INTERACTIVE VOICE RESPONSE SYSTEM

BY:

ARVIND SINGH DOTIYAL - 031404
RAJAT GULATI - 031277
SAURABH PAL - 031232

JAYPEE UNIVERSITY OF
INFORMATION TECHNOLOGY

MAY 2007

Submitted in partial fulfillment of the Degree of Bachelor of Technology

DEPARTMENT
OF
COMPUTER SCIENCE

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY-WAKNAGHAT

CERTIFICATE

This is to certify that the work entitled, "**Interactive Voice Response System**" submitted by **Arvind Singh Dotiyal(031404), Rajat Gulati(031277)** and **Saurabh Pal(031232)** in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science of Jaypee University of Information Technology has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

28.05.07

Mr. Ajay Kumar Singh
(Senior Lecturer)
Jaypee University Of Information Technology-Waknaghat

TABLE OF CONTENTS

LIST OF FIGURES PAGE NO.

LIST OF ABBREVIATIONS

IVR - Interactive voice response system

TAPI - Telephone application program interface

PABX – Private branch exchange

CSV - Comma separated value

EPABX - Electronic Private Automatic Branch Exchange

PSTN -Public switched telephone network

DAO - Data Access Objects

RDO - Remote Data Objects

ADO - ActiveX Data Objects

COM - Component Object Model

SOAP - Object Access Protocol

ASP - Active server pages

HTML - Hypertext markup language

XML - Extensible markup language

SOAP - Simple object access protocol

API - Application program interface

CLS - Common Language Specification

TSPI - Telephony Service Provider Interface

POTS – Plain old telephone service

ISDN - Integrated Services Digital Network

TSP - TAPI Service Provider

VB – Visual basic

WOSA - Windows Open Services Architecture

DFD – Data flow diagram

PROJECT DEFINITION

ABSTRACT

IVR - Interactive Voice Response is a blanket term for automated call handling systems where the user interacts with a computer controller voice signal (either recorder real speech or computer generated). The interaction can be through the use of a touch tone telephone or through speech recognition.

Interactive voice response: the link between people using the phone and computer databases. This technology (along with automatic speech recognition, ASR) allows callers to speak in their natural voice to complete transactions or queries over the phone.

The system can be helpful in providing an automated twenty four hours services that do not require an operator to deal with the queries. Only a certain number of queries can be answered over the phone as it is an automated system.

DEFINITION

IVRS answers enquiries by prompting callers to input data onto the touch-tone keypad, looking up the record in a database and speaking back information. A typical IVRS application initiates a predefined sequence of verbal prompts, providing options and instructions to a caller. It allows a caller to respond to verbal prompts and instructions through telephone touch-tone keys. Thus, users can interact with the IVRS by pressing buttons on their telephone or by Voice Recognition- simply speaking commands.

The user calls on a telephone number, the Voice System answers the call, greets the customer and prompts for instructions via spoken menu. In response to the caller's commands, database information may be retrieved or service requests executed. Responses to the caller are pre-recorded via a digitized voice.

The system here greets the caller appropriately and asks the caller to enter a valid roll number corresponding to which he or she wishes to retrieve their percentage achieved in the current semester. If the user enters an invalid roll number then his/her percentage will not be spoken out and a blank outgoing message is spoken out.

The retrieval of information is done through a database which can be easily updated by the administration using a friendly easy to use front end. The database can be viewed by the administration at any point of time and new entries can be made into it.

TYPICAL IVR SYSTEM DESIGN

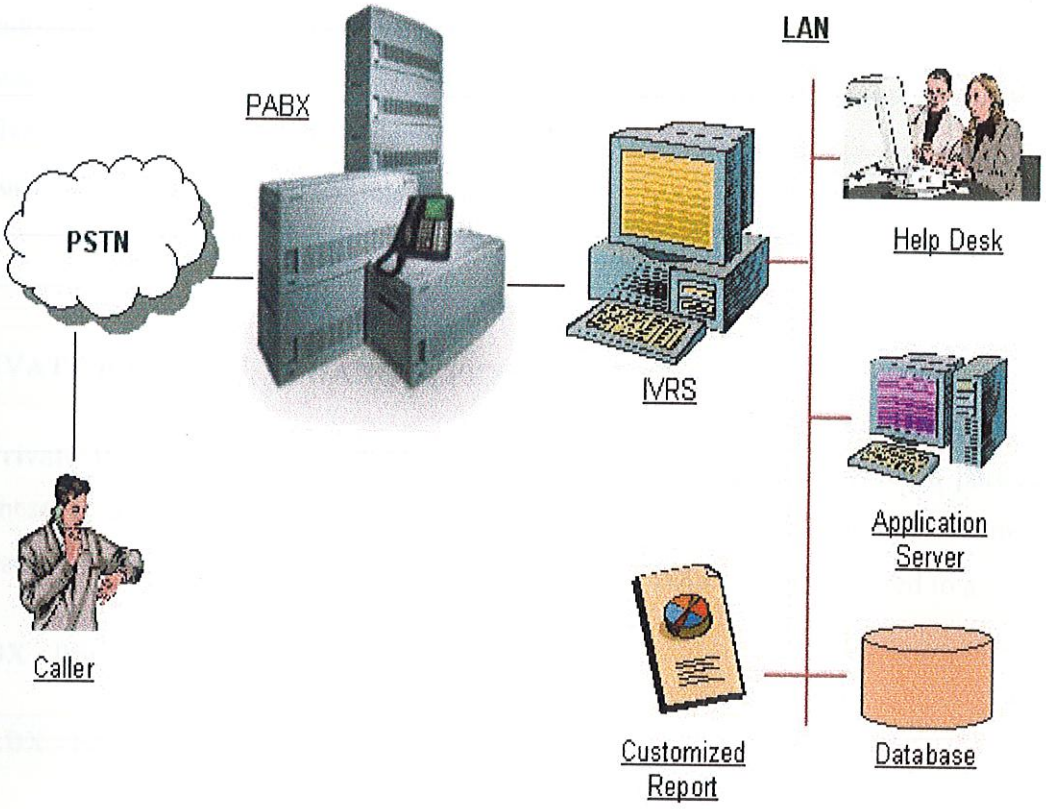


Fig 1: IVR SYSTEM DESIGN

The IVR system is setup on a computer system that may be or may not be connected to a LAN. The IVR system may gather the database information over the LAN, customized reports can also be generated.

The database at any time can be updated for new records. At any point of time the IVR system can transfer the call to the help desk where additional unanswered queries can be answered.

On the other hand the system may be connected to a PABX that helps the system handle more than one call using a single telephone line. The PABX is further connected to a telephone line over which the caller calls.

PUBLIC SWITCHED TELEPHONE NETWORK (PSTN)

The **public switched telephone network** (PSTN) is the network of the world's public circuit-switched telephone networks, in much the same way that the Internet is the network of the world's public IP-based packet-switched networks. Originally a network of fixed-line analog telephone systems, the PSTN is now almost entirely digital, and now includes mobile as well as fixed telephones.

PRIVATE BRANCH EXCHANGE (PBX)

A **Private Branch eXchange** (PBX) is a telephone exchange that serves a particular business or office, as opposed to one that a common carrier or telephone company operates for many businesses or for the general public. PBXs are also referred to as:

PABX - Private Automatic Branch eXchange

EPABX - Electronic Private Automatic Branch Exchange

AIM AND SCOPE

The aim of the project is to provide as many Interactive Voice functionalisms as possible. The project aim at providing easy to use service that just requires a caller to enter his/her roll number through there phones keypad and the system will retrieve the percentage corresponding to that particular roll number.

The project is also aimed to provide convenience to the administration, who at any time can update the database containing the result. The result can also be viewed by the administration at any time and updates can be made.

The scope of the project is limited up to providing service to a single user at a time as hardware requirement grows and becomes a lot more expensive in providing services to more than one user at a time.

The project aims at providing:

- Automatically handles every call from customers and agents.
- Answers calls on the first ring, 7 days a week, 24 hours a day.
- Quick response to customer requests.
- Speaks to callers in a human tone of voice.
- Provides accurate and reliable information, up-to-date.
- Resources the organization's present personnel for more productive work

The IVRS system can be expanded to include all types of complaints to eliminate the use of online operators. It can also be expanded to include help and information regarding the company

FEASIBILITY SCOPE

ECONOMICAL FEASIBILITY

The proposed system will be a lot faster, easier to handle and error free.The system cost will be far less distinct from the benefits it will bring. Thus the system is economically feasible.

TECHNICAL FEASIBILITY

The software and hardware required for the implementation of the system are available. Thus the system is technically feasible.

OPERATIONAL FEASIBILITY

The proposed system will reduce the workload on operators and will make the work a lot easier. As there is no large number of operators working, there will less operational cost during the whole process.

PROJECT DESIGN

The project design constitutes of a front end that allows administration of an institution to update the result database, this database contains the result information of all the students. This result can also be viewed at any point of time and updated.

FRONT END

WHY .NET?

.NET initiatives can be thought as convergence of several distinct but loosely tied goals, the most important of which are overcoming the limitations of the COM(Component Object Model) programming model and finding a common programming paradigm for Internet-related applications.

ISSUES WITH TODAY'S SOFTWARE DEVELOPMENT

WINDOWS INCONSISTENCIES

The windows platform has evolved chaotically in recent years. We have at least three different programming models for producing graphic-intensive applications (GDI, DirectX, and OpenGL). Microsoft has also produced several programming models for accessing databases – Data Access Objects (DAO), Remote Data Objects (RDO), and ActiveX Data Objects (ADO).

Windows itself comes in so many versions – Windows 95, 98, Me, NT, 2000, CE, XP and not all versions support all features.

Security is another area of concern that's heavily affected by the Windows platform developers. To make things worst, a developer must often account for the difference between Windows security and the "classic" COM security model, which in turns is different from the COM+ security model. With all these different models to consider, it's no surprise that security is often an afterthought in most enterprise applications, which makes them vulnerable to attacks.

COM SHORTCOMINGS

The Component Object Model (COM) has proven to be versatile enough to work as the infrastructure for many enterprises and distributed applications, but at the same time it has shown itself to be just too complex for developers who want to focus on the business problem. COM applications are inherently fragile because they depend heavily on the information stored in the system Registry, which become corrupter easily.

COM applications suffer from another serious problem: Versioning. As we know, each newer version of a CO component overwrites older version. In theory, this overwriting should encompass all the functionality of previous versions. In practice, however, this means that an application that uses older version can suddenly stop working if the user installs later versions because of minor incompatibility among these versions.

A third defect of COM is in deployment. Installing a COM-based application isn't a simple task because you have to install all the necessary components in system directories, register them in the Registry, and configure them, and so on. Writing installation programs has become an art of its own, with tricks and secrets.

OBJECT-ORIENTED ISSUES

Programming would be much easier if developers have a simple and secure way to reuse and extended code that they wrote, but the COM programming model offers very little in this respect. Inheriting and extending Windows functions is even more difficult because the Windows API doesn't expose classes.

As all experienced developers know, two COM objects that establish a circular reference to each other aren't automatically released when they go out of scope, such as when you have two objects in parent-child relationship or two Person objects that reference each other through their Spouse property. The COM memory manager relies on the concept of a reference counter and has no way to detect when the main program has no reference to a given object if the object is also pointed to by another object. As a result, the two objects continue to remain in the main memory until the application ends.

CROSS-LANGUAGE INTEROPERABILITY

The Windows platform makes it extremely difficult to integrate pieces of code written in different languages because each language has its own call conversions, limitations, and idiosyncrasies.

Because integrating different languages is so difficult, many software shops prefer to standardize on a single language, thus missing the particular advantages that each language can offer. If cross-language interoperability were easier, each portion of a large application might be developed with the language that fit better, and software development would be less expensive in terms of both time and money.

THE PROBLEMS OF ACTIVE SERVER PAGES

The most important goal for pushing Microsoft towards a new programming model was to provide a consistent platform for delivering robust Internet application.

After many attempts to provide credible Internet programming tools – a list that includes ActiveX controls, ActiveX documents, and Web Classes – Microsoft discovered that its most widely accepted platform for the ASP has one great advantage: it delivers dynamic page content that works well with any browser. This technology is fueling many Internet applications of all sizes. The many shortcomings of ASP are following:

- You can use only script languages, so you're subject to many limitations. Programs are interpreted and not compiled, so they execute slower; code uses late binding, and any misspelled method name raises an error at run time instead of compile time; script languages can't access advanced features in the Windows API.
- ASP code is intermixed with user-interface (UI) code – that is, the HTML sent to the browser. This makes it exceedingly difficult to separate the business logic from the UI code and makes it nearly impossible to let a graphic designer update the UI. In addition, debugging an ASP application is a nightmare.
- Although you can fix script-code shortcomings by using compiled COM components (COM components can access virtually any API, including Windows functions), COM components are all problematic. First of all, you can't replace a COM component with a new version without stopping and restarting the Web site. On

top of that, COM components make the deployment of the Web application more difficult because they must be registered in the system Registry.

- Large-scale code reuse is virtually impossible under ASP. In practice, the only way you can reuse code in ASP is either by using include files or by using compiled COM components.
- Most ASP Web sites store information about individual connected clients in Session variables because ASP programming is stateless, which means that values aren't prevented between consecutive requests to the server. Session variables offer storage for that client's state, but they have a couple of remarkable limitations: they don't work on Web farms (that is, when the site runes on multiple computers), and they don't work if users disabled cookie support in their browsers.

THE CHALLENGES OF THE INTERNET

Even if we don't worry about the problems of programming of ASP we probably agree that Internet programming is still in its infancy and therefore overly difficult. One of the crucial problems is the lack of standardization in how information can be shared over the Internet.

HTML dictates only how information is displayed in browsers but makes it difficult to extract that information for use by a program.

Microsoft and other large software companies – including Sun and IBM – have joined with the W3C committee to produce the Simple Object Access Protocol (SOAP) standard, which is based on XML. Using SOAP, you can query a site and get a return value through TCP/IP and XML. XML has a well-defined syntax and can be parsed unambiguously, unlike HTML. You can use SOAP even without switching to .NET but Microsoft has taken an additional step by formulating the concept of XML Web services.

THE SOLUTIONS ACCORDING TO MICROSOFT .NET

LEVELING WINDOWS PLATFORMS

Microsoft .NET offers an object-oriented view of the Windows operating system and includes hundreds of classes that encapsulate all the most important Windows kernel objects.

. NET security model is independent of the specific version of Windows the application is running on.

. NET components and applications are inherently safer than COM components and "old-style" Windows applications.

.NET AS A BETTER COM

A .NET application can consist of one or more *assemblies*. Assemblies are the unit of versioning and logical deployment. All the files in an assembly have the same version number.

All the information related to an application and the components is stored in configuration files held in the application's main directory and not in the Registry. There are two kinds of .NET components: private and shared. Private components are stored in the application's main directory and aren't visible to other applications. Shared components are visible to all the .NET applications and are typically stored in a central repository named global assembly cache (GAC) located under the C:\winNT\Assembly directory.

A new version of the .NET Framework can be installed simultaneously along with the previous version. .NET versioning is more flexible than COM's.

.NET also solves the problem of deployment. Because .NET applications can use private assemblies, you can install more .NET applications by using the so-called *XCOPY deployment*, that is, simply by copying all the files to a directory on the target system.

THE .NET FRAMEWORK CLASS HIERARCHY

The .NET Framework uses the concept of inheritance. All the objects form a hierarchy with a single root, Object Class, from which all the other classes derive. These classes provide functionality in area, including the user interface, data access, Internet, programming, XML processing, security, and cross-machine communication. This approach encourages code reuse.

The .NET Framework takes a novel approach to mutual object references, which gets rid of the circular reference problem. .NET classes aren't reference counter and that they aren't responsible for their lifetime. All .NET objects inherit from System.Object the ability to be released when the main application doesn't hold a reference to them. This technique works both with direct reference (the application has a variable pointing to the object) and with indirect references (when there are intermediate objects between the application and the object in question).

ALL .NET LANGUAGES ARE BORN EQUAL

.NET moves most of the functionality from the language to the .NET Framework itself. A portion of the .NET Framework named Windows Forms offers classes that can create windows and controls. All V languages can use these classes. Microsoft provides several languages with .NET, including Visual Basic .NET, C#, Managed C++, and Jscript.

Because all the objects belong to the .NET object hierarchy or extend objects in that hierarchy, you can easily manipulate them with any .NET language, which offers a degree of cross-language interoperability.

WEB FORMS, THE SUCCESSOR TO ACTIVE SERVER PAGES

ASP.NET comprises two distinct but tightly related technologies: Web Forms and XML Web services. Web Forms are used for Internet applications with a user interface and are

meant to replace ASP applications. Web Services are for Internet applications without a user interface.

ASP.NET applications are written in full-featured, compiled languages so ASP.NET code run faster. You can use early binding and strongly typed variables and have full access to components and functions in the windows API.ASP.NET offers the gamut of debugging features, including break points sat inside the IDE and the ability to print tracking and profiling information about the pages being browsed.ASP.NET permits and promotes separation between the user interface and the code that makes the application work. According to the concept of *code behind* modules, we can split an ASP.NET page into two distinct files, one containing the HTML code and controls and the other containing the source code.ASP.NET is based on the familiar event-driven programming model. ASP.NET uses compiled code, so we don't need to write components frequently. We can overwrite a component even while an application is using it. This feature is known as *shadow copying*. Under ASP.NET, Session variables can be held on the machine that hosts IIS, on other machines in the same network, or inside a SQL Server database. We can decide to create Session objects that don't rely on client side cookie support.

XML WEB SERVICES, THE INTERNET OF THE FUTURE

Microsoft and other software companies are trying to remedy the situation of no standard way to query millions of sites around the world to get information by introducing XML web services. An *XML Web service* is an application that listens to requests coming to a TCP socket and reacts to the commands each request contains. Both the requests and the result are sent and received using SOAP.

XML Web services aren't based on any proprietary technology. Protocols and technologies used by XML Web services are SOAP, XML, HTTP and TCP/IP.

.NET ARCHITECTURE

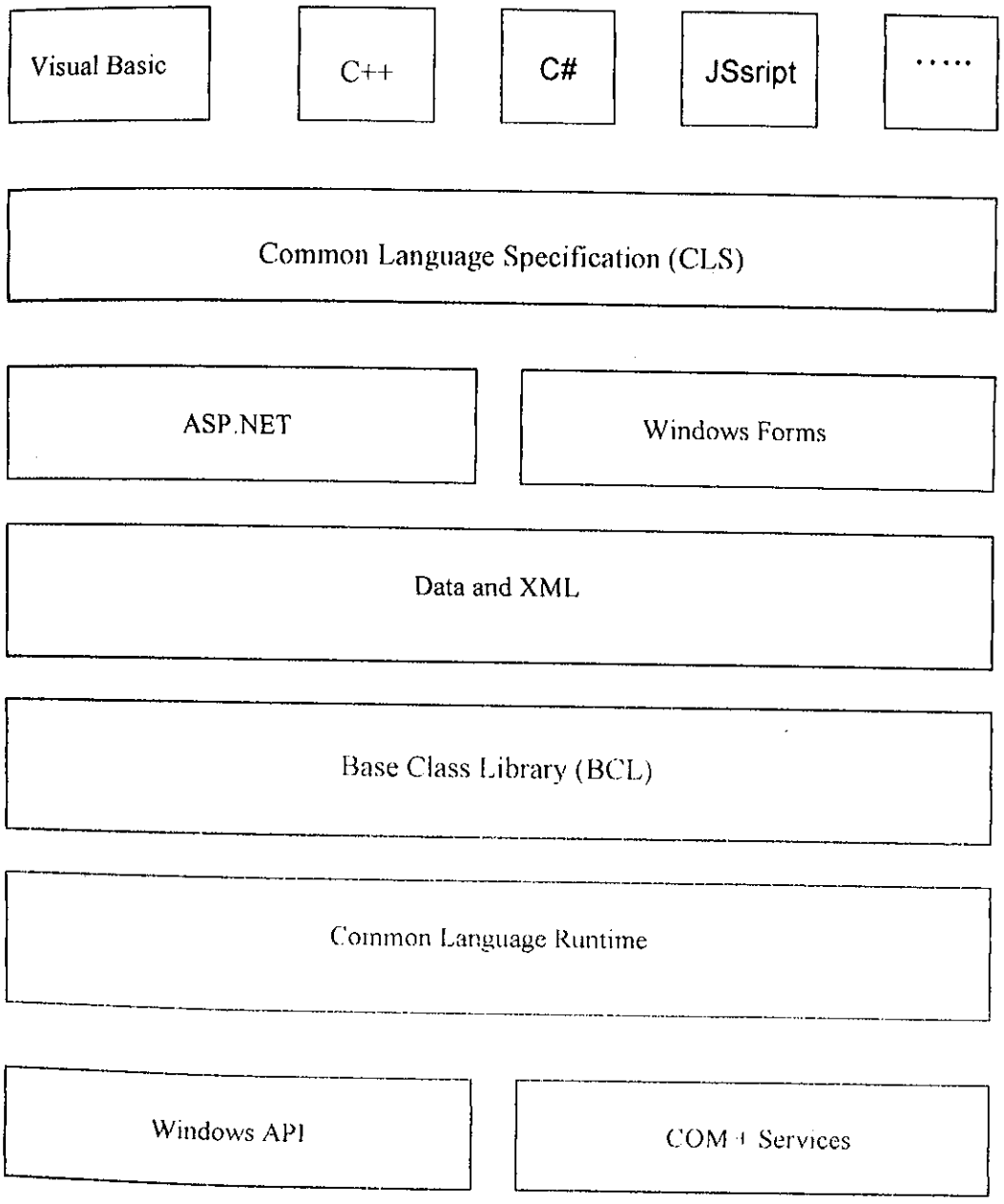


Fig 2 : .NET ARCHITECTURE

INTERFACE FOR UPDATING RESULT DATABASE

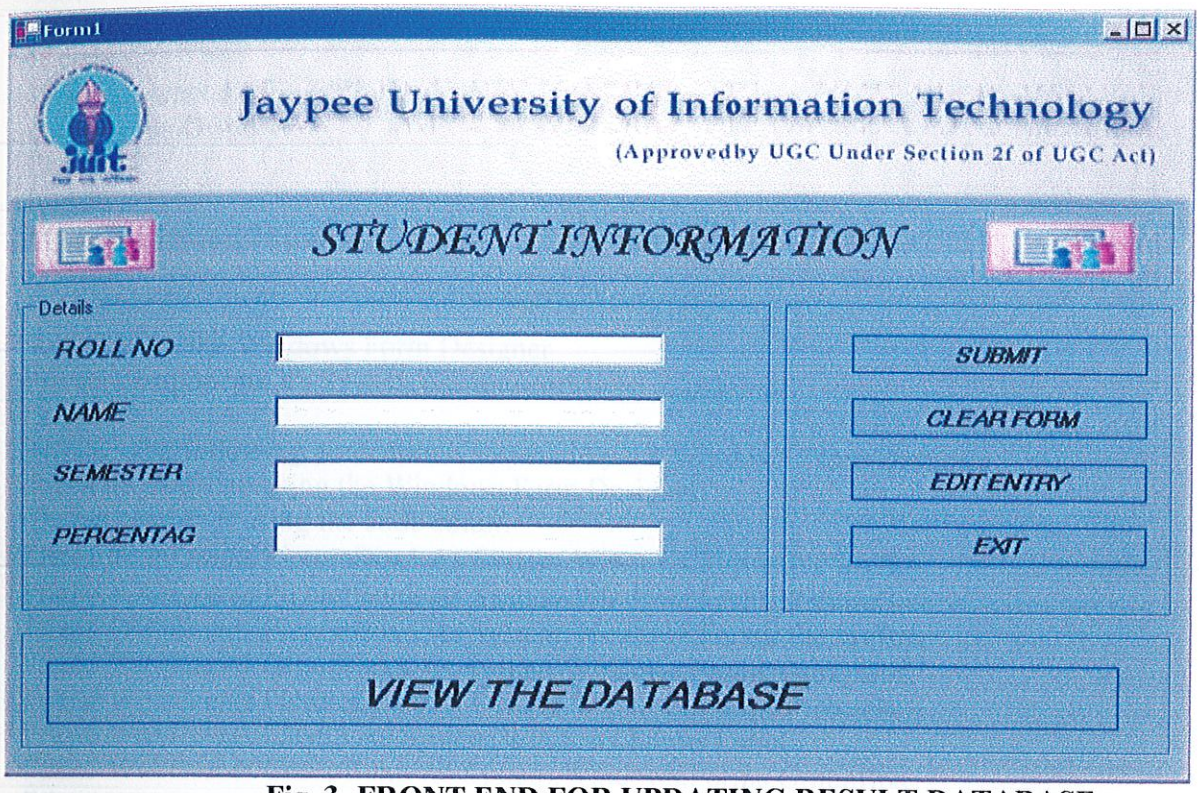


Fig. 3 FRONT END FOR UPDATING RESULT DATABASE

VB. NET CODE FOR UPDATING RESULT DATABASE

```
Imports System.Data.OleDb

Public Class Form1
Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

Public Sub New()
MyBase.New()

'This call is required by the Windows Form Designer.
InitializeComponent()

'Add any initialization after the InitializeComponent() call

End Sub
```

```
'Form overrides dispose to clean up the component list.
Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
If disposing Then
If Not (components Is Nothing) Then
components.Dispose()
End If
End If
MyBase.Dispose(disposing)
End Sub

'Required by the Windows Form Designer
Private components As System.ComponentModel.IContainer

'NOTE: The following procedure is required by the Windows Form Designer
'It can be modified using the Windows Form Designer.
'Do not modify it using the code editor.
Friend WithEvents PictureBox1 As System.Windows.Forms.PictureBox
Friend WithEvents PictureBox2 As System.Windows.Forms.PictureBox
Friend WithEvents PictureBox3 As System.Windows.Forms.PictureBox
Friend WithEvents GroupBox1 As System.Windows.Forms.GroupBox
Friend WithEvents PictureBox4 As System.Windows.Forms.PictureBox
Friend WithEvents Label2 As System.Windows.Forms.Label
Friend WithEvents GroupBox2 As System.Windows.Forms.GroupBox
Friend WithEvents PictureBox5 As System.Windows.Forms.PictureBox
Friend WithEvents GroupBox3 As System.Windows.Forms.GroupBox
Friend WithEvents Label1 As System.Windows.Forms.Label
Friend WithEvents TextBox1 As System.Windows.Forms.TextBox
Friend WithEvents Label3 As System.Windows.Forms.Label
Friend WithEvents TextBox2 As System.Windows.Forms.TextBox
Friend WithEvents Label4 As System.Windows.Forms.Label
Friend WithEvents TextBox3 As System.Windows.Forms.TextBox
Friend WithEvents Label5 As System.Windows.Forms.Label
Friend WithEvents TextBox4 As System.Windows.Forms.TextBox
Friend WithEvents GroupBox4 As System.Windows.Forms.GroupBox
Friend WithEvents Button1 As System.Windows.Forms.Button
Friend WithEvents Button2 As System.Windows.Forms.Button
Friend WithEvents Button3 As System.Windows.Forms.Button
Friend WithEvents Button4 As System.Windows.Forms.Button
Friend WithEvents GroupBox5 As System.Windows.Forms.GroupBox
Friend WithEvents OleDbConnection1 As System.Data.OleDb.OleDbConnection
Friend WithEvents Button5 As System.Windows.Forms.Button
<System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
Dim resources As System.Resources.ResourceManager = New
System.Resources.ResourceManager(GetType(Form1))
Me.PictureBox1 = New System.Windows.Forms.PictureBox
Me.PictureBox2 = New System.Windows.Forms.PictureBox
```

```
Me.PictureBox3 = New System.Windows.Forms.PictureBox
Me.GroupBox1 = New System.Windows.Forms.GroupBox
Me.PictureBox4 = New System.Windows.Forms.PictureBox
Me.Label2 = New System.Windows.Forms.Label
Me.PictureBox5 = New System.Windows.Forms.PictureBox
Me.GroupBox2 = New System.Windows.Forms.GroupBox
Me.GroupBox3 = New System.Windows.Forms.GroupBox
Me.TextBox4 = New System.Windows.Forms.TextBox
Me.Label5 = New System.Windows.Forms.Label
Me.TextBox3 = New System.Windows.Forms.TextBox
Me.Label4 = New System.Windows.Forms.Label
Me.TextBox2 = New System.Windows.Forms.TextBox
Me.Label3 = New System.Windows.Forms.Label
Me.TextBox1 = New System.Windows.Forms.TextBox
Me.Label1 = New System.Windows.Forms.Label
Me.GroupBox4 = New System.Windows.Forms.GroupBox
Me.Button3 = New System.Windows.Forms.Button
Me.Button2 = New System.Windows.Forms.Button
Me.Button1 = New System.Windows.Forms.Button
Me.Button4 = New System.Windows.Forms.Button
Me.GroupBox5 = New System.Windows.Forms.GroupBox
Me.OleDbConnection1 = New System.Data.OleDb.OleDbConnection
Me.Button5 = New System.Windows.Forms.Button
Me.GroupBox1.SuspendLayout()
Me.GroupBox2.SuspendLayout()
Me.GroupBox3.SuspendLayout()
Me.GroupBox4.SuspendLayout()
Me.GroupBox5.SuspendLayout()
Me.SuspendLayout()
'
'PictureBox1
'
Me.PictureBox1.Image = CType(resources.GetObject("PictureBox1.Image"),
System.Drawing.Image)
Me.PictureBox1.Location = New System.Drawing.Point(0, 0)
Me.PictureBox1.Name = "PictureBox1"
Me.PictureBox1.Size = New System.Drawing.Size(100, 104)
Me.PictureBox1.TabIndex = 2
Me.PictureBox1.TabStop = False
'
'PictureBox2
'
Me.PictureBox2.Image = CType(resources.GetObject("PictureBox2.Image"),
System.Drawing.Image)
Me.PictureBox2.Location = New System.Drawing.Point(96, 0)
Me.PictureBox2.Name = "PictureBox2"
```

```
Me.PictureBox2.Size = New System.Drawing.Size(336, 120)
Me.PictureBox2.TabIndex = 3
Me.PictureBox2.TabStop = False
'
'PictureBox3
'
Me.PictureBox3.Image = CType(resources.GetObject("PictureBox3.Image"),
System.Drawing.Image)
Me.PictureBox3.Location = New System.Drawing.Point(432, 0)
Me.PictureBox3.Name = "PictureBox3"
Me.PictureBox3.Size = New System.Drawing.Size(304, 104)
Me.PictureBox3.TabIndex = 4
Me.PictureBox3.TabStop = False
'
'GroupBox1
'
Me.GroupBox1.Controls.Add(Me.PictureBox4)
Me.GroupBox1.Controls.Add(Me.Label2)
Me.GroupBox1.Controls.Add(Me.PictureBox5)
Me.GroupBox1.Location = New System.Drawing.Point(8, 96)
Me.GroupBox1.Name = "GroupBox1"
Me.GroupBox1.Size = New System.Drawing.Size(688, 56)
Me.GroupBox1.TabIndex = 5
Me.GroupBox1.TabStop = False
'
'PictureBox4
'
Me.PictureBox4.Image = CType(resources.GetObject("PictureBox4.Image"),
System.Drawing.Image)
Me.PictureBox4.Location = New System.Drawing.Point(8, 16)
Me.PictureBox4.Name = "PictureBox4"
Me.PictureBox4.Size = New System.Drawing.Size(72, 32)
Me.PictureBox4.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage
Me.PictureBox4.TabIndex = 56
Me.PictureBox4.TabStop = False
'
'Label2
'
Me.Label2.Font = New System.Drawing.Font("Monotype Corsiva", 24.0!,
System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label2.Location = New System.Drawing.Point(168, 8)
Me.Label2.Name = "Label2"
Me.Label2.Size = New System.Drawing.Size(384, 40)
Me.Label2.TabIndex = 0
Me.Label2.Text = "STUDENT INFORMATION "
```

```
'
'PictureBox5
'
Me.PictureBox5.Image = CType(resources.GetObject("PictureBox5.Image"),
System.Drawing.Image)
Me.PictureBox5.Location = New System.Drawing.Point(576, 16)
Me.PictureBox5.Name = "PictureBox5"
Me.PictureBox5.Size = New System.Drawing.Size(88, 32)
Me.PictureBox5.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage
Me.PictureBox5.TabIndex = 57
Me.PictureBox5.TabStop = False
'
'GroupBox2
'
Me.GroupBox2.Controls.Add(Me.PictureBox1)
Me.GroupBox2.Controls.Add(Me.PictureBox2)
Me.GroupBox2.Controls.Add(Me.PictureBox3)
Me.GroupBox2.Location = New System.Drawing.Point(0, 0)
Me.GroupBox2.Name = "GroupBox2"
Me.GroupBox2.Size = New System.Drawing.Size(736, 96)
Me.GroupBox2.TabIndex = 6
Me.GroupBox2.TabStop = False
'
'GroupBox3
'
Me.GroupBox3.Controls.Add(Me.TextBox4)
Me.GroupBox3.Controls.Add(Me.Label5)
Me.GroupBox3.Controls.Add(Me.TextBox3)
Me.GroupBox3.Controls.Add(Me.Label4)
Me.GroupBox3.Controls.Add(Me.TextBox2)
Me.GroupBox3.Controls.Add(Me.Label3)
Me.GroupBox3.Controls.Add(Me.TextBox1)
Me.GroupBox3.Controls.Add(Me.Label1)
Me.GroupBox3.Location = New System.Drawing.Point(8, 160)
Me.GroupBox3.Name = "GroupBox3"
Me.GroupBox3.Size = New System.Drawing.Size(448, 200)
Me.GroupBox3.TabIndex = 7
Me.GroupBox3.TabStop = False
Me.GroupBox3.Text = "Details"
'
'TextBox4
'
Me.TextBox4.BackColor = System.Drawing.SystemColors.Info
Me.TextBox4.Location = New System.Drawing.Point(152, 144)
Me.TextBox4.Name = "TextBox4"
```

```
Me.TextBox4.Size = New System.Drawing.Size(232, 20)
Me.TextBox4.TabIndex = 7
Me.TextBox4.Text = ""
'
'Label5
'
Me.Label5.Font = New System.Drawing.Font("Microsoft Sans Serif", 9.75!,
CType((System.Drawing.FontStyle.Bold Or System.Drawing.FontStyle.Italic),
System.Drawing.FontStyle), System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label5.Location = New System.Drawing.Point(16, 144)
Me.Label5.Name = "Label5"
Me.Label5.Size = New System.Drawing.Size(96, 16)
Me.Label5.TabIndex = 6
Me.Label5.Text = "PERCENTAGE"
'
'TextBox3
'
Me.TextBox3.BackColor = System.Drawing.SystemColors.Info
Me.TextBox3.Location = New System.Drawing.Point(152, 104)
Me.TextBox3.Name = "TextBox3"
Me.TextBox3.Size = New System.Drawing.Size(232, 20)
Me.TextBox3.TabIndex = 5
Me.TextBox3.Text = ""
'
'Label4
'
Me.Label4.Font = New System.Drawing.Font("Microsoft Sans Serif", 9.75!,
CType((System.Drawing.FontStyle.Bold Or System.Drawing.FontStyle.Italic),
System.Drawing.FontStyle), System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label4.Location = New System.Drawing.Point(16, 104)
Me.Label4.Name = "Label4"
Me.Label4.Size = New System.Drawing.Size(96, 23)
Me.Label4.TabIndex = 4
Me.Label4.Text = "SEMESTER"
'
'TextBox2
'
Me.TextBox2.BackColor = System.Drawing.SystemColors.Info
Me.TextBox2.Location = New System.Drawing.Point(152, 64)
Me.TextBox2.Name = "TextBox2"
Me.TextBox2.Size = New System.Drawing.Size(232, 20)
Me.TextBox2.TabIndex = 3
Me.TextBox2.Text = ""
'
'Label3
'
```

```
Me.Label3.Font = New System.Drawing.Font("Microsoft Sans Serif", 11.25!,
CType((System.Drawing.FontStyle.Bold Or System.Drawing.FontStyle.Italic),
System.Drawing.FontStyle), System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label3.Location = New System.Drawing.Point(16, 24)
Me.Label3.Name = "Label3"
Me.Label3.Size = New System.Drawing.Size(96, 24)
Me.Label3.TabIndex = 2
Me.Label3.Text = "ROLL NO"
'
'TextBox1
'
Me.TextBox1.BackColor = System.Drawing.SystemColors.Info
Me.TextBox1.ForeColor = System.Drawing.SystemColors.WindowText
Me.TextBox1.Location = New System.Drawing.Point(152, 24)
Me.TextBox1.Name = "TextBox1"
Me.TextBox1.Size = New System.Drawing.Size(232, 20)
Me.TextBox1.TabIndex = 1
Me.TextBox1.Text = ""
'
'Label1
'
Me.Label1.Font = New System.Drawing.Font("Microsoft Sans Serif", 11.25!,
CType((System.Drawing.FontStyle.Bold Or System.Drawing.FontStyle.Italic),
System.Drawing.FontStyle), System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label1.Location = New System.Drawing.Point(16, 64)
Me.Label1.Name = "Label1"
Me.Label1.Size = New System.Drawing.Size(96, 23)
Me.Label1.TabIndex = 0
Me.Label1.Text = "NAME"
'
'GroupBox4
'
Me.GroupBox4.Controls.Add(Me.Button5)
Me.GroupBox4.Controls.Add(Me.Button3)
Me.GroupBox4.Controls.Add(Me.Button2)
Me.GroupBox4.Controls.Add(Me.Button1)
Me.GroupBox4.Location = New System.Drawing.Point(464, 160)
Me.GroupBox4.Name = "GroupBox4"
Me.GroupBox4.Size = New System.Drawing.Size(232, 200)
Me.GroupBox4.TabIndex = 8
Me.GroupBox4.TabStop = False
'
'Button3
'
Me.Button3.FlatStyle = System.Windows.Forms.FlatStyle.Popup
```

```
Me.Button3.Font = New System.Drawing.Font("Microsoft Sans Serif", 9.75!,
CType((System.Drawing.FontStyle.Bold Or System.Drawing.FontStyle.Italic),
System.Drawing.FontStyle), System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button3.Location = New System.Drawing.Point(40, 144)
Me.Button3.Name = "Button3"
Me.Button3.Size = New System.Drawing.Size(176, 24)
Me.Button3.TabIndex = 2
Me.Button3.Text = "EXIT"
'
'Button2
'
Me.Button2.FlatStyle = System.Windows.Forms.FlatStyle.Popup
Me.Button2.Font = New System.Drawing.Font("Microsoft Sans Serif", 9.75!,
CType((System.Drawing.FontStyle.Bold Or System.Drawing.FontStyle.Italic),
System.Drawing.FontStyle), System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button2.Location = New System.Drawing.Point(40, 64)
Me.Button2.Name = "Button2"
Me.Button2.Size = New System.Drawing.Size(176, 24)
Me.Button2.TabIndex = 1
Me.Button2.Text = "CLEAR FORM"
'
'Button1
'
Me.Button1.FlatStyle = System.Windows.Forms.FlatStyle.Popup
Me.Button1.Font = New System.Drawing.Font("Microsoft Sans Serif", 9.75!,
CType((System.Drawing.FontStyle.Bold Or System.Drawing.FontStyle.Italic),
System.Drawing.FontStyle), System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button1.Location = New System.Drawing.Point(40, 24)
Me.Button1.Name = "Button1"
Me.Button1.Size = New System.Drawing.Size(176, 24)
Me.Button1.TabIndex = 0
Me.Button1.Text = "SUBMIT"
'
'Button4
'
Me.Button4.FlatStyle = System.Windows.Forms.FlatStyle.Popup
Me.Button4.Font = New System.Drawing.Font("Microsoft Sans Serif", 18.0!,
CType((System.Drawing.FontStyle.Bold Or System.Drawing.FontStyle.Italic),
System.Drawing.FontStyle), System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button4.Location = New System.Drawing.Point(16, 24)
Me.Button4.Name = "Button4"
Me.Button4.Size = New System.Drawing.Size(640, 40)
Me.Button4.TabIndex = 9
Me.Button4.Text = "VIEW THE DATABASE"
'
'GroupBox5
```

```
'
Me.GroupBox5.Controls.Add(Me.Button4)
Me.GroupBox5.Location = New System.Drawing.Point(8, 368)
Me.GroupBox5.Name = "GroupBox5"
Me.GroupBox5.Size = New System.Drawing.Size(688, 80)
Me.GroupBox5.TabIndex = 10
Me.GroupBox5.TabStop = False
'
'OleDbConnection1
'
Me.OleDbConnection1.ConnectionString = "Jet OLEDB:Global Partial Bulk Ops=2;Jet
OLEDB:Registry Path=;Jet OLEDB:Database L" & "ocking Mode=1;Jet
OLEDB:Database Password=;Data Source=""C:\Documents and Setting" & "s\Rajat
Gulati\My Documents\Visual Studio Projects\STUDENT DATA BASE\student.mdb" &
_ """;Password=;Jet OLEDB:Engine Type=5;Jet OLEDB:Global Bulk
Transactions=1;Provide" & "r=""Microsoft.Jet.OLEDB.4.0"";Jet OLEDB:System
database=;Jet OLEDB:SFP=False;Exten" & "ded Properties=;Mode=Share Deny
None;Jet OLEDB:New Database Password=;Jet OLEDB:" & "Create System
Database=False;Jet OLEDB:Don't Copy Locale on Compact=False;Jet OL" &
"EDB:Compact Without Replica Repair=False;User ID=Admin;Jet OLEDB:Encrypt
Databas" & "e=False"
'
'Button5
'
Me.Button5.FlatStyle = System.Windows.Forms.FlatStyle.Popup
Me.Button5.Font = New System.Drawing.Font("Microsoft Sans Serif", 9.75!,
CType((System.Drawing.FontStyle.Bold Or System.Drawing.FontStyle.Italic),
System.Drawing.FontStyle), System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button5.Location = New System.Drawing.Point(40, 104)
Me.Button5.Name = "Button5"
Me.Button5.Size = New System.Drawing.Size(176, 24)
Me.Button5.TabIndex = 3
Me.Button5.Text = "EDIT ENTRY"
'
'Form1
'
Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
Me.BackColor = System.Drawing.Color.LightSteelBlue
Me.ClientSize = New System.Drawing.Size(704, 461)
Me.Controls.Add(Me.GroupBox5)
Me.Controls.Add(Me.GroupBox4)
Me.Controls.Add(Me.GroupBox3)
Me.Controls.Add(Me.GroupBox1)
Me.Controls.Add(Me.GroupBox2)
Me.Name = "Form1"
Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen
```

```
Me.Text = "Form1"
Me.GroupBox1.ResumeLayout(False)
Me.GroupBox2.ResumeLayout(False)
Me.GroupBox3.ResumeLayout(False)
Me.GroupBox4.ResumeLayout(False)
Me.GroupBox5.ResumeLayout(False)
Me.ResumeLayout(False)

End Sub

#End Region
Dim fm As New Form2

Private Sub Button3_Click (ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
End
End Sub


Private Sub Button2_Click (ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click

TextBox1.Text = ""
TextBox2.Text = ""
TextBox3.Text = ""
TextBox4.Text = ""

End Sub

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
Try
OleDbConnection1.Open()
Dim comm As New OleDb.OleDbCommand
comm = OleDbConnection1.CreateCommand

comm.CommandText = "insert into stu values(" & TextBox1.Text & ",'" &
TextBox2.Text & "'," & TextBox3.Text & "," & TextBox4.Text & ")"

comm.Connection = OleDbConnection1
MessageBox.Show(comm.CommandText)
comm.ExecuteNonQuery()
MsgBox("SAVED SUCCESSFULLY...!", MsgBoxStyle.Information)
OleDbConnection1.Close()
Catch ex As Exception
MessageBox.Show(ex.Message)
```

```
End Try
Button2.PerformClick()
End Sub

Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button4.Click
Me.Hide()
fm.Show()
End Sub

Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button5.Click
Dim fm3 As New Form3
Me.Hide()
fm3.Show()
End Sub
End Class
```

INTERFACE FOR VIEWING RESULT DATABASE

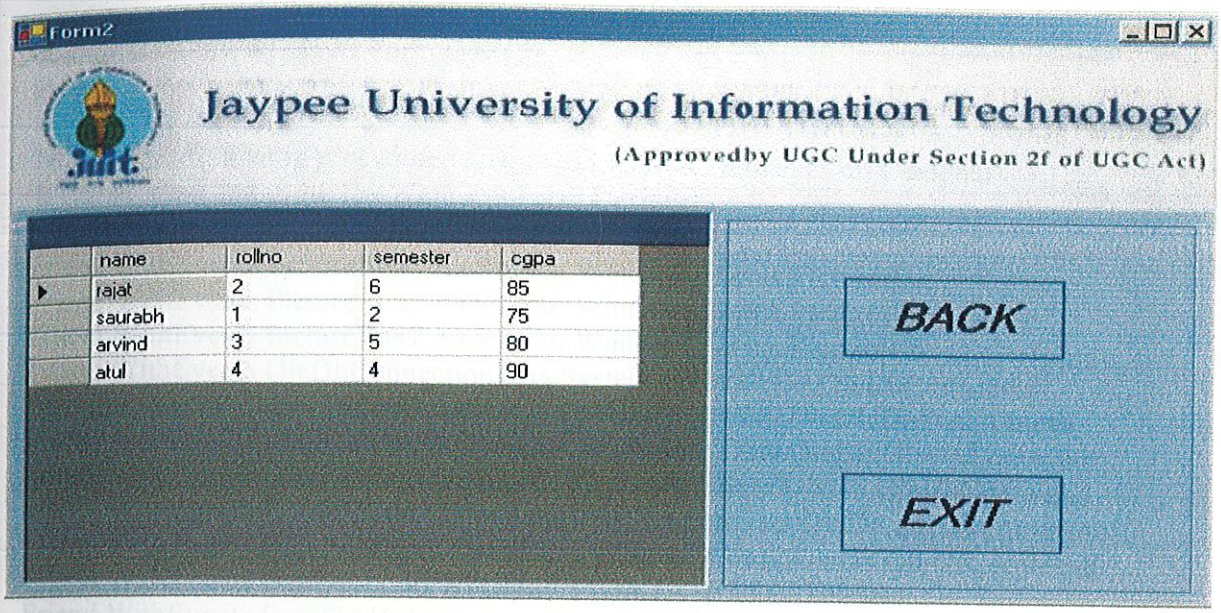


Fig. 4: FRONT END FOR VIEWING THE STUDENT DATABASE

VB.NET CODE FOR VIEWING RESULT DATABASE

```
Imports System.Data.OleDb

Public Class Form2
Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

Public Sub New()
MyBase.New()

'This call is required by the Windows Form Designer.
InitializeComponent()

'Add any initialization after the InitializeComponent() call

End Sub

'Form overrides dispose to clean up the component list.
Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
If disposing Then
If Not (components Is Nothing) Then
components.Dispose()
End If
End If
MyBase.Dispose(disposing)
```

```
End Sub

'Required by the Windows Form Designer
Private components As System.ComponentModel.IContainer

'NOTE: The following procedure is required by the Windows Form Designer
'It can be modified using the Windows Form Designer.
'Do not modify it using the code editor.
Friend WithEvents PictureBox1 As System.Windows.Forms.PictureBox
Friend WithEvents PictureBox3 As System.Windows.Forms.PictureBox
Friend WithEvents PictureBox2 As System.Windows.Forms.PictureBox
Friend WithEvents OleDbConnection1 As System.Data.OleDb.OleDbConnection
Friend WithEvents OleDbDataAdapter1 As System.Data.OleDb.OleDbDataAdapter
Friend WithEvents OleDbSelectCommand1 As System.Data.OleDb.OleDbCommand
Friend WithEvents OleDbInsertCommand1 As System.Data.OleDb.OleDbCommand
Friend WithEvents OleDbUpdateCommand1 As System.Data.OleDb.OleDbCommand
Friend WithEvents OleDbDeleteCommand1 As System.Data.OleDb.OleDbCommand
Friend WithEvents DataSet11 As STUDENT_DATA_BASE.DataSet1
Friend WithEvents DataGrid1 As System.Windows.Forms.DataGrid
Friend WithEvents GroupBox1 As System.Windows.Forms.GroupBox
Friend WithEvents Button1 As System.Windows.Forms.Button
Friend WithEvents Button2 As System.Windows.Forms.Button
<System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
Dim resources As System.Resources.ResourceManager = New
System.Resources.ResourceManager(GetType(Form2))
Me.PictureBox1 = New System.Windows.Forms.PictureBox
Me.PictureBox3 = New System.Windows.Forms.PictureBox
Me.PictureBox2 = New System.Windows.Forms.PictureBox
Me.OleDbConnection1 = New System.Data.OleDb.OleDbConnection
Me.OleDbDataAdapter1 = New System.Data.OleDb.OleDbDataAdapter
Me.OleDbDeleteCommand1 = New System.Data.OleDb.OleDbCommand
Me.OleDbInsertCommand1 = New System.Data.OleDb.OleDbCommand
Me.OleDbSelectCommand1 = New System.Data.OleDb.OleDbCommand
Me.OleDbUpdateCommand1 = New System.Data.OleDb.OleDbCommand
Me.DataSet11 = New STUDENT_DATA_BASE.DataSet1
Me.DataGrid1 = New System.Windows.Forms.DataGrid
Me.GroupBox1 = New System.Windows.Forms.GroupBox
Me.Button2 = New System.Windows.Forms.Button
Me.Button1 = New System.Windows.Forms.Button
CType(Me.DataSet11, System.ComponentModel.ISupportInitialize).BeginInit()
CType(Me.DataGrid1, System.ComponentModel.ISupportInitialize).BeginInit()
Me.GroupBox1.SuspendLayout()
Me.SuspendLayout()
'
'PictureBox1
'
```

```
Me.PictureBox1.Image = CType(resources.GetObject("PictureBox1.Image"),
System.Drawing.Image)
Me.PictureBox1.Location = New System.Drawing.Point(0, 0)
Me.PictureBox1.Name = "PictureBox1"
Me.PictureBox1.Size = New System.Drawing.Size(100, 104)
Me.PictureBox1.TabIndex = 5
Me.PictureBox1.TabStop = False
'
'PictureBox3
'
Me.PictureBox3.Image = CType(resources.GetObject("PictureBox3.Image"),
System.Drawing.Image)
Me.PictureBox3.Location = New System.Drawing.Point(400, 0)
Me.PictureBox3.Name = "PictureBox3"
Me.PictureBox3.Size = New System.Drawing.Size(304, 104)
Me.PictureBox3.TabIndex = 7
Me.PictureBox3.TabStop = False
'
'PictureBox2
'
Me.PictureBox2.Image = CType(resources.GetObject("PictureBox2.Image"),
System.Drawing.Image)
Me.PictureBox2.Location = New System.Drawing.Point(64, 0)
Me.PictureBox2.Name = "PictureBox2"
Me.PictureBox2.Size = New System.Drawing.Size(336, 104)
Me.PictureBox2.TabIndex = 6
Me.PictureBox2.TabStop = False
'
'OleDbConnection1
'
Me.OleDbConnection1.ConnectionString = "Jet OLEDB:Global Partial Bulk Ops=2;Jet
OLEDB:Registry Path=;Jet OLEDB:Database L" & _
"ocking Mode=1;Jet OLEDB:Database Password=;Data Source=""G:\documents\Visual
Stud" & _
"io Projects\STUDENT DATA BASE\student.mdb"";Password=;Jet OLEDB:Engine
Type=5;Jet" & _
" OLEDB:Global Bulk Transactions=1;Provider=""Microsoft.Jet.OLEDB.4.0"";Jet
OLEDB:S" & _
"ystem database=;Jet OLEDB:SFP=False;Extended Properties=;Mode=Share Deny
None;Je" & _
"t OLEDB:New Database Password=;Jet OLEDB:Create System Database=False;Jet
OLEDB:" & _
"Don't Copy Locale on Compact=False;Jet OLEDB:Compact Without Replica
Repair=Fals" & _
"e;User ID=Admin;Jet OLEDB:Encrypt Database=False"
'
```

```
'OleDbDataAdapter1
'
Me.OleDbDataAdapter1.DeleteCommand = Me.OleDbDeleteCommand1
Me.OleDbDataAdapter1.InsertCommand = Me.OleDbInsertCommand1
Me.OleDbDataAdapter1.SelectCommand = Me.OleDbSelectCommand1
Me.OleDbDataAdapter1.TableMappings.AddRange(New
System.Data.Common.DataTableMapping() {New
System.Data.Common.DataTableMapping("Table", "stu", New
System.Data.Common.DataColumnMapping() {New
System.Data.Common.DataColumnMapping("name", "name"), New
System.Data.Common.DataColumnMapping("rollno", "rollno"), New
System.Data.Common.DataColumnMapping("semester", "semester"), New
System.Data.Common.DataColumnMapping("cgpa", "cgpa")})})
Me.OleDbDataAdapter1.UpdateCommand = Me.OleDbUpdateCommand1
'
'OleDbDeleteCommand1
'
Me.OleDbDeleteCommand1.CommandText = "DELETE FROM stu WHERE (name = ?)
AND (rollno = ?) AND (semester = ?) AND (cgpa = " & _
"? OR ? IS NULL AND cgpa IS NULL)"
Me.OleDbDeleteCommand1.Connection = Me.OleDbConnection1
Me.OleDbDeleteCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Original_name",
System.Data.OleDb.OleDbType.VarWChar, 50, System.Data.ParameterDirection.Input,
False, CType(0, Byte), CType(0, Byte), "name", System.Data.DataRowVersion.Original,
Nothing))
Me.OleDbDeleteCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Original_rollno",
System.Data.OleDb.OleDbType.Integer, 0, System.Data.ParameterDirection.Input, False,
CType(0, Byte), CType(0, Byte), "rollno", System.Data.DataRowVersion.Original,
Nothing))
Me.OleDbDeleteCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Original_semester",
System.Data.OleDb.OleDbType.Integer, 0, System.Data.ParameterDirection.Input, False,
CType(0, Byte), CType(0, Byte), "semester", System.Data.DataRowVersion.Original,
Nothing))
Me.OleDbDeleteCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Original_cgpa",
System.Data.OleDb.OleDbType.Double, 0, System.Data.ParameterDirection.Input,
False, CType(0, Byte), CType(0, Byte), "cgpa", System.Data.DataRowVersion.Original,
Nothing))
Me.OleDbDeleteCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Original_cgpa1",
System.Data.OleDb.OleDbType.Double, 0, System.Data.ParameterDirection.Input,
False, CType(0, Byte), CType(0, Byte), "cgpa", System.Data.DataRowVersion.Original,
Nothing))
```

```
'
'OleDbInsertCommand1
'
Me.OleDbInsertCommand1.CommandText = "INSERT INTO stu(name, rollno,
semester, cgpa) VALUES (?, ?, ?, ?)"
Me.OleDbInsertCommand1.Connection = Me.OleDbConnection1
Me.OleDbInsertCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("name",
System.Data.OleDb.OleDbType.VarWChar, 50, "name"))
Me.OleDbInsertCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("rollno", System.Data.OleDb.OleDbType.Integer,
0, "rollno"))
Me.OleDbInsertCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("semester",
System.Data.OleDb.OleDbType.Integer, 0, "semester"))
Me.OleDbInsertCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("cgpa", System.Data.OleDb.OleDbType.Double, 0,
"cgpa"))
'
'OleDbSelectCommand1
'
Me.OleDbSelectCommand1.CommandText = "SELECT name, rollno, semester, cgpa
FROM stu"
Me.OleDbSelectCommand1.Connection = Me.OleDbConnection1
'
'OleDbUpdateCommand1
'
Me.OleDbUpdateCommand1.CommandText = "UPDATE stu SET name = ?, rollno = ?,
semester = ?, cgpa = ? WHERE (name = ?) AND " & _
"(rollno = ?) AND (semester = ?) AND (cgpa = ? OR ? IS NULL AND cgpa IS NULL)"
Me.OleDbUpdateCommand1.Connection = Me.OleDbConnection1
Me.OleDbUpdateCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("name",
System.Data.OleDb.OleDbType.VarWChar, 50, "name"))
Me.OleDbUpdateCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("rollno", System.Data.OleDb.OleDbType.Integer,
0, "rollno"))
Me.OleDbUpdateCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("semester",
System.Data.OleDb.OleDbType.Integer, 0, "semester"))
Me.OleDbUpdateCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("cgpa", System.Data.OleDb.OleDbType.Double, 0,
"cgpa"))
Me.OleDbUpdateCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Original_name",
System.Data.OleDb.OleDbType.VarWChar, 50, System.Data.ParameterDirection.Input,
```

```
False, CType(0, Byte), CType(0, Byte), "name", System.Data.DataRowVersion.Original,
Nothing))
Me.OleDbUpdateCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Original_rollno",
System.Data.OleDb.OleDbType.Integer, 0, System.Data.ParameterDirection.Input, False,
CType(0, Byte), CType(0, Byte), "rollno", System.Data.DataRowVersion.Original,
Nothing))
Me.OleDbUpdateCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Original_semester",
System.Data.OleDb.OleDbType.Integer, 0, System.Data.ParameterDirection.Input, False,
CType(0, Byte), CType(0, Byte), "semester", System.Data.DataRowVersion.Original,
Nothing))
Me.OleDbUpdateCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Original_cgpa",
System.Data.OleDb.OleDbType.Double, 0, System.Data.ParameterDirection.Input,
False, CType(0, Byte), CType(0, Byte), "cgpa", System.Data.DataRowVersion.Original,
Nothing))
Me.OleDbUpdateCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Original_cgpa1",
System.Data.OleDb.OleDbType.Double, 0, System.Data.ParameterDirection.Input,
False, CType(0, Byte), CType(0, Byte), "cgpa", System.Data.DataRowVersion.Original,
Nothing))
'
'DataSet11
'
Me.DataSet11.DataSetName = "DataSet1"
Me.DataSet11.Locale = New System.Globalization.CultureInfo("en-US")
'
'DataGrid1
'
Me.DataGrid1.DataMember = "stu"
Me.DataGrid1.DataSource = Me.DataSet11
Me.DataGrid1.HeaderForeColor = System.Drawing.SystemColors.ControlText
Me.DataGrid1.Location = New System.Drawing.Point(8, 104)
Me.DataGrid1.Name = "DataGrid1"
Me.DataGrid1.ReadOnly = True
Me.DataGrid1.Size = New System.Drawing.Size(376, 232)
Me.DataGrid1.TabIndex = 8
'
'GroupBox1
'
Me.GroupBox1.Controls.Add(Me.Button2)
Me.GroupBox1.Controls.Add(Me.Button1)
Me.GroupBox1.Location = New System.Drawing.Point(392, 104)
Me.GroupBox1.Name = "GroupBox1"
Me.GroupBox1.Size = New System.Drawing.Size(256, 232)
```

```
Me.GroupBox1.TabIndex = 9
Me.GroupBox1.TabStop = False
'
'Button2
'
Me.Button2.FlatStyle = System.Windows.Forms.FlatStyle.Popup
Me.Button2.Font = New System.Drawing.Font("Microsoft Sans Serif", 18.0!,
CType((System.Drawing.FontStyle.Bold Or System.Drawing.FontStyle.Italic),
System.Drawing.FontStyle), System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button2.Location = New System.Drawing.Point(64, 160)
Me.Button2.Name = "Button2"
Me.Button2.Size = New System.Drawing.Size(120, 48)
Me.Button2.TabIndex = 1
Me.Button2.Text = "EXIT"
'
'Button1
'
Me.Button1.FlatStyle = System.Windows.Forms.FlatStyle.Popup
Me.Button1.Font = New System.Drawing.Font("Microsoft Sans Serif", 18.0!,
CType((System.Drawing.FontStyle.Bold Or System.Drawing.FontStyle.Italic),
System.Drawing.FontStyle), System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button1.Location = New System.Drawing.Point(64, 40)
Me.Button1.Name = "Button1"
Me.Button1.Size = New System.Drawing.Size(120, 48)
Me.Button1.TabIndex = 0
Me.Button1.Text = "BACK"
'
'Form2
'
Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
Me.BackColor = System.Drawing.Color.LightSteelBlue
Me.ClientSize = New System.Drawing.Size(656, 341)
Me.Controls.Add(Me.GroupBox1)
Me.Controls.Add(Me.DataGrid1)
Me.Controls.Add(Me.PictureBox1)
Me.Controls.Add(Me.PictureBox2)
Me.Controls.Add(Me.PictureBox3)
Me.Name = "Form2"
Me.Text = "Form2"
CType(Me.DataSet11, System.ComponentModel.ISupportInitialize).EndInit()
CType(Me.DataGrid1, System.ComponentModel.ISupportInitialize).EndInit()
Me.GroupBox1.ResumeLayout(False)
Me.ResumeLayout(False)

End Sub
```

```
#End Region

Private Sub Form2_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

OleDbDataAdapter1.Fill(DataSet11)
End Sub

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
Dim fm As New Form1
Me.Hide()
fm.Show()
End Sub

Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
End
End Sub
End Class
```

INTERFACE FOR EDITING THE DATABASE

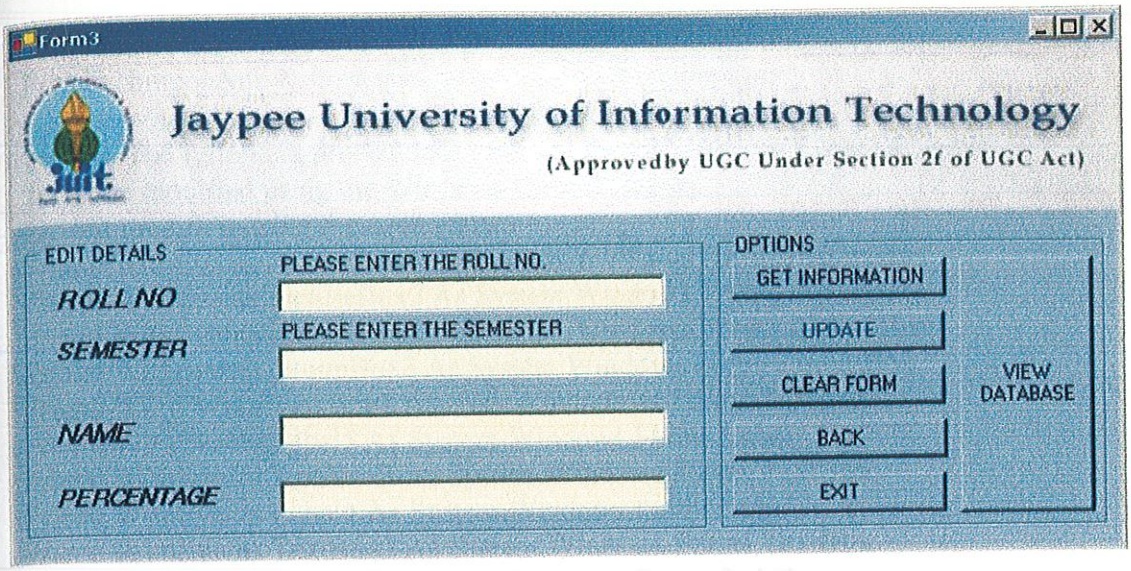


Fig. 5: FRONT END FOR EDITING THE STUDENT DATABASE

VB.NET CODE FOR VIEWING RESULT DATABASE

```
Imports System.Data.OleDb
Public Class Form3
Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

Public Sub New()
MyBase.New()

'This call is required by the Windows Form Designer.
InitializeComponent()

'Add any initialization after the InitializeComponent() call

End Sub

'Form overrides dispose to clean up the component list.
Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
If disposing Then
If Not (components Is Nothing) Then
components.Dispose()
End If
End If
MyBase.Dispose(disposing)
```

```
End Sub

'Required by the Windows Form Designer
Private components As System.ComponentModel.IContainer

'NOTE: The following procedure is required by the Windows Form Designer
'It can be modified using the Windows Form Designer.
'Do not modify it using the code editor.
Friend WithEvents PictureBox1 As System.Windows.Forms.PictureBox
Friend WithEvents PictureBox2 As System.Windows.Forms.PictureBox
Friend WithEvents PictureBox3 As System.Windows.Forms.PictureBox
Friend WithEvents GroupBox3 As System.Windows.Forms.GroupBox
Friend WithEvents TextBox4 As System.Windows.Forms.TextBox
Friend WithEvents Label5 As System.Windows.Forms.Label
Friend WithEvents TextBox3 As System.Windows.Forms.TextBox
Friend WithEvents Label4 As System.Windows.Forms.Label
Friend WithEvents TextBox2 As System.Windows.Forms.TextBox
Friend WithEvents Label3 As System.Windows.Forms.Label
Friend WithEvents TextBox1 As System.Windows.Forms.TextBox
Friend WithEvents Label1 As System.Windows.Forms.Label
Friend WithEvents Button1 As System.Windows.Forms.Button
Friend WithEvents Button2 As System.Windows.Forms.Button
Friend WithEvents Button3 As System.Windows.Forms.Button
Friend WithEvents GroupBox1 As System.Windows.Forms.GroupBox
Friend WithEvents Button4 As System.Windows.Forms.Button
Friend WithEvents Label2 As System.Windows.Forms.Label
Friend WithEvents Label6 As System.Windows.Forms.Label
Friend WithEvents Button5 As System.Windows.Forms.Button
Friend WithEvents OleDbConnection1 As System.Data.OleDb.OleDbConnection
Friend WithEvents OleDbCommand1 As System.Data.OleDb.OleDbCommand
Friend WithEvents Button6 As System.Windows.Forms.Button
<System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
Dim resources As System.Resources.ResourceManager = New
System.Resources.ResourceManager(GetType(Form3))
Me.PictureBox1 = New System.Windows.Forms.PictureBox
Me.PictureBox2 = New System.Windows.Forms.PictureBox
Me.PictureBox3 = New System.Windows.Forms.PictureBox
Me.GroupBox3 = New System.Windows.Forms.GroupBox
Me.TextBox4 = New System.Windows.Forms.TextBox
Me.Label5 = New System.Windows.Forms.Label
Me.TextBox3 = New System.Windows.Forms.TextBox
Me.Label4 = New System.Windows.Forms.Label
Me.TextBox2 = New System.Windows.Forms.TextBox
Me.Label3 = New System.Windows.Forms.Label
Me.TextBox1 = New System.Windows.Forms.TextBox
Me.Label1 = New System.Windows.Forms.Label
```

```
Me.Button1 = New System.Windows.Forms.Button
Me.Button2 = New System.Windows.Forms.Button
Me.Button3 = New System.Windows.Forms.Button
Me.GroupBox1 = New System.Windows.Forms.GroupBox
Me.Button4 = New System.Windows.Forms.Button
Me.Label2 = New System.Windows.Forms.Label
Me.Label6 = New System.Windows.Forms.Label
Me.Button5 = New System.Windows.Forms.Button
Me.OleDbConnection1 = New System.Data.OleDb.OleDbConnection
Me.OleDbCommand1 = New System.Data.OleDb.OleDbCommand
Me.Button6 = New System.Windows.Forms.Button
Me.GroupBox3.SuspendLayout()
Me.GroupBox1.SuspendLayout()
Me.SuspendLayout()
'
'PictureBox1
'
Me.PictureBox1.Image = CType(resources.GetObject("PictureBox1.Image"),
System.Drawing.Image)
Me.PictureBox1.Location = New System.Drawing.Point(-8, 0)
Me.PictureBox1.Name = "PictureBox1"
Me.PictureBox1.Size = New System.Drawing.Size(100, 104)
Me.PictureBox1.TabIndex = 8
Me.PictureBox1.TabStop = False
'
'PictureBox2
'
Me.PictureBox2.Image = CType(resources.GetObject("PictureBox2.Image"),
System.Drawing.Image)
Me.PictureBox2.Location = New System.Drawing.Point(56, 0)
Me.PictureBox2.Name = "PictureBox2"
Me.PictureBox2.Size = New System.Drawing.Size(336, 104)
Me.PictureBox2.TabIndex = 9
Me.PictureBox2.TabStop = False
'
'PictureBox3
'
Me.PictureBox3.Image = CType(resources.GetObject("PictureBox3.Image"),
System.Drawing.Image)
Me.PictureBox3.Location = New System.Drawing.Point(392, 0)
Me.PictureBox3.Name = "PictureBox3"
Me.PictureBox3.Size = New System.Drawing.Size(304, 104)
Me.PictureBox3.TabIndex = 10
Me.PictureBox3.TabStop = False
'
'GroupBox3
```

```
'
Me.GroupBox3.Controls.Add(Me.Label6)
Me.GroupBox3.Controls.Add(Me.Label2)
Me.GroupBox3.Controls.Add(Me.TextBox4)
Me.GroupBox3.Controls.Add(Me.Label5)
Me.GroupBox3.Controls.Add(Me.TextBox3)
Me.GroupBox3.Controls.Add(Me.Label4)
Me.GroupBox3.Controls.Add(Me.TextBox2)
Me.GroupBox3.Controls.Add(Me.Label3)
Me.GroupBox3.Controls.Add(Me.TextBox1)
Me.GroupBox3.Controls.Add(Me.Label1)
Me.GroupBox3.Location = New System.Drawing.Point(8, 112)
Me.GroupBox3.Name = "GroupBox3"
Me.GroupBox3.Size = New System.Drawing.Size(408, 176)
Me.GroupBox3.TabIndex = 11
Me.GroupBox3.TabStop = False
Me.GroupBox3.Text = " EDIT DETAILS"
'
'TextBox4
'
Me.TextBox4.BackColor = System.Drawing.SystemColors.Info
Me.TextBox4.Location = New System.Drawing.Point(152, 144)
Me.TextBox4.Name = "TextBox4"
Me.TextBox4.Size = New System.Drawing.Size(232, 20)
Me.TextBox4.TabIndex = 7
Me.TextBox4.Text = ""
'
'Label5
'
Me.Label5.Font = New System.Drawing.Font("Microsoft Sans Serif", 9.75!,
CType((System.Drawing.FontStyle.Bold Or System.Drawing.FontStyle.Italic),
System.Drawing.FontStyle), System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label5.Location = New System.Drawing.Point(16, 144)
Me.Label5.Name = "Label5"
Me.Label5.Size = New System.Drawing.Size(104, 16)
Me.Label5.TabIndex = 6
Me.Label5.Text = "PERCENTAGE"
'
'TextBox3
'
Me.TextBox3.BackColor = System.Drawing.SystemColors.Info
Me.TextBox3.Location = New System.Drawing.Point(152, 64)
Me.TextBox3.Name = "TextBox3"
Me.TextBox3.Size = New System.Drawing.Size(232, 20)
Me.TextBox3.TabIndex = 5
Me.TextBox3.Text = ""
```

```
'
'Label4
'
Me.Label4.Font = New System.Drawing.Font("Microsoft Sans Serif", 9.75!,
CType((System.Drawing.FontStyle.Bold Or System.Drawing.FontStyle.Italic),
System.Drawing.FontStyle), System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label4.Location = New System.Drawing.Point(16, 56)
Me.Label4.Name = "Label4"
Me.Label4.Size = New System.Drawing.Size(96, 23)
Me.Label4.TabIndex = 4
Me.Label4.Text = "SEMESTER"
'
'TextBox2
'
Me.TextBox2.BackColor = System.Drawing.SystemColors.Info
Me.TextBox2.Location = New System.Drawing.Point(152, 104)
Me.TextBox2.Name = "TextBox2"
Me.TextBox2.Size = New System.Drawing.Size(232, 20)
Me.TextBox2.TabIndex = 3
Me.TextBox2.Text = ""
'
'Label3
'
Me.Label3.Font = New System.Drawing.Font("Microsoft Sans Serif", 11.25!,
CType((System.Drawing.FontStyle.Bold Or System.Drawing.FontStyle.Italic),
System.Drawing.FontStyle), System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label3.Location = New System.Drawing.Point(16, 24)
Me.Label3.Name = "Label3"
Me.Label3.Size = New System.Drawing.Size(96, 24)
Me.Label3.TabIndex = 2
Me.Label3.Text = "ROLL NO"
'
'TextBox1
'
Me.TextBox1.BackColor = System.Drawing.SystemColors.Info
Me.TextBox1.ForeColor = System.Drawing.SystemColors.WindowText
Me.TextBox1.Location = New System.Drawing.Point(152, 24)
Me.TextBox1.Name = "TextBox1"
Me.TextBox1.Size = New System.Drawing.Size(232, 20)
Me.TextBox1.TabIndex = 1
Me.TextBox1.Text = ""
'
'Label1
'
```

```
Me.Label1.Font = New System.Drawing.Font("Microsoft Sans Serif", 11.25!,
CType((System.Drawing.FontStyle.Bold Or System.Drawing.FontStyle.Italic),
System.Drawing.FontStyle), System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label1.Location = New System.Drawing.Point(16, 104)
Me.Label1.Name = "Label1"
Me.Label1.Size = New System.Drawing.Size(96, 23)
Me.Label1.TabIndex = 0
Me.Label1.Text = "NAME"
'
'Button1
'
Me.Button1.Location = New System.Drawing.Point(431, 160)
Me.Button1.Name = "Button1"
Me.Button1.Size = New System.Drawing.Size(128, 23)
Me.Button1.TabIndex = 12
Me.Button1.Text = "UPDATE"
'
'Button2
'
Me.Button2.Location = New System.Drawing.Point(431, 192)
Me.Button2.Name = "Button2"
Me.Button2.Size = New System.Drawing.Size(128, 23)
Me.Button2.TabIndex = 13
Me.Button2.Text = "CLEAR FORM"
'
'Button3
'
Me.Button3.Location = New System.Drawing.Point(431, 256)
Me.Button3.Name = "Button3"
Me.Button3.Size = New System.Drawing.Size(128, 23)
Me.Button3.TabIndex = 14
Me.Button3.Text = "EXIT"
'
'GroupBox1
'
Me.GroupBox1.Controls.Add(Me.Button6)
Me.GroupBox1.Controls.Add(Me.Button5)
Me.GroupBox1.Controls.Add(Me.Button4)
Me.GroupBox1.Location = New System.Drawing.Point(424, 112)
Me.GroupBox1.Name = "GroupBox1"
Me.GroupBox1.Size = New System.Drawing.Size(232, 176)
Me.GroupBox1.TabIndex = 15
Me.GroupBox1.TabStop = False
Me.GroupBox1.Text = "OPTIONS"
'
'Button4
```

```
'
Me.Button4.Location = New System.Drawing.Point(8, 112)
Me.Button4.Name = "Button4"
Me.Button4.Size = New System.Drawing.Size(128, 23)
Me.Button4.TabIndex = 0
Me.Button4.Text = "BACK"
'
'Label2
'
Me.Label2.Location = New System.Drawing.Point(152, 8)
Me.Label2.Name = "Label2"
Me.Label2.Size = New System.Drawing.Size(176, 16)
Me.Label2.TabIndex = 8
Me.Label2.Text = "PLEASE ENTER THE ROLL NO."
'
'Label6
'
Me.Label6.Location = New System.Drawing.Point(152, 48)
Me.Label6.Name = "Label6"
Me.Label6.Size = New System.Drawing.Size(184, 16)
Me.Label6.TabIndex = 9
Me.Label6.Text = "PLEASE ENTER THE SEMESTER"
'
'Button5
'
Me.Button5.Location = New System.Drawing.Point(8, 16)
Me.Button5.Name = "Button5"
Me.Button5.Size = New System.Drawing.Size(128, 23)
Me.Button5.TabIndex = 1
Me.Button5.Text = "GET INFORMATION"
'
'OleDbConnection1
'
Me.OleDbConnection1.ConnectionString = "Jet OLEDB:Global Partial Bulk Ops=2;Jet
OLEDB:Registry Path=;Jet OLEDB:Database L" & _
"ocking Mode=1;Jet OLEDB:Database Password=;Data Source=""G:\documents\Visual
Stud" & _
"io Projects\STUDENT DATA BASE\student.mdb"";Password=;Jet OLEDB:Engine
Type=5;Jet" & _
" OLEDB:Global Bulk Transactions=1;Provider=""Microsoft.Jet.OLEDB.4.0"";Jet
OLEDB:S" & _
"ystem database=;Jet OLEDB:SFP=False;Extended Properties=;Mode=Share Deny
None;Je" & _
"t OLEDB:New Database Password=;Jet OLEDB:Create System Database=False;Jet
OLEDB:" & _
```

```
"Don't Copy Locale on Compact=False;Jet OLEDB:Compact Without Replica
Repair=Fals" & _
"e;User ID=Admin;Jet OLEDB:Encrypt Database=False"
'
'OleDbCommand1
'
Me.OleDbCommand1.CommandText = "SELECT name, cgpa FROM stu WHERE
(rollno = 'Textbox1.text') AND (semester = 'text" & _
"box3.text')"
Me.OleDbCommand1.Connection = Me.OleDbConnection1
'
'Button6
'
Me.Button6.Location = New System.Drawing.Point(144, 16)
Me.Button6.Name = "Button6"
Me.Button6.Size = New System.Drawing.Size(80, 152)
Me.Button6.TabIndex = 2
Me.Button6.Text = "VIEW DATABASE "
'
'Form3
'
Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
Me.BackColor = System.Drawing.Color.LightSteelBlue
Me.ClientSize = New System.Drawing.Size(664, 301)
Me.Controls.Add(Me.Button3)
Me.Controls.Add(Me.Button2)
Me.Controls.Add(Me.Button1)
Me.Controls.Add(Me.GroupBox3)
Me.Controls.Add(Me.PictureBox1)
Me.Controls.Add(Me.PictureBox2)
Me.Controls.Add(Me.PictureBox3)
Me.Controls.Add(Me.GroupBox1)
Me.Name = "Form3"
Me.Text = "Form3"
Me.GroupBox3.ResumeLayout(False)
Me.GroupBox1.ResumeLayout(False)
Me.ResumeLayout(False)

End Sub

#End Region

Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
End
End Sub
```

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click

TextBox1.Text = ""
TextBox2.Text = ""
TextBox3.Text = ""
TextBox4.Text = ""

End Sub


Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button4.Click

Dim fm As New Form1
Me.Hide()
fm.Show()

End Sub

Private Sub Form3_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

TextBox2.Enabled = False
TextBox4.Enabled = False
Button1.Enabled = False

End Sub

Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button5.Click

Dim myCommand As OleDbCommand
Dim dr As OleDbDataReader
Dim nm As String
Dim cgp As Double
Try
OleDbConnection1.Open()
Dim comm As New OleDb.OleDbCommand
comm = OleDbConnection1.CreateCommand

comm.CommandText = "SELECT name, cgpa FROM(stu) WHERE rollno = " &
TextBox1.Text & " AND semester = " & TextBox3.Text & ""
```

```
comm.Connection = OleDbConnection1
MessageBox.Show(comm.CommandText)

dr = comm.ExecuteReader
dr.Read()
nm = dr("name")
cgp = dr("cgpa")
TextBox2.Text = nm
TextBox4.Text = cgp
OleDbConnection1.Close()

Catch ex As Exception
MessageBox.Show(ex.Message)
End Try
TextBox2.Enabled = True
TextBox4.Enabled = True
Button1.Enabled = True
End Sub

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

OleDbConnection1.Open()
Dim comm1 As New OleDbCommand
comm1 = OleDbConnection1.CreateCommand

comm1.CommandText = "update stu set cgpa = " & TextBox4.Text & ", name='" &
TextBox2.Text & "' Where rollno=" & TextBox1.Text & " and semester = " &
TextBox3.Text & ""

comm1.Connection = OleDbConnection1
MessageBox.Show(comm1.CommandText)
comm1.ExecuteNonQuery()
MsgBox("SAVED SUCCESSFULLY...!", MsgBoxStyle.Information)
OleDbConnection1.Close()

End Sub

Private Sub Button6_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button6.Click
Dim fm2 As New Form2
Me.Hide()
fm2.Show()
End Sub

End Class
```

THE DATABASE

The database is meant to store the result information of all the students, his information can be retrieved using the roll number, corresponding to which the percentage achieved could be retrieved. The database creation is done using Microsoft access.

ABOUT MS ACCESS

Microsoft Access is a relational database management system from Microsoft which combines the relational Microsoft Jet Database Engine with a graphical user interface. It is a member of the 2007 Microsoft Office system.

Access can use data stored in Access/Jet, Microsoft SQL Server, Oracle, or any ODBC-compliant data container. Skilled software developers and data architects use it to develop application software. Relatively unskilled programmers and non-programmer "power users" can use it to build simple applications. It supports some object-oriented (OO) techniques but falls short of being a fully OO development tool.

Access was also the name of a communications program from Microsoft, meant to compete with ProComm and other programs. This Access proved a failure and was dropped. Years later Microsoft reused the name for its database software.

One of the benefits of Access from a programmer's perspective is its relative compatibility with SQL (structured query language) —queries may be viewed and edited as SQL statements, and SQL statements can be used directly in Macros and VBA Modules to manipulate Access tables. In this case, "relatively compatible" means that SQL for Access contains many quirks, and as a result, it has been dubbed "Bill's SQL" by industry insiders. Users may mix and use both VBA and "Macros" for programming forms and logic and offers object-oriented possibilities.

MSDE (Microsoft SQL Server Desktop Engine) 2000, a mini-version of MS SQL Server 2000, is included with the developer edition of Office XP and may be used with Access as an alternative to the Jet Database Engine.

Unlike a complete RDBMS, the Jet Engine lacks database triggers and stored procedures. Starting in MS Access 2000 (Jet 4.0), there is a syntax that allows creating queries with parameters, in a way that looks like creating stored procedures, but these procedures are limited to one statement per procedure.Microsoft Access does allow forms to contain code that is triggered as changes are made to the underlying table (as long as the modifications are done only with that form), and it is common to use pass-through queries and other techniques in Access to run stored procedures in RDBMSs that support these.

In ADP files (supported in MS Access 2000 and later), the database-related features are entirely different, because this type of file connects to a MSDE or Microsoft SQL Server, instead of using the Jet Engine. Thus, it supports the creation of nearly all objects in the underlying server (tables with constraints and triggers, views, stored procedures and UDF-s). However, only forms, reports, macros and modules are stored in the ADP file (the other objects are stored in the back-end database).

- **TABLES:** A Table is a collection of data about a specific topic, such as products or suppliers. Using a separate Table for each topic means that you store that data only once, which makes your Database more efficient, and reduces data-entry errors.

- **QUERIES:** You use Queries to view, change, and analyze data in different ways. You can also use them as the source of records for forms, reports, and Data Access Pages. The most common type of Query is a Select Query. A Select Query retrieves data from one or more Tables by using criteria you specify and then displays it in the order you want.

- **REPORTS:** A Report is an effective way to present your data in a printed format. Because you have control over the size and appearance of everything on a Report, you can display the information the way you want to see it. Most of the information in a Report comes from an underlying Table, Query, or SQL statement, which is the source of the Report's data. Other information in the Report is stored in the Report's design.

- **DATA ACCESS PAGES :** A Data Access Page is a special type of Web page designed for viewing and working with data from an Internet or intranet — data that is stored in a Microsoft Access Database or a Microsoft SQL Server database. The Data Access Page may also include data from other sources, such as Microsoft Excel. Data Access Pages can supplement the forms and reports that you use in your Database application.

- **MODULES:** A Module is a collection of Visual Basic for Applications declarations and procedures that are stored together as a unit.

THE COMMA SEPERATED VALUE FILE

The **comma-separated values** (or **CSV**; also known as a **comma-separated list** or **Comma-Separated Variable**) file format is a file type that stores tabular data. The format dates back to the days of mainframe computing. For this reason, CSV files are common on all computer platforms.

CSV is one implementation of a delimited text file, which uses a comma to separate values. However CSV differs from other delimiter separated file formats in using a " (double quote) character around fields that contain reserved characters (such as commas or newlines). Most other delimiter formats either use an escape character such as a backslash, or have no support for reserved characters.

In computer science terms, this type of format is called a "flat file" because only one table can be stored in a CSV file. Most systems use a series of tables to store their information, which must be "flattened" into a single table, often with information repeated over several rows, to create a delimited text file.

Many informal documents exist that describe the CSV format. How To: The Comma Separated Value (CSV) File Format provides an overview of the CSV format in the most widely used applications and explains how it can best be used and supported.

The basic rules are as follows:

CSV is a delimited data format that has fields/columns separated by the comma character and records/rows separated by newlines. Fields that contain a special character (comma, newline, or double quote), must be enclosed in double quotes. However, if a line contains a single entry which is the empty string, it may be enclosed in double quotes. If a field's value contains a double quote character it is escaped by placing another double quote character next to it. The CSV file format does not require a specific character encoding, byte order, or line terminator format

```
31224,6,"vishal",6

31232,75,"saurabh",8

31233,82,"abhishek",4

31234,78,"gopal",7

31235,87,"rajesh",4

312404,88,"arvind",8

31277,90,"RAJAT",8
```

The Plugin

A plugin (or plug-in) is a computer program that interacts with a main application to provide a certain, usually very specific, function.
The main application provides services which the plugins can use, including a way for plugins to register themselves with the main application and a protocol by which data is exchanged with plugins. Plugins are dependent on these services provided by the main application and do not usually work by themselves. Conversely, the main application is independent of the plugins, making it possible for plugins to be added and updated dynamically without changes to the main application.

An IVM Plugin is a small external program that IVM can run to process or obtain data during a call. For example, if you are making a interactive voice response system to tell the caller the current temperature, a plugin can be used to read from the hardware and return the temperature to IVM. A plugin could also be used to restart the computer, access a database, process credit card orders and much more.

The plugin our case is written to accept values that are entered through the telephone keypad and corresponding to that value the percentage is retrieved.

The find data plugin Returns information from a simple database stored as a comma delimited list. Can be used to return information to a caller based on what they dial in.

FINDDATA PLUGIN :

Finddatax searches and gets data from a comma7 delimited text file (or csv file).It can be useful in many circumstances where you need to get data into an telephone IVR system.
Getdatax searches for the item "SearchItemColumn1" in column 1 of the comma delimited list.

If it finds the item it returns with "[FoundReturnVariableName] = [FoundVariableValue] & [ReturnVariableNameColumn2] = DataInColumn2&[ReturnVariableNameColumn3] = DataInColumn3...".If it does not find the item it returns with "[FoundReturnVariableName] = [NotFoundVariableValue]".Usually you can use %[FoundReturnVariableName]% to determine the next OGM (depending on whether matching data was found).

Usage:

finddatax.exe "DataFileFullPath.csv" SearchItemColumn1 [FoundReturnVariableName] [FoundVariableValue] [NotFoundVariableValue] [ReturnVariableNameColumn2] [ReturnVariableNameColumn3] [ReturnVariableNameColumn4] .....

where:

DataFileFullPath.csv is the full pathname of the file to be read, including the extension.

SearchItemColumn1 is the value to search for in column 1.

FoundReturnVariableName is the name of the variable to be returned, default is data.

FoundVariableValue is the value to be assigned to the FoundReturnVariableName if a match is made, default is found. NotFoundVariableValue is the value to be assigned to the FoundReturnVariableName if no match is made, default is not found. ReturnVariableColumn2.... is the name of the variable to be returned with the relevant column data.

PLUGIN CODE :

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>

// Format finddatax.exe DataFilePath.csv SearchItemColumn1 [FoundReturnVariableName]
```

```
//[FoundVariableValue] [NotFoundVariableValue] [ReturnVariableNameColumn2]
//[ReturnVariableNameColumn3] [ReturnVariableNameColumn4] .....
// ** GetCommaDelString
void GetCommaDelString(const char* szInString, size_t& iidx, char* szOutString,
size_t maxstr)
{
        size_t oidx = 0;
        maxstr--;
        while (1)
                {
                        char c = szInString[iidx];
                        if ((c == 0) || (c == ',') || (c == '\n') || (c == '\r'))
                                        {
                                                if (c == ',') iidx++;
                                                    szOutString[oidx] = 0;
                                                return;
                                        }
                        iidx++;
                        if (oidx < maxstr) szOutString[oidx++] = c;
                }
}

// ** TrimString

void TrimString(char* szString)
{
        size_t ist = 0;
        while (szString[ist] == ' ' || szString[ist] == 34) ist++;

        if (ist)
            {
```

```
            int ost = 0;
            char c;
            do {
                    c = szString[ist++];
                    szString[ost++] = c;
                } while (c);
        }
    size_t ien = strlen(szString);
    while ((ien > 0) && (szString[ien - 1] == ' ' || szString[ien - 1] == 34)) ien--;
    szString[ien] = 0;
}
int main(int argc, char *argv[], char *[])
  {
    setvbuf(stdout, NULL, _IONBF, 0);
    const char* szDataVariable = ((argc > 3) ? argv[3] : "data");
    const char* szFoundVariable = ((argc > 4) ? argv[4] : "found");
    const char* szNotFoundVariable = ((argc > 5) ? argv[5] : "not found");

    char szStatusValue[256];
    strcpy(szStatusValue,szNotFoundVariable);

    if (argc > 2)
        {
            char szValue[256];
            strcpy(szValue, argv[2]);
            TrimString(szValue);
            FILE* fh;
            int cntr=0;
            time_t szSTime,szCTime;
            while(!(fh = fopen(argv[1], "r")) && cntr<5)
                {
```

```
            time(&szSTime);
            time(&szCTime);
            while(difftime(szCTime, szSTime)<1)
            time(&szCTime);
            cntr++;
        }
if (fh)
        {
            while (!feof(fh))
                {
                    size_t idx = 0;
                    char szLine[512];
                    if (fgets(szLine, 512, fh) == NULL) break;
                    char szCurrentValue[256];
                    GetCommaDelString(szLine, idx, szCurrentValue,
                                                sizeof szCurrentValue);
                    TrimString(szCurrentValue);
                    if (stricmp(szCurrentValue, szValue) == 0)
                        {
                            strcpy(szStatusValue,szFoundVariable);
                            fclose(fh);
                            printf("%s=%s", szDataVariable,
                                                szSt atusValue);
                           for( int i=6;i<argc;i++){
                               char szData[256];
                               GetCommaDelString(szLine, idx, szData,
                                                sizeof szData);
                                        TrimString(szData);
                                printf("&%s=%s", argv[i], szData);
                           }
                        return 0;
```

```
                        }
                }
                fclose(fh);
        }
        else
                printf("Error - File open failed\n");
    }
    printf("%s=%s", szDataVariable, szStatusValue);
    return 1;
}
```

RUNNING THE PLUGIN

A plugin is usually obtained as an exe file, the file path is then given by clicking the add new exe tab

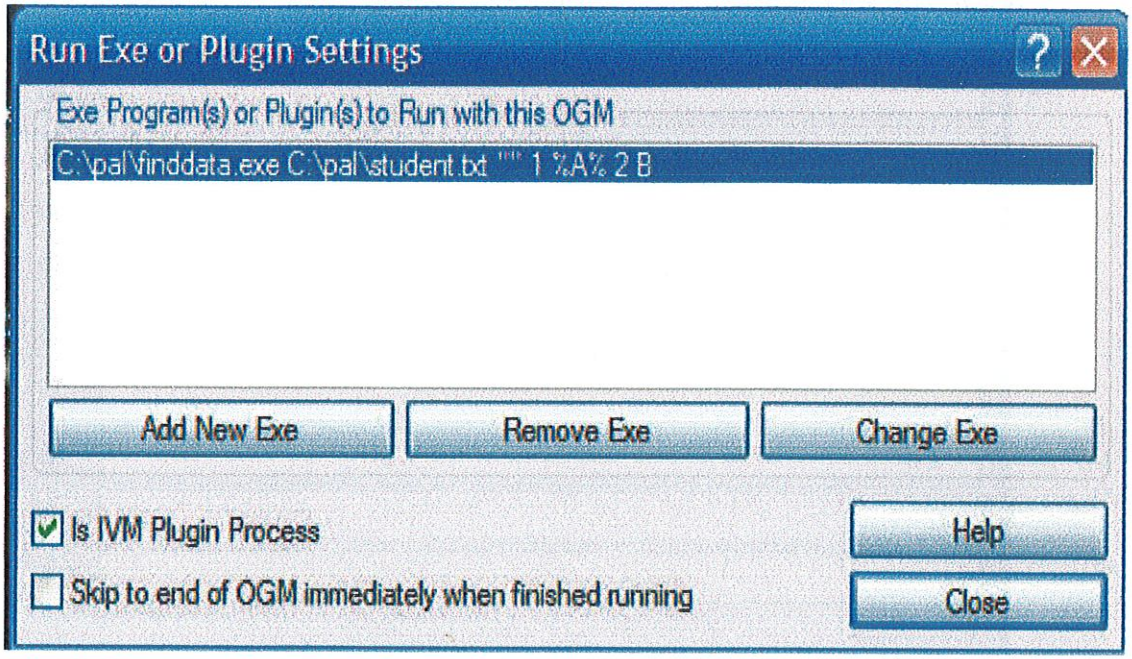


Fig. 6: WINDOW FOR RUNNING THE PLUGIN

- The first argument to the plugin is the Path of the CSV file that is exported from the MS ACCESS database.

- The second argument to the plugin is the variable into which the roll number entered by the caller is cached.

- The third argument to the plugin is the variable into which the percentage achieved by the student is retrieved.

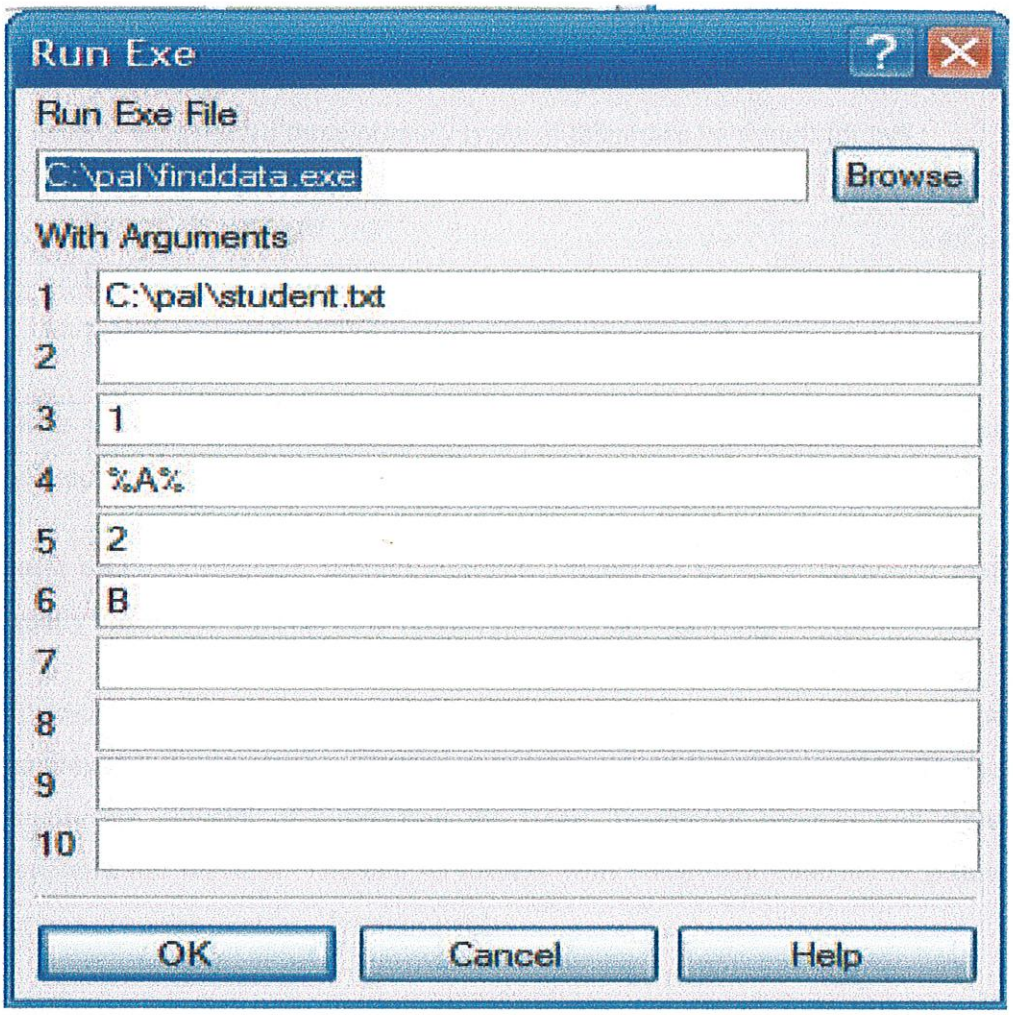


Fig. 7: WINDOW FOR SUPPLYING THE PARAMETERS

THE MODEM

In general we detected three primary types of telephone hardware for pcs :

- Basic data modems
- Voice-data modems
- Telephony cards

These three types of interface cards provide a wide range of telephony service for desktop workstations.

Basic data modems can support Assisted Telephony services (outbound dialing) and usually are able to support only limited inbound call handling.

Voice-data modems are a new breed of low-cost modems that provide additional features that come close to that of the higher-priced telephony cards. These modems usually are capable of supporting the Basic Telephony services and many of the Supplemental services. The key to success with voice-data modems is getting a good service provider interface for your card.

Finally, telephony cards offer the greatest level of service compatibility. Telephony cards usually support all of the Basic Telephony and all of the Supplemental Telephony services, including phone device control. Most telephony cards also offer multiple lines on a single card. This makes them ideal for supporting commercial-grade telephony applications.

We also look at how modems work and how Win95 and WinNT use modem drivers to communicate with hardware devices.

All TAPI services are routed through some type of modem. These modems also depend on the Windows operating system to supply device drivers to communicate between programs and the device itself. While a detailed discussion of device drivers is beyond the scope of this book, it is a good idea to have a general understanding of how Windows uses device drivers and how modems work. In this section you'll get a quick review of

modem theory and a short discussion of the Universal Modem Driver that ships with Win95 and WinNT.

In This Project the modem we have used is Dlink- Model DFM-560 IS +++ which is an internal PCI modem consisting of the CONEXANT CHIPSET " HSFi CX 11252-11", works well with IVM.

A QUICK REVIEW OF HOW MODEMS WORK

Before getting into the details of how the three types of telephony hardware differ, it is important to do a quick review of how modems work. If you've seen all this before, you can skip to the next section.

Sending computer data over voice-grade phone lines is a bit of a trick. All data stored on a pc (documents, programs, graphics, sound and video files, and so on) is stored as 1s and 0s-binary data. However, standard telephone lines are not capable of sending binary data-only sounds. That means that any information sent over the telephone line has to be in the form of sound waves. In order to accomplish this feat, hardware was invented to convert digital information into sound (that is, to *modulate* it), then back again from sound into digital information (*demodulate* it). This process of modulating and demodulating is how the device got its name: mo-dem (*mo*dulate-*dem*odulate).

Sending data over phones lines involves three main steps. First, a connection must be established between two modem devices over a telephone line. This is typically done by having one modem place a telephone call to the other modem. If the second modem answers the telephone call, the two modems go through a process of determining if they understand each other called *handshaking*. If that is successful, then information can be passed. In the second step, the digital information is modulated into sound and then sent over the voice-grade telephone line to the second modem. In the last step, the modem at the other end of the call converts (demodulates) the sound back into digital information and presents it to the computer for processing (view the graphic, save the file, play the video or audio, and so on).

THE UNIVERSAL MODEM DRIVERS AND TAPI SERVICE PROVIDERS

TAPI requires each workstation to have not just a TAPI-compliant application, but also a Telephony Service Provider Interface (TSPI). This interface talks directly to the hardware to convert your TAPI service requests into commands understood by the hardware. The TSPI is usually supplied by the hardware vendor, but Microsoft Win95 ships with a simple TSPI called the UniModem Driver (Universal Modem Driver). The UniModem driver is designed to support Assisted Telephony and some Basic Telephony. You can build simple applications that allow users to place and receive voice and data calls using basic data modems and the UniModem driver that ships with Win95 and WinNT.

Microsoft has released a modem driver that supports additional voice features including playing and recording audio files. This driver is called the UniModemV Driver (Universal Modem for Voice). This driver supports the use of voice commands along with recording and playing back voice files. It can also handle caller ID and some other added service features. Exactly what the UniModemV driver can do is also dependent on the hardware. The telephony hardware must recognize any advanced features and be able to communicate them to the driver.

Basic Data Modems

The most basic type of hardware that supports TAPI is the basic data modem. This type of modem is designed to use analog phone lines to send digital data. Any computer that can access online services (BBS, Internet, commercial information services, and so on) has at least this level of modem hardware. You can get basic data modems with speeds of 14,400 to 56,000bps (bits per second).

Almost all basic data modems recognize a common set of control codes. This set of control codes is called the Hayes or AT command set. This set of controls was developed by the makers of the Hayes modem. The first command in the set (AT) is the "attention" command. This tells the device you are about to send control codes directly to the

hardware. The command set is known by the original author's name ("Hayes") or by the first command in the set ("AT").

Basic data modems support Assisted Telephony services without any problem (that is, placing outbound calls). Most basic modems are capable of supporting some of the Basic Telephony services, including accepting inbound calls. However, if you want to perform any of the more advanced TAPI services, such as playing or recording audio files, you'll need more advanced hardware. Also, if you want to access advanced features available for voice telephones such as caller ID, call hold, park, forward, and so on, you'll need more than a basic data modem. Figure below shows the TAPI service levels and telephony hardware classes. The highlighted areas give you an idea of how basic data modems do in supporting TAPI services.

If you are designing applications that allow users to select names or phone numbers and then place outbound voice or data calls, basic modems will work just fine. In fact, unless you are planning to add voice recording, playback, or other advanced telephony features to your application, the basic modem will provide all your TAPI needs.

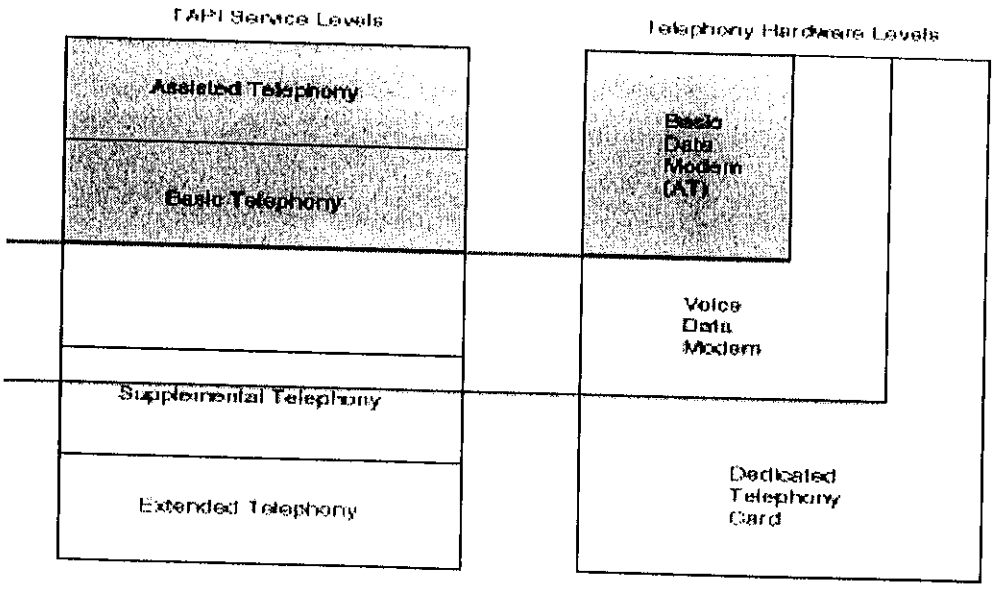


Fig. 8: BASIC MODEM SUPPORT FOR TAPI SERVICES

Data Modems with Voice

There is a new type of modem available that offers all the services of a data modem, but also has added support for voice services. These modems are often called voice-data modems (or data-voice modems). This hardware has additional programming built into the chips that will support advanced telephone features such as caller ID, call hold, park, forward, and so on. Just as basic data modems use the AT command set, the voice-data modems use an extension of that set called the *AT+V* command set (*AT plus Voice*).

AT+V modems cost a bit more than basic data modems. You can find them in the U.S. packaged with sound cards and other multimedia hardware.

Voice-data modems also require a TAPI-compliant modem driver in order to work with TAPI services. This driver is usually supplied by the hardware vendor. Microsoft also supplies a modem driver that supports voice services-the UniModemV driver. If your modem does not ship with a TAPI-compliant driver, you might be able to install the UniModemV driver to enable your voice features.

A word of caution is in order when purchasing a voice-data modem. There are several modems on the market that offer voice, voice-mail, telephone answering, and other TAPI-like services for pcs. The thing to keep in mind is that many of them are not TAPI-compliant. While you may get a modem that can do all the things you want, it may not do it using the TAPI calls and you many not be able to program it using TAPI services.

As of the writing of this book, there are a handful of voice-data modem vendors that have announced the release of TAPI-compliant hardware. Here is a list of some vendors currently offering TAPI-compliant voice-data modems:

- Compaq Presario Systems
- Creative Labs Phone Blaster
- Logicode 14.4 pcMCIA

- Diamond Telecommander 2500
- Cirrus Logic
- Aztech Systems

Voice-data modems with supporting TAPI drivers offer a wide range of access to TAPI services. You can use voice-data modems to perform both outbound and inbound call handling, play and record voice files, and (if the feature is available on the telephone line) support caller ID and other advanced services for single-line phones. Figure belowshows how voice-data modems shape up in their support of TAPI services.

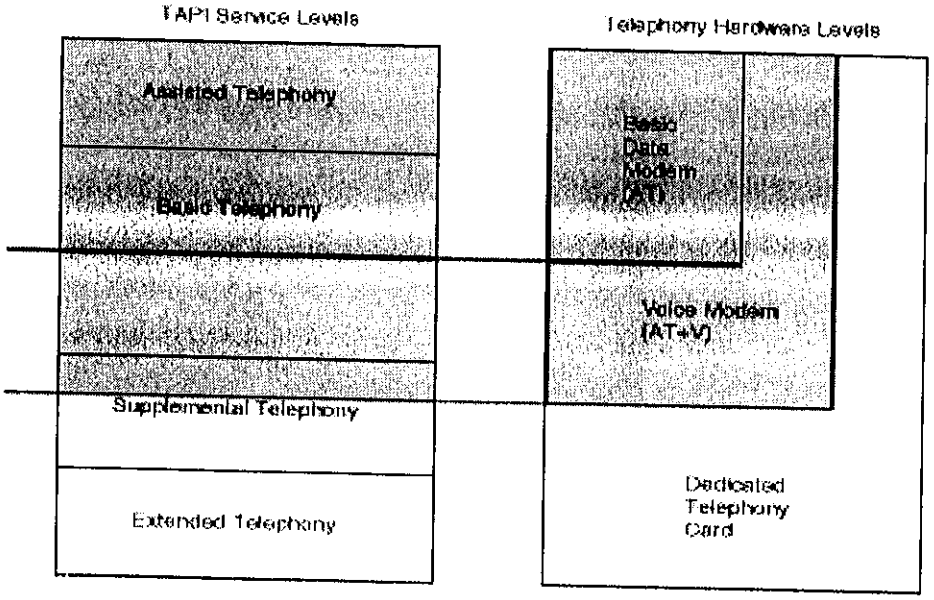


Fig. 9: VOICE-DATA MODEM SUPPORT FOR TAPI SERVICES

Telephony Cards

The most advanced level of hardware you can get for TAPI services on a desktop pc is a dedicated telephony card. This is a piece of hardware dedicated to handling telephone services. Most telephony cards are designed to handle more than one line at a time, too.

If you are planning an application that must answer several phone lines or perform any line transfers, and so on, you'll need a telephony card.

Most telephony cards are sold as part of a kit. You can get software development tools, cards for the pc, cables, and documentation all for one price. This price usually starts at around $1000 U.S. and can easily climb depending on the number of lines you wish to support. Even though the price is a bit high, if you are doing any serious TAPI work, you'll need this kind of equipment.

As with other telephony hardware, telephony cards need an accompanying TAPI driver in order to recognize TAPI calls from your program. While most telephony card vendors are working on TAPI drivers, not all of them supply one as of this writing. It is important to check the specifications of the hardware and supporting materials before you buy.

It is also important to point out that there are lots of very sophisticated hardware and software tools for handling telephony services that are not TAPI-based. It is possible that you will be able to find the right hardware and software to meet your needs without using TAPI services at all. The only drawback is that you'll be using a proprietary system that may (or may not) become obsolete in the future. If it is possible, it is a good idea to use TAPI-compliant products since the power of Microsoft and the Windows operating system is likely to support interfaces like TAPI for quite some time.

Telephony cards (along with TAPI drivers to match) offer the greatest access to TAPI services. You can support all the Assisted TAPI and Basic TAPI functions along with access to Supplemental TAPI services. Also, if the driver supports it, you will be able to use Extended TAPI services to gain access to vendor-specific functions unique to the installed hardware. Figure below shows how telephony cards support all levels of TAPI services.

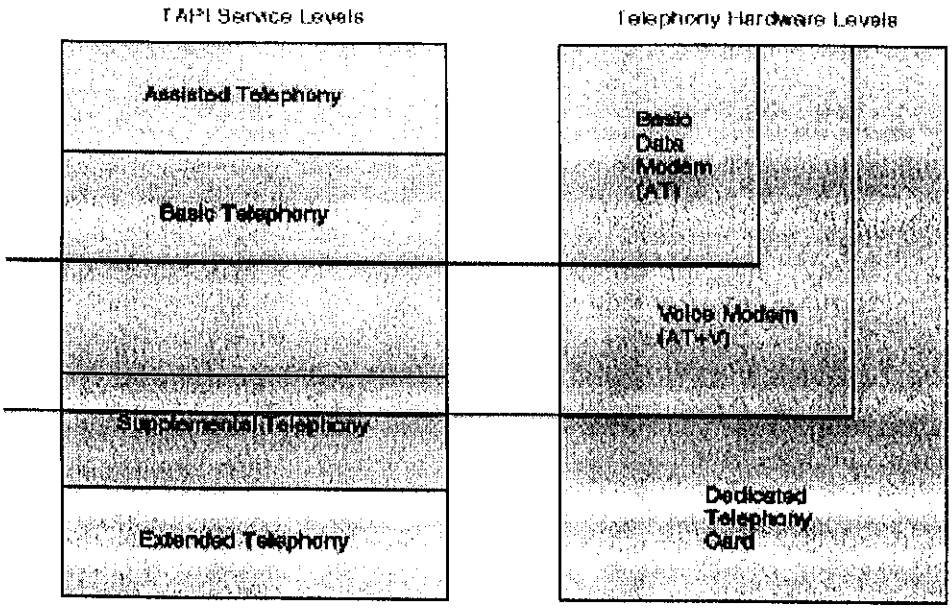


Fig.10: TELEPHONE CARDS CAN SUPPORT ALL LEVELS OF TAPI SERVICES

To sum up we can say that :

- **Basic data modems** support Assisted Telephony services (outbound dialing) and can support only limited inbound call handling. Use this type of hardware if you are building simple outbound dialing applications.
- **Voice-data modems** are capable of supporting the Assisted Telephony and Basic Telephony services and many of the Supplemental services. Use this type of hardware if you want to provide both inbound and outbound services on a single-line phone.
- **Telephony cards** support all of the Basic Telephony and all of the Supplemental Telephony services, including phone device control. Most telephony cards also offer multiple lines on a single card. This make them ideal for supporting commercial-grade telephony applications.

TELEPHONY APPLICATION PROGRAMMING INTERFACE (TAPI)

The **Telephony Application Programming Interface (TAPI)** is a Microsoft Windows API, which provides computer telephony integration and enables PC's running Microsoft Windows to use telephone services. Different versions of TAPI are available on different versions of Windows. TAPI was introduced in 1993 as the result of joint development by Microsoft and Intel. The first publicly available version of TAPI was version 1.3, which was released as a patch on top of Microsoft Windows 3.1. Version 1.3 is no longer supported, although some MSDN development library CDs still contain the files and patches.

With Microsoft Windows 95, TAPI was integrated into the operating system. The first version on Windows 95 was TAPI 1.4. TAPI 1.4 had support for 32-bit applications.

The TAPI standard supports both connections from individual computers and LAN connections serving any number of computers.

TAPI 2.0 was introduced with Windows NT 4.0. Version 2.0 was the first version on the Windows NT platform. It made a significant step forward by supporting ACD and PBX-specific functionality.

In 1997, Microsoft released TAPI version 2.1. This version of TAPI was available as a downloadable update and was the first version to be supported on both the Microsoft Windows 95 and Windows NT/2000 platforms.

TAPI 3.0 was released in 1999 together with Windows 2000. This version enables IP telephony (VoIP) by providing simple and generic methods for making connections between two (using H.323) or more (using IP Multicast) computers and now also offers the ability to access any media streams involved in the connection.

Windows XP included both TAPI 3.1 and TAPI 2.2. TAPI 3.1 supports the Microsoft Component Object Model and provides a set of COM objects to application programmers. This version uses File Terminals which allow applications to record

streaming data to a file and play this recorded data back to a stream. A USB Phone TSP (Telephony Service Provider) was also included which allows an application to control a USB phone and use it as a streaming endpoint.

The Telephony Server Application Programming Interface (TSAPI) is a similar standard developed by Novell for NetWare servers.

TAPI is based on the principles of the Windows Open Services Architecture (WOSA). TAPI provides some central services and holds some global state. Its main purpose, however, is to provide connections between TAPI Service Providers (TSPs) and TAPI applications. Applications are programmed using TAPI. TSPs implement the Telephony Service Provider Interface (TSPI) functions that are used by the TAPI implementation. Each TSP then uses whatever interface is appropriate to control its telephony hardware. Together, it looks like this:

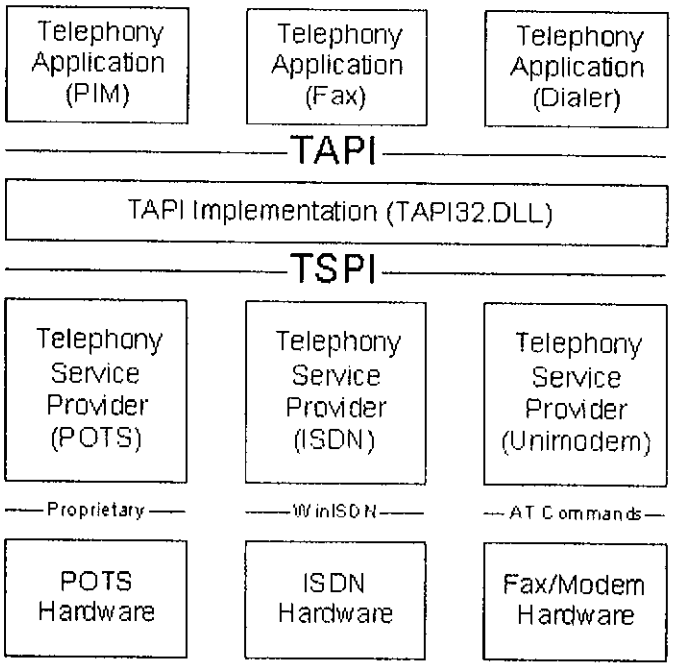


Fig.11 TAPI ARCHITECTURE

This layered approach makes it possible for an application to be developed without worrying about the specific hardware provided on a particular machine. Any telephony hardware vendor can then implement the appropriate parts of the TSPI without worrying about what telephony applications have been installed. This separation gives both applications and hardware independence from each other. Applications and hardware can come and go without directly affecting one another.

The WOSA model is used to achieve the goals of the TAPI, which are:

1. Call control focused.
2. Access to data via existing standard APIs.
3. Network independence.
4. Connection model independence.
5. Platform independence where possible.
6. Sharing of lines between multiple applications.

TAPI 2.x vs TAPI 3.x

It is a common misconception that **TAPI 3.0** (or **TAPI 3.1**) replaces **TAPI 2.x**. TAPI 2.x (and all earlier versions) is written in C/C++ and requires applications to make heavy use of C style pointer arithmetic. This makes TAPI fast and easy to access from C/C++ applications, but it also makes it difficult to use from many other programming languages.

On the other hand, TAPI 3.x was designed with a COM (Component Object Model) interface. This was done with the intent of making it accessible from managed languages like Visual Basic, VB Scriptcode, Java or other environments that provide easy access to COM but don't deal with C-style pointers.

TAPI 3.x has a slightly different set of functionality than TAPI 2.x. The addition of integrated media control was the most significant addition. But TAPI 3.x doesn't include all functionality that TAPI 2.x does, like support for the Phone class.

One very notable issue with TAPI 3.x is the lack of support for managed code (.NET environment). As documented in Microsoft KB Article 841712, Microsoft currently has no plans to support TAPI 3.x directly from .Net programming languages.

One often overlooked reason an application developer might choose between TAPI 2.x and TAPI 3.x should be the hardware vendors recommendation. Even though TAPI provides an abstract model of phone lines, telephony applications are still heavily impacted by the specific behavior of the underlying hardware. Troubleshooting behavior issues usually requires both software and hardware vendors to collaborate. Because there is almost a 1:1 relationship between the TAPI Service Provider (TSP) interface and the

TAPI 2.x interface, collaboration is often easier if the application is designed using TAPI 2.x. Experience with TAPI 3.x varies significantly between hardware vendors

TAPI COMPLIANT HARDWARE

Telephony hardware that supports TAPI includes most voice modems and some telephony cards such as Dialogic boards.

DATA FLOW DIAGRAMS

FIRST LEVEL DATA FLOW DIAGRAM

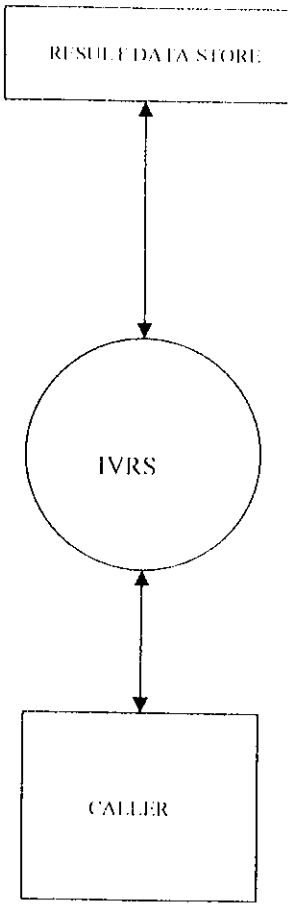


Fig. 12: FIRST LEVEL DFD

SECOND LEVEL DATA FLOW DIAGRAM

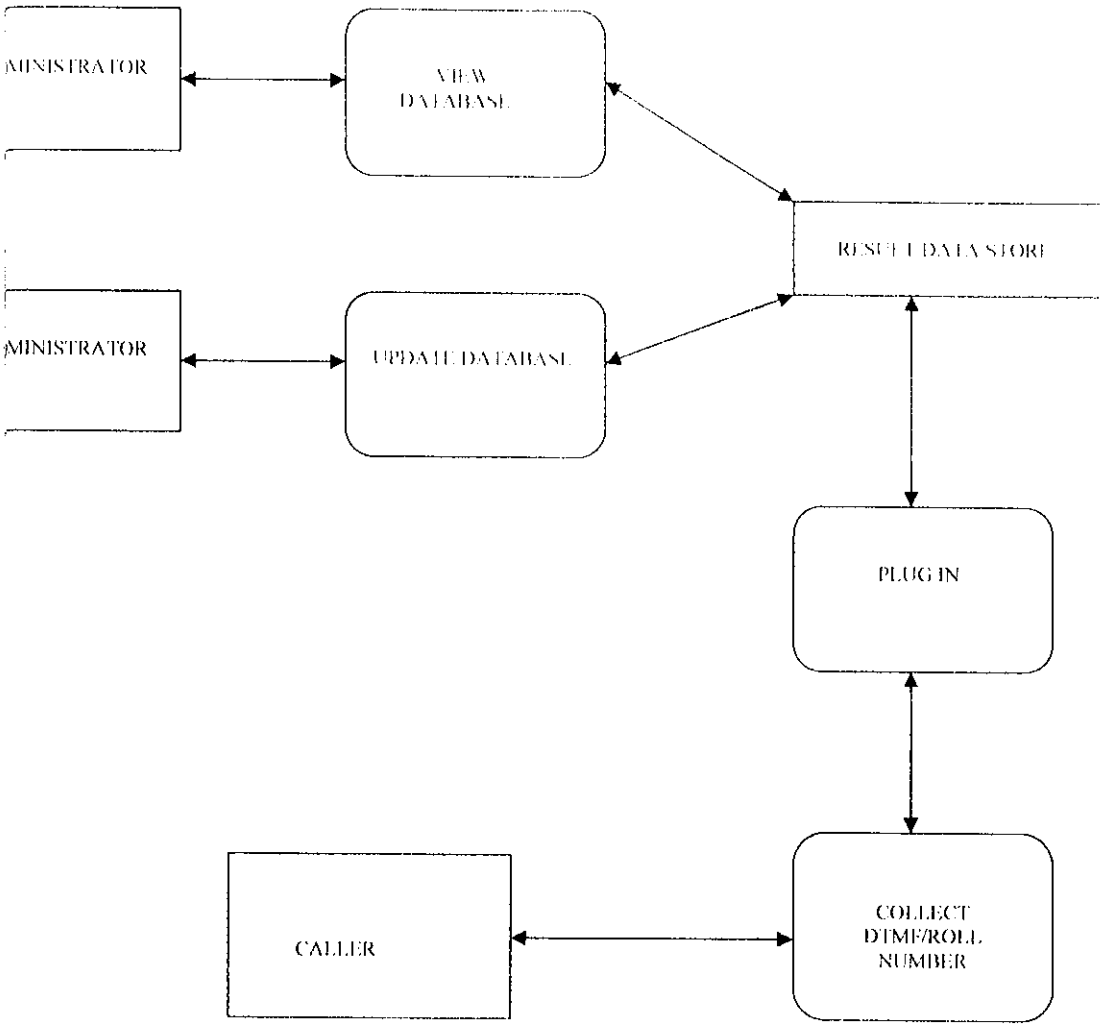
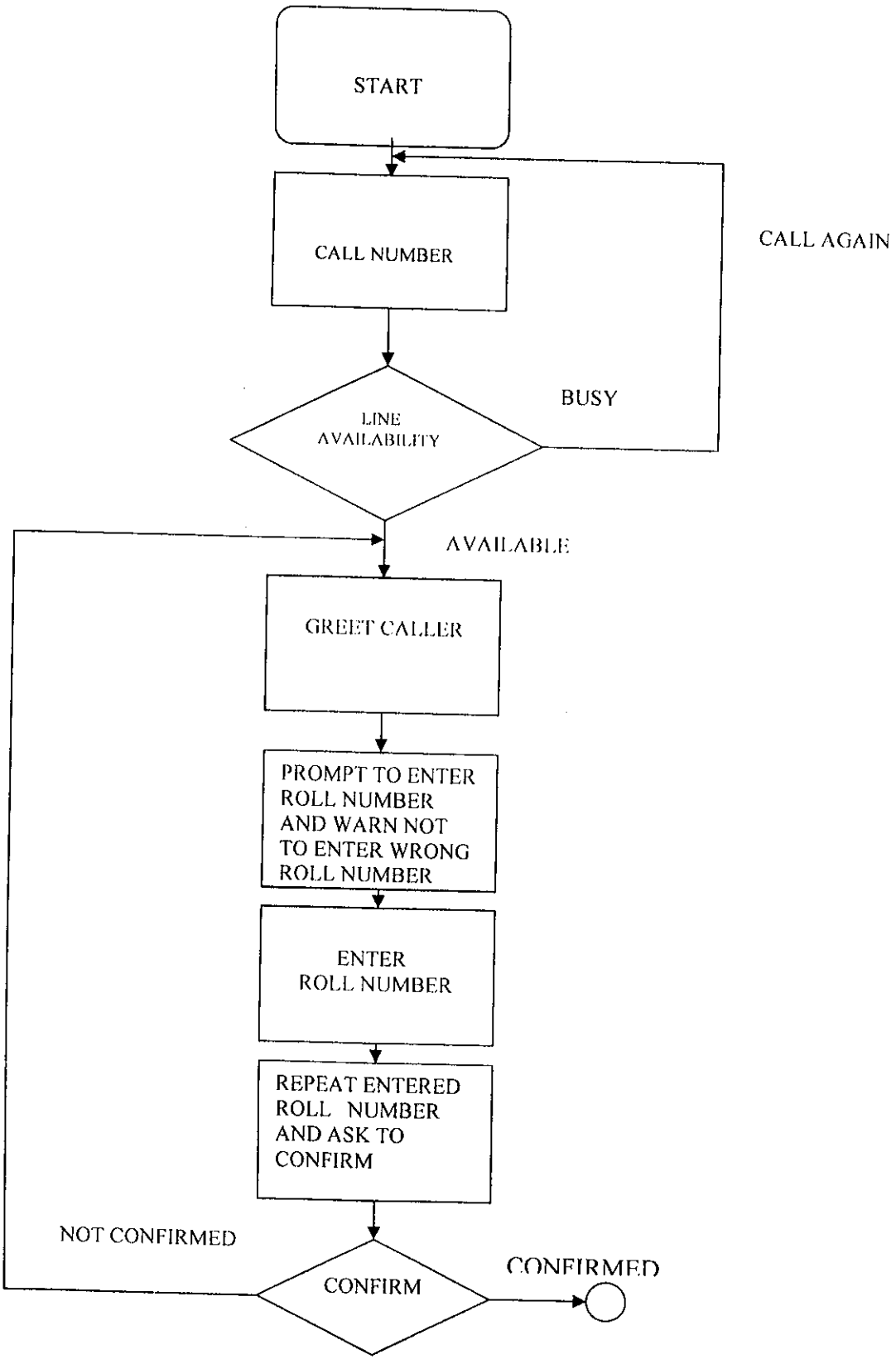


Fig.13 : LEVEL SECOND DATA FLOW DIAGRAM


START
CALL NUMBER
CALL AGAIN
LINE AVAILABILITY
BUSY
AVAILABLE
GREET CALLER
PROMPT TO ENTER ROLL NUMBER AND WARN NOT TO ENTER WRONG ROLL NUMBER
ENTER ROLL NUMBER
REPEAT ENTERED ROLL NUMBER AND ASK TO CONFIRM
NOT CONFIRMED
CONFIRM
CONFIRMED

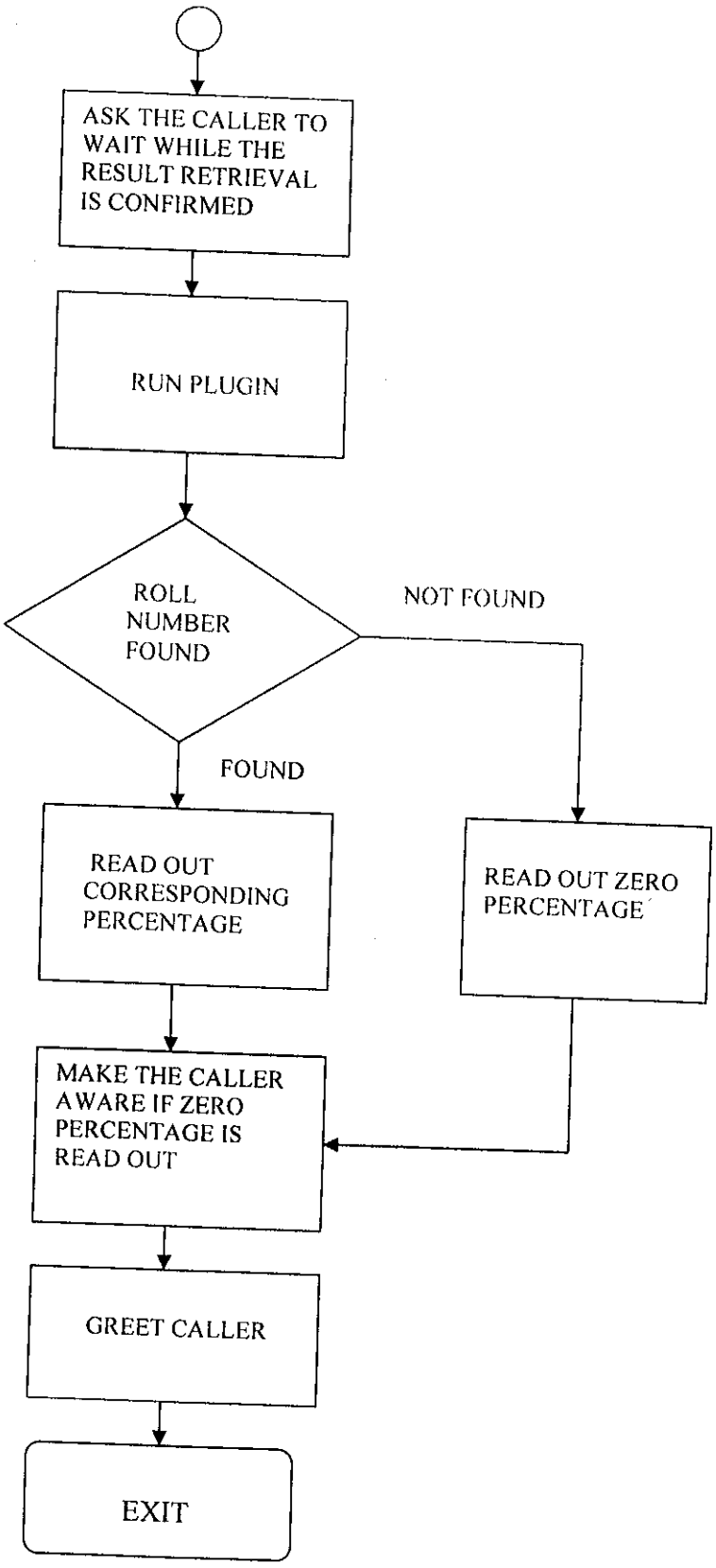


Fig. 14: FLOWCHART

CHALLENGES AND DRAWBACKS

- The biggest challenge has been gathering the desired hardware ie the appropriate Voice modem that is TAPI compliant. The issues related to finding such a modem are that IVM uses the TAPI (Telephony Application Interface) standard to connect to a wide range of telephony hardware devices (including voice modems or professional telephony voice cards). Often when IVM does not work, it is because the hardware is not TAPI compliant or because there is some problem with the drivers or driver installation

- The second biggest challenge for us has been directly coding with the modem to detect the DTMF tone manually. The problem with this approach is that we need to write a separate TAPI driver that uses MS ActiveX controls and for this we will have to develop a separate library in VB.net. As this at present level is out of our scope so it was a challenge for us and a future enhancement.

- The drawback that we have faced is that every time we need to convert the database into CSV file as it cannot be done automatically, the plugin only works with the CSV file.

ENHANCEMENTS

1. Using voice recognition replacing the use of touch tone keypad to enter the commands to IVRS system.

2. The use of multiple languages make the IVR System a truly information system for one and all.

3. Using Dialogic cards for high quality audio and good DTMF tone detection.

4. Create an inbuilt module to detect the DTMF tones

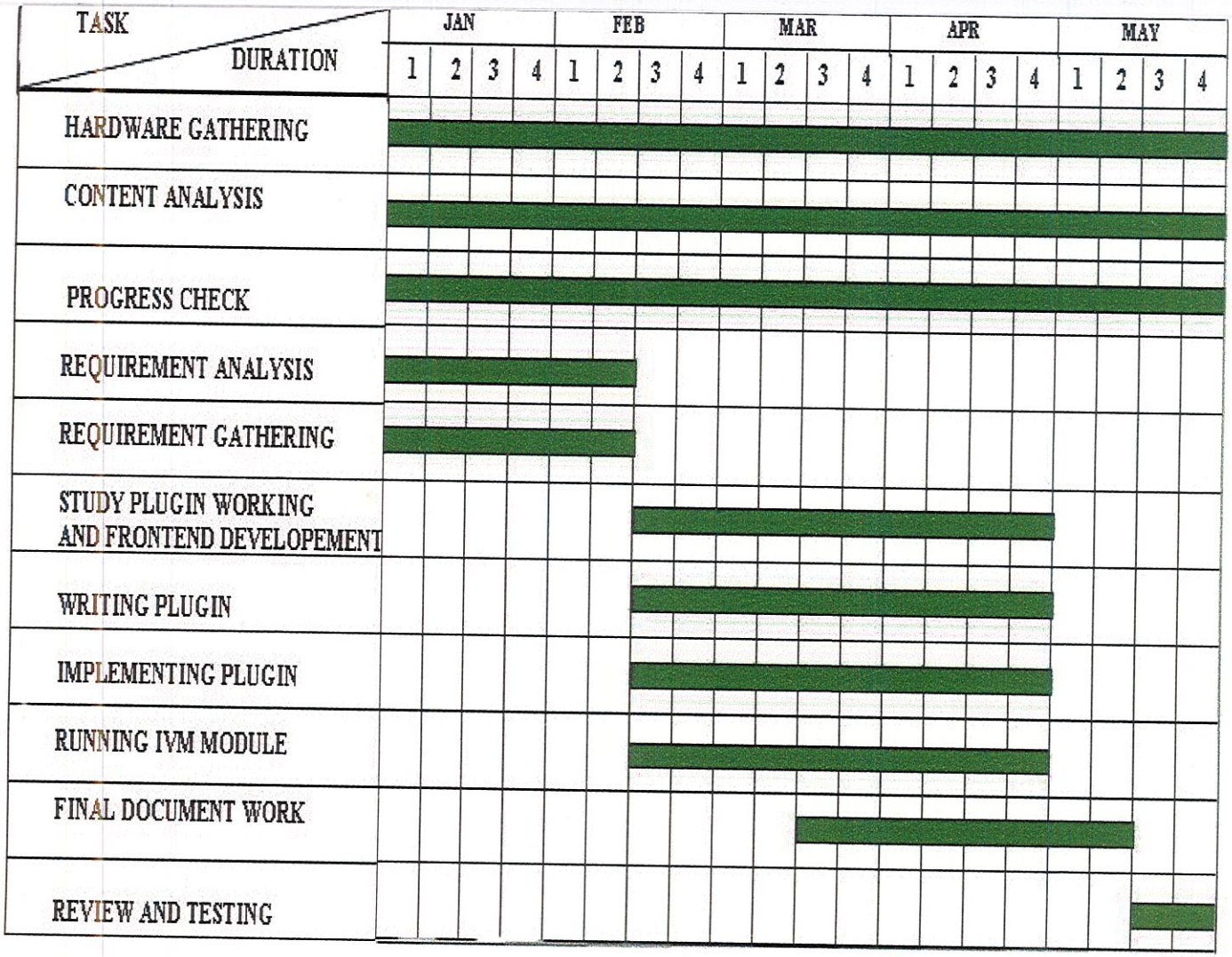


Fig. 15: GANTT CHART

CONCLUSION

In the end we can conclude that the project has been successfully executed and most of the targets that were set in the beginning of the project were accomplished on time. The Gantt Chart shows the progress of the project and tasks accomplished from the beginning till the end.

BIBLIOGRAPHY

1. www.voiceguide.com
2. www.nch.com
3. www.google.com
4. Books :
   - Object Oriented Programming in C++ by Robert Lafore.
   - Let us C by Yashwant Kanitkar.
   - Introduction to C programming by Dietal and Dietal.
   - Black book for VB.NET.
   - Mastering VB.NET
   - Professional VB.NET, 2nd Edition by Fred Barwell
   - Database Programming With VB.NET
   - Visual Basic Design Patterns VB 6.0 and VB.NET by James W. Cooper.
   - Visual Basic .NET Programming by Harold Davis