Cyber Café Manager (CCM)

By

Muhammad Irfan

A project report submitted as a partial fulfillment of the requirements for the degree of M.Sc. Computer Sciences to Department of Computer Science Quaid-e-Azam University Islamabad.

(January 2002)

In The Name of Allah, The Most Beneficent, The Most Merciful

Praise be to Allah, Lord of the worlds, who blessed me to accomplish this task.
It is only because of the blessings of Allah that I have been able to complete my project and my M.Sc.
Finally, I pray to Almighty Allah that I might be able to keep high the name of my country, my university, my parents and my teachers.

All my love is for Allah

Dedicated to

My *Loving parents, brother and Sisters.*
Whose prayers, support and love
enable me to accomplish this task.

Acknowledgement

I have no words at my command to express my deepest sense of gratitude to Almighty **ALLAH** who enabled me to complete my project against all odds and adversities.

Would that I have words to pay tribute to my loving parents, brother and sisters whose invaluable prayers, productive advice and bolster attitude kept my spirit alive to strive for knowledge and integrity which enabled me to reach this milestone.

It is with genuine pleasure that I acknowledge my debt to my teachers and internal supervisor **Mr. Khalid Saleem** who provided me personal stimulation support and guidance. His experience, searching questions and arguments have contribution in designing and developing system and writing this report.

And finally my words cannot say gratitude to my senior and friend **Mr. Kashif Maqbool** (a software engineer at Askeri (Pvt.) Ltd. Islamabad) for providing me essential and useful guideline in analyzing, designing, and developing the system. His efforts and devotion has always encouraged me. Askari information Systems Limited

This acknowledgement will remain incomplete if I didn't mention my loving friends Tahir, Waqas, Altaf, Qadeer, Asif, Nasir, Imran Bukhari ,Imran Rafiq , Soomro, Javeed and Zaheer Qureeshi .I have really enjoyed their friendship during hostel life at **QAU**.

January 31, 2002. Muhammad Irfan

QUAID-I-AZAM UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Dated:__________________

FINAL APPROVAL

This is to certify that we have read the project report submitted by Mr. Muhammad Irfan and it is our judgement that this report is of sufficient standard to warrant its acceptance by the Quaid-I-Azam University, Islamabad for the degree of Master of Science in Computer Science.

COMMITTEE:

1. **External Examiner** [illegible]

   Mr. Arshad Ali Shahid
   National University of Computer Science and Engineering Science
   FAST House, Rohtas Road
   G-9/4,
   Islamabad

2. **Supervisor** [illegible]

   Mr. Khalid Saleem
   Assistant Professor
   Dept. of Computer Science
   Quaid-I-Azam University
   Islamabad.

3. **Chairman** [illegible]

   Dr. Masud Ahmad Malik
   Chairman
   Deptt. of Computer Science
   Quaid-I-Azam University
   Islamabad

Project in Brief

Project Title:	Cyber Café Manager
ndertaken By:	Muhammad Irfan
upervised By:	Mr. Khalid Saleem Assistant Professor, Department of Computer Science, Quiad-e-Azam University Islamabad.
Starting Date:	September, 2001.
Completion Date:	January, 2002.
Software Tools:	Visual C++6.0, MS SQL Server 7.0, MS Access.
Operating System:	Windows NT, Windows 2000.
stem Used:	Intel Pentium III, 773 MHz, 128 MB RAM

Abstract

Cyber Café Manager is general-purpose software that manages all the activities of cyber café.

The system is developed to cater for the scratch card activities in a cyber café. Scratch card's working is similar to PTCL calling card system. Each card has unique number and pin code which enables the user to login the system. Each card has specific price which is reduced as the user is working on the system. The need of such a system is to use the resources and time in an efficient way. The most encouraging feature is to keep track of the activities of all the users who are logged in, using a scratch card. Any workstation can be restarted or shutdown from the server side.

PREFACE

Chapter 1:	This chapter gives an introduction to Scratch Card, Project and Application Area of the System.
Chapter 2:	This chapter consists of requirements analysis, use case identification, description and use case view diagram.
Chapter 3:	This chapter introduces the Object-Oriented design of the system. It includes Classes, Static and behavioral diagrams of the system.
Chapter 4:	This chapter describes the technologies and terminologies that are used in the implementation of the system.
Chapter 5:	This chapter is about the implementation of the system. It illustrates the description of components and division of classes into different components.
Chapter 6:	This chapter details the testing of the system.
Appendix:	It contains screens of user interface.

Contents

Chapter 1

Introduction

INTRODUCTION

1.1 Scratch Card System

There are many systems which are currently working under scratch card facility. Actually a scratch card is a simple card having unique number that is coated with some material making the number invisible. Each scratch card has some price which is associated with it. PTCL calling card, UFONE and ISP cards are examples of scratch cards. Scratch card system is helpful in using the resources in efficient way to save money and time of both consumer and provider. Usage of the scratch card by the customer is very simple, scratch the coated material to get the number, enter the number into specific device which is associated with card. Card number, its balance and its expiry date are verified and this information about card is stored into the database which is normally placed at server side. After verification the user is allowed to enter into the system to use its services and card balance is automatically reduced depending upon the time the card is utilized.

So, we can say that scratch card systems save the money, time, increase the probability to use the system resources in well-organized manner and reduce the conflicts between consumer and provider.

1.2 Project Introduction

The purpose of this Software application is to facilitate the management of Cyber Café environment. There are two components of this software:

- Server Software.
- Client Software.

Server Software works on the Internet / Proxy Server and controls all activities of all clients. The Administrator on the server can perform the following activities:

- Generate the new Scratch cards with unique numbers.

- Maintain the card information.
- Update LAN setting for proxy server.
- Restart / Shutdown any remote machine from server.
- Manage the user accounts for Cyber Café Manager.
- Give the facility to view all users who are currently login through the card number and pin code.
- Prepare system related reports.
- Manage the log information for each machine and for each user who are using the internet through his card number.

Client software works for remote machine so that user can login into the system through the card number and pin code. Following are the major activities of client software:

- It provides graphical identification and authentication user interface like Window NT and Window 2000 to login the system.
- It includes a Window NT service that runs in the background to capture the card usage time, detects card balance and sends all this record to the database at server side.
- It captures the user activities for log information.
- It includes a facility for user to view his current card balance and change his pin code.

1.3 Application Area *of the system*

- It is applicable in any cyber café.
- It is applicable in student's laboratories to capture student activities and time management of the machines for the students.
- It also is applicable in organizations having LAN network to capture the employee's activities and to keep track of their login and logout times.

Chapter 2

Analysis of Cyber Café Manager

2.1 Problem Definition

Software is needed that will manage all the activities of cyber cafes and implements the scratch card security system for browsing charges of the internet. This software is divided into two parts (1) Server Software (2) Client Software.

Server Software will be on the Internet Proxy Server and will be controlling all the activities of the clients. The administrator on the server will be able to perform the following tasks:

- Generates new unique card numbers and manage all other card related activities.
- Restart or shutdown any workstation on the network and also send message to any workstation.
- Configure the internet port for every workstation.

Client Software will be able to capture the user activities for every workstation and report these activities to server database. It will also be able to provide the graphical identification and authentication interface for user's login on workstation through card number for internet purpose.

2.2 Requirement Analysis

Requirements of the system are divided into three parts with respect to functionality of the system. These are the following parts

- Café Employee Requirements
- Administrator Requirements.
- Card User Requirements.
- System Requirements.

Café Employee Requirements

A cafe employee is a person who manages some activities of the system. He has following requirements

Functional Requirements:

The café employee of the system should be able to

- Shutdown or restart any remote machine.
- Retrieve information of account status of any card on the basis of card number and print out this account status if user demands.
- View the current logged in users.
- Generate reports of different sorts e.g. list of generated card in particular month, list of expiry cards, and list of card usage history.
- Manage the log information; it includes which applications user started at which machine and at what date and time. It also includes retrieving and deleting log information for any card.
- He should be able to send message to any remote machine.
- The café employee should be able to change his password.

Non-Functional Requirements:

- He should be able to send informative messages before restarting or shutting down any remote machine. User will be given some time before any of the above operation is started.
- During this specified time he can be able to abort the shutdown operation.
- He should use the login name and password to log into the system.
- The length of password should be of at least 5 characters.
- The characters of password should not be visible.

Administrator Requirements

The administrator is a person who manages all activities of the system. Administrator will also be a café employee with following special requirements.

Functional requirements:

Administrator should be able to

- Manage cards information, which includes adding card code information, generating new card numbers and also printing this list of newly generated card numbers.
- Create the café user accounts.
- Update the café user accounts
- Delete the café user accounts.
- Set password of any card if user demands.
- Delete the expired cards.
- He should be able to update LAN settings for proxy server

Card User Requirements

Card user is user who logs in through his card number on any machine to use the system resources for internet or any other purposes. He has following requirements.

Functional Requirements:

- The user should remain at login screen until he/she is verified properly.
- User can change his card password.
- User card balance is shown on client machine during his session.
- User card should have unique number.
- User should be allowed to log off at any time.

Non-Functional Requirements:

- User card is valid up to expiry date.

System Requirements

These are the requirements which system should automatically meet without human intervention.

Functional Requirements:

- It verifies card number and password.
- When user logs into the system it should start timer and perform deduction of card balance on the server side. Deduction of card balance is based upon its browsing rate and usage time.
- It keeps track of expiry date of each card.
- System should be able to maintain sessions for multiple users.
- It should be able to logoff any user when his respective card balance has finished.
- It captures the user activities that include which applications are started by the user on which machine at what time and also adds this information into database.
- When user logs off or shuts down the system its timer and deduction of card balance is stopped automatically on the server side.
- The system should provide Graphical Identification and Authentication (GINA) interface just like that of Windows NT that includes login screen, change password screen, system locked screen and a screen that appears when user presses ALT+CTRL+DEL sequence of keys to logoff, shutdown or view the list of tasks.

Non-Functional Requirements:

- System should store password in encrypted form.
- System should be able to declare user card invalid if it is not used within specified time period.
- System should be able to show popup screen on user desktop if balance is about to finish.

Future Enhancement

- The Client machine should be protected from file deletions.
- After every session, System and configuration files should be restored, leaving the system as it was before last use.

2.3 Use-Case Modeling

A use case is modeling technique used to describe what a new system should do. So Use cases show the functionality which system should provide. Each use case specifies a complete functionality. The primary components of use-case model are use-cases and actors. In use case modeling the system is looked upon as a "black box" that provides use cases. How the system does this, how the use cases are implemented and how they work internally is not important .In fact, when the use-case modeling is done early in the project ,the developers have no idea how the use cases will be implemented .The primary purpose for use case are:

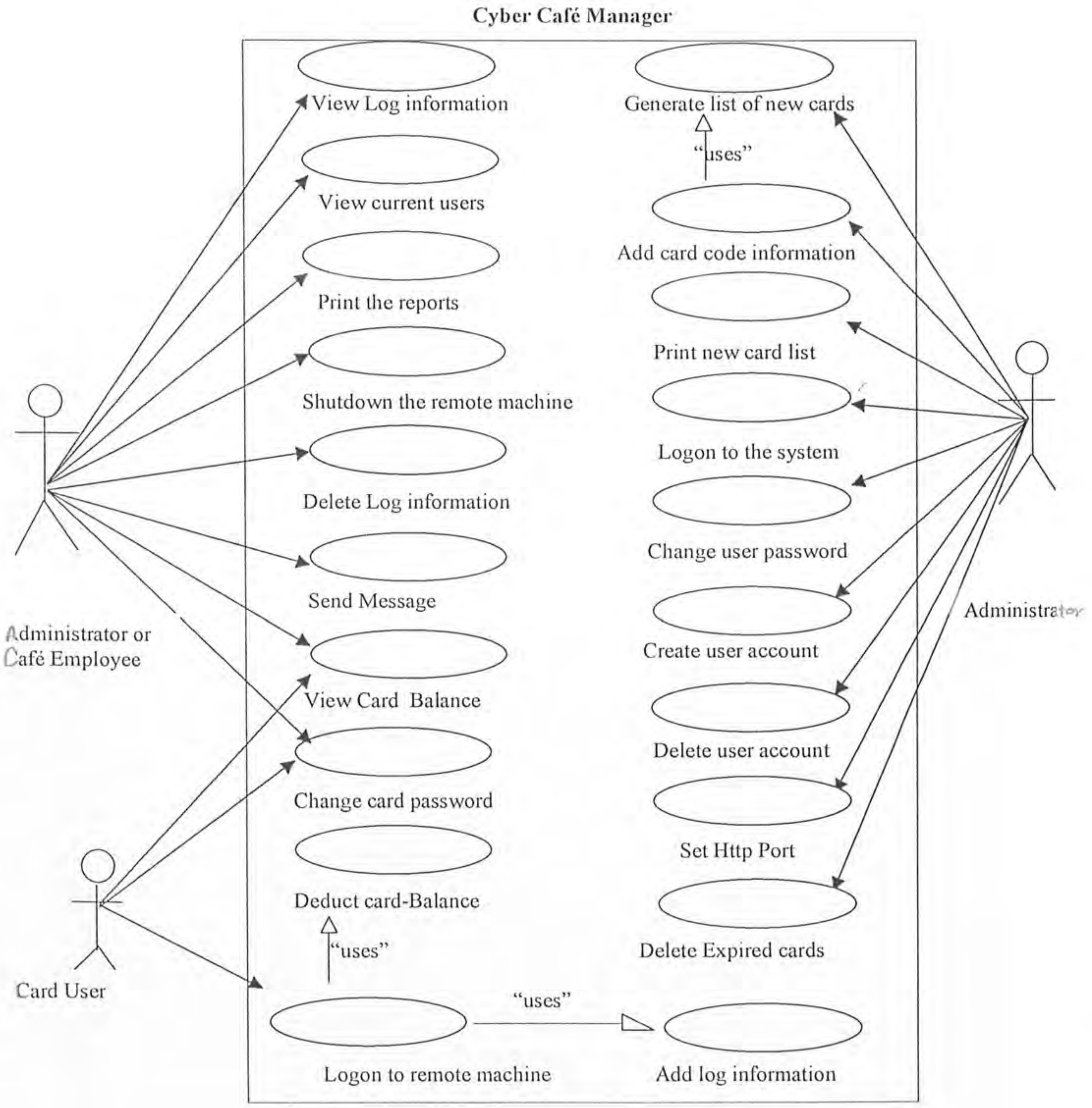
- To decide and describe the functional requirements of the system, resulting in an agreement between the customer (end users) and the software developers who are building the system.
- To give a clear and consistent description of what the system should do, so that the model is used throughout the development process to communicate to all developers those requirements, and to provide the basis for further design modeling that delivers the requested functionality.

2.4 Use Case Identification

These are the following use cases of my system "Cyber Café Manager"

- Add card code information.
- Generate new card numbers.
- Print new card list
- Delete Expired cards.
- View card balance.
- Change card password.
- Deduct card balance.
- View current users.
- Logon into system.
- Print the reports.
- Add log information.
- Delete log information.
- View log information
- Send message.
- Set http port.
- Create user accounts.
- Delete user accounts.
- Shutdown the remote machine.
- Change user's password.
- Login to remote machine

2.5 Use-Case view Diagram



2.6 Actors

An actor is someone or something that interacts with the system it's who or what uses the system. By "interacts with the system". We mean that the actor send or receive the messages to and from the system or exchange information with the system .In short, actors carry out use cases. A use case is always initiated by an actor that sends or receive message to it. This is sometimes called stimulus.

These are the following actors of my system" Cyber Café Manager"

- Administrator
- Café Employee
- Card User

2.7 Use Case Description

UC1.

Name: Add Card Code Information

Initiator: Administrator

Trigger: This use case is trigger only when administrator selects the card dialog to add the card code information.

Process:

1.1 Enter the card code

1.2 Enter the card type.

1.3 Enter the card strength.

1.4 Enter the browsing rate.

1.5 If this information is correct then save to database.

1.6 Initiate the use case "Generate list of new cards".

UC2.

Name: Generate new card numbers.

Initiator: Administrator

Trigger: This use case is trigger only when administrator selects the card dialog and card code information is added successfully

Process:

1.1 Get current dates that is apply date.

1.2 Add 6 month into current date that becomes expiry date.

1.3 Generate unique card numbers on the base of card strength.

1.4 If this information is correct then save to database.

UC3.

Name: Print new cards.

Initiator: Administrator

Trigger: This use case is trigger only when administrator selects the card dialog, card code information is added successfully and list of new generated cards are appeared.

Process:

1.1 Select the printer.

1.2 Set page formatting.

1.3 Select the **print** button for printing or select the **cancel** button to cancel the printing.

1.4 If printing is cancel then remove the list of new generated cards from the database.

UC4.

Name: Delete expired cards.

Initiator: Administrator

Trigger: This use case is trigger only when administrator selects the card dialog.

Process:

1.1 Select card dialog.

1.2 Select the Delete Expired card option from the menu.

1.3 Views the list of expired cards. Select the **delete** button for deleting the cards permanently or select the **cancel** button to cancel the deletion process.

1.4 If delete button is selected then all expired cards are deleted from the database.

UC5.

Name: View card balance.

Initiator: Administrator, Café Employee

Trigger: This use case is trigger only when initiator selects the card dialog or user select view balance option from the popup menu that is appeared on user machine

Process:

1.1 Select the view balance option.

1.2 Enter the card number whose balance is required.

1.3 Retrieve the balance from the database.

1.4 Display the balance in message box.

UC6.

Name: Logon into system.

Initiator: Administrator, Café employee

Trigger: This use case is trigger only when administrator starts the system.

Process:

1.1 Enter the login name.

1.2 Enter the password.

1.3 Select the **ok** button for login or select the **cancel** button to cancel the login process.

1.4 If **ok** button is pressed then his login name and password is verified and then allow to login into the system.

UC7.

Name: Logon into remote machine.

Initiator: card user.

Trigger: This use case is trigger by a user when machine is in logoff condition.

Process:

1.1 Enter the unique card number.

1.2 Enter the pin code or password.

1.3 Select the **ok** button for login or select the **shutdown** button to shutdown the system.

1.4 If **ok** button is pressed then his card number and password is verified and then allow to login into the system.

1.5 If **ok** button is pressed then his current card balance is shown in message dialog.

1.6 Create new text file and write the card number in it.

1.7 Set login status true in the database.

1.8 Initiate two new use cases "Add log information" and "Update card balance"

UC8.

Name: Deduct card balance.

Initiator: Card user.

Trigger: This use case is trigger only when initiator login to the remote machine successfully.

Process:

1.1 Read the card number from test file that is generated in login process.

1.2 Get the browsing rate from the database through card number.

1.3 Get the card balance from the database through card number.

1.4 Calculate the machine charges per minute with the help of browsing rate.

1.5 Detect these charges from current balance.

1.6 Update the balance into the database .through card number.

UC9.

Name: Add log information.

Initiator: Card User.

Trigger: This use case is trigger only when initiator login to the remote machine successfully.

Process:

1.1 Read the card number from test file that is generated in login process.

1.2 Get the application name with full path if user runs or open.

1.3 Get the current time and date.

1.4 Get the computer name on the network.

1.5 Add this all above information into the database with the help of card number.

UC10.

Name: Change password.

Initiator: Administrator, Café Employee, Card user.

Trigger: This use case is trigger only when initiator selects the change password dialog to change the password when he login to the system successfully.

Process:

1.1 Enter the card number or Login name.

1.2 Enter the old password.

1.3 Enter the new password.

1.4 Confirm the new password.

1.5 Press the **ok** button to change password or select the **cancel** button to cancel this operation.

1.6 If **ok** button is selected then verified the passwords, update the old password with new password and save the changes into the database.

1.7 Show the message if password has been changed successfully.

UC11.

Name: Print the reports.

Initiator: Administrator, Café Employee.

Trigger: This use case is trigger only when initiator selects the report dialog after successful login into the system.

Process:

1.1 Select the report type.

1.2 Display the report contents.

1.3 Press the **print** button for printing.

1.4 Select the printer.

1.4 Set the page formatting.

1.5 Select the **ok** button to print the report or select the **cancel** button to cancel the printing.

UC12.

Name: View log information

Initiator: Administrator, Café Employee.

Trigger: This use case is trigger only when initiator selects the view dialog after successful login into the system.

Process:

1.1 Select the log information type.

1.2 Enter one of these Machine name, Card number or select the specific date.

1.3 Press the **view** button.

1.4 Displays the log information contents on the base of its type.

UC13.

Name: Delete log information

Initiator: Administrator, Café Employee.

Trigger: This use case is trigger only when initiator selects the view dialog after successful login into the system.

Process:

1.1 Select the log information type.

1.2 Enter one of these Machine name, Card number or select the specific date.

1.3 Press the **view** button.

1.4 Displays the log information contents on the base of its type.

1.5 Delete the log information.

UC14.

Name: Shutdown the remote machine.

Initiator: Administrator, Café Employee

Trigger: This use case is trigger only when initiator selects the machine dialog after successful login into the system.

Process:

1.1 Select the computer you want to shutdown.

1.2 Type the message for that remote machine.

1.3 Enter the abort time in seconds.

1.4 Select option to close the remote application forcibly.

1.4 Press the **Shutdown** button to perform this operation.

1.5 A message dialog is appeared on the affected machine.

1.6 A small dialog is also appeared on for initiator so that he can abort the operation within specified abort time.

UC15.

Name: Send message.

Initiator: Administrator, Café Employee

Trigger: This use case is trigger only when initiator selects the message dialog after successful login into the system.

Process:

1.1 Select the computer name.

1.2 Type your message.

1.3 Press **Send** button to send message for selected machine.

1.4 Message dialog is appeared on affected computer.

UC16.

Name: Set HTTP port.

Initiator: Administrator

Trigger: This use case is trigger only when administrator selects the internet dialog after successful login into the system.

Process:

1.1 Enter the Proxy server address

1.2 Enter the port number.

1.3 Press the **ok** button for these settings.

1.4 Then the system registry edit for each user for internet connection.

UC17.

Name: Create user account.

Initiator: Administrator

Trigger: This use case is trigger only when administrator selects the user dialog after successful login into the system.

Process:

1.1 Enter the login name of new user.

1.2 Enter the password for new user.

1.3 If specify information is valid this information is added into the database.

UC18.

Name: Delete user account.

Initiator: Administrator

Trigger: This use case is trigger only when administrator selects the user dialog after successful login into the system.

Process:

1.1 Enter the login name of new user.

1.2 Press the **delete** button.

1.3 If login name is valid then his account is removed form the database.

UC19.

Name: View current users.

Initiator: Administrator, Café Employee

Trigger: This use case is trigger only when initiator selects the view dialog after successful login into the system.

Process:

1.1 Select the view dialog.

1.2 Select the current user's option

1.3 Press the **view** button to get information about current users.

1.4 Information of current users is displayed into the view window.

Chapter 3

Design of Cyber Café Manager

3.1 Introduction to Design

Design is the process of determining effective and efficient implementation to carry out the function and store the data to fulfill the requirements. During the design the appropriate computers, devices, software services, and the data storage strategies required for system development is selected. The designer's goal is to produce a model or representation of any entity that will later be built. The design must contain the following characteristics:

- The design ensures the accurate translation of customer requirements.
- It should be readable and understandable.
- It should provide complete picture of the software.
- Design forms basis for programming and maintenance.

Object-Oriented Approach

The object-oriented approach automatically provides all the features that are required for the software design. The object-oriented design is preferred because there is much correspondence between the object-oriented model and real world. In object-oriented, there is more concentration on the identification of classes and there objects. In this approach the smallest unit of programming is class. There is complete correspondence between the object-oriented analysis and object-oriented design. The object oriented design has the following characteristics:

- The object-oriented software is easy to maintain.
- The object-oriented analysis translates to object oriented design easily.
- In object-oriented approach, the lower level implementation details are hidden. It means it provide information hiding facility.
- The object-oriented approach leads to important technology called "Reusability" that is time and resource saving.

3.2 Object-Oriented Concepts

Following are the main object-oriented concepts:

Class

The class is logical concept that encapsulates data and procedures that are required to model the contents and behavior of some real world entity.

Object

The instance of class is called object.

Attributes and Methods

Each object has its own attributes, which are mostly local to the class. The attributes are called data members. The methods are algorithms that process the attributes, which is simulated by means of message.

3.3 Object-Oriented Features

3.3.1 Encapsulation

Encapsulation defines as "packaging of collection of items". Here collection of items means attributes (and other classes of aggregate objects) and operations that class should possess. So encapsulation provides the information hiding and higher level of abstraction. Abstraction is mechanism that enables the designer to focus on the essential details of program components (either data or process) without concerns for lower-level details. The internal state of any object is not directly accessible from outside.

3.3.2 Inheritance

The process in which the characteristics of one class are completely inherited by other class is called inheritance. The class from which the data is inherited is called the super class. The data, which is public, protected and privation will be prompted to inherited class as it is. Inheritance may be simple or multiple.

3.3.3 Polymorphism

It allows different objects to respond to the same message in different ways. It allows the same features having the same meaning.

3.3.4 Relationships

There are mainly three types of relationships:

3.3.5 Association

It represents a "uses a" relationship. It is also called instance relationship.

3.3.6 Aggregation

It represents a "part of" relationship. Other type of aggregation is a composition which is also called "whole of" relationship.

3.3.7 Generalization

It represents "is a" relationship or "kind of" relationship.

3.4 Classes Identification

These are the following classes of CCM system

- WorkStation.
- ShellHook.
- Service.
- HttpConnection
- Gina
- SystemTrayIcon
- Message
- CardLogInfo
- Card.
- CardCode
- CardUsedInfo
- User.
- Report
- CCMProxyServer

3.5 Classes Description

Class Name: WorkStation

Purpose: This class responsibility is to maintain the data about the workstations on the server.

Attributes:

Private:

WorkStationName: String
Message: String
AbortTime: Time
bFlag: Boolean
bForceFlag: Boolean

Methods:

+Create ( ): WorkStation
+**ShutDown** (WorkStationName: String, Msg: String, AbortTime: Time, bFlag: Boolean, bForceFlag: Boolean): Boolean
+**Restart** (WorkStationName: String, Msg: String, AbortTime: Time, bFlag: Boolean, bForceFlag: Boolean): Boolean
+**AbortShutDown** (WorkStationName: String): Boolean
+**AbortRestart** (WorkStationName: String): Boolean
+**Destroy** ( )

Ref: UML Tool Kit by Hanis-Rrik Pinker (+ for Public, - for Private, # for Derived)

Class Name: ShellHook

Purpose: This class responsibility is to capture the shell activities when users log on the workstation.

Attributes:

Private:

ShellExecuteInfo* pSeInfo
LPVerb: String
SzText: String

Methods:

+Create ( ): ShellHook
+**Execute** (ShellExecuteInfo* pSeInfo)**: void**
+**RegisterClass** (Classid: String)**:** Boolean
+**Destroy ( )**

Class Name: Service

Purpose: This class responsibility is to set timer and detect the balance of card when user log on the workstation with card number. Its object is created at boot time.

Attributes:

Private:

ServiceName: String
ErrorMsg: String
Timer: Time
ServiceStatus: Boolean
bService: Boolean

Methods:

+Create ( ): Service
+**Start** (none)**:** void
+**Run** (none)**:** void
+**SetTimer** (TimerId: integer, time: Time)**:** void
+**Install** (none)**:** Boolean
+**TimerOfExitWindow** (uMsg: integer, idEvent: integer, dwTime: integer)
+**BalanceTimerProc** (uMsg: integer, idEvent: integer, dwTime: integer)
+**ErrorMessage** (error: String)**:** Boolean
+**Destroy** ( )

Class Name: HttpConnection

Purpose: This class responsibility is to keep track of http port configuration for internet connection of specific workstation through proxy server.

Attributes:

Private:

ProxyServerAddress: String
PortNumber: Integer
bProxyEnable: Boolean

Methods:

+Create ( ): HttpConnection
+**GettProxyAddress** (none)**:** String
+**GetPortNumber** (none)**:** Integer
+**ChekProxyEnable** (bProxyEnable: Boolean)**:** Boolean
+**SetProxyAddress** (ProxyAddress: String)**:** Boolean
+**SetPortNumber** (PortNumber**:** Integer)**:** Boolean
+**EditRegistory** (none)**:** void
+**Destroy ( )**

Class Name: Message

Purpose: This class responsibility is to send message for any machine on LAN.

Attributes:

Private:

SenderName: String
DestinationName: String
MessageBuffer: String

Methods:

+Create ( ): Message
+**SendMessage** (SenderMachine: String, DestiMachine: String, Message: String, MessageLength: Integer)**:** Boolean
+**GetMachineName** (none)**:** String
+**Destroy ( )**

Class Name: Gina

Purpose: This class responsibility is to provide security for user to login on the workstation. So its object is created by operating system at boot time.

Attributes:

Private:

UserName: String
Passwd: String
Domain: String
hToken: String
hGlobalWlx: String

Methods:

+Create ( ): Gina
+**LoginUser** (LoginName: String, Passwd: String, Domain: String): Boolear
+**GetTokenInformation** (hToken: String): String
+**AllocAndCaptureText** (hdlg: Hwnd, Idctrl: Integer): String
+**CreateProcessAsUser** (Processid: Integer, ProcessName: String): Boolea
+**ImpersonateLoggedOnUser** (hToken: String): Boolean
+**GenerateTextFile** (FilePath: String): Boolean
+**WlxNegotiate** (none): Boolean
+**WlxInitialize** (WinlogonVersion: Integer, pDllVersion: Integer): Boolean
+**WlxDisplaySASNotice** (none): void
+**WlxLoggedOutSAS** (none): void
+**WlxActivateUserShell** (hToken: String): Boolean
+**WlxCreateUserDesktop** (hToken: String): Boolean
+**WlxStartApplication** (AppName: String): Boolean
+**WlxLoggedOnSAS** (none): Boolean
+**WlxDisplayLockedNotice** (none): Boolean
+**WlxWkstaLockedSAS** (none): Boolean
+**WlxIsLockOk** (none): Boolean
+**WlxIsLogoffOk** (none): Boolean
+**WlxLogoff** (none): Integer
+**WlxShutdown** (none): Integer
+**WlxUseCtrlAltDel** (none): Void
+**WlxDialogBoxParam** (none): void
+**WlxSetTimeout** (time: Time): Boolean
+**Destroy ( )**

Class Name: SystemTrayIcon

Purpose: This class responsibility is to create an icon in the system tray for user interface. So its object is also created by the operating system when user's desktop is created.

Attributes:

Private:

IConid: Integer
IConSize: Integer
IConText: String

Methods:

+Create ( ): SystemTrayIcon
+TrayMessage ( ): Boolean
+**ReadTextFile** (Path: String, strFile: String)**:** Boolean
+**DrawIcon** (Iconid: Integer)**:** void
+**DeleteIcon** (Iconid: Integer)**:** void
+**SetIconText** (Iconid: Integer, Test: String)**:** void
+**Destroy ( )**

Class Name: Report

Purpose: This class responsibility is to handle the reports.

Attributes:

Private:

Reportid: Integer
PageLength: Interger
PageWidth: Interger
ReportHeader: String
ReportHeader: String
TotalColumns: Integer
TotalRows: Integer
Type: String
Date: Date

Methods:

+Create ( ): Report
+**SetReportHeader** (header: String): void
+**SetReportFooter** (footer: String): void
+**PopulateReportContents** (none): void
+**SetDate** (date: Date): void
+**Print** (Reportid: Integer): void
+**Destroy** ( )

Class Name: CardLogInfo

Purpose: This class responsibility is to store the information about card logs in to the data base.

Attributes:

Private:

Logid: Integer
MachineName: String
ApplicationName: String
OperationType: String
StartTime: Time
Date: Date

Methods:

+Create ( ): CardLogInfo
+AddLogInformation (Cardno: integer, Mname: String, AppName: String, type: String, time: Time, date: Date)**:** vc
+**GetLogid** (none)**:** Integer
+**DeleteLogInformation** (Cardno: Integer)**:** void
+**DeleteLogInformation** (MachineName: String)**:** void
+**DeleteLogInformation** (date: Date)**:** void
+**GetLogInformation** (Cardno: Integer)**:** return LogInformation
+**GetLogInformation** (MachineName: String)**:** return LogInformation
+ **GetLogInformation** (date: Date)**:** return LogInformation
+**Destroy ( )**

Class Name: CardCode

Purpose: This class responsibility is to store information about card code in to the data base.

Attributes:

Private:

Codeid: Integer
CardType: String
BrowsingRate: Integer
ApplyDate: Date
Strength: Integer
CardPrice: integer

Methods:

+Create ( ): CardCode
+**AddCardCodeInfo** (Codeid: Integer, CardType: String, Rate: Integer, AppDate: Date, Strength: Integer, CardPrice: Integer): vc
+**GetCardCodeInfo** (CodeId: Integer): return CardcodeInformation
+**DeleteCardCodeInfo** (CodeId: Integer): Boolean
+**Destroy ( )**

Class Name: CardUsedInfo

Purpose: This class responsibility is to store card used information in to the database

Attributes:

Private:

CardNumber: Integer
LoginDate: Date
LoginTime: Time
LogoutTime: Time
MachineName: String

Methods:

+**<u>Create ( ): CardUsedInfo</u>**
+**AddCardUsedInfo** (Cardid: Integer, LoginDate: Date, LoginTime: Tim LogoutTime: Time, MachineName: String)**:** vc
+**SetLogoutTime** (Cardid: Integer, LoginDate: Date, LoginTime: Time, LogoutTime: Time)**:** voi
+**DeleteCardUsedInfo** (Cardid: Integer)**:** Boolean
+**GetCardUsedInfo** (Cardid: Integer)**:** return CardUsedInfo
+**Destroy ( )**

Class Name: User

Purpose: This class responsibility is to store the information about user in to the data base

Attributes:

Private:

LoginName: String
Password: String
UserType: String

Methods:

+**<u>Create ( ): User</u>**
+**CreateUserAccount** (LoginName: String, Passwd: String, Type: String): voi
+**DeleteUserAccunt** (LoginName: String): void
+**IsUserValid** (LoginName**:** String**,** Passwd**:** String): Boolean
+**ChangePasswd** (LoginName: String, NewPasswd: String): void
+**SetUserType** (LoginName: String, Type: String): void
+**Destroy ( )**

Class Name: Card

Purpose: This class responsibility is to store the information about card in to the data base.

Attributes:

Private:

CardNumber: Integer
CardPasswd: String
CardBalance: Integer
IssueDate: Date
ExpiryDate: Date
LoginStatus: Boolean
LoginTime: Time

Methods:

<u>**+Create ( ): Card**</u>
+AddCard (Cardid: Integer, Passwd: String, Balance: Real, IssueDate: Da
LoginStatus: Boolean, LoginTime: Time)**:** void
+DeleteCard (Cardid: Integer)**:** Boolean
+GetBalance (Cardid: Integer)**:** Integer
+SetBalance (Cardid: Integer, Balance: Real)**:** Boolean
+SetLoginStatus (Cardid: Integer, LoginStatus: Boolean)**:** Boolean
+GetLoginStatus (Cardid: Integer)**:** Boolean
+SetLoginTime (Cardid: Integer, LoginTime: Time)**:** void
+GetLoginTime (Cardid: Integer)**:** return Time
+IsCardValid (Cardid: Integer, Passwd: String Boolean
+ChangePasswd (Cardid: Integer, NewPasswd: String)**:** void
+GetExpiryDate (Cardid: Integer)**:** return date
+IsCardExpired (Cardid: Integer)**:** Boolean
+GenerateCardNumbers (none)**:** void
+GetCardInfo (Cardid: Integer): return CardInformation
+Destroy ( )

Class Name: CCMProxyServer

Purpose: This class works like proxy/stub between classes and establish communication between the classes.

Attributes: This class has no attributes.

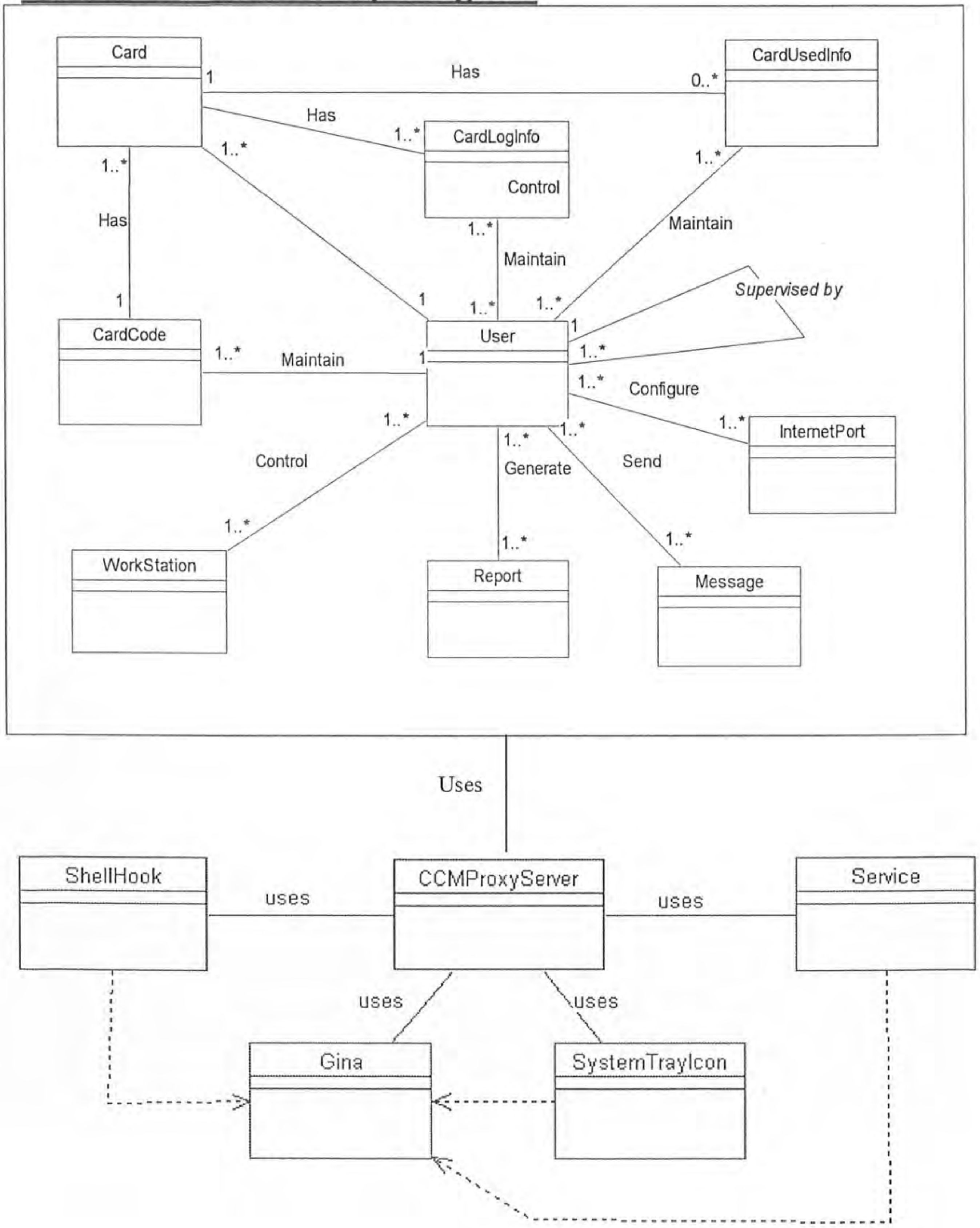
Interfaces:

This class has only interfaces which are used by other classes to communicate each other through this class.

Methods:

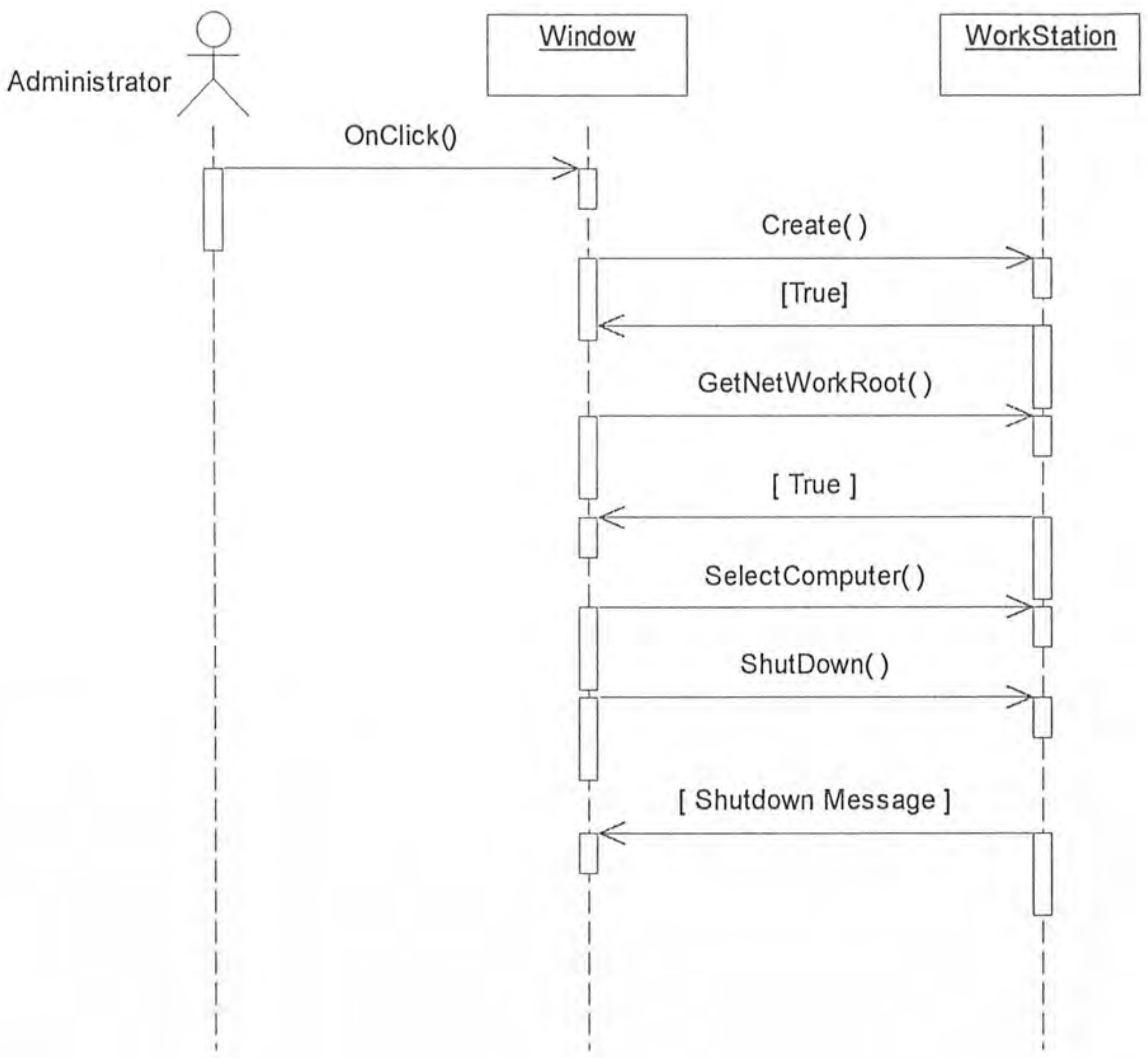
<u>**+Create ( ): CCMProxyServer**</u>
+isCardValidProxy ( ): Boolean
+GetBalanceProxy ( ): integer
+GetBrowsingRateProxy ( ): integer
+SetBalanceProxy ( ): Boolean
+AddLogInfoProxy ( ): Boolean
+SetPasswordProxy ( ): Boolean
+SetLoginStatusProxy ( ): Boolean
+Destroy ( )

3.6 Class Relationship Diagram

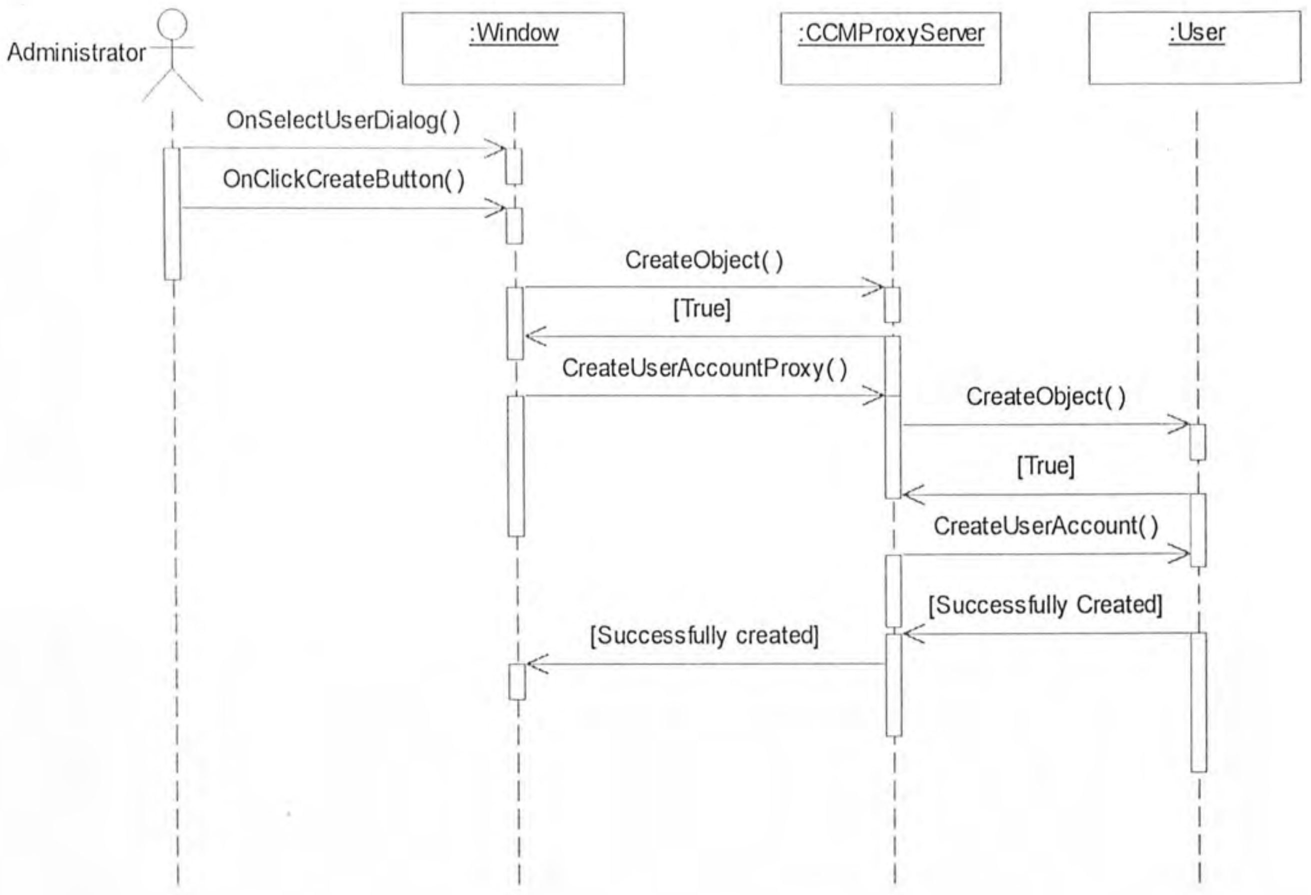


3.7 Sequence Diagrams

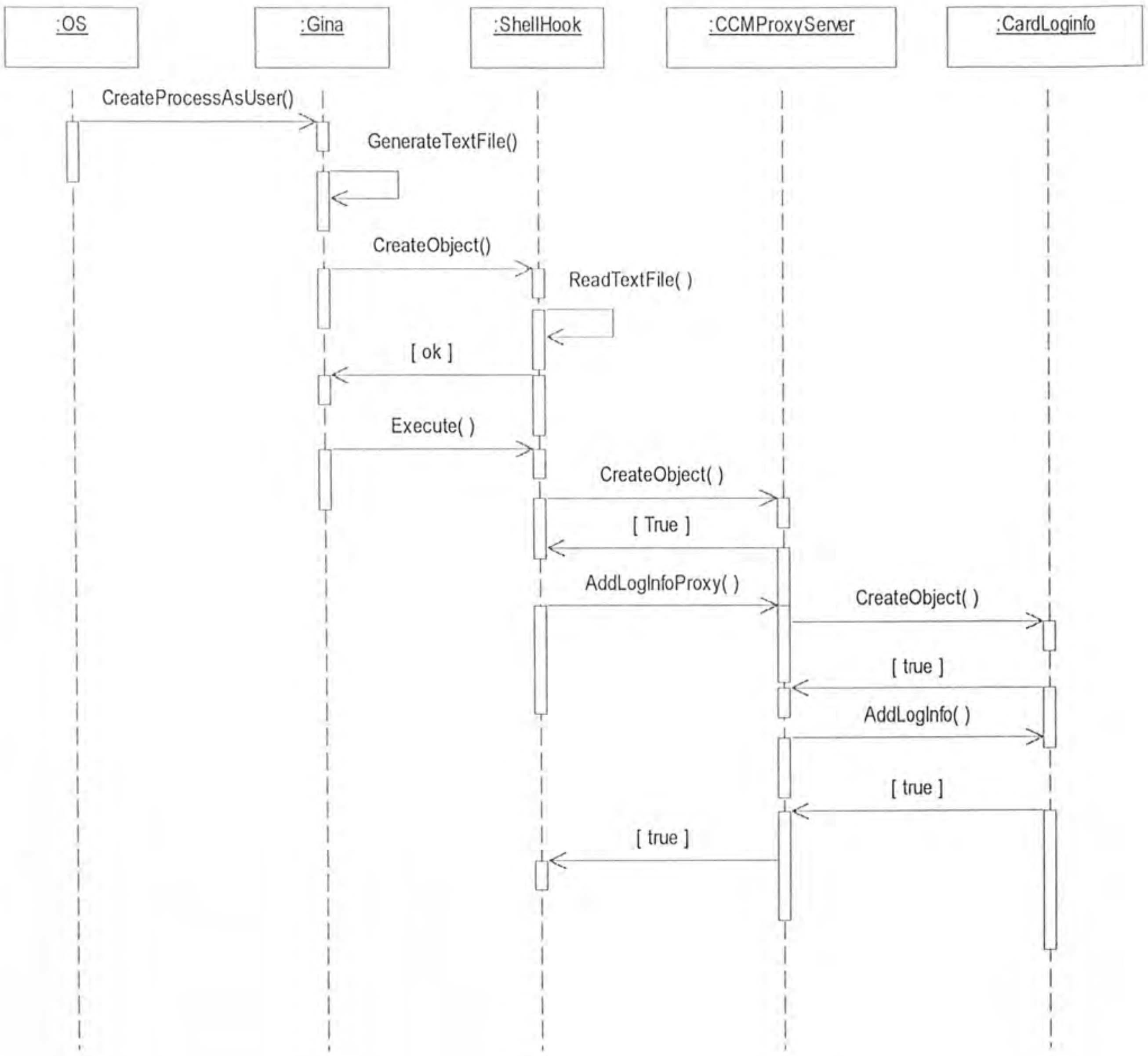
1) Sequence diagram for shutting down the WorkStation



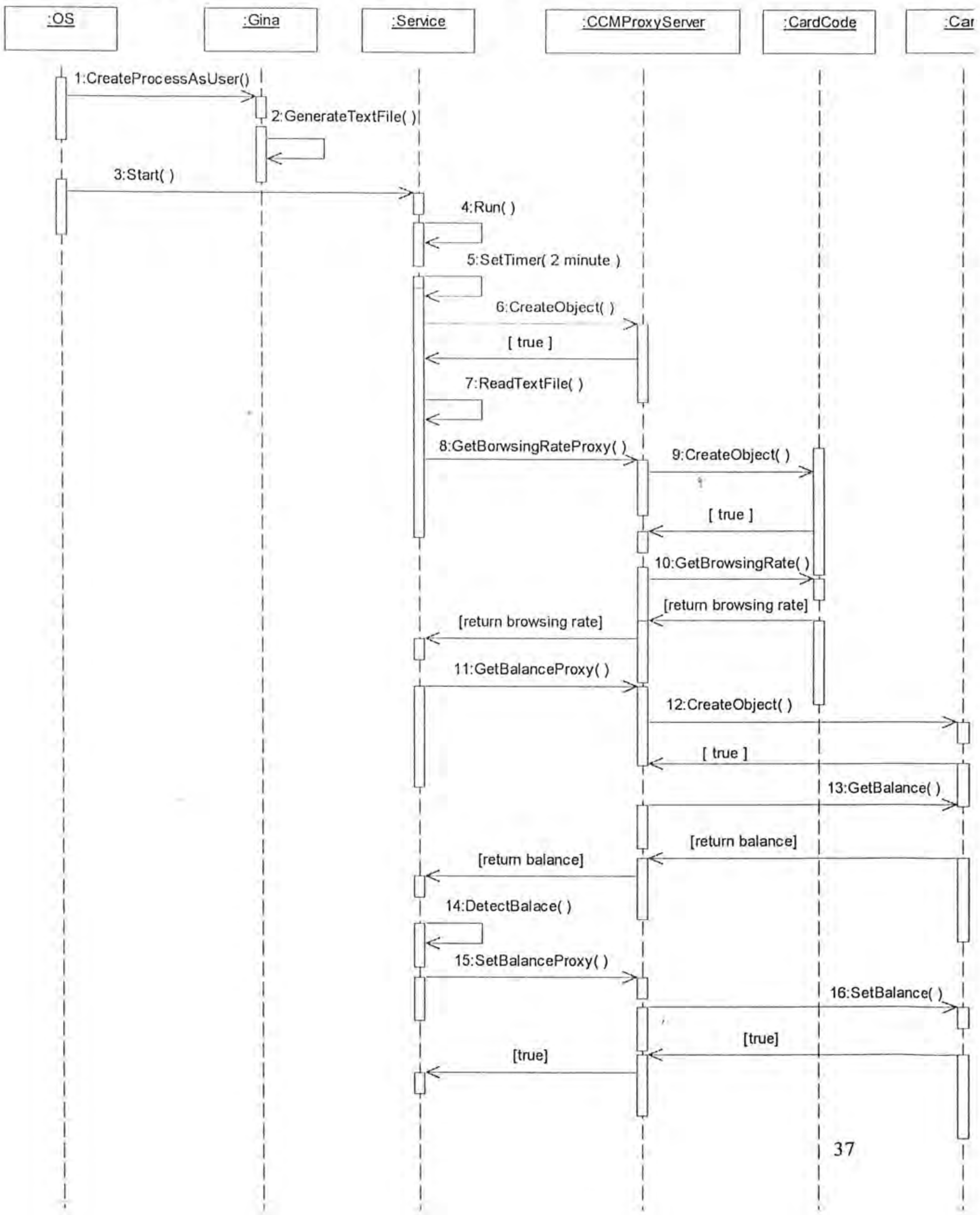
2) Sequence diagram for create new user account.



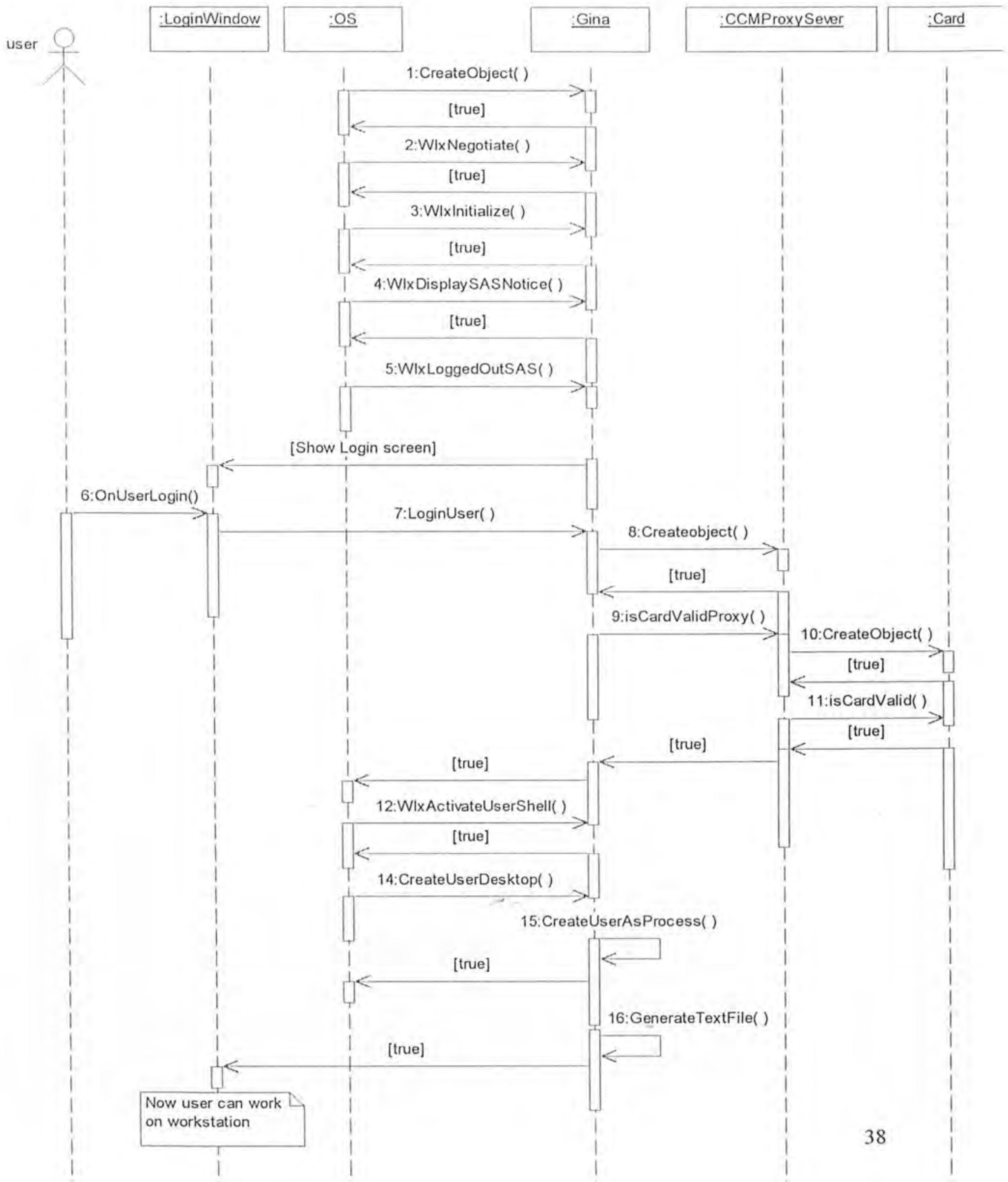
3) Sequence diagram for add log Information into the database.



4) Sequence diagram for deducting the card balance



Sequence diagram for login on remote machine on remote



3.8 Database Design

Database design is extracted from the class relationship diagram. Corresponding classes are converted into database tables and class attributes are treated as field name of table.
These are following classes which are converted into tables.

- Card
- User
- CardLogInfo
- CardUsedInfo
- CardCode

Table Name: Card

Field Name	Data Type	Length	Primary Key
Id	Auto		yes
CodeId	Integer		Foreign key (CardCode)
CardNumber	VARCHAR	15	
Password	VARCHAR	81	
Balance	Currency		
LoginTime	Date/Time		
LoginStatus	Integer		
StartDate	Date/Time		
ExpiryDate	Date/Time		

Table Name: CardCode

Field Name	Data Type	Length	Primary Key
CodeId	Integer		yes
CardType	VARCHAR	10	
BrowsingRate	Currency		
ApplyDate	Date/Time		
Strength	Integer		
CardPrice	Currency		

Table Name: User

Field Name	Data Type	Length	Primary Key
Id	Auto		yes
LoginName	VARCHAR	81	
Password	VARCHAR	81	
UserType	VARCHAR	25	

Table Name: CardUsedInfo

Field Name	Data Type	Length	Primary Key
Id	Auto		yes
CardNumber	VARCHAR	15	Foreign Key (Card)
LoginDate	Date/Time		yes
LoginTime	Date/Time		yes
LogoutTime	Date/Time		
MachineName	VARCHAR	81	

Table Name: CardLogInfo

Field Name	Data Type	Length	Primary Key
Logid	Auto		yes
CardNumber	VARCHAR	15	Foreign Key (Card)
Date	Date/Time		
StartTime	Date/Time		
MachineName	VARCHAR	81	
ApplicationName	VARCHAR	255	
OperationType	VARCHAR	80	

Chapter 4

Technologies and Terminologies

4.1 Dynamic Link Library (DLL)

A file containing executable code and data bound to a program at load time or run time, rather than during linking. The code and data in a DLL can be shared by several applications simultaneously. DLLs are similar to library modules in that they both contain set of functionality that has been packaged for use by the applications. The difference is when the applications link to the library. With the library module (LIB), the application is linked to the functionality in the library during the compile and builds process. The functionality contained in the library file becomes part of the application executable file. With a DLL, the application links to the functionality in the library file when the application is run. DLLs reduce the size of application executable files by placing the functionality that is used by the multiple applications into a DLL that are shared by all the applications. As the same copy of DLLs functions are called simultaneously by multiple applications at the same time so, the functionality in the DLL must be thread-safe. The DLL is loaded into the memory when a process request for it and is remained in memory until at least one process is using it.

4.2 Component Object Model (COM)

It describes a methodology for implementing components: reusable objects that can be attached in different way to create different applications.

4.2.1 Purpose

The Component Object Model (COM) is a platform-independent; distributed, Object-oriented, system for creating **binary software** components that can interact. COM is the foundation technology for Microsoft's OLE (compound documents), ActiveX (internet enabled components).

A binary standard that enables objects to interoperate in a networked environment regardless of the language in which they were developed or on which computers they reside. COM allows an object to expose its functionality to other components and to host applications.

It is not an object-oriented language, but a standard. Nor does COM specify how an application should be structured. Language, structure, and implementation details are let to the application programmer. COM does specify an object model and programming requirements that enable COM objects (also called COM components, or sometimes simply objects) to interact with other objects. These objects can be within a single process, in other processes, even on structurally quite dissimilar. That is why COM is referred to as a binary standard. It is standard that applies after a program has been translated to binary machine code.

4.2.2 Where Applicable

The only language requirement for COM is that code is generated in a language that can create structures of pointers and, either explicitly or implicitly, calls functions through pointers. Object-oriented languages such as C++ and Smalltalk provide programming mechanisms that simplify the implementations of COM objects, but languages such as C, Pascal, Ada, Java, and even BASIC programming environments can create and use COM objects.

COM defines the essential nature of a COM objects. In general, a software object is made up of a set of data and the functions that manipulate the data. A COM object is one in which access to an object's data is achieved exclusively through one or more sets of related functions. These function sets are called **interfaces**, and functions of an interface are called methods. Further, COM requires that the only way to gain access to the methods of an interface be through a pointer to the interface.

Besides specifying the basic binary object standard, COM defines certain basic interfaces that provide functions common to all COM-based technologies. It also provides a small number of API functions that all components requires. COM has now expanded its scope to define how objects work together over a distributed environment, and added security features to ensure system and component integrity.

4.2.3 Using COM Objects

Briefly described, the Component Object Model (COM) is an open architecture for cross-platform development of client/server applications based on object-oriented technology. COM is a way of creating reusable software components. Clients have access to an object through interfaces implemented on the object. COM is language-neutral, so any language that produces ActiveX components can also produce COM applications.

4.2.4 Interfaces, classes, servers

The fundamental idea is that some objects provide services, and are the "**servers**", and some objects use those services, and are the "**clients**". The client needs to have access to the services of the server. A COM object is associated to a number of interfaces, each of which represents a distinct role of the object. Access to a COM object always occurs through the methods of its interfaces. Interfaces and implementations are neatly separated.

Interfaces are implements by COM classes. A single COM class may implement several interfaces (similar to Java). COM classes are packaged into COM servers.

4.2.5 Types of COM Servers

There are two main types of servers **in process** and **out-of-process.** In-process servers are implemented in a dynamic-linked library (DLL), which is loaded into the client's process the first time the client accesses a server's class (fore example, an ActiveX control is an in-process COM server). Out-of-process servers are implemented in an .exe file, and it can run on the same machine of the client or on a remote machine.

All components implemented in Microsoft Transaction Server (MTS) are in-process servers. An out-of-process server is an object application implemented in an executable file that runs in a separate process space. One example of an out-of-process server is a client application calling an .exe file on a server.

Running a component in process with applications often provides superior performance to out-of-process servers.

An interface defines a contract between a COM object and its clients. Every COM interface is derived from the IUnknown root interface and inherits its methods: QueryInterface, which allows to obtain an interface pointer to an interface; AddRef, to increment the counter of clients in an object (the reference counter); Release, to decrease that counter. The first methods allow a client to ask whether an object implements a specific interface. The other two help in the housecleaning of objects (an object always knows how many clients are using it). Access to an object interface occurs via an interface pointer. This points to a table (the "virtual table") of methods pointers, which in turn reference the actual implementations. There are no objects, only interface pointers to objects. When a client requests COM to create an object, COM returns an interface pointer. Therefore, COM objects are not persistent. Objects are addressed and located via a registration process, i.e. via the system registry.

Clients create objects by invoking factory objects, which manufacture objects of different types. Each class is generally associated with a meta-class, or class factory, which is responsible for creating COM objects. The class factory allows clients to create new instances of the class through the IClassFactory interface. The class factory also serves as storage for static data. The client calls the class factory via CoGetClassObject and then creates an instance via CoGetClassObject. Note that COM does not support inheritance but only aggregation (reuse of interfaces).

4.2.6 The Broker pattern: proxies and stubs

If the interface is not physically linked as a DLL to the client that wants to access it (i.e., client and server belong to two separate processes), a proxy is used for this purpose. This is the out-of-process model, as opposed to the in-process model (i.e., DLLs).

COM uses the Broker pattern: clients do not access the server directly but use proxies instead. A proxy implements the same interface as the server object that it represents and it is linked as a DLL to the client. The server has t stub to match the client's proxy. In other words, COM loads a proxy object in the client process space and a stub object in the server process space. The proxy is in process; the stub is out of process. The proxy packages (**marshals**) the parameters and forwards any method call via RPC to the server's stub (which **unmarshals** the parameters and makes the actual call to the server object).

4.2.7 MIDL and COM

Microsoft IDL allows the definition of COM interfaces (identified by an IID), interface categories (identified by a CATID), type libraries (identified by a LIBID) and COM classes (identified by a CLSID). Each interface, class and object is identified by a "globally unique identifier" (GUI). COM generates proxies and stubs based on these definitions.

4.2.8 COM-specific Data Types

The data types specific to COM are: **Variant**, **BSTR**, and **SafeArray.**

Variant

A Variant is a structured data type that contains a value member and a data type member. Actually VARIANT is simply a discriminated union that contains an integer denoting the type (the discriminator) and union containing all of the possible data types. A Variant may contain a wide range of other data types including another Variant, BSTR, Boolean, IDispatch or IUnknown pointer, currency, data, and so on. COM also provides methods that make it easy to convert one data type to another. The **_variant_t** class encapsulates and manages the Variant data type. The need for this data type was due to the different architecture of the machines communicating with each other. The data is marshaled at one side at then successfully unmarshaled at the other side giving the correct value.

BSTR

A (Basic STRing) is a structured data type that contains a Unicode string, string's length and a null terminator. COM provides methods to allocate, manipulate, and free a BSTR. The **_bstr_t** class encapsulates and manages the BSTR data type.

SafeArray

A SafeArray is a structured data type that contains an array of other data types. A SafeArray is called *safe* because it contains information about the bounds of each array dimension, and limits access to array elements within those bounds.

4.2.1 Advantages of components

- Reusability
- Flexibility
- Language independence
- Performance
- Run-time Requirements
- Runs on a wide variety of operating systems.

4.3 Serialization

The ability to serialize data, sometimes referred to as persistence, allows the control to write the value of its properties to persistent storage. Controls can then be re-created by reading the object's state from the storage. A control is not responsible for obtaining access to the storage medium. Instead, the control's container is responsible for providing the control with a storage medium to use at the appropriate times.

4.4 Graphical Identification and Authentication (GINA)

Winlogon is a component of the Microsoft® Windows NT®/Windows® 2000 operating system that provides interactive logon support. Winlogon is designed around an interactive logon model that consists of three components. The Winlogon executable program, a Graphical Identification and Authentication dynamic-link library (DLL)—referred to as the GINA. Winlogon handles interface functions that are independent of authentication policy.

The GINA is a replaceable DLL component that is loaded by Winlogon. The GINA implements the authentication policy of the interactive logon model, and is expected to

Perform all identification and authentication user interactions. For example, replacement GINA DLLs can implement smart-card, retinal-scan, or other authentication mechanisms in place of the standard Windows NT/Windows 2000 user name and password authentication.

Chapter 5

Implementation of Cyber Café Manager

5.1 Language Selection

Language selection is very difficult job and it depends upon the requirements of the system to be developed. Visual languages provide a visual interface and many other facilities, which help the developer making an application, which are more users friendly and attractive. The most common visual languages are Visual C++ and Visual Basic.

Now the selection between these visual languages is problem and depends on the requirements of the application. The key points of the selection may be following.

1. The languages provide the data structure, language constructs for implementing the requirements e.g. a language may not support pointers and arrays and an application required to use these data structures should not use this language.
2. The language should support the approach selected by analyst so that the analysis and design could be fully implemented by the developer.
3. The speed of application required should be fulfilled by the language selected e.g. in the implementation of real time systems some language slower for implementation of these problems.

5.2 Language Selection for Project

For implementation of Cyber Café Manager (CCM) the language "Visual C++ 6.0" is used. The reason for this selections are listed below,

1.Visual C++ provides many built in classes which are helpful for implementation.

2.Visual C++ provides simple way to build user interface.

3.Visual C++ was requirement of the client.

4.It is very fast language and provide facility to build components like COM, DCOM.

5.3 Changes from Design to implementation

The beautiful nature of object-oriented nature is that classes can be added and removed at any time according to the requirements of the user. The classes added in the implementation are of different nature some classes are interface classes, which are added to provide user friendly interface. Some classes are used as wrapper of other classes that enable the communication between classes. In other words if one class is used the functionality of other class it is used the wrapper class as medium way for communication.

5.4 User Interface Classes

These classes are identified for user interface including

- CLoginDlg
- CReportDlg
- CUserDlg
- CMainDlg
- CPrintDlg
- CCardDlg
- CGenerateCardDlg
- CCardUsedDlg
- CMenuDlg
- CViewDlg
- CEventDlg
- CShutDownDlg
- CClientDlg
- CMessageDlg
- CLogViewDlg

5.5 Buil$\overset{t}{\text{d}}$ in Classes

These are building classes which are used in the system.

- CDialog
- ClistCtrl
- COleDateTime
- CButton
- CTab
- CString
- CImageList
- CRegKey

These are main classes of the visual C++ and many other control classes are also used in the system.

5.6 Wrapper Class

This is class that is used to establish communication between two classes.

- CCMProxyServer

5.7 Classes Division for Components

System classes are divided into different components. These components are listed below including classes for each.

- **CafeCom DLL**

  This COM base DLL is used to communicate with database includes these classes.

  1. Card
  2. CardCode
  3. CardUsedInfo
  4. CardLogInfo
  5. Message
  6. User
  7. CCardFile
  8. CLogfileView
  9. CCardUsedFile
  10. CafeCom

- **CCMProxyServer**

  Actually this component is DCOM which is used as wrapper for CafeCom DLL to communicate with clients remotely. CCMProxySever has interface classes only which is used for marshalling the data over the entire network.

- **Main**

  This component is including user interface related classes. It is used for administrators and café employees.

- **Gina DLL**

  This DLL is used for client side interface it work with operating system to provide security for login into the system it includes two classes CClientDlg and Gina.

- **Service**

  This COM base service is running in background for remote machine and includes only one class Service.

- **Shell Hook**

  This COM base DLL is used for client to capture the remote machine activities it includes ShellHook class only.

- **System Tray Icon**

  This exe is also used for client machine to create the system tray icon. It includes two classes Menu and SystemTrayIcon.

5.8 Implementation Detail of Components

- **Café Com DLL**

  Common attributes and methods for each COM class are added at implementation time those are listed below.

Attributes:

```
CObArray  m_obArray;
 BSTR      m_ErrorMessage;
```

Methods:

VariantToArray (VARIANT variant);

This method converts the data in variant to corresponding class attribute

VariantToMem (const VARIANT &vtVariant, HGLOBAL* phData)

This method converts the variant to memory file such that it can be converted to class attributes.

FileToVariant (CFile &file, VARIANT &vrResult)

This method converts the attribute of class into variant data type

So, above these methods are used for marshalling and unmarshalling of data.

GetLastError (BSTR* error)

This method gets the error message if occur in the DLL.

Some classes are added in CaféCom DLL at implementation time so the description of these classes are listed below

Class Name: CCardFile

Purpose: This class is used to serialize the class data members of Card and CardCode classes which are passed through its constructors. This serialize data is used for marshalling purpose.

Methods:

CCardFile ();

~CCardFile ();

Void Serialize (CArchive &ar);

Void SerializeOne (CArchive &ar);

Void SerializeTwo (CArchive &ar);

CCardFile (CString Cardid, CString Machine, CString Balance,
COleDateTime LoginTime);

CCardFile (CString Cardid, CString Balance, CString Cardtype,
COleDateTime IssueDate, COleDateTime ExpirDate);

CCardFile (int code, CString cardtype, int strength, int rate,
COleDateTime IssueDate, COleDateTime ExpirDate);

Attributes:

CString m_strCardid, m_strCardType, m_strMachine, m_strBalance;
COleDateTime m_strLoginTime, m_strIssueDate, m_strExpirDate;
Int m_iCode, m_iRate, m_iStrength;

Class Name: CLogfileView

Purpose: This class is used to serialize the class data members of CardLogInfo class which are passed through its constructors. This serialize data is used for marshalling purpose.

Methods:

CLogfileView ();

virtual ~CLogfileView ();

void Serialize (CArchive &ar);

CLogfileView (CString Cardno, CString Machine, COleDateTime Date, COleDateTime Time, CString Type, CString AppName);

Attributes:

CString m_strCardno, m_strMachine, m_strType, m_strAppName;

COleDateTime m_strDate, m_strTime;

Class Name: CCadUsedFile

Purpose: This class is used to serialize the class data members of CardUsedInfo class which are passed through its constructors. This serialize data is used for marshalling purpose.

Methods:

CCadUsedFile ( );

virtual ~ CCadUsedFile ( );

void Serialize (CArchive &ar);

CCadUsedFile (CString Cardid, COleDateTime Logindate, COleDateTime starttime, COleDateTime endtime, CString machine);

Attributes:

CString m_strCardid, m_strMachineName;

COleDateTime m_strLogindate, m_strStartTime m_strEndTime;

- **CCMProxyServer** (DCOM)

This component has the following list of interfaces which are used for proxy/stub purpose. It is also used to communicate among clients and server side DLL (CafeCom.DLL).

1. interface IProxyCard : IDispatch
2. interface IProxyCardCode : IDispatch
3. interface IProxyLogfile : IDispatch
4. interface IProxyMessage : IDispatch
5. interface IProxyCardUsed : IDispatch
6. interface IProxyUser : IDispatch

- **Main (Administrator Interface)**

This component is used for administrator interface. It includes only interface dialogs. All dialogs have these common methods and derived from CDialog class.

1. virtual void DoDataExchange(CDataExchange* pDX);
2. virtual BOOL OnInitDialog( );
3. afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
4. afx_msg void OnPaint( );
5. afx_msg HCURSOR OnQueryDragIcon( );
6. afx_msg HBRUSH OnCtlColor(CDC* pDC, CWnd* pWnd, UINT nCtlColor)
7. OnOk( );
8. OnCancel( );

Note: Remaining component classes are used with little modification in design so their descriptions and purposes are described in the design chapter.

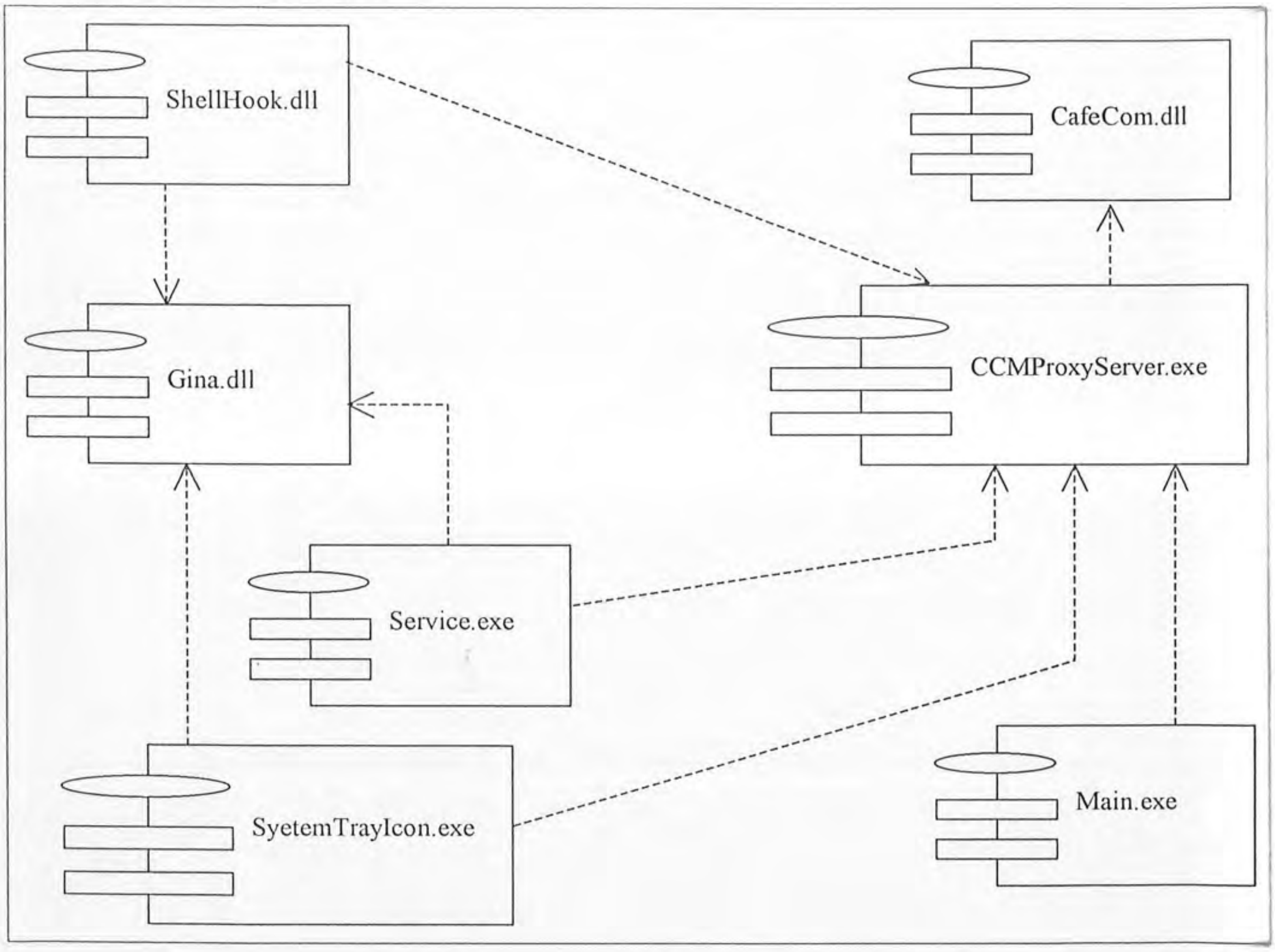
5.9 MSADO15.dll

This COM base system DLL is used for database connectivity.

5.10 Implementation Diagrams

Component Diagram

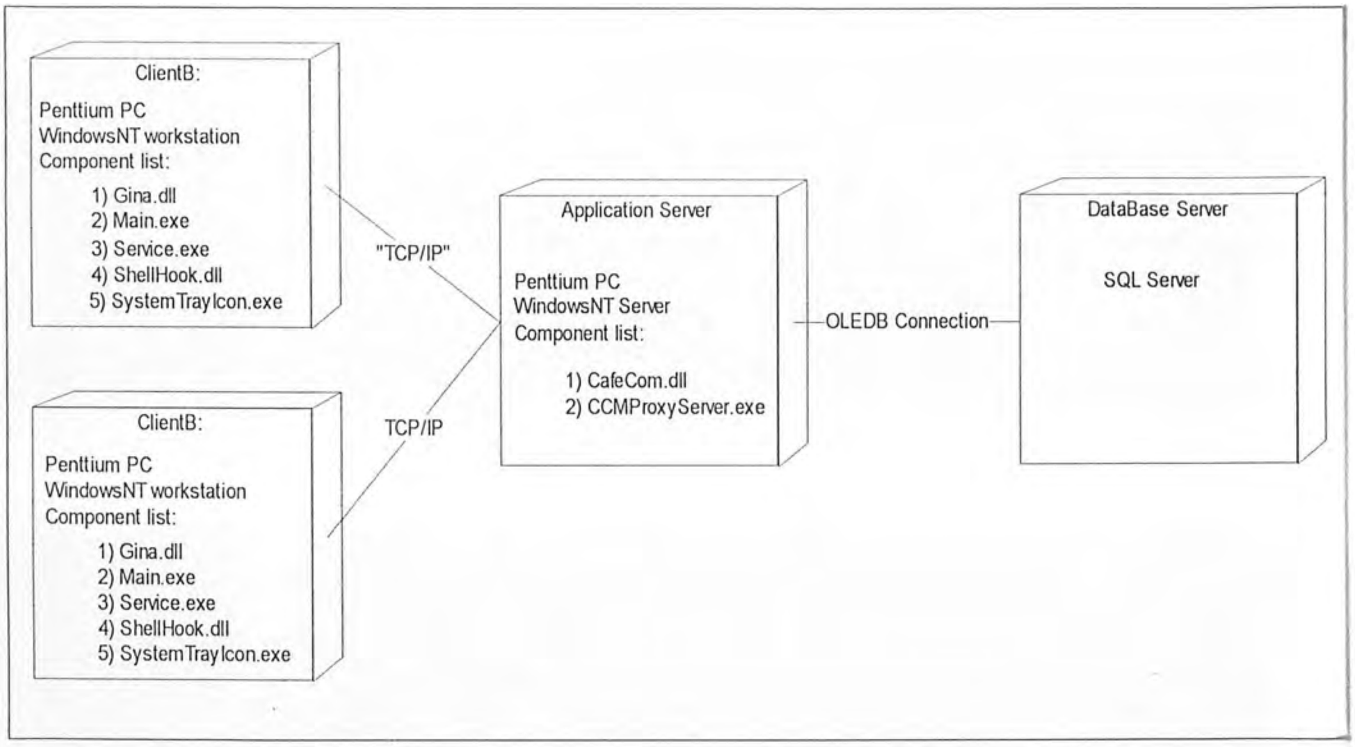
A component diagram shows the physical structure of the code in terms of code components. A component can be a source code component, a binary component, or an executable component. A component contains information about the logical class or classes it implements, thus creating a mapping from the logical view to the component view. Dependencies between the components are shown, making it easy to analyze how other components are affected by a change in one component. It is also called implementation diagram.



A Component Diagram of CCM

Deployment Diagram

The deployment diagram shows the physical architecture of the hardware and software in the system. It also shows the actual computers and devices (nodes), along with the connection they have to each other; it also shows the type of connections. Inside the nodes, executable components and objects are allocated to show which software units are executed on which nodes.



Deployment Diagram of CCM

Chapter 6

Testing of Cyber Café Manager

6.1 Introduction to Testing

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design, and coding. Testing a software system typically occupies a large percentage of the time spent developing the complete system. Yet, the amount of scientific effort devoted to this stage of software production is tiny by comparison with those of analysis, specification, design and coding.

6.2 Testing Process

Testing is often presented as if it is a separate and separable process in the software lifecycle. In fact, testing can (and should) be carried out at each stage in the production and maintenance of a piece of software. There are two testing philosophies that represent two extremes in software testing Bottom-up and Top-down.

In top-down testing, we produce and test the highest-level structure of system first and individual module is tested later. But bottom-up testing, as the name suggests, proceeds in the opposite direction. Individual modules are tested first, and then integrated. ***I follow the bottom-up approach for testing the system.***

6.3 Testing Strategies

Since program testing can only demonstrate the presence of errors, a successful test strategy is one that finds bugs. This is as Somerville notes (p. 407), an inherently destructive process. There are two strategies of testing Black-box and White-box testing. In black-box testing only input and output is considered or in other words to check the requirements of the system fulfill or not. But white-box testing is used check the logic and code of each module. So it is internal level testing or code testing. ***I follow the Black-box testing only***.

6.4 Unit Testing

In this testing methodology each component is testing individually.

Test Case 1

I first check the Café Com DLL that it works correctly it loads at run time into the client process area efficiently. So it is loaded and working properly with client exe.

Test Case 2

In this test case the DCOM server (CCMProxyServer) is tested. Register the component on the server and workstation and then it is accessed remotely. So it is working properly.

Test Case 3

This test case is used for Service component. Install the service on the client machine and then check it is started when operating system is loaded. So, it starts properly when operating system is started.

Test Case 4

This test case is used for Main component which is used for administrator activities. It works properly and fulfills the requirements of the administrator.

Test Case 5

This test case is used for ShellHook DLL component which is used to capture the activities of the user on each workstation. First of all it register on the client machine and then start shell application so it works properly and capture name, time, date and operation type related to this shell application.

Test Case 6

This test case is used to capture the bugs in the GINA DLL. When operating system starts it loads MyGina.DLL into memory and it works properly with winlogon.exe of the operating system. It also fulfills all the security features of system.

Test Case 7

This test case is used to draw the icon in the system tray when operating system is started. So it is created accurately in the system tray and receives all mouse events.

6.5 Integration Testing

In integration testing all modules are integrated into one and check the behavior of the system. So all modules are integrated into one on the single machine and also check on the network so it works well. Five components are installed on client machine and two components are installed on the server machine and then these five components (MyGina.DLL, SystemTaryIcon.exe, ShellHook.DLL, Service.exe and Main.exe) access DCOM (CCMProxyServer.exe) object from server side so its object is created on server machine and gives services to client machine components accurately.

6.6 Validation testing

Validation testing phase comes after all the components are integrated together. Validation testing is performed to check user viable input to get recognizable output of the system. It is performed to find if the software conforms to the system requirements. Validation testing errors are removed.

6.7 Regression testing

Regression testing is performed after an error has been removed from the system. So that the removal of errors have not created other errors. Regression testing is performed after each testing activity. Some new errors are found and removed from the system. So that system is working well.

6.8 Other Quality Attributes

A well tested system results into a quality software. But there are other quality attributes which should be tested in order to confirm the quality of software. Other quality attributes are explained below

6.8.1 Reliability

The system is designed to reliable after meeting the user requirements.

6.8.2 Modularity

System is implemented using object oriented approach, which is modular approach for software development. Moreover design is made in the way to make the development more modular.

6.8.3 Portability

System was required to run on any 32-bit operating system. The system is tested on Window 95, Window 98, Window NT, Window 2000 professional and window 2000 advance server and it is working correctly.

6.8.4 Usability

CCM is designed as user-friendly software. The interface is designed according to Microsoft standards, so the user of the windows operating systems will not get frustrated with the Cyber Café Manager.

6.8.5 Efficiency

Efficiency is the first consideration while developing this software. The system is developed efficient enough that user don't feel any delay in any activity involved in the usage of Cyber Café Manager.

6.9 Errors in system

The system is developed with the best efforts to give quality software. But fact is nothing is perfect. So there will be errors in the software also.

6.10 Card Security Testing

These test cases are generated to confirm the security of card.

Test Case 1: Login with correct card number and password.
Input:
Cardno: 121956808597.
Password: Irfan.

Output: Login successfully.

Test Case 2: Login with invalid card number
Input:
Cardno: 1219536598123 5808597.
Password: Irfan.

Output: Invalid card number and not allow to login.

Test Case 3: Login with invalid password.

Input:

Cardno: 121956808597.
Password: lpspss@*.

Output: System can't allow to login due to invalid password.

Test Case 4: Login with spaces.

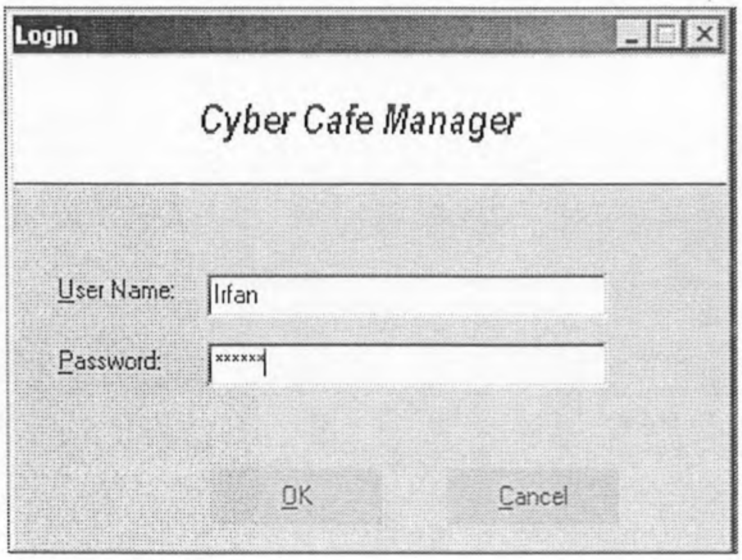
Input:

Cardno: Spaces
Password: Spaces

Output: System can't allow to login due to invalid card number and password.

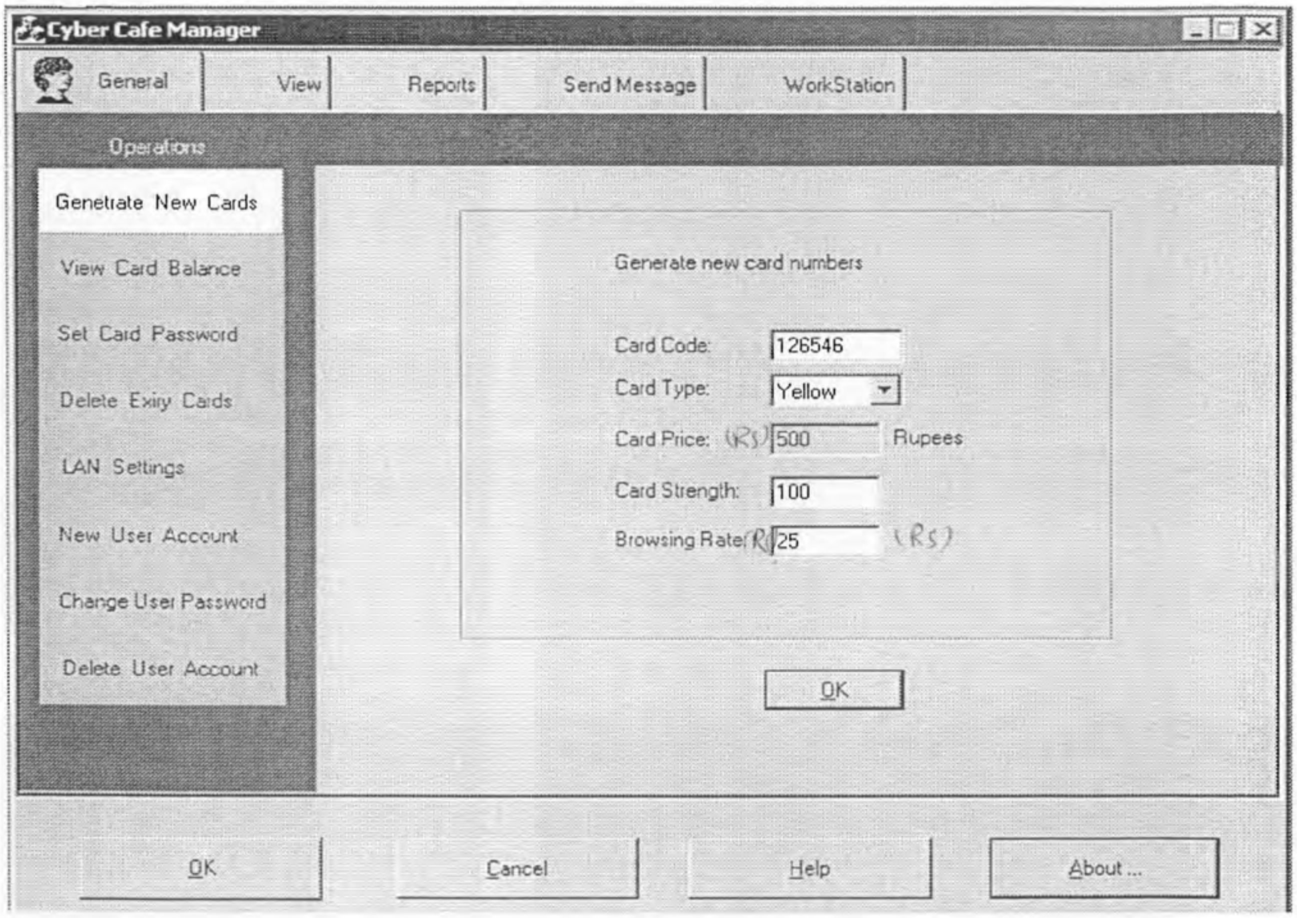
Appendix

This dialog is used for users to login into the system.



User Name: Irfan
Password: ******

This dialog is used to generate new card numbers.



Generate new card numbers

Card Code: 126546

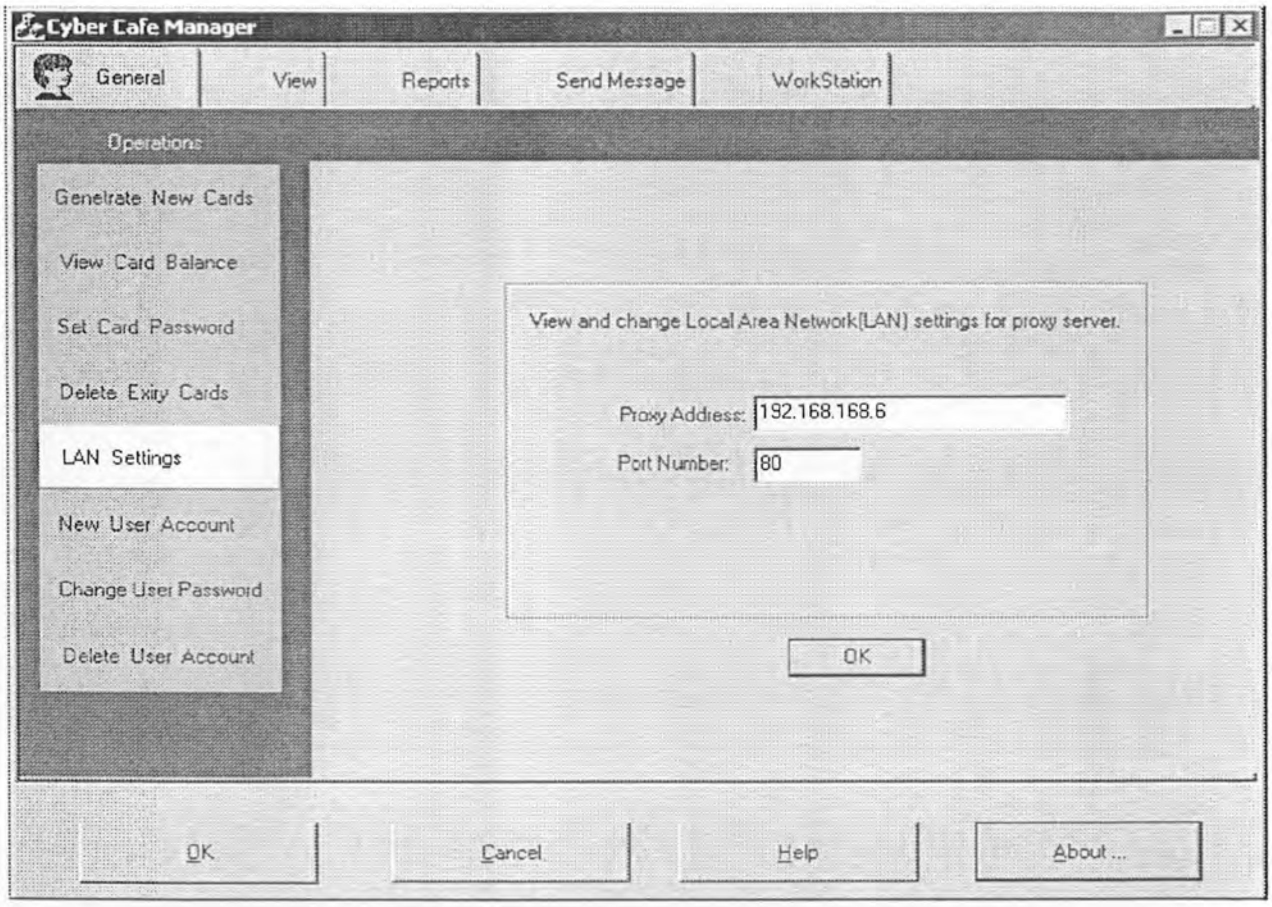
Card Type: Yellow

Card Price: (Rs) 500 Rupees

Card Strength: 100

Browsing Rate (Rs) 25 (Rs)

This window is used to update LAN settings for proxy server.

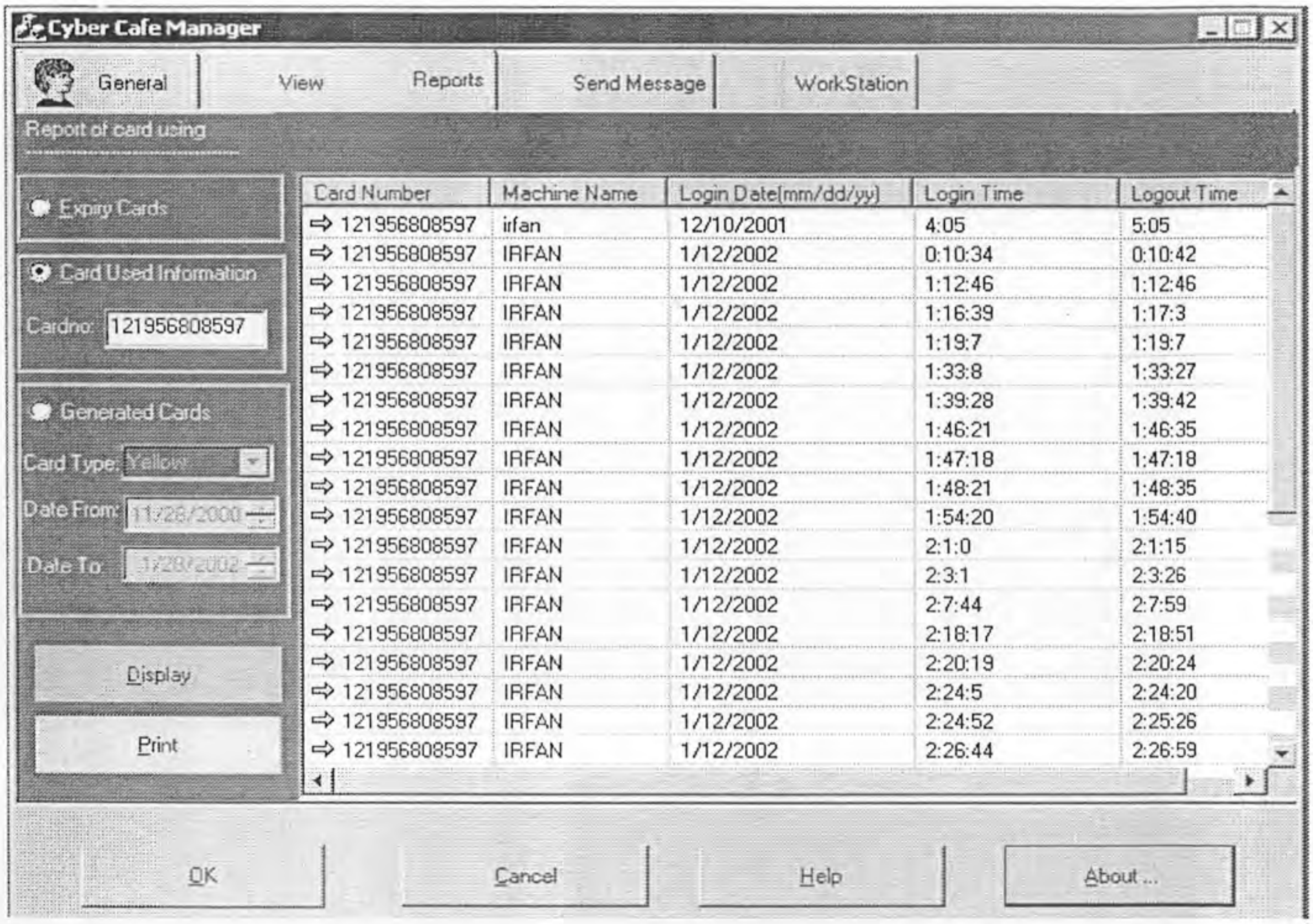


View and change Local Area Network(LAN) settings for proxy server.

Proxy Address: 192.168.168.6

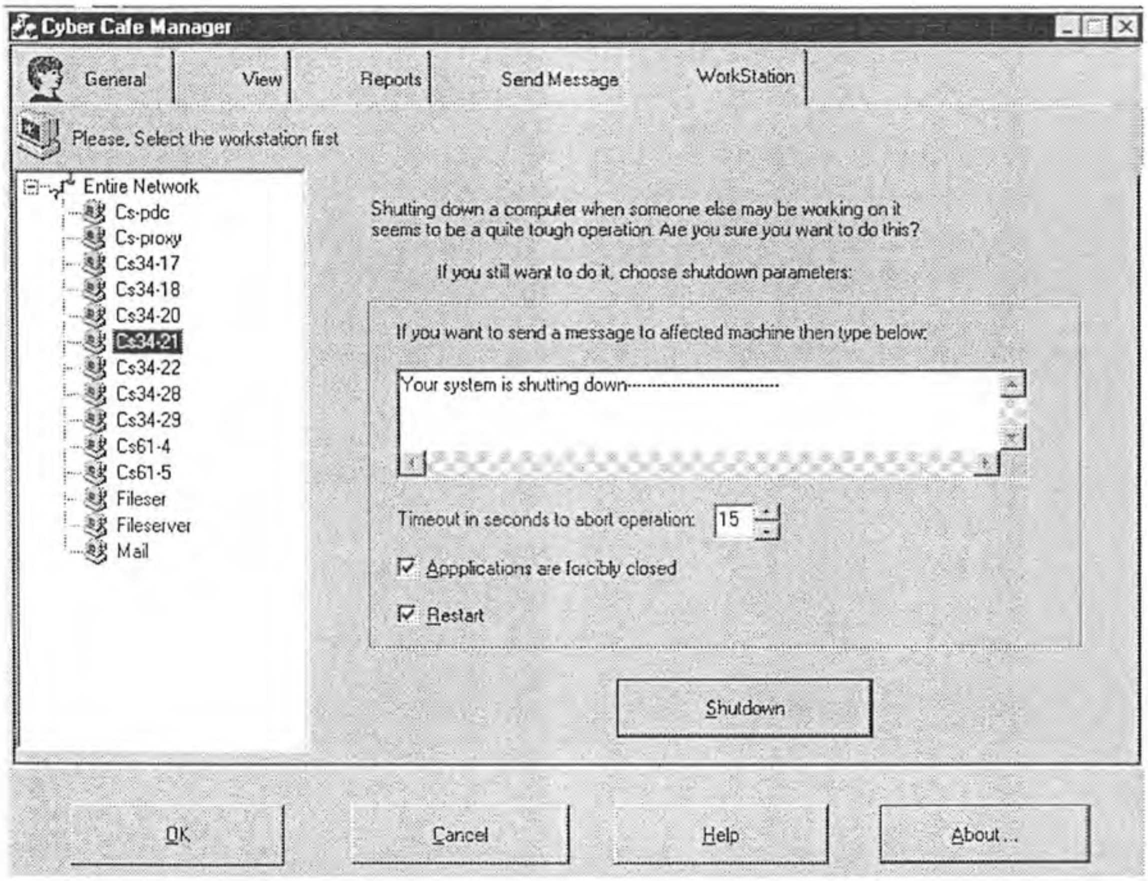
Port Number: 80

This window is used for to display and print the reports



Card Number	Machine Name	Login Date(mm/dd/yy)	Login Time	Logout Time
⇨ 121956808597	irfan	12/10/2001	4:05	5:05
⇨ 121956808597	IRFAN	1/12/2002	0:10:34	0:10:42
⇨ 121956808597	IRFAN	1/12/2002	1:12:46	1:12:46
⇨ 121956808597	IRFAN	1/12/2002	1:16:39	1:17:3
⇨ 121956808597	IRFAN	1/12/2002	1:19:7	1:19:7
⇨ 121956808597	IRFAN	1/12/2002	1:33:8	1:33:27
⇨ 121956808597	IRFAN	1/12/2002	1:39:28	1:39:42
⇨ 121956808597	IRFAN	1/12/2002	1:46:21	1:46:35
⇨ 121956808597	IRFAN	1/12/2002	1:47:18	1:47:18
⇨ 121956808597	IRFAN	1/12/2002	1:48:21	1:48:35
⇨ 121956808597	IRFAN	1/12/2002	1:54:20	1:54:40
⇨ 121956808597	IRFAN	1/12/2002	2:1:0	2:1:15
⇨ 121956808597	IRFAN	1/12/2002	2:3:1	2:3:26
⇨ 121956808597	IRFAN	1/12/2002	2:7:44	2:7:59
⇨ 121956808597	IRFAN	1/12/2002	2:18:17	2:18:51
⇨ 121956808597	IRFAN	1/12/2002	2:20:19	2:20:24
⇨ 121956808597	IRFAN	1/12/2002	2:24:5	2:24:20
⇨ 121956808597	IRFAN	1/12/2002	2:24:52	2:25:26
⇨ 121956808597	IRFAN	1/12/2002	2:26:44	2:26:59

This window is used for shutdown or restarts any workstation



Shutting down a computer when someone else may be working on it seems to be a quite tough operation. Are you sure you want to do this?

If you still want to do it, choose shutdown parameters:

If you want to send a message to affected machine then type below:

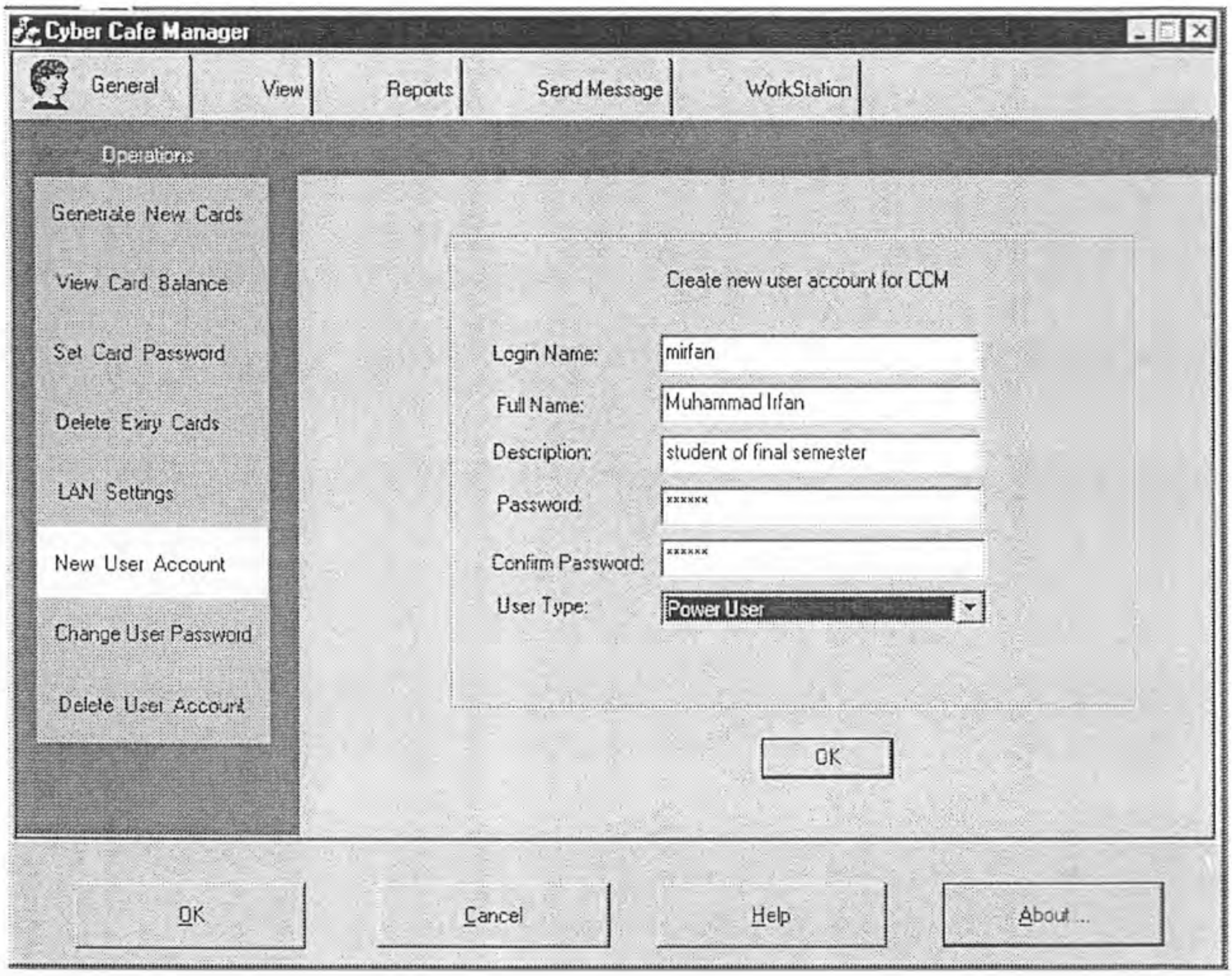
Your system is shutting down------------------------------

Timeout in seconds to abort operation: 15

☑ Applications are forcibly closed

☑ Restart

This window is used to create new café user account.



Create new user account for CCM

Login Name:	mirfan
Full Name:	Muhammad Irfan
Description:	student of final semester
Password:	xxxxxx
Confirm Password:	xxxxxx
User Type:	Power User

OK

Font
verdana 8

Bibliography

1. Horton, Ivor (1998). Beginning Visual C++6.0 SHROF Publication.
2. Gregory, Kalt (1998). Using Visual C++6.0 Prentice Hall.
3. David white Kenn Scubner, Eugence olofsen, etal. (1999) MFC Programming with visual C++ 6.0 Tec media.
4. Wynkoop, Stephen (1999). Special edition using Microsoft SQL server7.0 Prentice Hall.
5. Pressman, Roger's (1997). Software engineering a practitioner' approach fourth edition.
6. Hans-Erik Eriksson Magnus Pinker (1998). UML Toolkit.
7. Rob Pooly and Perdita Stevens (1999). Using UML Software engineering with objects and components.
8. Lan Somerville (Lancaster University) (1998). Software Engineering (fifth edition).
9. VISUAL C++ WINDOWS SHELL PROGRAMMING by Dino Esposito, Wrox Press Ltd.
10. Beginning ATL 3 COM Programming By Dr Richard Grimes, Wrox Press Ltd.
11. MSDN online help.
12. http://www.codeguru.com
13. http://www.codeproject
14. http://www.Microsoft.com
15. http://www.Jbulider.com
16. http://www.google.com