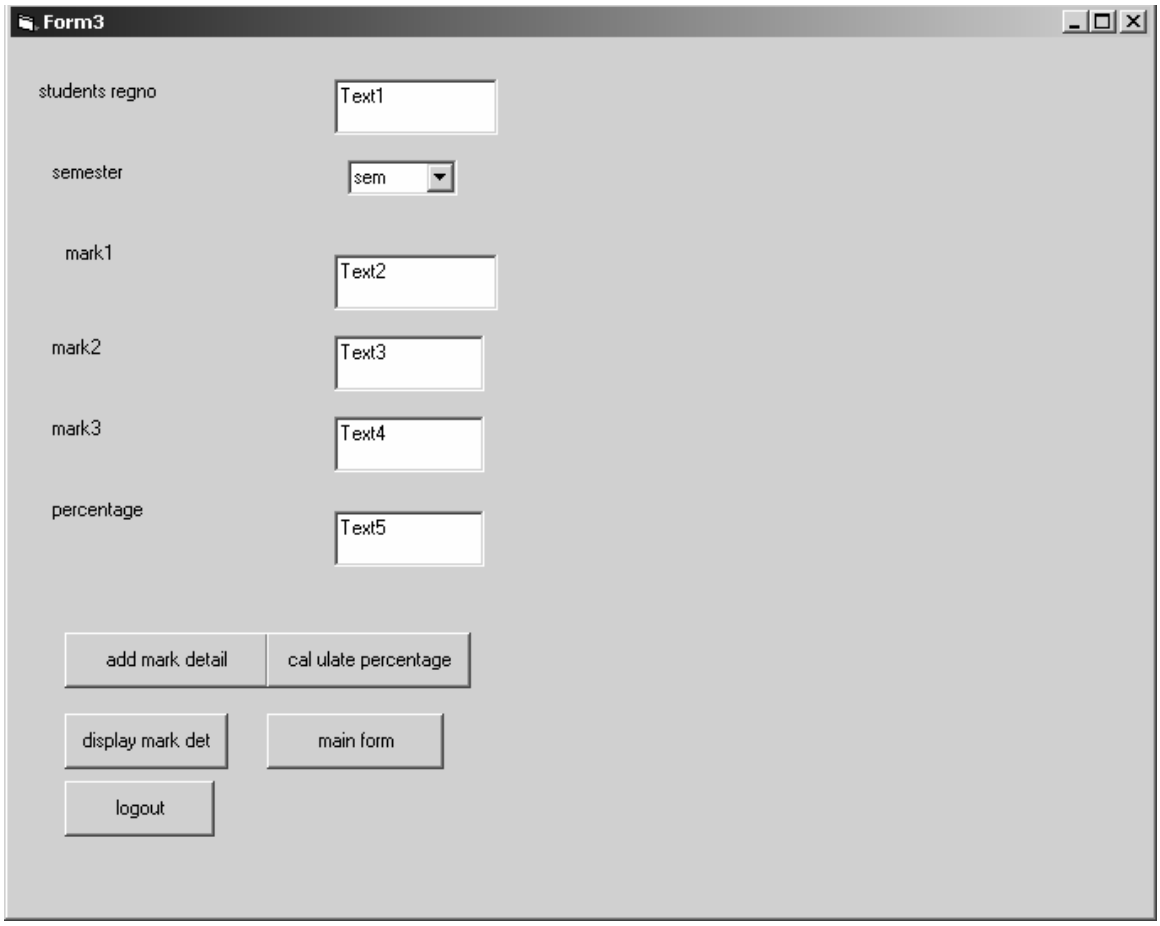


FORM 2 CODING:

```
Dim s_per As stud_per_det

Private Sub Command1_Click()
Set s_per = New stud_per_det

Call s_per.add_per_detail(Text1.Text, Text2.Text, Combo1.Text, Text4.Text)

End Sub

Private Sub Command2_Click()
Set s_per = New stud_per_det
Call s_per.display_per_det

End Sub

Private Sub Command3_Click()
Form2.Hide
Form3.Show
Form3.Command1.Visible = False
Form3.Command2.Visible = False
Form3.Text1.Text = Form2.Text3.Text

End Sub
```

```
Private Sub Command4_Click()
Form2.Hide
Form1.Show

End Sub

Private Sub Command5_Click()
End
End Sub

Private Sub Form_Load()
Text3.Text = Form1.Text1.Text

End Sub
```

FORM3 DESIGN:



FORM 3 CODING:

```
Dim s_mark As stud_mark

Private Sub Command1_Click()
Set s_mark = New stud_mark

Call s_mark.add_mark_detail(Trim(Text1.Text), Combo1.Text,
Int(Trim(Text2.Text)), Int(Trim(Text3.Text)), Int(Trim(Text4.Text)))

End Sub

Private Sub Command2_Click()
Set s_mark = New stud_mark
Call s_mark.cal_percentage(Trim(Text1.Text), Combo1.Text)
End Sub

Private Sub Command3_Click()
Set s_mark = New stud_mark
Call s_mark.display_mar_det(Trim(Text1.Text), Combo1.Text)
End Sub

Private Sub Command4_Click()
Form3.Hide
Form1.Show

End Sub

Private Sub Command5_Click()
End
End Sub
```

ADMIN CLASS MODULE:

```
Option Explicit

'##ModelId=4AB45CA30389
Private uname As String

'##ModelId=4AB45CA801D4
Private pwd As String
Dim db As Database
Dim rs As Recordset
```

```
'##ModelId=4AB45CD702A5
Public Sub login_verify(aname As String, apwd As String)
Set db = OpenDatabase("C:\Documents and Settings\cse1\Desktop\vino\studdb.mdb")
Set rs = db.OpenRecordset("admin")
rs.MoveFirst
Dim flag As Boolean
flag = False
Do While Not rs.EOF
If Trim(rs(0).Value) = aname Then
   If Trim(rs(1).Value) = apwd Then
      Form3.Show
      Form3.Command1.Visible = True
Form3.Command2.Visible = True
      Form1.Hide

      Exit Do
   End If
End If
rs.MoveNext

Loop

rs.Close
db.Close
End Sub
```

STUDENT PERSONAL DETAIL CLASS MODULE:

```
Option Explicit

'##ModelId=4AB1F3320269
Private regno As String

'##ModelId=4AB1F33A0375
Private pwd As String

'##ModelId=4AB1F33F01FE
Private name As String

'##ModelId=4AB1F3450065
Private dob As Date

'##ModelId=4AB1F34C02C5
Private dept As String

'##ModelId=4AB1F34E033B
Private address As String
```

```
'##ModelId=4AB1F3D20371
Public NewProperty As stud_mark
Dim db As Database
Dim rs As Recordset

'##ModelId=4AB1F36F0067
Public Sub login_verify(regno As String, pwd As String)
Set db = OpenDatabase("C:\Documents and Settings\cse1\Desktop\vino\studdb.mdb")
Set rs = db.OpenRecordset("stud_per")
rs.MoveFirst
Dim flag As Boolean
flag = False
Do While Not rs.EOF
If Trim(rs(0).Value) = regno Then
   If Trim(rs(1).Value) = pwd Then
      flag = True

      Exit Do
   End If
End If
rs.MoveNext

Loop

Call display_avail(flag)

rs.Close
db.Close
End Sub

'##ModelId=4AB1F3780097
Public Sub add_per_detail(sname As String, sdate As String, sdept As String, sadd As
String)
Set db = OpenDatabase("C:\Documents and Settings\cse1\Desktop\vino\studdb.mdb")
Set rs = db.OpenRecordset("stud_per")
rs.MoveFirst
Do While Not rs.EOF
If Trim(rs(0).Value) = Trim(Form2.Text3.Text) Then

rs.Edit
rs(2).Value = sname
rs(3).Value = sdate
rs(4).Value = sdept
rs(5).Value = sadd
rs.Update
MsgBox "det added"
Exit Do
End If
rs.MoveNext
```

```
Loop
rs.Close
db.Close
End Sub

'##ModelId=4AB1F4760303
Public Sub display_avail(flag As Boolean)
If flag = True Then
   MsgBox "valid student"
   Form1.Hide
   Form2.Show
Else
   MsgBox "invalid login"
   End If
End Sub

'##ModelId=4AB1F724028B
Public Sub display_per_det()
Set db = OpenDatabase("C:\Documents and Settings\cse1\Desktop\vino\studdb.mdb")
Set rs = db.OpenRecordset("stud_per")
rs.MoveFirst
Do While Not rs.EOF
If Trim(rs(0).Value) = Trim(Form2.Text3.Text) Then
Form2.Text1.Text = rs(2).Value
Form2.Text2.Text = rs(3).Value
Form2.Combo1.Text = rs(4).Value
Form2.Text4.Text = rs(5).Value
Exit Do
End If
rs.MoveNext
Loop
rs.Close
db.Close

End Sub
```

STUDENT MARK DETAIL CLASS MODULE:

```
Option Explicit

'##ModelId=4AB1F358025B
Private regno As String

'##ModelId=4AB1F35B00A1
Private sem As String

'##ModelId=4AB1F363004A
Private mark1 As Integer
```

```
'##ModelId=4AB1F36600BF
Private mark2 As Integer

'##ModelId=4AB1F3680146
Private mark3 As Integer
Dim db As Database
Dim rs As Recordset
'##ModelId=4AB1F3890359
Public Sub add_mark_detail(sregno As String, ssem As String, sm1 As Integer, sm2
As Integer, sm3 As Integer)
Set db = OpenDatabase("C:\Documents and Settings\cse1\Desktop\vino\studdb.mdb")
Set rs = db.OpenRecordset("stud_mark")
MsgBox "inside"
rs.AddNew
rs(0).Value = sregno
rs(1).Value = ssem
rs(2).Value = sm1
rs(3).Value = sm2
rs(4).Value = sm3
rs(5).Value = 0
rs.Update
MsgBox "added"

End Sub

'##ModelId=4AB1F3B000B6
Public Sub display_mar_det(sregno As String, ssem As String)
Set db = OpenDatabase("C:\Documents and Settings\cse1\Desktop\vino\studdb.mdb")
Set rs = db.OpenRecordset("stud_mark")
rs.MoveFirst
Do While Not rs.EOF
If rs(0).Value = sregno Then
   If rs(1).Value = ssem Then
      Form3.Text2.Text = rs(2).Value
      Form3.Text3.Text = rs(3).Value
      Form3.Text4.Text = rs(4).Value
      Form3.Text5.Text = rs(5).Value
      Exit Do
   End If
End If
rs.MoveNext
Loop
End Sub

'##ModelId=4AB1F3B701D3
Public Sub cal_percentage(sregno As String, ssem As String)
Set db = OpenDatabase("C:\Documents and Settings\cse1\Desktop\vino\studdb.mdb")
Set rs = db.OpenRecordset("stud_mark")
rs.MoveFirst
Do While Not rs.EOF
```

```
If rs(0).Value = sregno Then
   If rs(1).Value = ssem Then
   rs.Edit
      rs(5).Value = (rs(2).Value + rs(3).Value + rs(4).Value) / 3

      Form3.Text5.Text = rs(5).Value
      rs.Update

      Exit Do
   End If
End If
rs.MoveNext
Loop
rs.Close
db.Close

End Sub
```

RESULT:

Thus the system for Student mark analysis is created and executed. The output is verified.

Ex. NO.: **Employee Payroll system**
DATE:

Aim

To create a computer system to process employee's payroll.

Problem statement

- This system is built for employee and manager.
- The employee can view his salary and the manager can add a new employee, update employee details and calculate and credit the salary.
- The employee and manager can log into the system providing their respective user names at login, if the entered input is incorrect an appropriate message is displayed.

System requirements

Microsoft visual basic 6.0 is used as front-end for our project and ms-access as our back-end.

Use-case diagram

The payroll use cases in our system are:

1 .Login
2. Add employee
3. Update employee details
4. Delete employee
5. Calculate salary
6. View employee details
7. Logout

Actors involved

1. Manager
2. Employee

Use-case name: Login

The user enters the username and password and chooses if the user is employee or administrator. If entered details are valid, the user's account becomes available.
If it is invalid, an appropriate message is displayed to the user.

Use-case name: Add employee

Only the manager has access to this use-case. The manager has to provide the details of the employee to create an account for a new employee. If the employee added already exits an appropriate message is displayed.

Use-case name: Update employee details

This use-case can be accessed only by the manager and not by the employee. The desired employee is searched for and the details of the employee are updated with new entries. If the employee searched for is not available an appropriate message is displayed.

Use-case name: Delete employee

This use-case can also be used only by the manager. It is used to delete the record of an employee if it is not necessary. The employee identity is provided by the manager which is searched for and if it is found, it is deleted. If it is not found an appropriate message is displayed.

Use-case: calculate salary

This use-case is used to calculate the salary of the employee after adding allowances and deducting as necessary. The net salary is displayed. This use-case can be used by both employee and by the administrator. The employee details are searched for and the net salary is calculated.

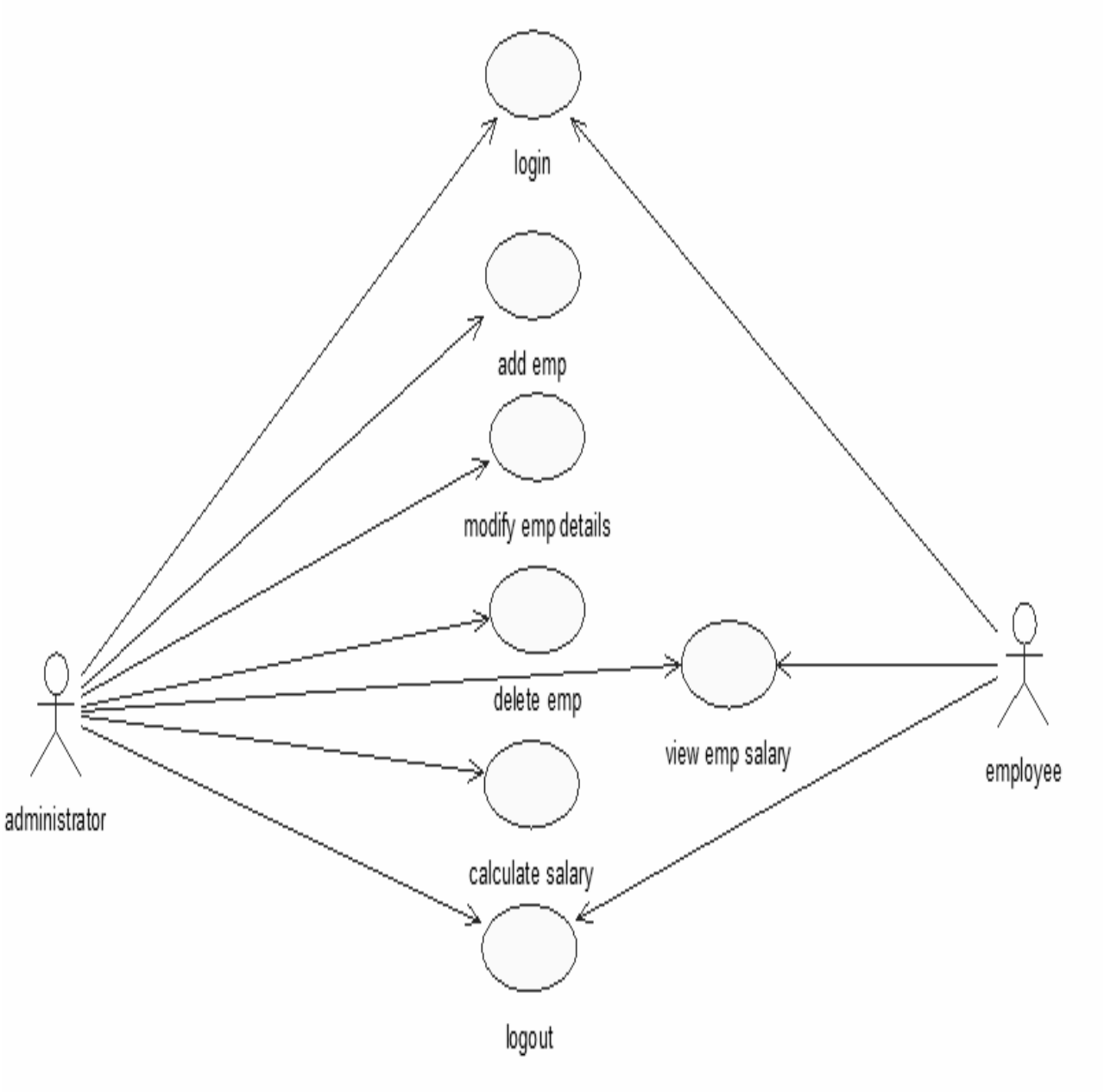
Use-case: View employee details

This use-case is used to display details of the employee. If the employee searched for is not available, an appropriate message is displayed.

Use-case: logout

After all the necessary operations are complete, this use-case is used to logout of the current account.

Use-case diagram for payroll system



Class diagram

The class diagram is a graphical representation of all the classes used in the system and their operations, attributes and relationships.

The payroll system makes use of the following classes:

1. System
2. Employee details
3. Employee salary

1) System

It consists of two attributes and two operations. The attributes are username and password. The operations used are login () and logout ().

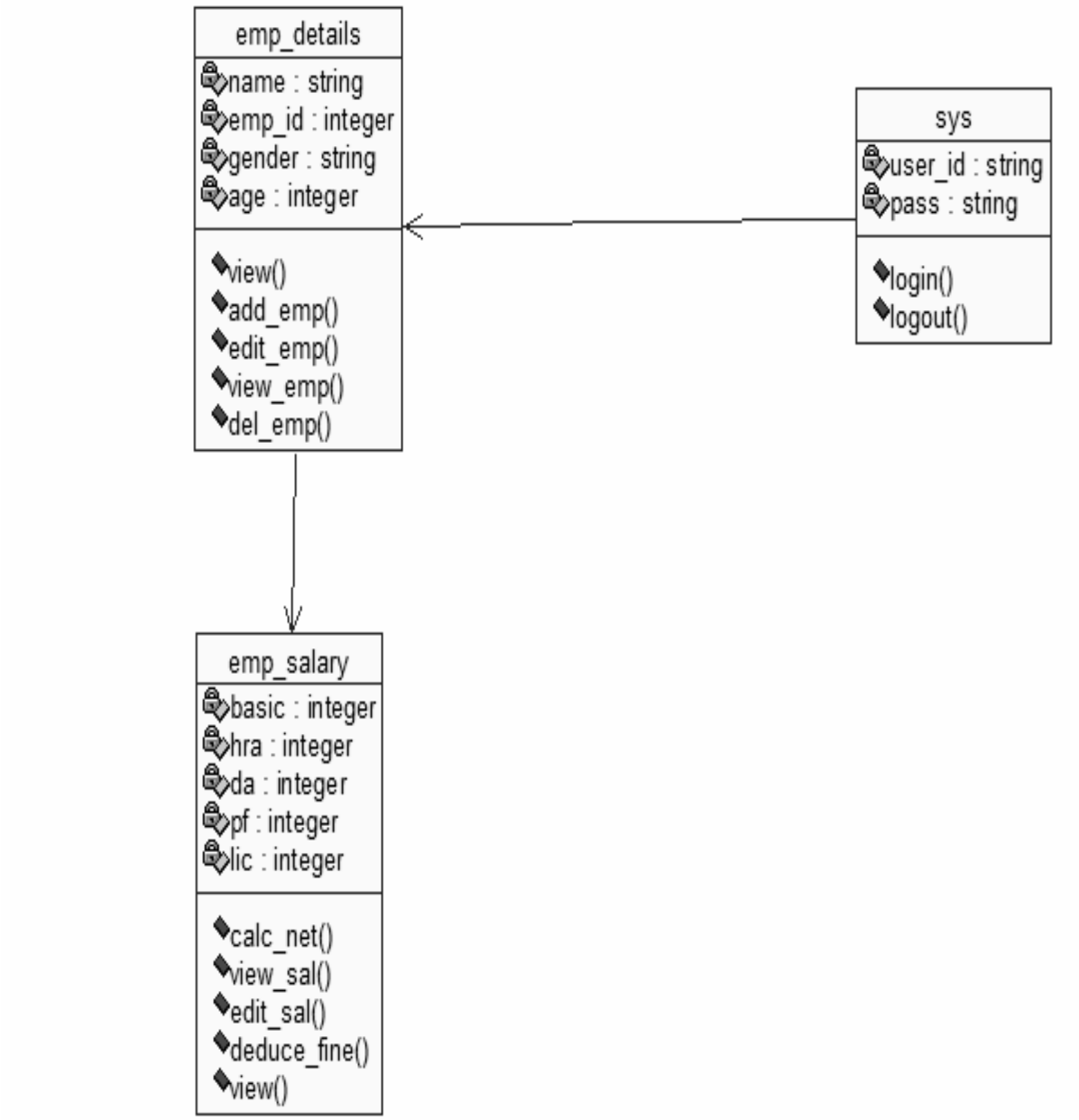
2) Employee details

It is used to store the personal details of the employee, such as name, employee id, gender and age. The operations available in this class are view(), add employee(), edit employee details(), view employee details(), delete employee details().

3) Employee salary

It is used to store the salary details of the employee such as basic pay, house rent allowance, dearness allowance, provident fund and insuranece. The operations available are calculate net salary(), view salary details(), update salary details() and deduce fine().

Class diagram for payroll system



Sequence diagram

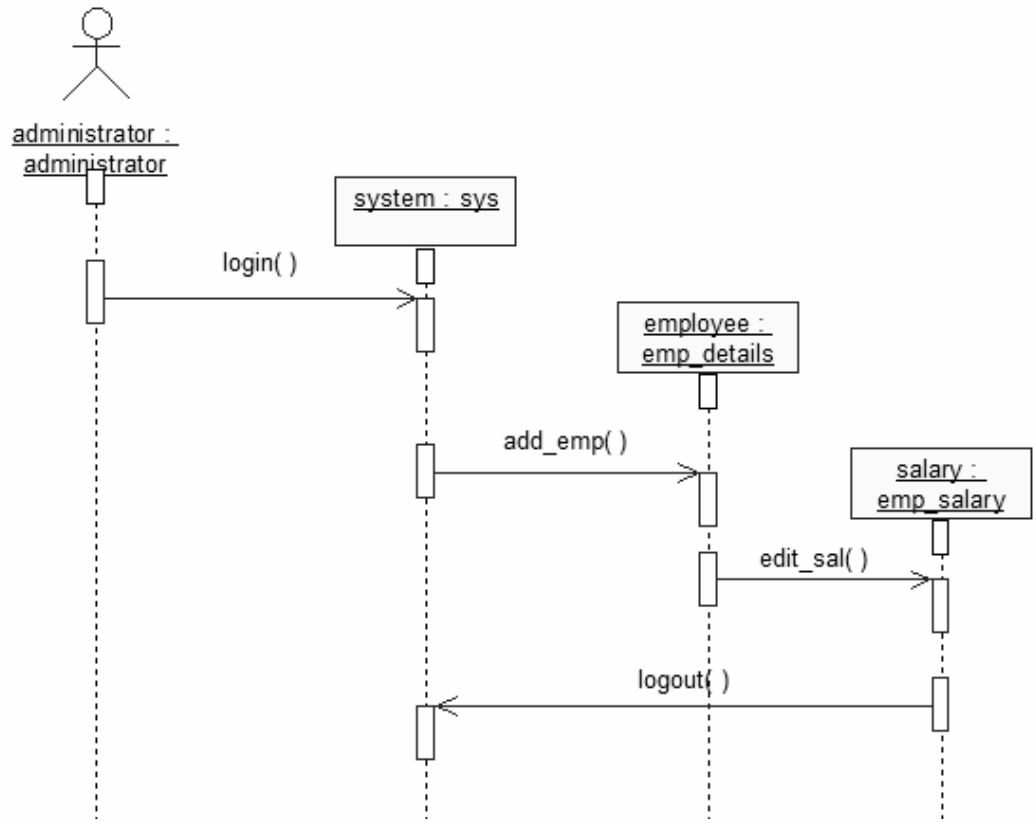
A sequence diagram represents the sequence and interactions of a given use-case or scenario. Sequence diagrams can capture most of the information about the system. Most object-to-object interactions and operations are considered events and events include signals, inputs, decisions, interrupts, transitions and actions to or from users or external devices.

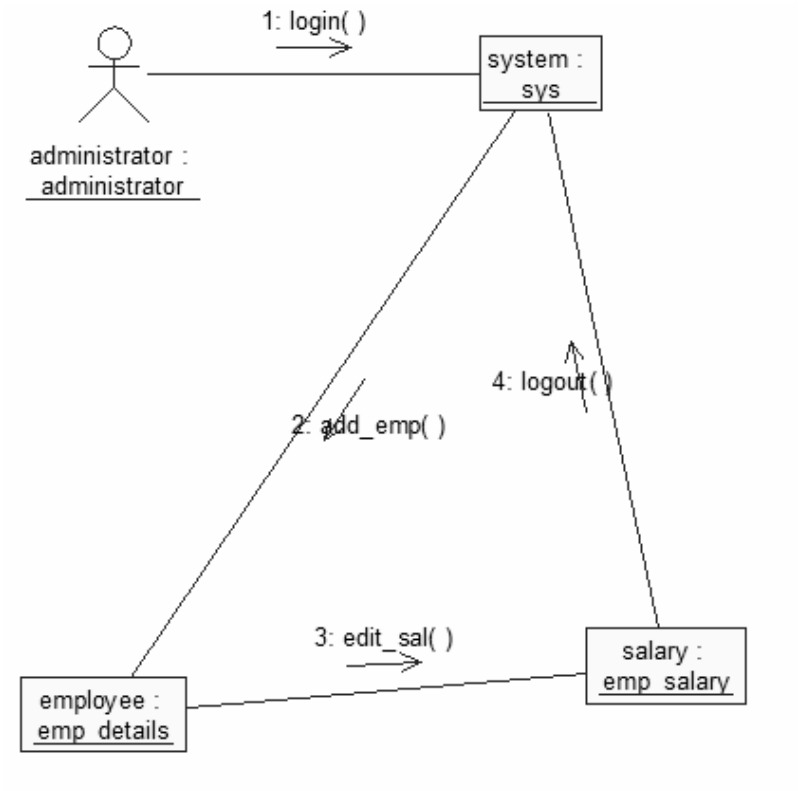
An event also is considered to be any action by an object that sends information. The event line represents a message from one object to another, in which the “from” object is requesting an operation be performed by the “to” object. The “to” object performs the operation using a method that the class contains.

It is also represented by the order in which things occur and how the objects in the system send message to one another.

The sequence diagram for each use-case that exists when a user logs in, adds, views, updates or deletes records in the system.

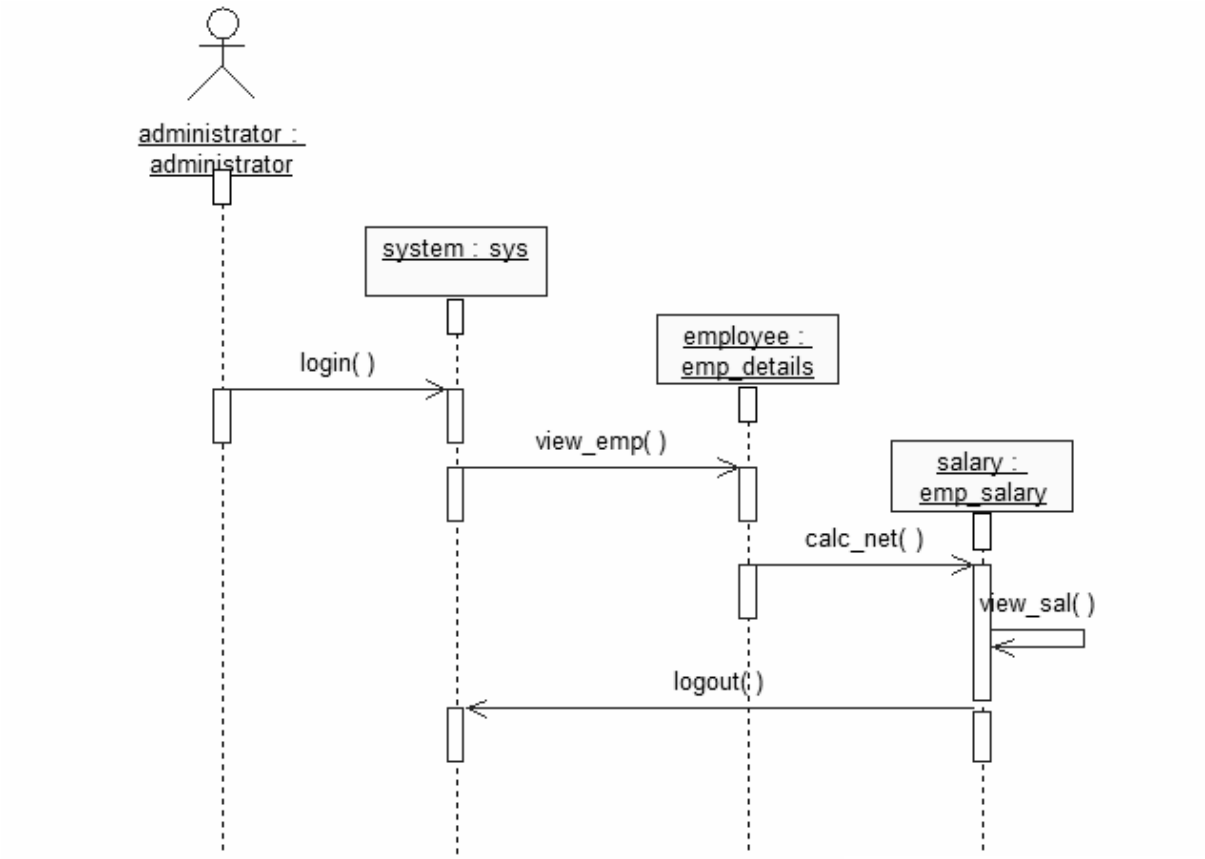
Sequence diagram and collaboration for adding an employee

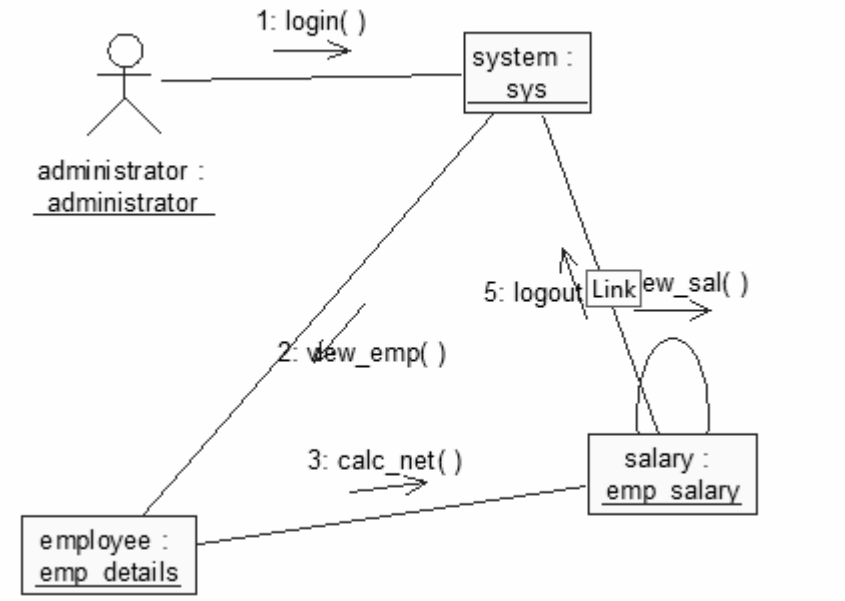




For the use case of adding an employee, a sequence and an interaction diagram are drawn. The diagrams show that administrator can only add a new employee.

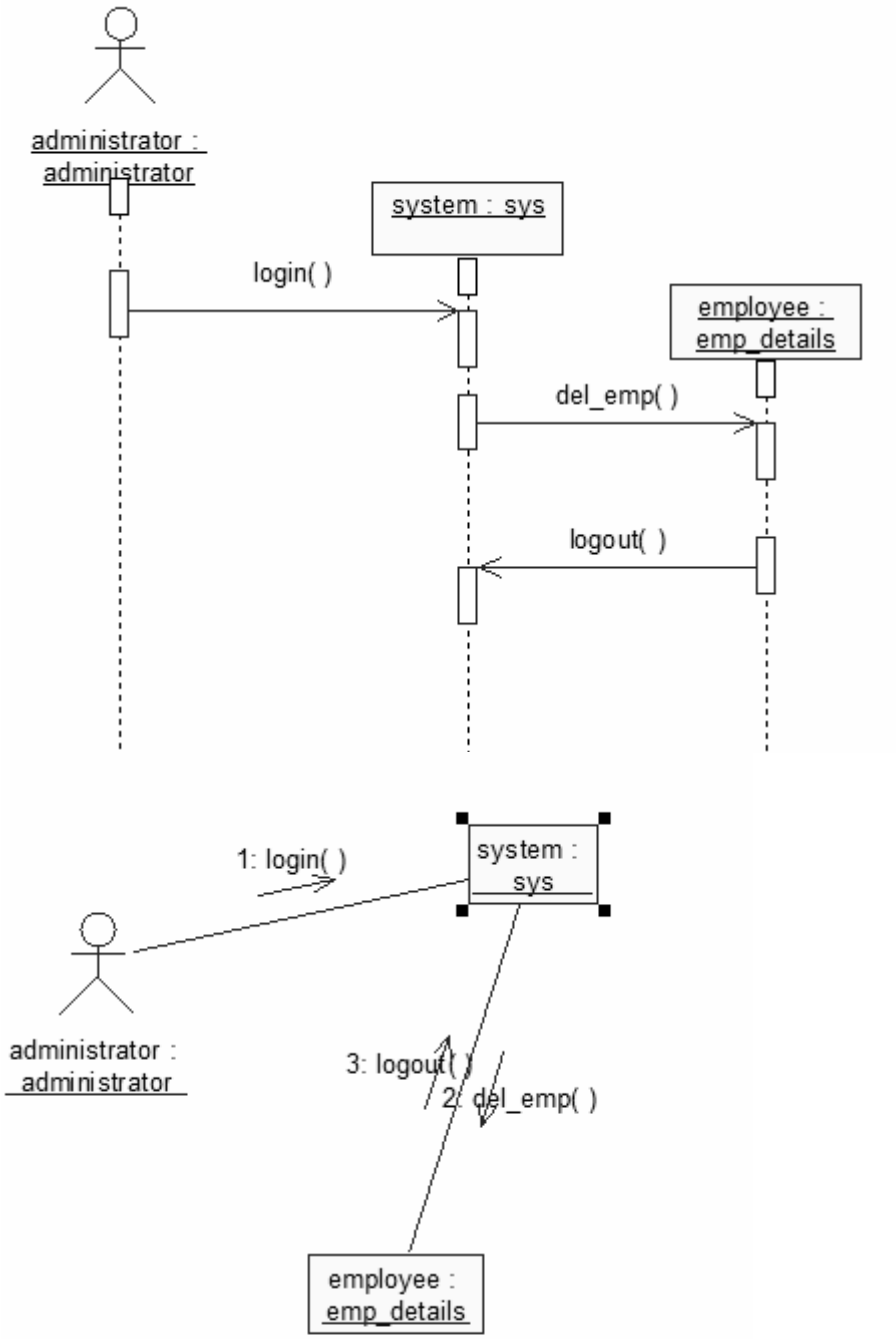
Sequence and collaboration diagram for calculating the net salary





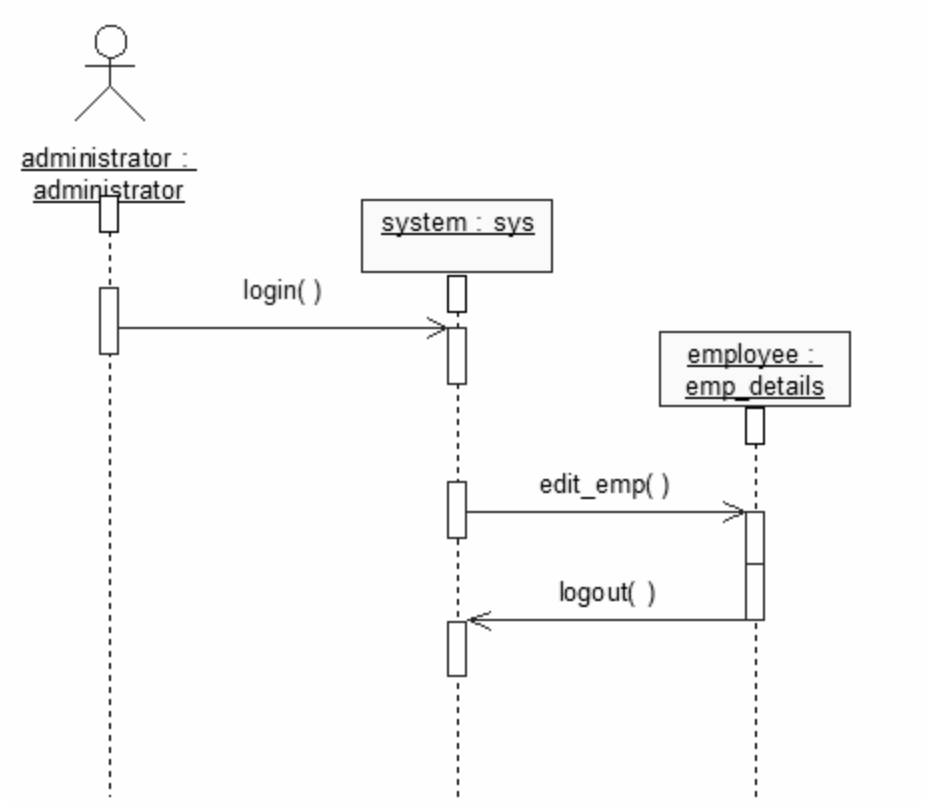
The net salary is calculated by adding the salary details such as basic hra, da, etc., and deducing the insurance and pf amount.

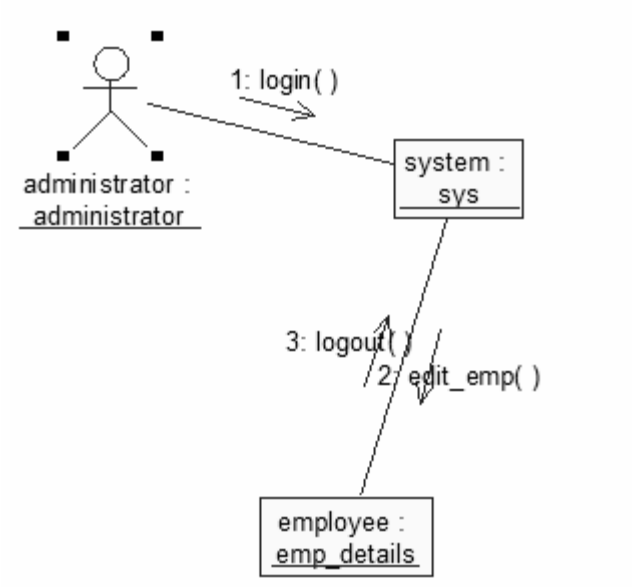
Sequence and collaboration diagram for deleting an employee



Only the administrator has privilege to delete an employee. The employee record which has to be deleted is selected by the administrator and deleted.

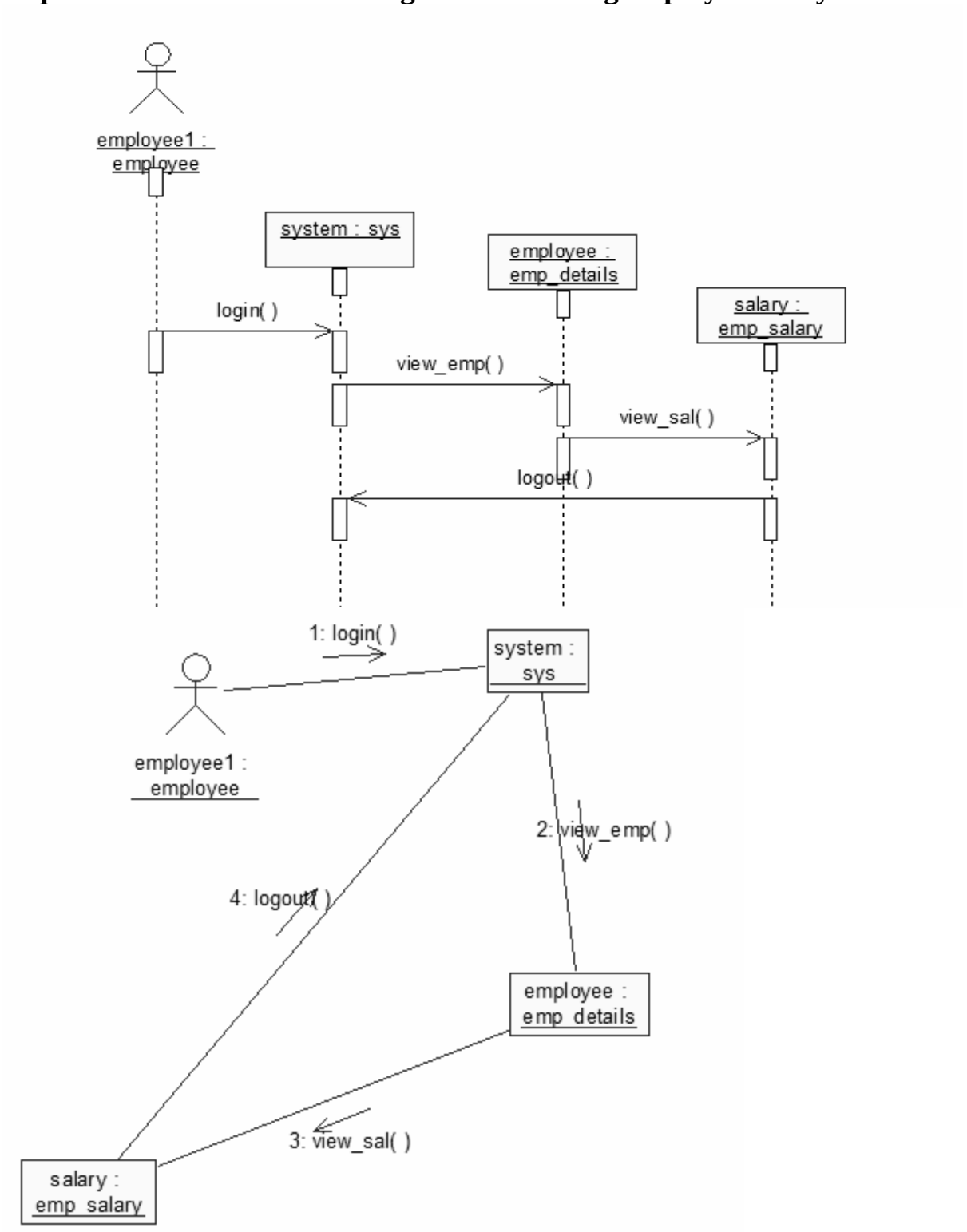
Sequence and collaboration diagram for editing an employee's details





Only the administrator has the privilege to edit employee details. The employee details are selected and edited by the administrator and saved to the database.

Sequence and collaboration diagram for viewing employee salary



The employee may wish to view his salary details. An employee may see only his own salary details and not that of anyone else.

RESULT:

Thus the documentation for employee payroll system is created. The output is verified.

Ex. NO.: **AIRLINE TICKET RESERVATION SYSTEM**
DATE:

Introduction:

The manual system of ticket reservation takes more time and the number of reservations per day is limited. To increase the efficiency of the process, we go for online ticket reservation system. This system supports online ticket booking.

Problem statement

This system is built for user to directly access the system online to book tickets. The user can book, print, delete tickets without the help of a clerk. The administrator has control over the adding flights available for booking and has control over deleting flights that are not necessary. The administrator and user can both enter the system using their respective login details

System requirements

Microsoft visual basic 6.0 is used as the front-end for our project and ms-access is used as the back-end.

Use-case diagram

The online ticket reservation system uses the following use cases:

1. Login
2. Book ticket
3. Print ticket
4. Cancel ticket
5. View flight
6. Add flight
7. Delete flight
8. Logout

Actors involved

1) Administrator
2) Passenger

Use-case name: login

The user enters a username and a password. And if the entered details are valid, the user's details are brought to the screen; if they are invalid then an appropriate message is displayed.

Use-case name: Book ticket

The user is allowed to book a ticket on the flight he requires and the date and time as is necessary for the user. The user has to provide details such as name, flight number, date of travel, departure time, and can view the price of the ticket.

Use-case name: Print ticket

The user after booking a ticket can print a copy of the ticket reserved. The user has to provide the details about ticket number for searching in the database and passenger name for confirming passenger identity.

Use-case name: Cancel ticket

A passenger can decide to cancel a ticket after the ticket is booked. The passenger has to provide details about ticket for searching and details about him for confirmation of identity.

Use-case name: View flight

The passenger can view the flights available in the database for deciding which flight's ticket he wishes to book. The passenger can view the details of flights such as, flight number, Flight Company, price, departure and arrival times.

Use-case name: Add flight

Only the administrator has privilege to add flights. The administrator can add the flight on which tickets can be booked by the passengers. The administrator has to provide details about a new flight such as flight number, flight company name, price, departure time, date of travel.

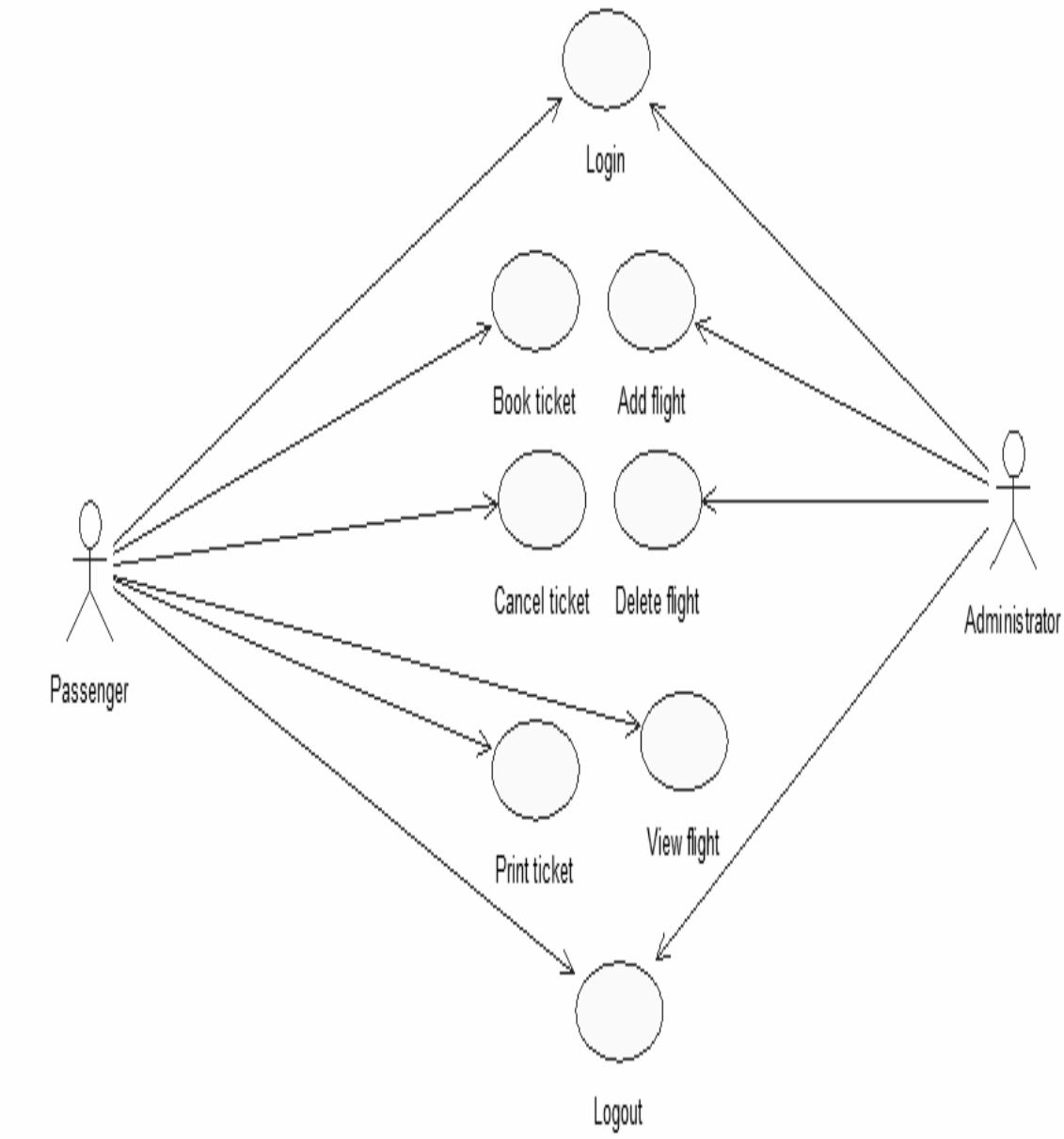
Use-case name: Delete flight

The administrator also has the privilege to delete flights that are not necessary. The administrator has to provide details about the flight for searching and inform any passengers that have booked tickets on the flight about the change and make necessary arrangements.

Use-case name: Logout

After the necessary operations have been performed on the system, the user can choose to logout from the system.

Use-case diagram for airline reservation



Class diagram

The class diagram is a graphical representation of all the classes used in the system and their operations, attributes and relationships.

The online ticket reservation system makes use of the following classes:

- Ticket system
- Flight details
- Ticket

Ticket system

It consists of two attributes and two operations. The attributes are username and password. The operations used are login () and logout ().

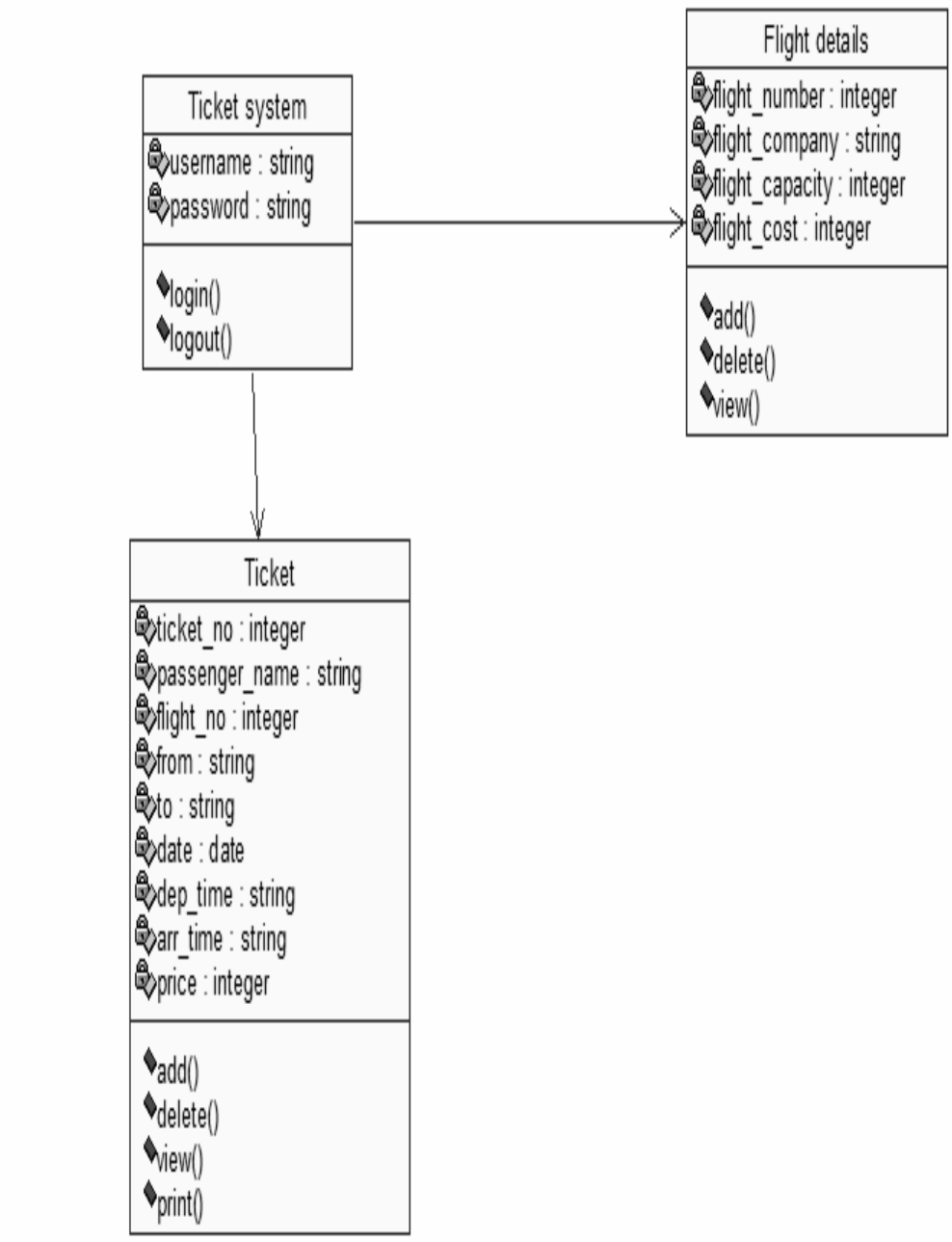
Flight details

It stores the details of all the flights such as flight number, Flight Company, flight capacity, and cost. The operations available are add (), delete () and view ().

Ticket

It records the details of every ticket booked such as ticket number, passenger name, and flight number, from place, to place, date of travel, departure time, arrival time, and price. The operations available are add (), delete (), view (), and print ().

Class diagram for airline reservation system



Sequence diagram

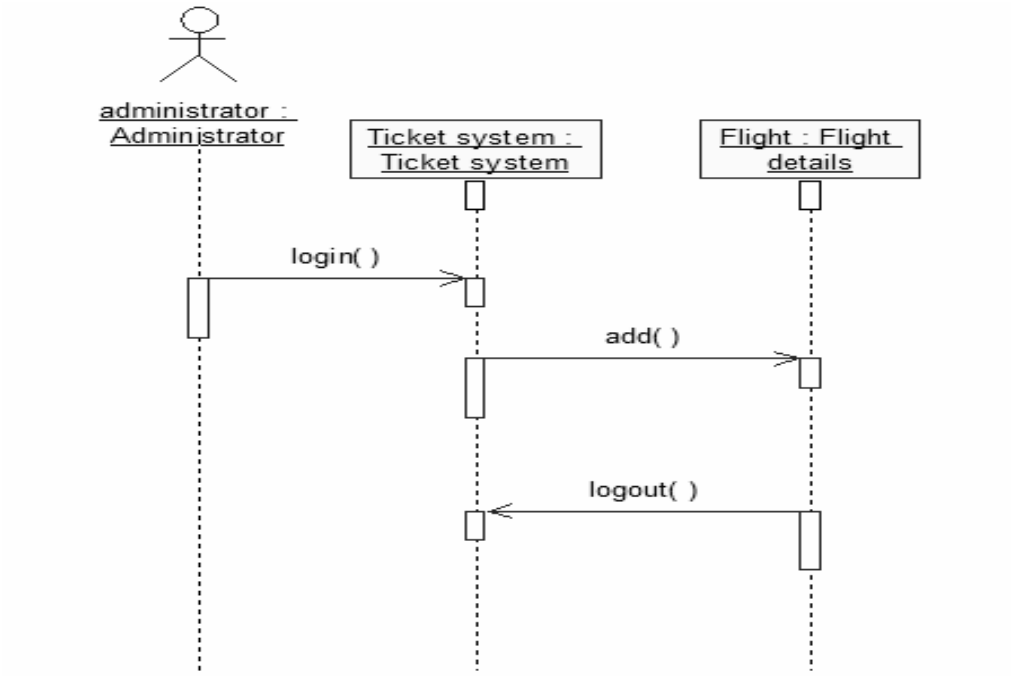
A sequence diagram represents the sequence and interactions of a given use-case or scenario. Sequence diagrams can capture most of the information about the system. Most object-to-object interactions and operations are considered events and events include signals, inputs, decisions, interrupts, transitions and actions to or from users or external devices.

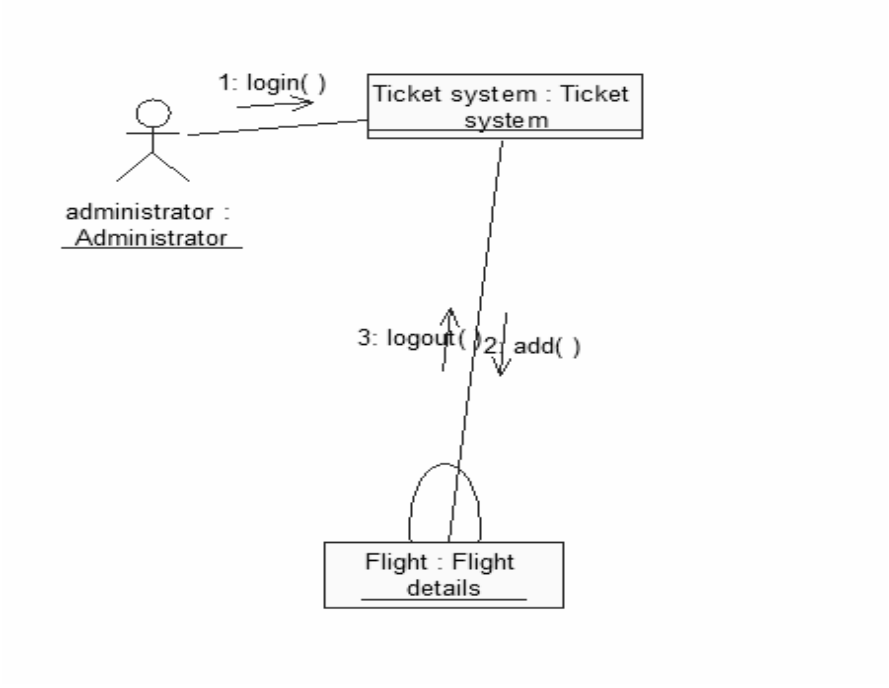
An event also is considered to be any action by an object that sends information. The event line represents a message from one object to another, in which the “from” object is requesting an operation be performed by the “to” object. The “to” object performs the operation using a method that the class contains.

It is also represented by the order in which things occur and how the objects in the system send message to one another.

The sequence diagram for each use-case that exists when a user logs in, adds, views, updates or deletes records in the system.

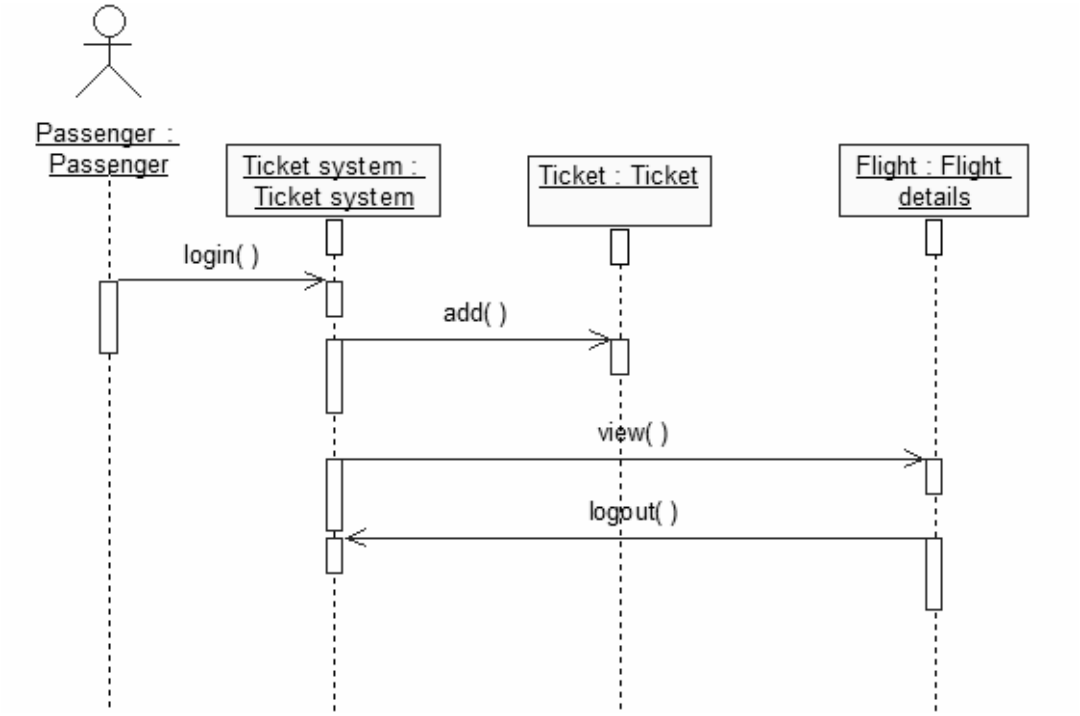
Sequence and collaboration diagram for adding a flight

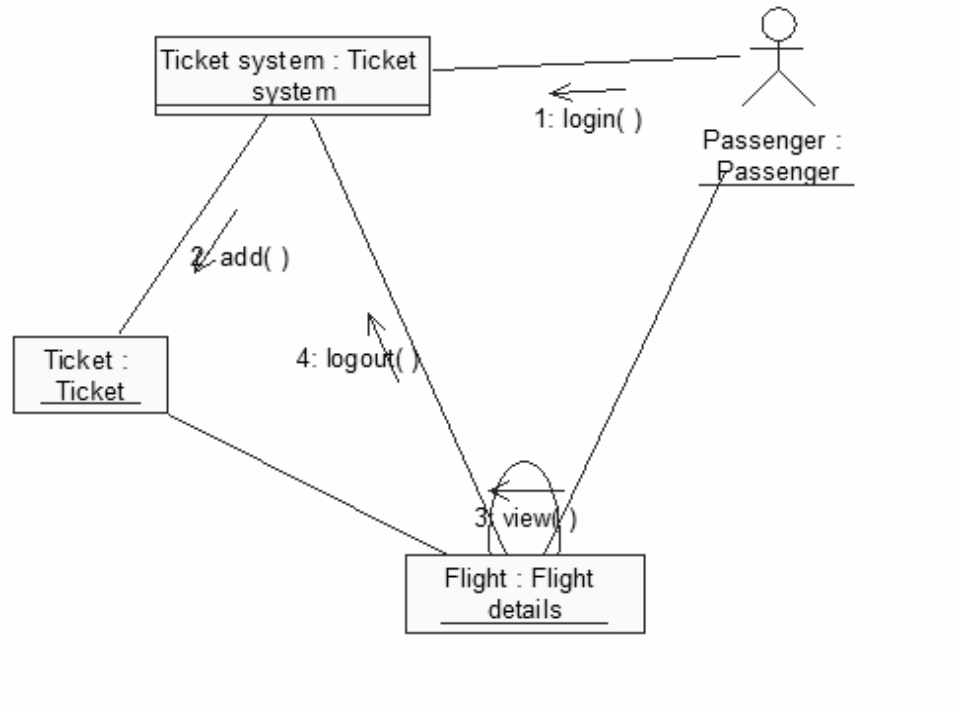




The Administrator has the privilege to add flight. He has to provide details about the new flight that is being created in the database.

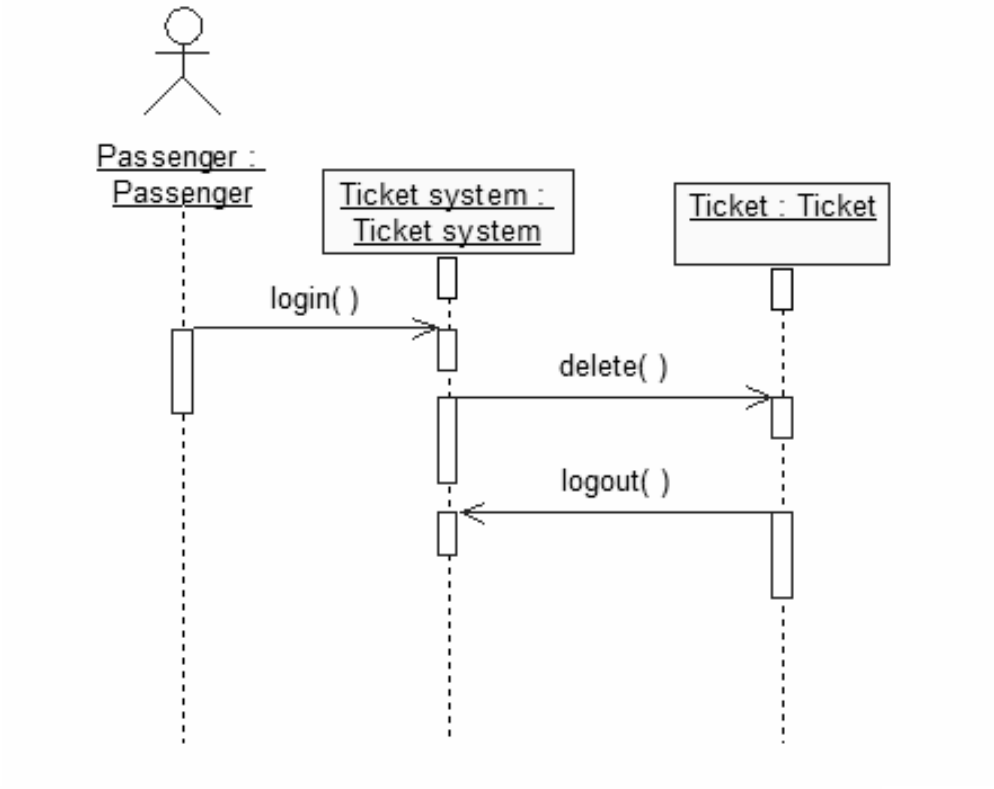
Sequence and collaboration diagram for booking a ticket

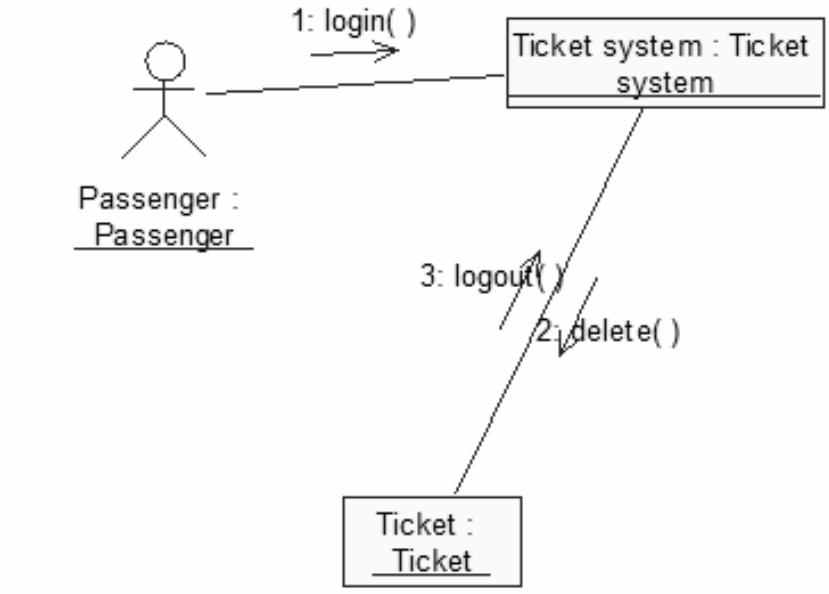




A passenger can book a ticket by himself. He can view the flight details he wants to book a ticket on and as per his necessity may book an appropriate ticket.

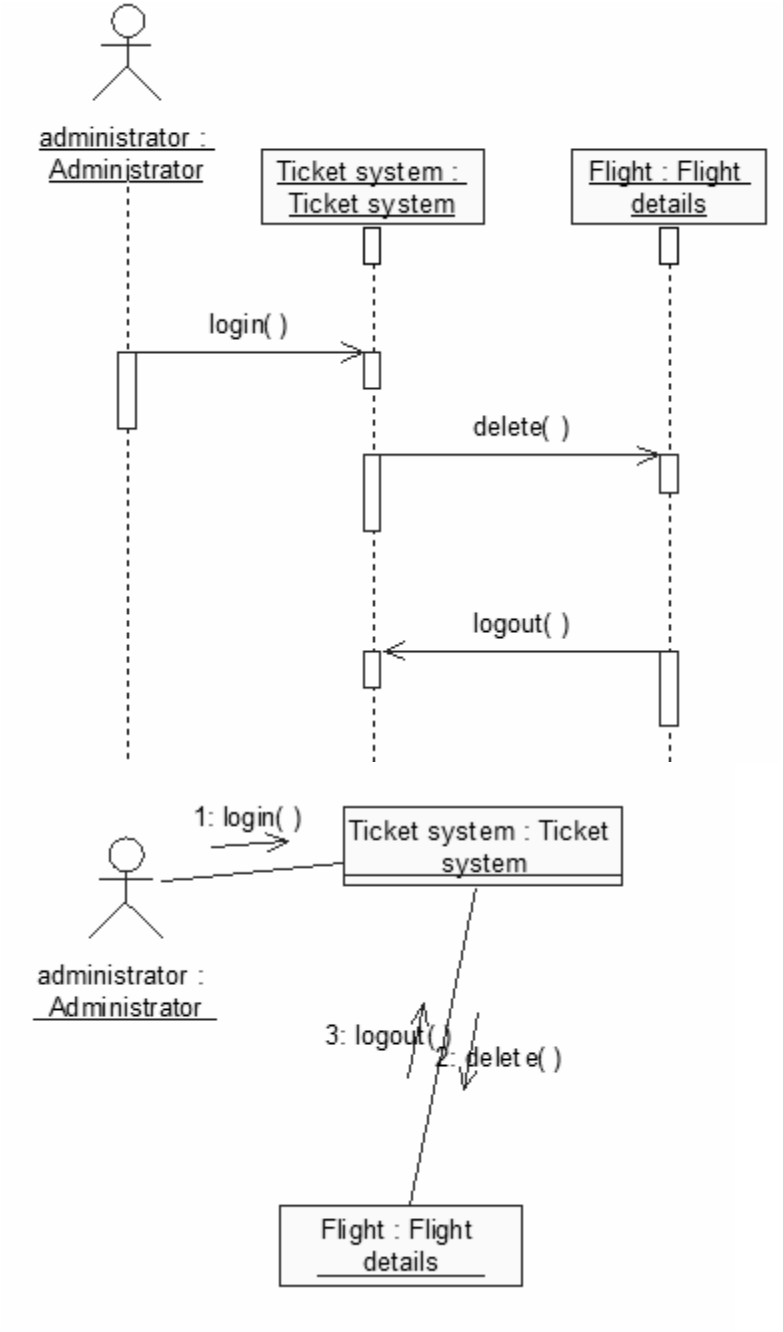
Sequence and collaboration diagram for canceling a ticket





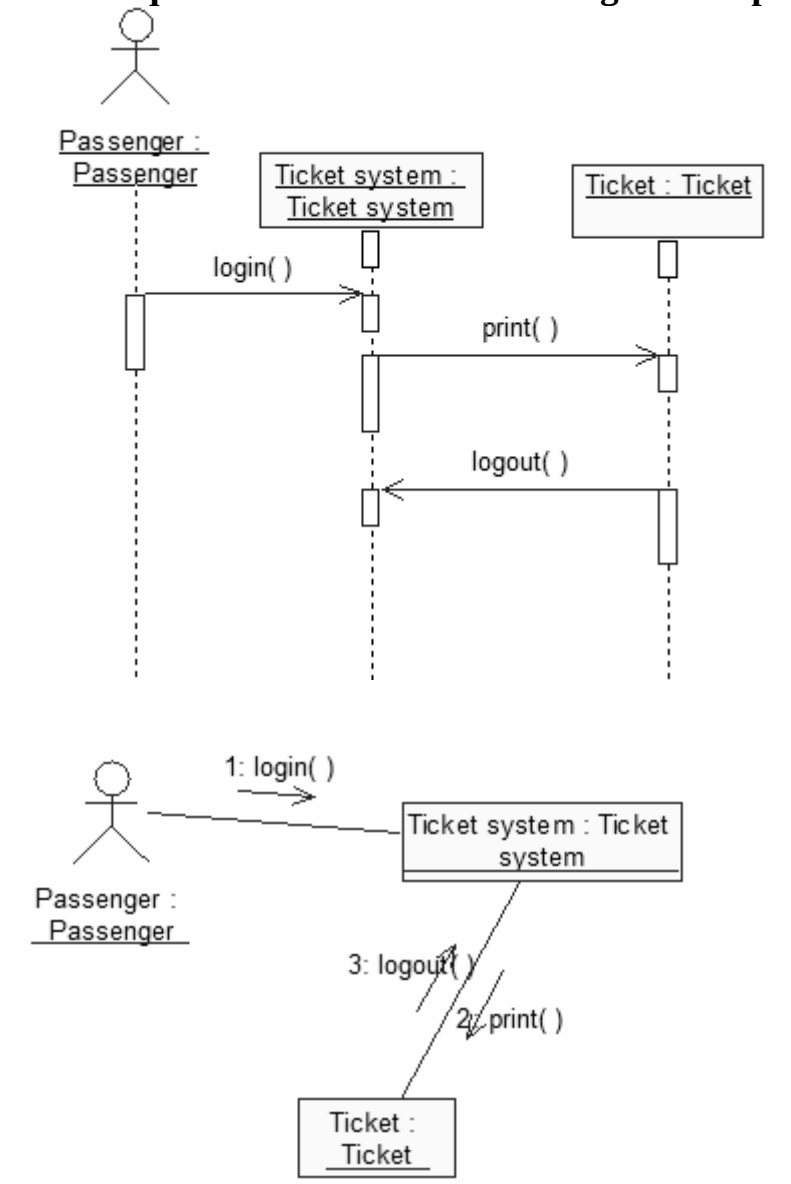
Canceling a ticket can be performed by the passenger himself. He may view the ticket he wants and cancel it.

Sequence and collaboration diagram for deleting a flight



A flight can be deleted only by the administrator. The flight to be deleted is selected and removed from the database.

Sequence and collaboration diagram for printing a ticket



The passenger may choose to print a ticket after booking a ticket. The ticket which is booked is selected and printed by the passenger.

RESULT:

Thus the documentation for airline reservation system is created. The output is verified.

Ex. NO.: **Course Registration System**
DATE:

Aim

To create a system through which students can register to the courses desired by them.

Problem statement

- The system is built to be used by students and managed by an administrator.
- The student and employee have to login to the system before any processing can be done.
- The student can see the courses available to him and register to the course he wants.
- The administrator can maintain the course details and view all the students who have registered to any course.

System requirements

Microsoft visual basic 6.0 is used as the front end of our project and ms-access as the back end.

Use-case diagram

The course registration system has the following use-cases

Login
View course details
Registration
Display details
Maintain course details
Logout

The actors involved in the system are

1. Student
2. Administrator

Use-case name: Login

The user enters the username and password and chooses if the user is student or administrator. If entered details are valid, the user's account becomes available.
If it is invalid, an appropriate message is displayed to the user.

Use-case name: View course details

In this use case, a student can search all the courses available to him and choose the best course he wants. The student can view the course duration, faculty and department of the courses he may choose.

Use-case name: Registration

When a student has successfully chosen a course, he can register to that course. Upon registration, the student's details are stored in the database.

Use-case name: Display details

After registration to any course, the student may see the details of his current course. He may wish to know details about fees and other information. The administrator also has the privilege to display details of the the students and the corresponding course for which they have registered.

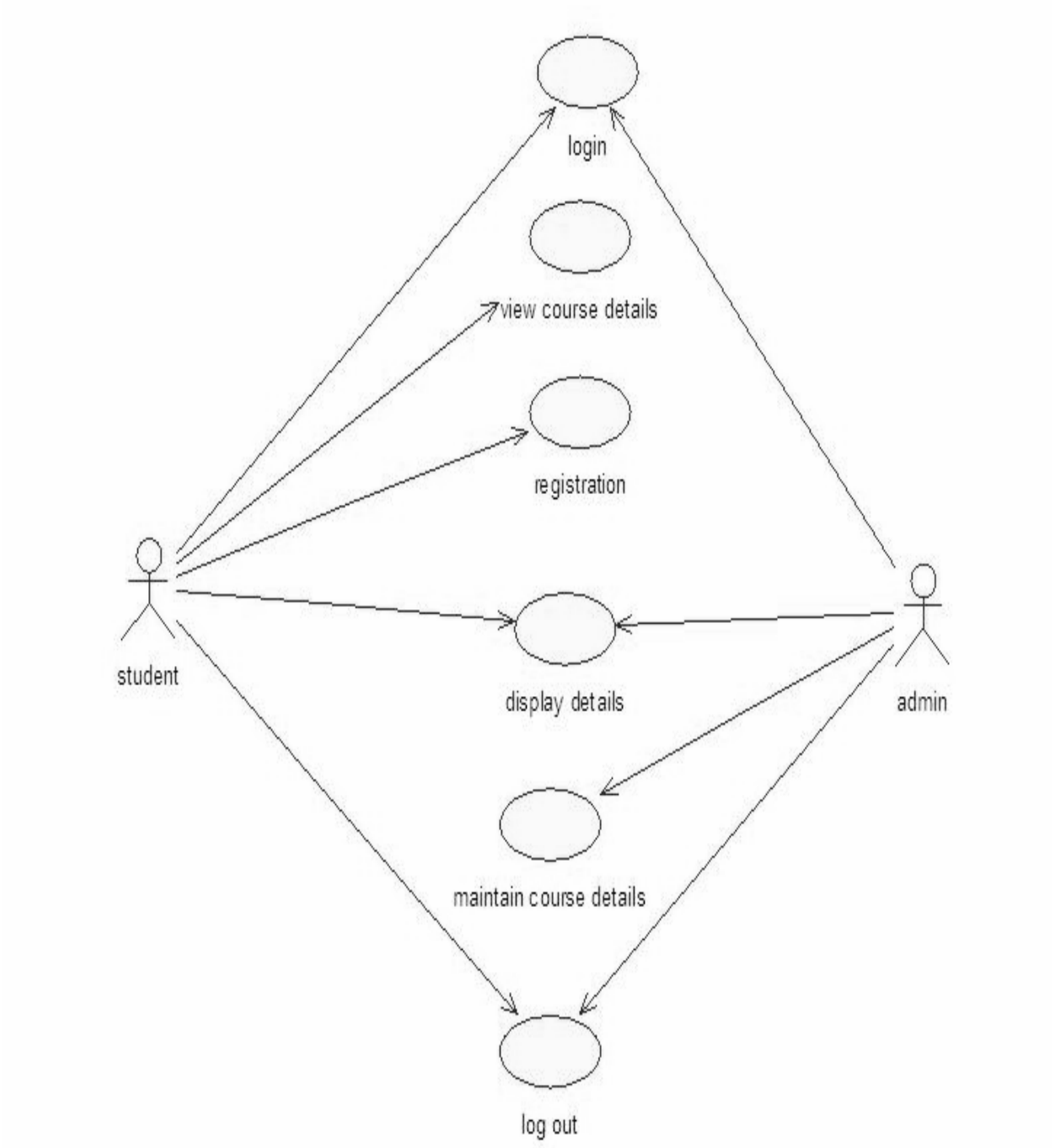
Use-case name: Maintain course details

The administrator has to perform the duties of maintaining the course details. Any change to the course structure is maintained by the administrator.

Use-case name: Logout

After all the desired transactions are made, the user may choose to logout from the system to save all he changes they have made.

Use-case diagram for course registration system



Class diagram

The class diagram is a graphical representation of all the classes used in the system and their operations, attributes and relationships.

The course registration system makes use of the following classes:

1. Stud(student details)
2. Administrator

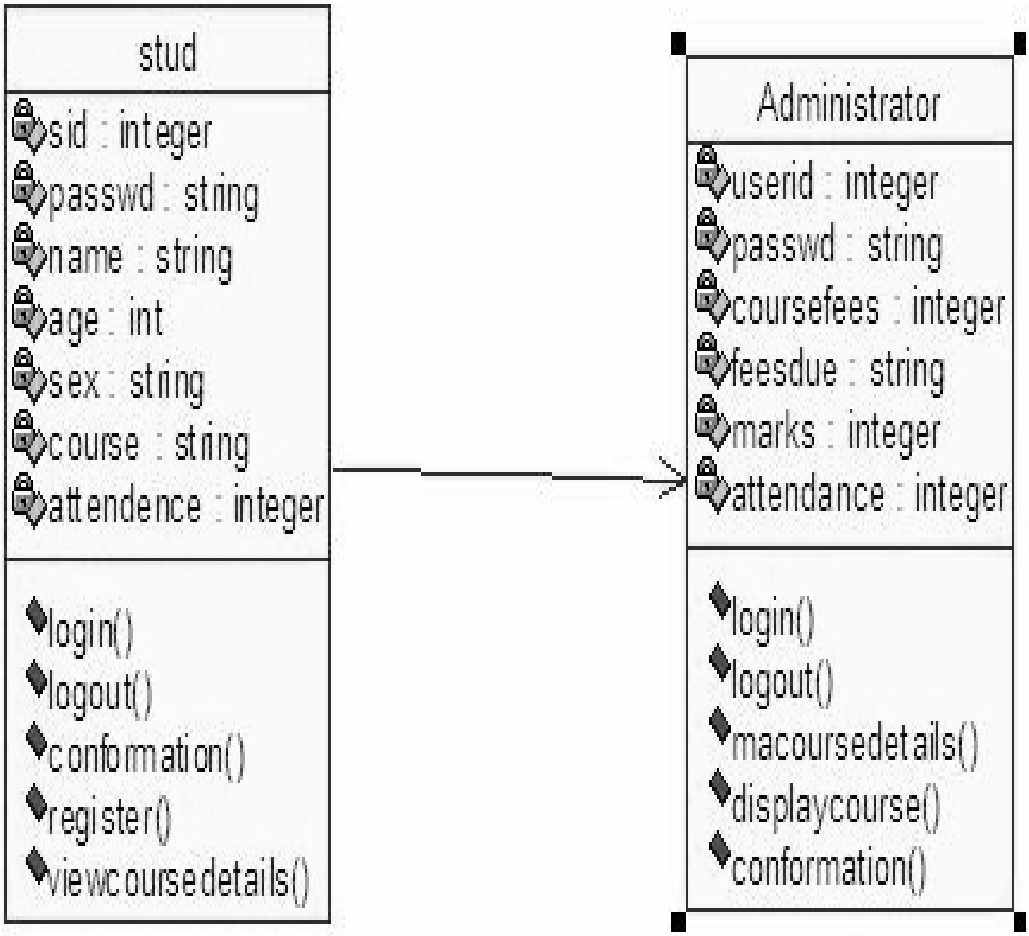
1) Stud

It consists of the details of all the students present in the database. The attributes present in this class are student id, password, name, age, sex, course and attendance. The object of this class is created as soon as the student registers to a course. The operations available to this class are login (), logout (), confirmation (), register (), and view course details ().

2) Administrator

It consists of details of all the courses available to the student. The attributes present in this class are username, password, course fees, fees due, marks, and attendance. The operations available to this class are login (), logout (), ma course details (), display course (), and confirmation ().

Class diagram for course registration system



Sequence diagram

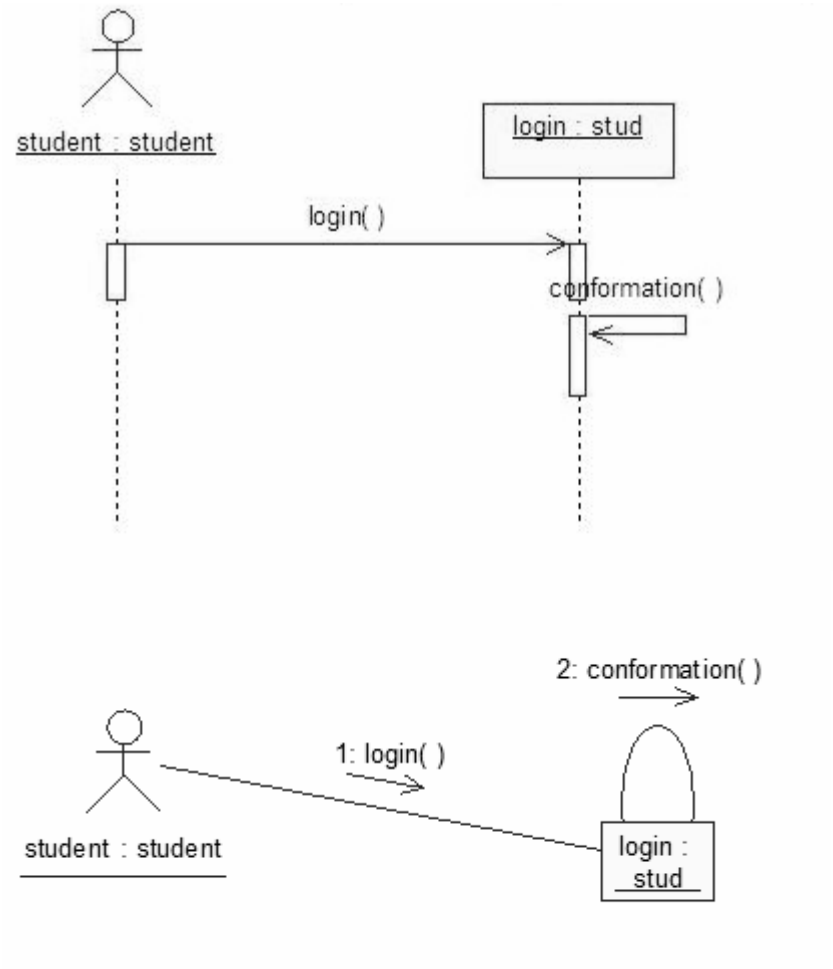
A sequence diagram represents the sequence and interactions of a given use-case or scenario. Sequence diagrams can capture most of the information about the system. Most object-to-object interactions and operations are considered events and events include signals, inputs, decisions, interrupts, transitions and actions to or from users or external devices.

An event also is considered to be any action by an object that sends information. The event line represents a message from one object to another, in which the “from” object is requesting an operation be performed by the “to” object. The “to” object performs the operation using a method that the class contains.

It is also represented by the order in which things occur and how the objects in the system send message to one another.

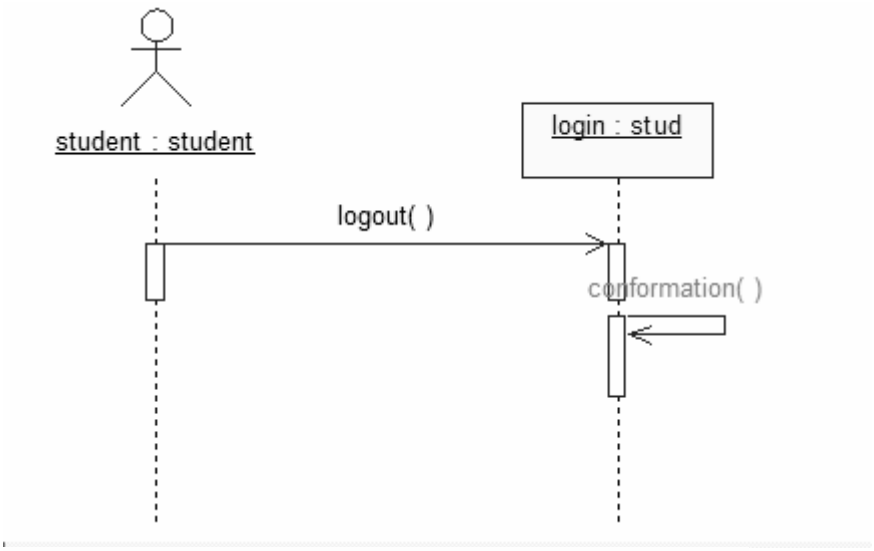
The sequence diagram for each use-case that exists when a user logs in, adds, views, updates or deletes records in the system.

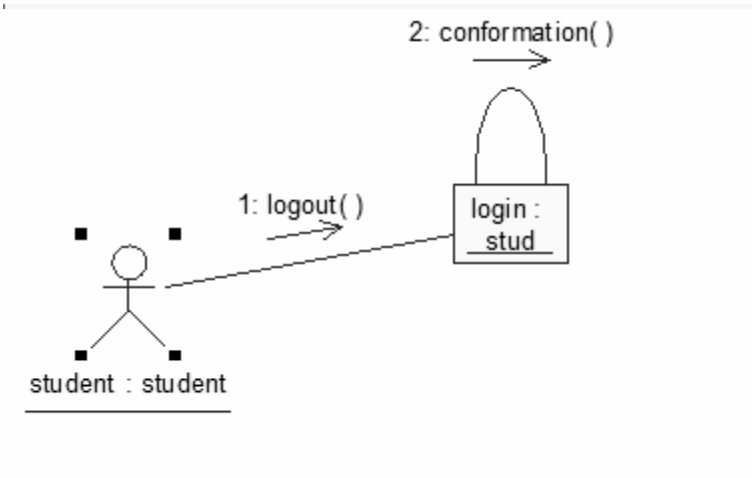
Sequence and collaboration diagram for login to the system



Users have to first login to the system before performing any operation. The user has to provide the necessary details to the system for login.

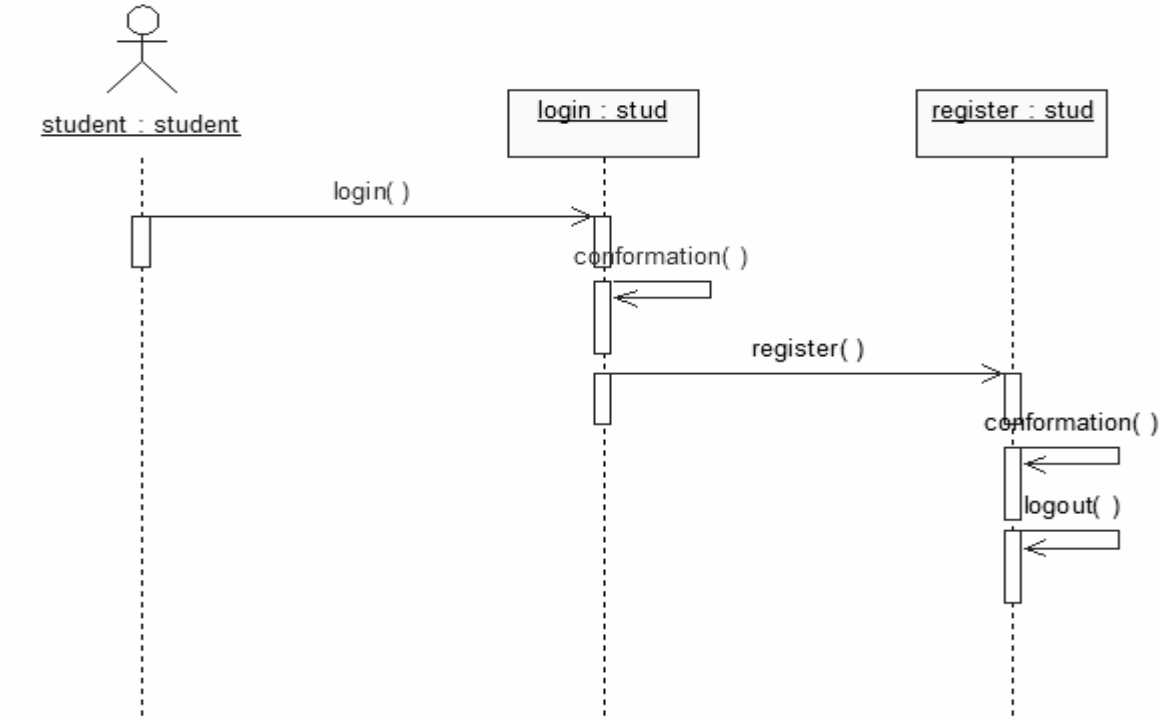
Sequnce and collabaration diagram for logout

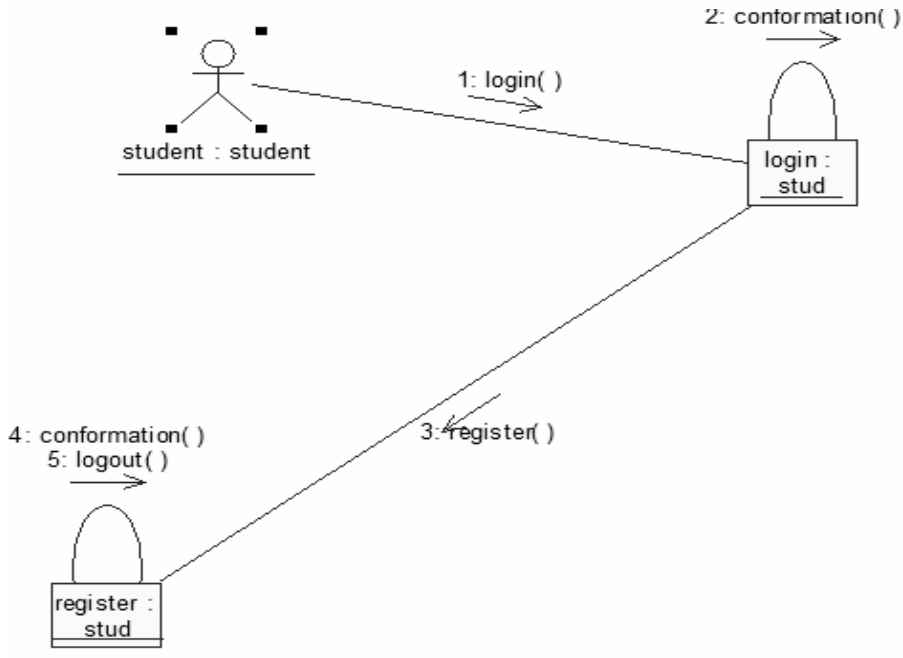




When the necessary operations have been performed on the system, the user may choose to save the changes and logout from the system.

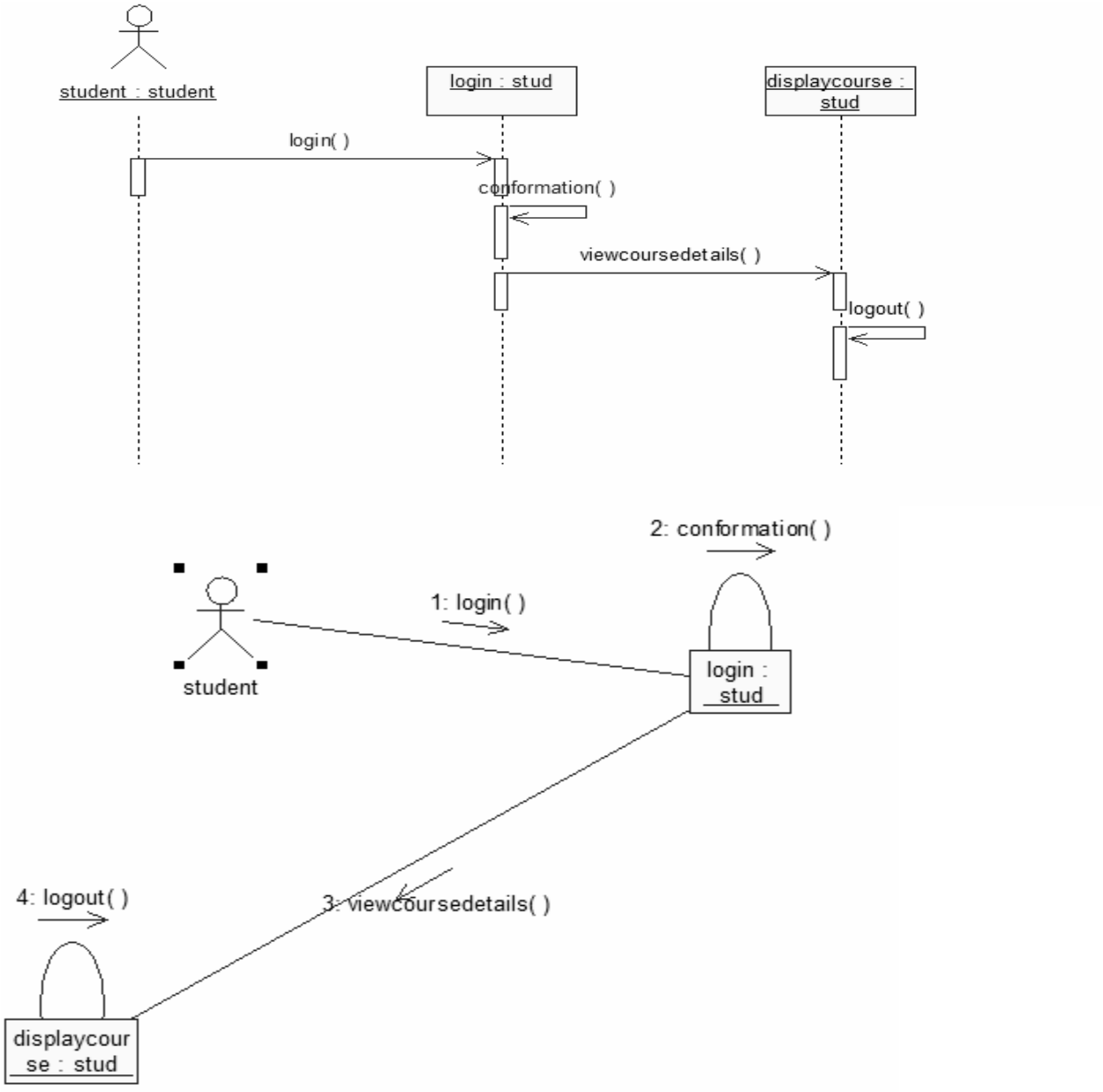
Sequence and collabaration diagram for registration to a course





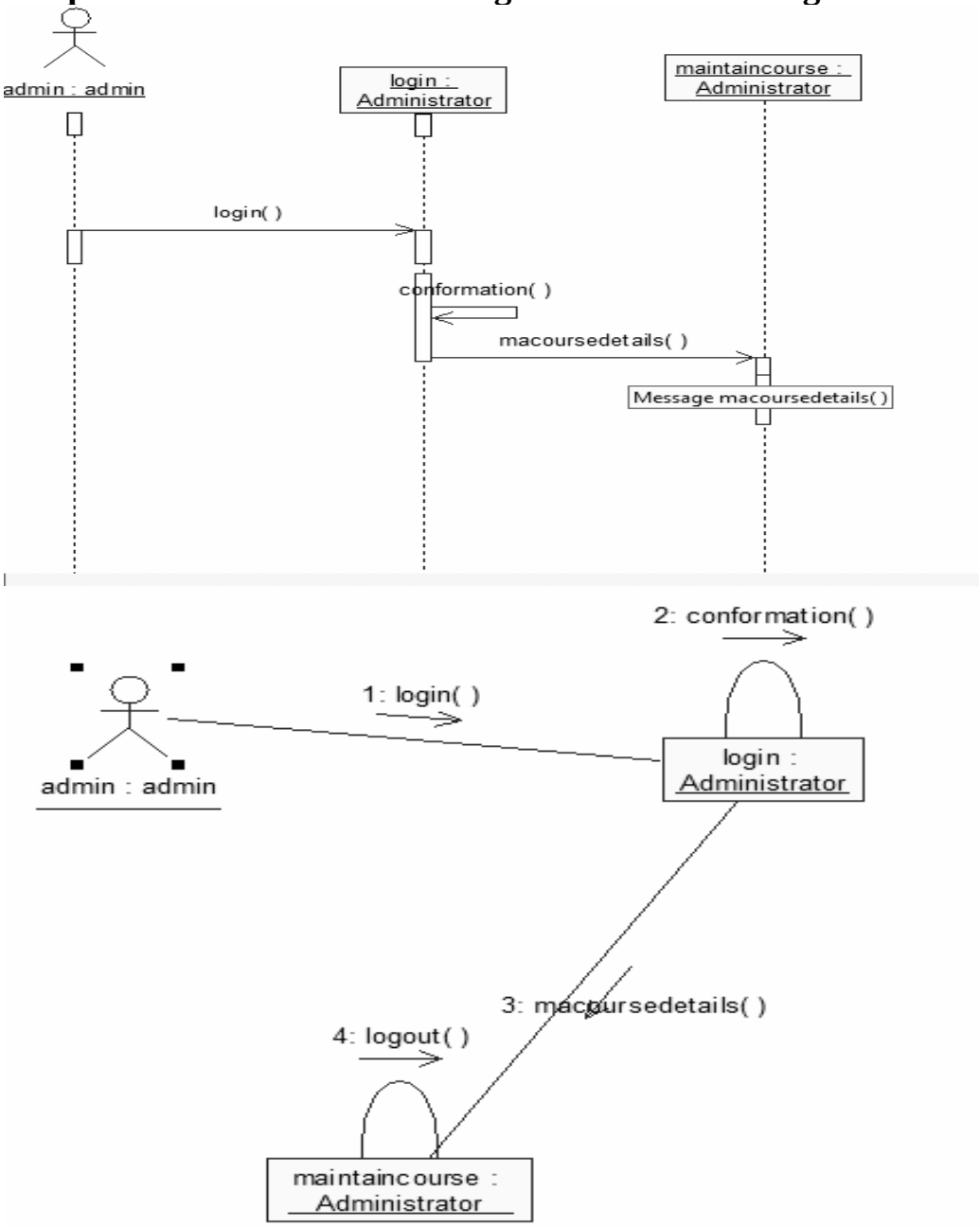
After login, the student has to register to a course of his choice. The student can view all the courses available to him and register to a course suitable to him. The student may view the course details before registration.

Sequence and collaboration diagram for viewing course details



A student may wish to view course details before registration. For this, the student has to first login and select the course details he wishes to see.

Sequence and collaboration diagram for maintaining course details



Course details may be changed as per the requirement every year. So the administrator can edit the details of the course as necessary.

RESULT:

Thus the documentation for course registration system is created. The output is verified.