ALAGAPPA UNIVERSITY

[Accredited with 'A+' Grade by NAAC (CGPA:3.64) in the Third Cycle
and Graded as Category–I University by MHRD-UGC]

(A State University Established by the Government of Tamil Nadu)

KARAIKUDI – 630 003

Directorate of Distance Education

B.Sc. [Information Technology]

VI - Semester

129 64

LAB: .NET PROGRAMMING

Author:

Dr. Preety Khatri, Assistant Professor-SOIT IMS, Noida

Vikas® is the registered trademark of Vikas® Publishing House Pvt. Ltd.

VIKAS® PUBLISHING HOUSE PVT. LTD.
E-28, Sector-8, Noida - 201301 (UP)
Phone: 0120-4078900 • Fax: 0120-4078999
Regd. Office: A-27, 2nd Floor, Mohan Co-operative Industrial Estate, New Delhi 1100 44
• Website: www.vikaspublishing.com • Email: helpline@vikaspublishing.com

Work Order No. AU/DDE/DE 12-27/Preparation and Printing of Course Materials/2020 Dated 12.08.2020 Copies - 500

SYLLABI-BOOK MAPPING TABLE

LAB: .NET PROGRAMMING

Syllabi

BLOCK 1

1. Building Simple Applications, Observe and Draw Visual.NET IDE Layout and Hands on Practice to Create, Save and Open the Project.
2. Working with Intrinsic Controls, Control Arrays, Sub Procedures and Functions.

BLOCK 2

3. Application with Multiple Forms
4. Application with Dialogs
5. Application with Menus
6. Application using Data Controls
7. Application using Common Dialogs

BLOCK 3

8. Drag and Drop Events, In-Built Functions, Mathematical and String Functions
9. Database Management
10. Creating ActiveX Controls
11. Database Object (DAO) and Properties
12. Active Data Objects (ADO) and OLE DB

BLOCK 4

13. Database: Bounded and Unbounded Mode, Connecting to the Database, Retrieving a Recordset, Creating a Query Dynamically, Using a Parameterized Query, Using Action Queries - Adding Records, Editing Records, Closing the Database Connection

BLOCK 5

14. Simple Application Development
    - (i) Library Information System
    - (ii) Students Mark Sheet Processing
    - (iii) Telephone Directory Maintenance
    - (iv) Gas Booking and Delivering
    - (v) Electricity Bill Processing
    - (vi) Bank Transaction
    - (vii) Pay Roll Processing
    - (viii) Personal Information System
    - (ix) Question Database and Conducting Quiz
    - (x) Personal Diary

NOTES

INTRODUCTION

The .NET Framework, pronounced as 'DOT NET' is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large class library called Framework Class Library (FCL) and provides language interoperability (each language can use code written in other languages) across several programming languages. Programs written for .NET Framework execute in a software environment (in contrast to a hardware environment) named the Common Language Runtime (CLR). The CLR is an application virtual machine that provides services, such as security, memory management, and exception handling. As such, computer code written using .NET Framework is termed as the 'Managed Code'. FCL and CLR together constitute the .NET Framework. FCL provides the user interface, data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications. Programmers produce software by combining their source code with .NET Framework and other libraries. The framework is intended to be used by most new applications created for the Windows platform. Microsoft also produces an Integrated Development Environment (IDE) for .NET software called Visual Studio.

This lab manual, *DOT NET Programming*, contains several programs based on DOT NET programming which includes building simple applications, observe and draw Visual.NET IDE layout and hands on practice to create, save and open the project, working with intrinsic controls, control arrays, sub procedures and functions, application with multiple forms, dialogs, menus, data controls, common dialogs, drag and drop events, in-built functions, mathematical and string functions, database management, creating ActiveX controls, Database Object (DAO) and properties, Active Data Objects (ADO) and OLE DB, connecting to the database, retrieving a record set, creating a query dynamically, parameterized query, action queries, simple application development, such as library information system, students mark sheet processing, telephone directory maintenance, gas booking and delivering, electricity bill processing, bank transaction, pay roll processing, personal information system, etc.

In addition, it will help students in coding and debugging their DOT NET programs. The manual provides all logical, mathematical and conceptual programs that can help to write programs easily. These exercises shall be taken as the base reference during lab activities for students.

BLOCK I : LAB .NET PROGRAMMING

NOTES

This block will cover the following topics:

1. Introduction of .NET framework and VB.NET IDE.
2. Create, save and open the project.
3. Work with intrinsic controls, control arrays, Sub Procedures and functions.

Introduction to .NET

.NET is a software framework which is designed and developed by Microsoft. The first version 1.0 of the .NET framework came in 2002. .NET is also defined as XML web services platform which allows to build .NET applications, through which users can interact with the internet using wide range of smart devices like tablets, smart phones etc. It is a virtual machine for compiling and executing programs written in various languages like C#, VB.NET, etc. .NET allows building and integrating the web services and which comes with different set of tools like Visual Studio to fully develop and build those applications.

There is a large variety of programming languages available on the .NET platform, for example VB.NET and C# which is used to build applications for Windows, phone, web, etc. It provides a lot of functionalities and also supports industry standards.

.NET Framework

.NET framework is a software platform. It is a language-neutral environment for building applications and developing .NET experiences that can easily operate within it. When developed applications are deployed, these applications will target .NET and will execute wherever .NET is implemented instead of depending on a particular Hardware/OS combination. The components that make the .NET platform are collectively called the .NET framework. The .NET framework is a managed as well as a type-safe environment for developing and executing various applications. The .NET framework manages all kinds of program execution for example, how to allocate the memory for the storage of data and instructions, managing execution of the application, granting permissions to the application, reallocation of memory etc. Basically, .NET framework is designed for cross-language compatibility, which is an application written in VB .NET may reference a DLL file written in C#. A VB.NET class might be derived from a C# class or vice versa. The .NET Framework consists of various components, some important components are:

a) Common Language Runtime (CLR)

b) Class Libraries

c) Common Language Specification (CLS)

NOTES

a) Common Language Runtime (CLR)

The CLR is an execution engine of .NET which provides the environment to run the program. CLR can manage the execution of programs. It also provides core services like memory allocation, code compilation, thread management, garbage collection etc. The software version of .NET is actually the CLR version. CLR is the virtual machine component of the .NET framework. It is the run-time environment in the .NET framework that runs the codes. . It also helps in code management.The code that targets the runtime is known as the managed code and code that doesn't target to runtime is known as unmanaged code. It helps in making the development process easier by providing the a variety of services like thread management, remoting, memory management, robustness, type-safety etc.. CLR is responsible for managing the execution of .NET programs instead of any .NET programming language.

b) Class Libraries

Class library is another component of .NET framework that is designed to integrate with the CLR. It provides the program access to runtime environment. The class library consists of classes, interfaces, namespaces and value types that can be used in the applications created in VB .NET and visual studio .NET. It contains the number of classes that serves the following functions:

1. It provides the base and user-defined data types.
2. It supports the exceptions handling.
3. It helps in managing I/O and stream operations.
4. It allows access to data.
5. It hels in creating the Windows-based GUI applications.
6. It supports in creating the web services.
7. It is useful in creating the web-client and server applications.

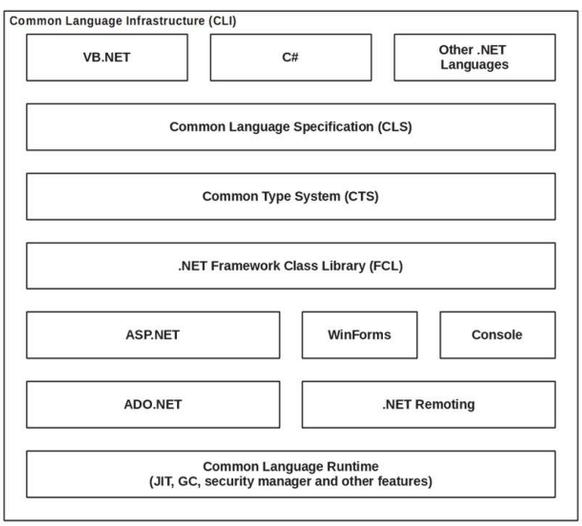


Fig. 1.1 *.NET Framework*

c) Common Language Specification (CLS)

NOTES

The CLS describes a set of rules and constraints that are common in different languages which runs in .NET framework. If we want the code which we write in a language to be used by programs in some other languages, then it should remain in CLS. It defines the minimum standards that .NET language compilers must confirm to ensure that any source code compiled by a .NET compiler and that code interoperate with the other language. Developers are building applications using the .NET framework due to following features:

1. To increase the performance
2. To improved reliability
3. To provide mobility support
4. XML web service support
5. To increase the developer productivity
6. To provide an environment that integrate with existing systems
7. Ease of deployment
8. To provide powerful security
9. Flexible data access

Basic Requirements to Install Visual Studio

The minimum requirements of a system for installing visual studio are:

1. RAM: 256 MB (Recommended)
2. Operating System: Windows 2000 or Windows XP
3. Processor: Pentium II 450 MHz
4. Hard Disk Space: 3.5 GB (Includes 500 MB free space on disk)

Visual Basic .NET

Visual basic .NET provides the easiest and most productive language and tool for building Windows and web applications. It comes with improved visual designers, a powerful integrated development environment (IDE) and increased application performance. It also supports creation of applications for internet-enabled and wireless hand-held devices. There are various features of VB.NET as follows:

a) Building Web-based Applications

With the help of VB.NET, we can build web applications using the shared web form designer. You can double-click and write code to respond to events. There is an enhanced HTML editor for working with complex web pages. We can also use IntelliSense technology and tag completion, or choose the WYSIWYG editor for visual authoring of interactive web applications.

NOTES

b) Powerful Windows-based Applications

VB.NET provides the features like forms designer, an in-place menu editor and automatic control docking and anchoring. VB.NET provides new productivity features for building robust applications very quickly. With the help of IDE environment, VB.NET provides automatic, fast formatting of code, improved IntelliSense, an enhanced object browser and XML designer.

c) Improved Coding

You can code faster and more effectively. A multitude of enhancements to the code editor, including enhanced IntelliSense, smart listing of code for greater readability and a background compiler for real-time notification of syntax errors transforms into a rapid application development (RAD) coding machine. You can tackle any data access scenario easily with ADO.NET and ADO data access. The flexibility of ADO.NET enables data binding to any database, as well as classes, collections, and arrays, and provides true XML representation of data. Seamless access to ADO enables simple data access for connected data binding scenarios. Using ADO.NET, VB.NET can gain high-speed access to MS SQL Server, Oracle, DB2, Microsoft access, and more.

d) Simplified Deployment With VB.NET

With the help of VB.NET, we can build applications more rapidly and maintain them very efficiently. Web auto-download and XCOPY-deployment of Windows-based applications combine the simplicity of web page deployment and maintenance with the power of rich and responsive Windows-based applications. Side-by-side versioning provides multiple versions of the same component to live safely on the same machine so that applications can use a specific version of a component.

e) Direct Access to the Platform

VB.NET provides direct access to the platform. It enables developers can have full access to the capabilities available in .NET framework. The new Windows service project template enables to build real Microsoft Windows NT services. Developers can easily program system services including performance counters, file system and event log.

f) Full Object-Oriented Constructs

VB.NET provides full object-oriented constructs. You can create enterprise, reusable-class code using full object-oriented constructs. Structured exception handling provides a global error handler and eliminates spaghetti code. Language features consists of full implementation encapsulation, polymorphism and inheritance.

VB Language

Visual basic is very popular language for its friendly working environment and it clearly states how widely used for developing applications. VB.NET is an extension of visual basic programming language having various features in it. VB.NET was designed to take advantage of the .NET framework runtime environment and base classes. It comes with power packed features that simplify application development. The changes from VB to VB .NET ranging from the change in syntax of the language to the types of projects and also depends on the way of designing applications. Following are the points which elaborate advancement fromVB to VB.NET.

NOTES

- One of the major changes from VB to VB .NET is that it is based on the concept of object-oriented.
- We can now create classes and objects, and also derive classes from other classes.
- It provides the advantage of code reusability with OOP.
- VB.NET supports multithreading.
- VB.NET adds console applications (that run in the DOS version) to it apart from Windows and web applications.
- VB.NET supports all OOP features i.e. abstraction, inheritance, polymorphism and encapsulation.
- Representing data in XML format allows us to send large amounts of data on the internet. It reduces network traffic when communicating with the database.
- VB.NET requires declaration of all the variables by default before using them.
- Web development is now an integral part of VB.NET making two major types of applications i.e. web forms and web services.
- VB.NET supports structured exception handling using Try...Catch...Finally.
- Various controls can be added to the toolbar which make application development more efficient.
- VB.NET uses ADO.NET, a new data handling model to communicate with databases on local machines or on a network and also it makes handling of data on the internet easy.
- Data in ADO.NET is represented in XML format and is exchanged in the same format.

Namespaces

A namespace is a collection of various classes. The namespace with all the built-in VB functionality is the system namespace. The VB applications are developed

NOTES

using classes from the .NET system namespace. All other namespaces are based on this system namespace.

- ***System****:* It includes necessary classes and base classes for commonly used data types, events and exceptions.
- ***System.Collections:*** It includes classes and interfaces which define collection of objects such as queues, list, arrays, hash tables etc.
- ***System.Globalization:*** It includes classes that specify culture-related information.
- ***System.IO:*** It includes classes for data access with Files System.NET that provides interface to protocols used on the internet.
- ***System.Diagnostics:*** It includes classes that allow to debug our application and to step through our code.
- ***System.Threading:*** It includes classes and interfaces to support multithreading.
- ***System.Data:*** It includes classes which lets us handle data from data sources.
- ***System.Data.OleDb:*** It includes classes that support the OLEDB .NET provider.
- ***System.Security:*** It includes classes to support the structure of common language runtime security system.
- ***System.Data.SqlClient:*** It includes classes that support the SQL Server .NET provider.
- ***System.Drawing:*** It provides access to drawing methods.
- ***System.Reflection:*** It includes classes and interfaces that return information about types, methods and fields.
- ***System.Windows.Forms:*** It includes classes for creating Windows based forms.
- ***System.Web:*** It includes classes and interfaces that support browser-server communication system.
- ***Web.Services*****:** It includes classes that let us build and use Web services.
- ***System.XML:*** It includes classes for XML support.

Console Applications

Console Applications are command-line oriented applications that allow us to read and characters from the console. Console applications are written in code and are supported by the System.Console namespace. The console applications are executed in the DOS.

An Example of Console Application

NOTES

Create a folder in C drive with any name and make sure the console applications which you open are saved there. The default location where all the .NET applications are saved is C:\Documents and Settings\Administrator\My Documents\Visual Studio Projects. The new project dialogue looks like the Figure 2.

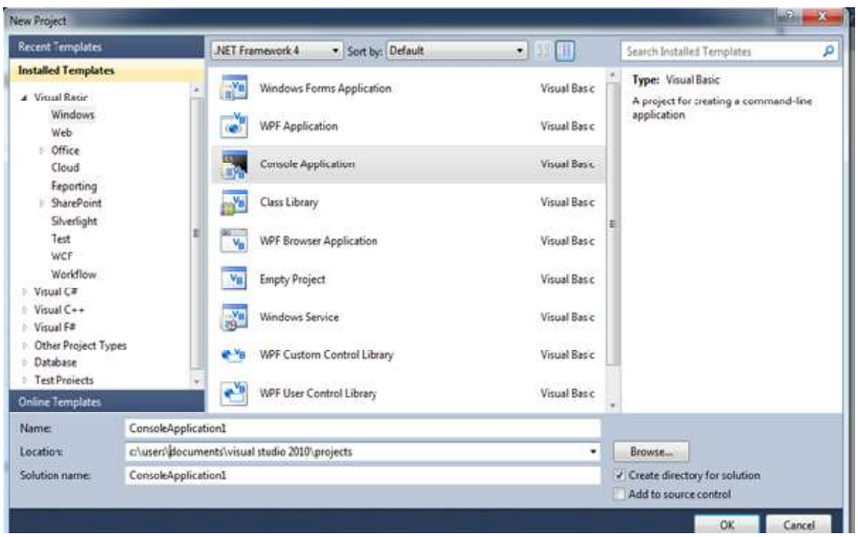


***Fig. 2** Starting Console Applications*

The following code is an example of a console application.

```
Module Module1
Sub Main()
System.Console.Write("Running program with Console Application")
End Sub
End Module
```

When, you run the code by selecting Debug'!Start from the main menu or by pressing F5 on the keyboard. The result "Running program with Console Application" is displayed on a DOS window. Alternatively, you can run the program using the VB compiler (vbc). To do that, go to the Visual Studio.NET command prompt on selecting from Start'!Programs'!Visual Studio.NET'!Visual Studio.NET Tools'!Visual Studio.NET Command Prompt and type: c:\examples>vbc example1.vb.

NOTES

The result "Running program with Console Application" is displayed on a DOS Window as shown in the screenshot given below.

Explanation

See the first line, here we are creating a VB Module and Modules are designed to hold code. The code which we write should be within the module. Next line starts with Sub Main () which is the entry point of the program. The third shows that we are using the Write method of the System.Console class to write to the console.

How to Comment the Code?

In VB.NET, comments start with a single quote (') character and the statements following that are ignored by the compiler. Comments are generally used to define that what is going in the program. It also provides an idea about the flow of the program. The general form looks like this:

```
Dim I as Integer
'declaring an integer
```

Visual Studio .NET IDE

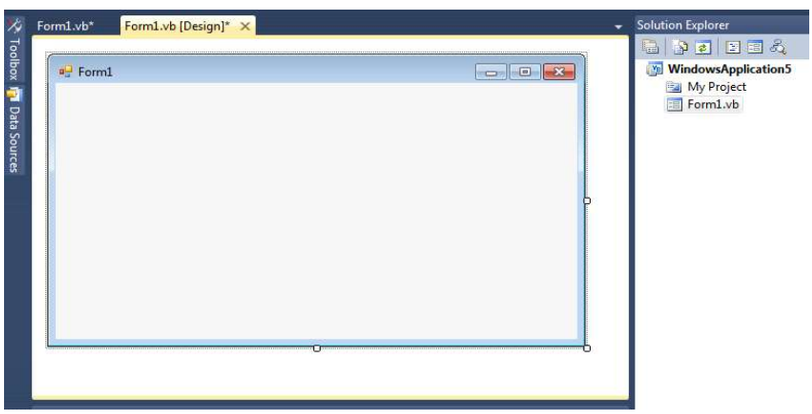
Visual Studio.NET IDE (Integrated Development Environment) provides the environment for developing the .NET based applications which come with various features. Visual Studio .NET IDE is an upgraded version of all previous IDE's by Microsoft. It provides many options and includes many features which simplify application development. Following the the important features of IDE.

1. IDE is Customizable: It can be customized based on your preferences and this can be done using My Profile settings. You can set the IDE screen the way you want and you can also filter the help files based on the language of your choice or set the way the keyboard behaves.
2. One IDE for all .NET Projects: It provides the same environment for developing all types of .NET applications. Applications can range from single windows applications to complex one.

3. Built-in Browser: The IDE have a built-in browser that helps to browse internet without launching another application. With the help of built-in-browser, you can look for source codes, online help files, additional resources etc.
4. Option to choose from Multiple Programming Languages: VS.NET provides various options to choose from multiple programming languages. You can also integrate multiple programming languages in one .NET solution and edit that with the IDE. You can choose the programming language of your choice to develop applications based on your expertise in that language.

You can open the VS.NET using the steps i.e. Start'!Programs'!Microsoft Visual Studio .NET'!Microsoft Visual Studio.NET. The start page also allows us to select from the most recent projects.

The Integrated Development Environment (IDE) is shown in the screenshot given below. It shows the interface with which we actually work with. In this IDE, there is toolbars towards the left side along with the Solution Explorer window towards the right.

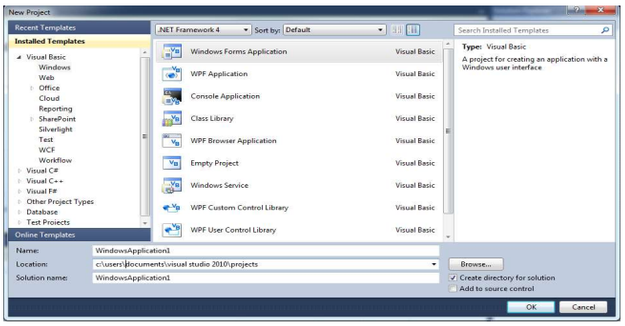


New Project Dialogue Box

The New Project dialogue box is used to create a new project which shows the name the project and also shows it's location on the disk where it is saved. The

NOTES

default location on the hard disk where all the projects are saved is C:\DocumentsandSettings\Administrator\MyDocuments\VisualStudioProjects.

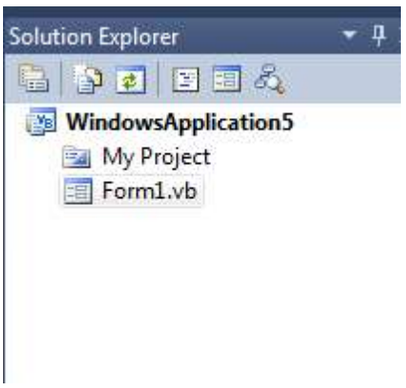


There are various templates under project types. Some are given below:

1. **Windows Application**: It is used to create standard windows based applications.
2. **Web Control Library**: It is used to create user-defined controls for the web.
3. **Windows Control Library:** It is used to create our own windows controls, where you group some controls, add it to the toolbox also.
4. **Console Application:** It is used to create command line based applications.
5. **Class Library:** It is used to provide functionality similar to Active X and DLL by creating classes that access other applications.
6. **ASP.NET Web Application:** It is used to create web-based applications, create web pages, web applications and web services using IIS.
7. **Windows Service:** They are designed for special purpose and will keep running and come to an end only when the system is shut down.
8. **ASP.NET Web Service:** It is used to create XML web services.

Solution Explorer Window

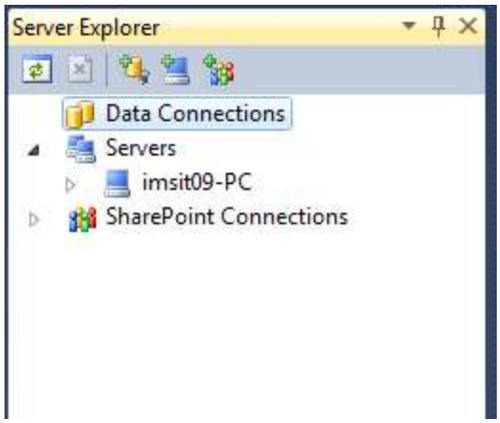
The Solution Explorer window provides an overview of the solution with which we are working and lists all the files in the project as shown below.



NOTES

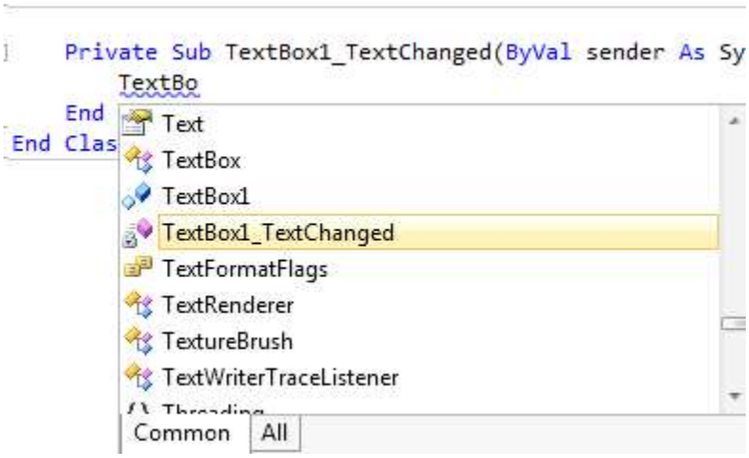
Server Explorer Window

The Server Explorer window is a great tool and it provides drag and drop feature. With the help of server explorer, it is easy to work with databases. If we drag and drop a database table onto a form, VB .NET automatically creates connection and command objects which are required to access the table.



Intellisense

Intellisense is responsible for the boxes that open when we type the code. It provides a list of options which make language references easily accessible. It helps us to find the required information.



Code Designer Window

Code Designer window is used to edit and write code. This window will open, when we double-click on a form or any control. This is the place where we write all the code for the application. The right box allows us to select the part of code that we want to work on and the left box allows us to select the object's code we

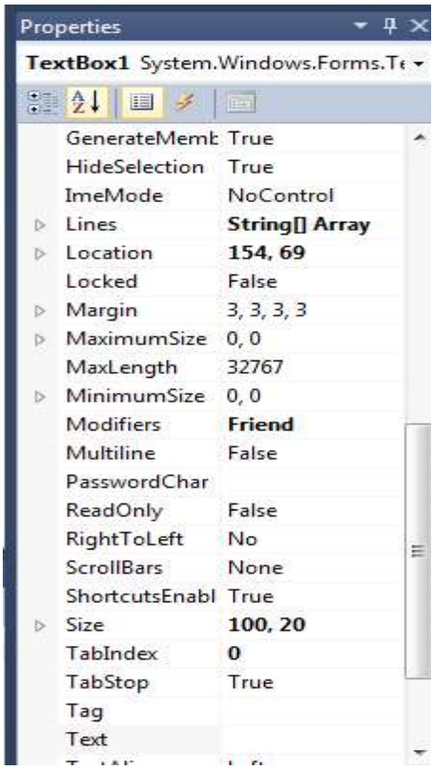
are working with. The "+" and "-" boxes are used to display code that is created in VB.NET.

NOTES

```
TextBox1                                    TextChanged
Public Class Form1

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

    End Sub

    Private Sub TextBox1_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox1.

    End Sub
End Class
```

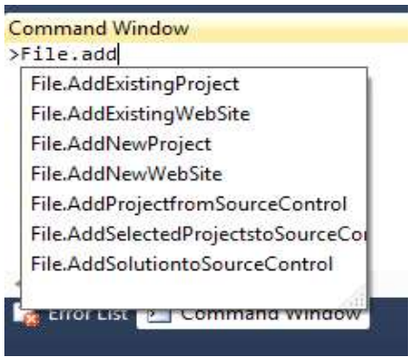
Properties Window

Properties window can be used to set properties for various objects at design time. The properties window can be viewed by pressing F4 on the keyboard or by selecting View'!Properties Window from the main menu. For example to change the name, text, font, font size, color etc. of various controls like textbox, button etc. which can be done easily using the properties window.



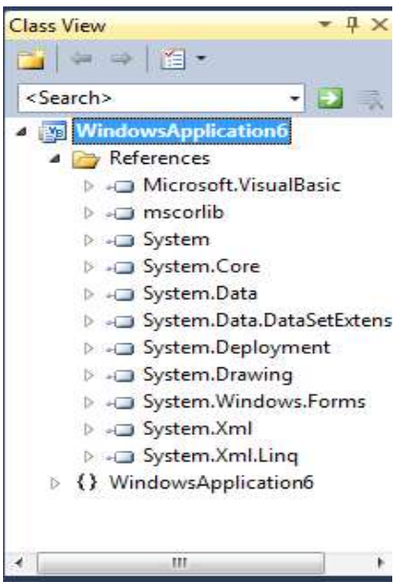
Command Window

You can add new item to the project, add new project and so on using the command window. The command window that is given below displays all possible commands with file. You can view the command window by selecting View'!Other Windows'!Command Window from the main menu.



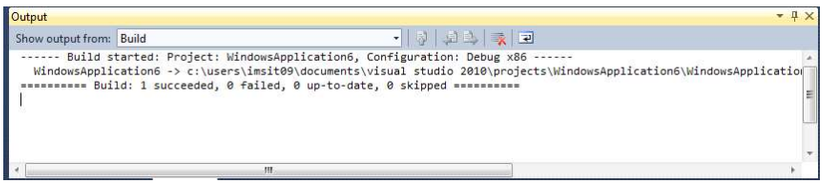
Class View Window

With the help of class view window, you can find a member of a class. The class view window presents projects and solutions in terms of the classes they contain and the members of these classes. The class view can be access through view'!class view. The class view window displayed all the methods and events for the controls which were available on the form.



Output Window

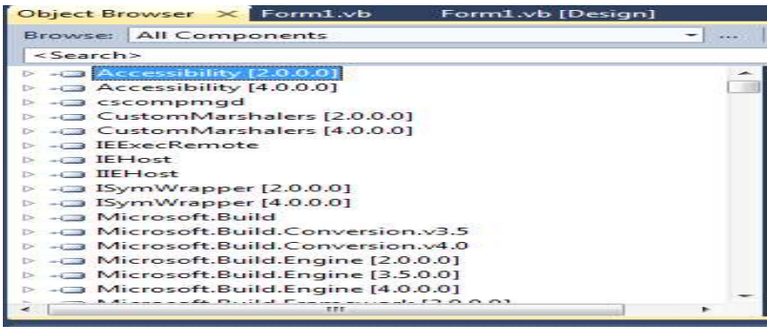
The output window as show below is used to displays the results of building and running applications.



NOTES

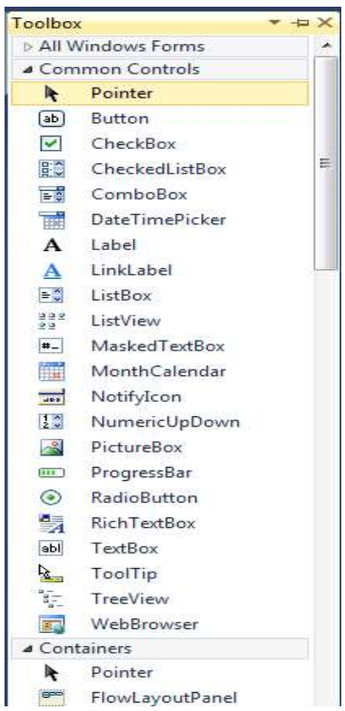
Object Explorer Window

The object explorer window lists all the objects in our code and gives us access to them. You can view the object explorer window by selecting View'!Other Windows'!Object Browser from the main menu. Or it can be access through view'!object browser.



Toolbox Window

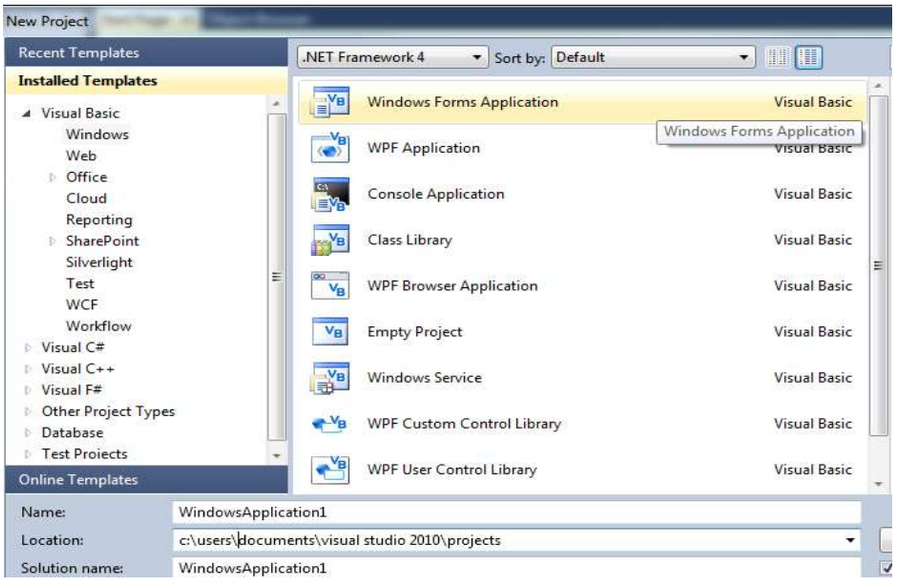
The toolbox window provides access to all components and controls. This window consists of various tabs like components, data, window forms, general etc. Data tab displays tools for creating datasets and making data connections. The Clipboard Ring tab displays recent items stored in the clipboard and allows us to select from them.The Windows Forms tab displays tools for adding controls to forms. The General tab is left empty by default. The Clipboard Ring tab displays recent items stored in the clipboard and allows us to select from them.



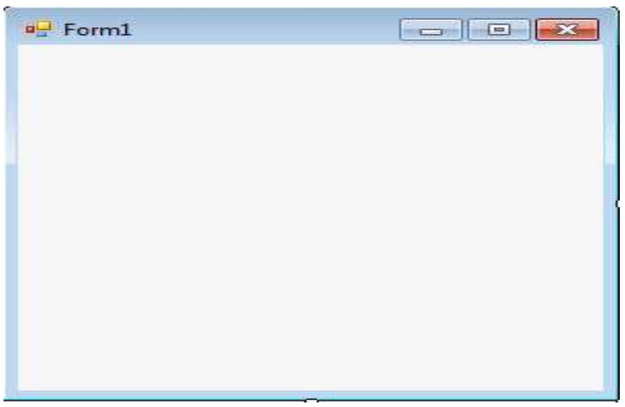
Windows Forms

NOTES

In VB .NET forms are based on the System.Windows.Forms namespace and the form class is System.Windows.Forms.Form. These forms are the base on which we develop and build our entire user interface. The form class is based on the Control class and it allows it to share many methods and properties with other controls. As shown below windows forms, it displays window form application. Once you click OK, then a new Form opens having title, Form1, towards the top-left side of the form. It also consists of close buttons, maximize, minimize towards the top right of the form. The main area of the form in which we work is called the Client Area. It's in this client area in which we design the user interface leaving all the code to the code behind file.



The Figure below shows that how a window form look like.



NOTES

Working with Intrinsic Controls

There are two types of controls in VB i.e. intrinsic and extrinsic. Intrinsic controls are the built-in controls that cannot be changed or removed from the toolbox. You can use them from the toolbox. The Table 1 below lists the intrinsic controls.

Table 1 *The visual basic 6 intrinsic controls*

Controls	Description
Label	It displays the text on a form.
Frame	This control Serves as a *container* to other controls
CheckBox	It enables the users to select or deselect from an option.
ComboBox	This control allows the users to add a new value or select from a list of items.
HscrollBar	It allows scrolling horizontally from a list of data in another control.
Timer	It allows the program to perform actions in real time, without user interaction.
DirListBox	It enables to select a directory or folder.
Shape	Reflect a shape on a form.
Image	It displays images on a form.
OLE Container	It enables you to add the functionality of another Control program to your program.
PictureBox	It can serve as a container and displays images on a form.
TextBox	It is used to display text and also enables users to enter or edit new or existing text.
CommandButton	Used to initiate actions.
OptionButton	It allows users select one choice from a group.
ListBox	It allows users to select from a list of items.
VscrollBar	It helps in scrolling vertically through a list of data in another control.
DriveListBox	Used to select a disk drive.
FileListBox	Selects a file.
Line	Displays a line.
Data	Used to connect to a database.

Adding and Removing Controls

Double-clicking and by drawing are the two ways to add controls on a form. Whenever you double-click an icon on the toolbar, the associated control appears on your form. You can put it wherever you want it, when you draw a control on your form. Following are the steps to draw a control on a form.

1. Click on the control's toolbox icon.
2. Whenever you move the mouse on your form the pointer shapes as crosshair instead of an arrow. Now click and hold the button at the position where you want the control to go.
3. Drag the mouse based on your choice.

4. When the control is in the proper size, let go of the mouse button.

Following are the steps to remove a control from a form.

1. Select the control you want to delete.
2. Press the Delete key.

NOTES

You can also delete a control by right-clicking from the context menu that appears and select delete.

Data Types in VB .NET

There are various data types in VB .NET based on their type and size as shown below.

Data Type	Size in Bytes	Description
Boolean	A Boolean type depends on the implementing platform	True or False
Byte	1 byte	Byte Range start from 0 to 255 (unsigned)
Char	2 bytes	Char Range start from 0 to 65535 (unsigned)
Date	8 bytes	Date range can be 0:00:0 (midnight) January 1, 0001 to 11:5959 PM of December 31, 9999.
Integer	4 bytes	-2,147,483,648 to 2,147,483,647 (signed)
Long	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 (9.2…E + 18) (signed)
Object	Object size based on the platform such as 4 bytes in 32-bit and 8 bytes in 64-bit platform	It can store any type of data defined in a variable of type Object
SByte	1 byte	-128 to 127 (signed)
Short	2 bytes	-32,768 to 32,767 (signed)
Single	4 bytes	-3.4028235E + 38 to -1.401298E-45 for negative values; And for positive value: 1.401298E-45 to 3.4028235E + 38.
String	String Datatype depend on the implementing platform	It accepts Unicode character from 0 to approximately 2 billion characters.
Decimal	16 bytes	Range from 0 to +/- 79,228,162,514,264,337,593,543,950,335 (+/-7.9…E+28) without any decimal point; And 0 to +/-7.9228162514226433759354395033 5 with 28 position to the right of the decimal
Double	8 bytes	-1.79769313486231570E+308 to -4.94-65645841246544E-324 for negative values; 4.94065645841246544E-324 to 1.79769313486231570E+308, for positive values

Access Specifiers

Access specifiers are used to specify how a variable, method, class can be used. Some of the access specifiers are given below:

- **Public:** It provides a variable public access, i.e. there is no restriction on their accessibility.
- **Private:** It provides a variable private access, i.e. they are accessible only within their declaration content

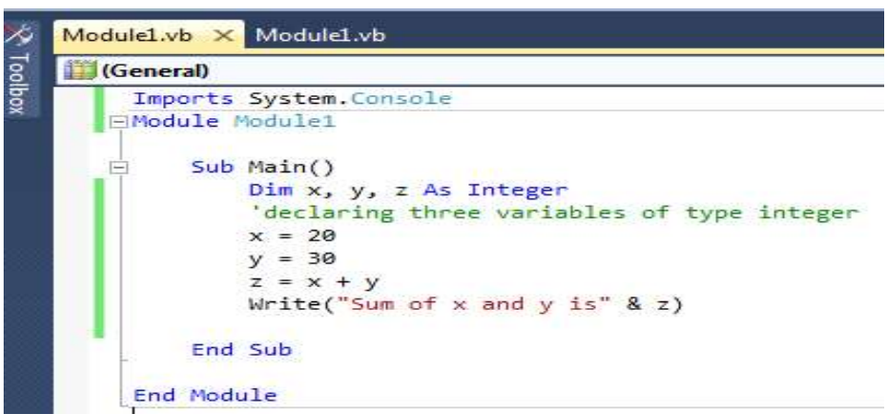
NOTES

- **Protected:** It provides a variable accessibility within their own class or a class derived from that class.
- **Friend:** It provides a variable friend access i.e. they are accessible within the program that contains their declaration.
- **Protected Friend:** It provides a variable both protected and friend access.
- **Static:** It makes a variable static i.e. the variable will hold the value even the procedure in which there declaration ends.
- **Shared:** It means a variable can be shared across many instances and not associated with a specific instance of a class or structure.
- **ReadOnly:** It makes a variable only to be read and cannot be written.

Variables

Variables are used to store data and each variable has a name. VB.NET needs variables to be declared before using them. Variables are declared with the Dim keyword. Dim stands for Dimension. For example:

```
Imports System.Console
Module Module1
Sub Main()
Dim a,b,c as Integer
'declaring three variables of type integer
x=20
y=30
z=x+y
Write("Sum of x and y is" & z)
End Sub
End Module
```



The output of the above code is shown below:

NOTES

Arrays

Arrays are the collection of variables of similar data types. Arrays are programming constructs that store data and allow us to access them by numeric index or subscript. Arrays in Visual Basic.NET inherit from the Array class in the system namespace. Arrays help us create shorter and simpler code in many situations. All arrays in VB are zero based i.e. index of the first element is zero and they are numbered sequentially. The number of array elements must be specified by indicating the upper bound of the array. The upper bound is the number that indicates the index of the last element of the array. An array can have one dimension (linear arrays) or more than one (multidimensional arrays). Arrays are declared using Dim, ReDim, static, private, public and protected keywords. The dimensionality of an array refers to the number of subscripts used to identify an individual element. In visual basic, we can specify up to 32 dimensions. Arrays don't have fixed size in visual basic. Consider an example given below:

```
Imports System.Console
Module Module1
Sub Main()
Dim fruit(5) As String
'declaring an array
fruit(0) = "Apple"
fruit (1) = "Banana"
fruit (2) = "Orange"
fruit (3) = "kiwi"
fruit (4) = "Guaua"
fruit (5) = "Pomegranate"
'storing values in the array
WriteLine("Name of the Fruit in the second location" & "
" & fruit(2))
'displaying value from array
End Sub
End Module
```

NOTES

```
Module1.vb  Module1.vb
(General)                                   (Declarations)
Imports System.Console
Module Module1
    Sub Main()
        Dim fruit(5) As String
        'declaring an array
        fruit(0) = "Apple"
        fruit(1) = "Banana"
        fruit(2) = "Orange"
        fruit(3) = "kiwi"
        fruit(4) = "Guaua"
        fruit(5) = "Pomegranate"
        'storing values in the array
        WriteLine("Name of the Fruit in the second location" & " " & fruit(2))
        'displaying value from array
    End Sub

End Module
```

The output of the above code is given below:

Reinitializing Arrays

We can change the size of an array after creating them. You can use ReDim statement to change the number of elements in an array. The ReDim statement assigns a completely new array object to the specified array variable. The following lines of code demonstrate that the code reinitializes the Test array declared above.

```
Dim Test(15) as Integer
ReDimTest(20) as Integer
'Reinitializing the array
```

When using the Redim statement all the data contained in the array gets lost. If you want to preserve existing data when reinitializing an array, then you should use the Preserve keyword which is given below:

```
Dim Test() as Integer={2,4,6}
'declares an array an initializes it with three members
ReDim Preserve Test(20)
'resizes the array
```

Multidimensional Arrays

All arrays which were mentioned above are one dimensional or linear array. There are two kinds of multidimensional arrays supported by the .NET framework i.e. rectangular arrays and jagged arrays.

NOTES

Rectangular arrays

Rectangular arrays are arrays in which each member of each dimension is extended in each other dimension by the same length. We declare a rectangular array by specifying additional dimensions at declaration. The following lines of code demonstrate the declaration of a multidimensional array.

```
Dim rectArray(4, 2) As Integer
'declares an array of 5 by 3 members which is a 15 member array
Dim rectArray(,) As Integer = {{2, 1, 4}, {5, 7, 9}, {12, 10, 14}}
'setting initial values
```

Jagged Arrays

Jagged Array is also multidimensional array in which the length of each array can differ. This array can be used is to create a table in which the number of columns differ in each row. Say, if row1 has 3 columns, row2 has 3 columns then row3 can have 4 columns, row4 can have 5 columns and so on. The following code demonstrates the concept of jagged arrays.

```
Dim fruit(2)() as String
'declaring an array of 3 arrays
fruit(0)=New String(){"apple","banana","orange"}
'initializing the first array to 3 members and setting values
fruit(1)=New  String(){"kiwi","Pomegranate,"guaua","banana"}
'initializing the second array to 4 members and setting values
fruit(2)=New  String(){"apple","banana","kiwi","guaua","orange"}
'initializing the third array to 5 members and setting values
```

Methods

A Method is a procedure which is built into the class. Methods are series of statements which are executed when called. Methods allow us to handle code in an organized fashion. There are two types of methods in VB.NET i.e. those that return a value (called functions) and those that do not return a value (Sub Procedures). Both of them are discussed below.

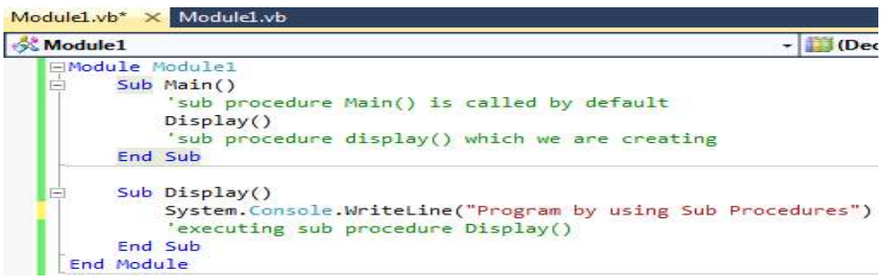
Sub Procedures

Sub procedures are methods that do not return a value. Sub Main (), the starting point of the program itself is a sub procedure. Every time when the Sub procedure is called the statements within it are executed until the End Sub is encountered.

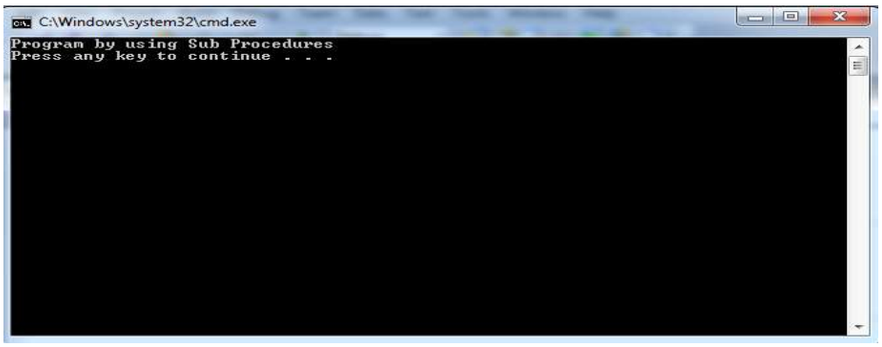
NOTES

The control is transferred to Main Sub procedure automatically which is called by default when the application starts execution. Consider the example given below:

```
Module Module1
Sub Main()
'sub procedure Main() is called by default
Display()
'sub procedure display() which we are creating
End Sub

Sub Display()
System.Console.WriteLine("Program by using Sub
Procedures")
'executing sub procedure Display()
End Sub
End Module
```



The output of the above code is given below:

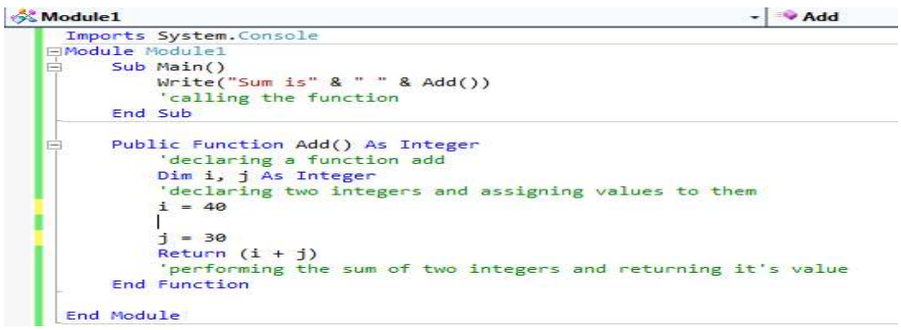


Functions

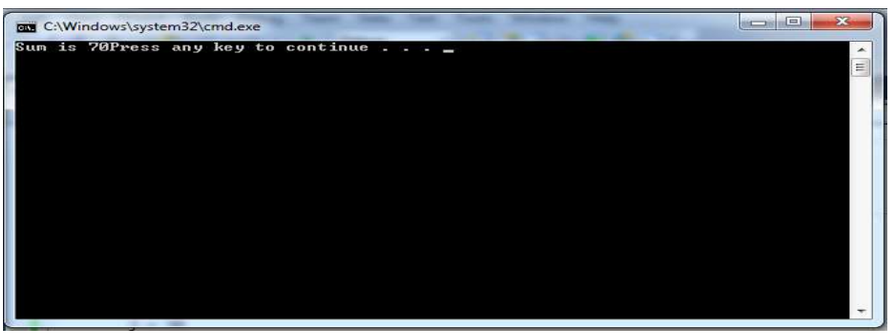
Function is a method that returns a value. Functions are used to evaluate, calculate and transform data. Declaring a function is similar to declaring a sub procedure. Functions are declared with the Function keyword. Consider the following example code:

NOTES

```
Imports System.Console
Module Module1
Sub Main()
Write("Sum is" & " " & Add())
 'calling the function
End Sub

Public Function Add() As Integer
'declaring a function add
Dim i, j As Integer
'declaring two integers and assigning values to them
i = 40
j = 30
Return (i + j)
'performing the sum of two integers and returning it's
value
End Function

End Module
```



The output from above code is given below.



BLOCK 2

NOTES

This block will cover the following topics:

1. Working with forms and dialogs.
2. Working with menus, data controls and common dialogs.

Working with Forms

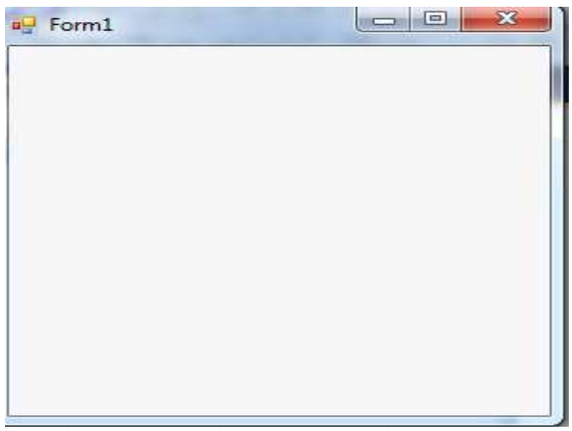
Before starting to work with form, you must know about the properties of window form.The default properties of the form can be found by selecting View'!Properties Window or by pressing F4 on the keyboard. Some of the properties are:

- **Appearance:** Appearance is used to make changes to the appearance of the form like background color, background image etc.
- **Layout:** With the help of layout, we can set the location of the form, maximize, minimize size of the form.
- **Behavior:** The behavior property is used to enable or disable the form by setting the property to True/False.
- **Form Event:** The default event of a form is to load event which looks like this in code given below:

```
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs)_ Handles MyBase.Load End
Sub
```

You can write code in the load event of the form just like you write for all other controls. An example is given below:

You can run the Form by pressing F5 on the keyboard or by selecting Debug'!Start from the main menu. When you run a blank form with no controls on it then nothing is displayed.



Now, add a TextBox and a Button to the form from the toolbox. After adding the TextBox and Button, you can run the program. The output window

displays a TextBox and a Button. But when you click the Button nothing happens. So to do the event for the button, get back to design view and double-click on the button.

```
Public Class Form1
Inherits System.Windows.Forms.Form
#Region " Windows Form Designer generated code "
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs)_ Handles MyBase.Load
End Sub
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As _ System.EventArgs) Handles Button1.Click
End Sub
End Class
```

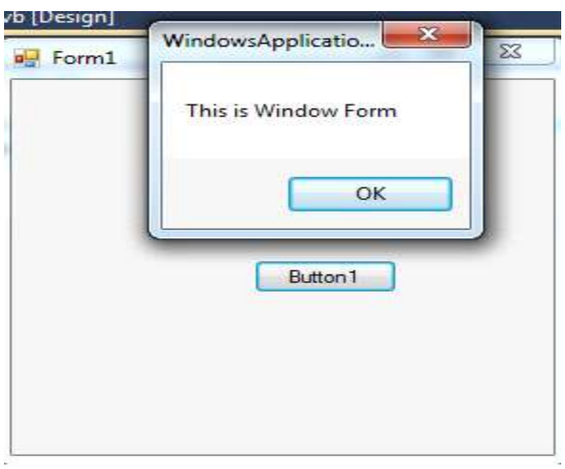
You can write the code TextBox1.Text=" This is Window Form " in the Click event of the Button and run the application. When you click the button the output " This is Window Form " is displayed in the TextBox.

Another way is that you can also use the MessageBox functions to display text when you click on the Button. So for that place a Button on the form and double-click on that to open its event. Write this line of code, MsgBox ("This is Window Form ").

It looks like the given below in the code.

```
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As _ System.EventArgs) Handles Button1.Click
MsgBox("This is Window Form ")
End Sub
```

When, you run the form and click the Button, a small message box displays, "Welcome to Forms". The output is given below:

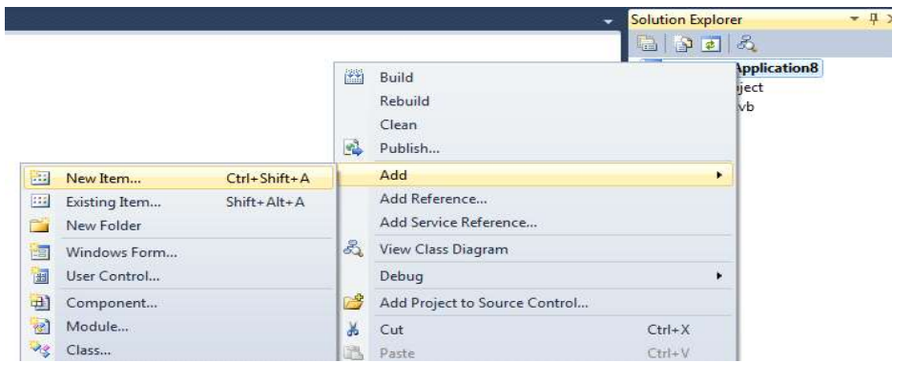


NOTES

NOTES

Adding a New Form to the Project

You can add a new form to the existing project. For adding a new form, with the solution explorer, just right-click on the project name in solution explorer and select Add'!Add Windows Form. After adding a new form, you need to set the new form as Startup object. To do that, right-click on the project name in solution explorer window and select properties which displays the Property Pages window. On this window click the drop-down box which is labeled as Startup Object. This will displays all the forms available in the project.



You can select the form which you want to be displayed, when you run the application and click Apply. So, when you run the application, the form you assigned as Startup object will be displayed.

Working with Multiple Forms

In visual Basic .NET, we can work with multiple forms. For example, take three forms in your application Form1, Form2 and Form3. Now drag a buttons form the toolbox on Form1 andForm2. Now, we want to open Form2 when a button on the Form1 is clicked and when we clicked the button on Form2, Form3 will displayed. Double click on Button1 on Form1 and place the code given below in the click event of the button. The code for that is given below:

```
Public Class Form1
    Dim F2 As New Form2
    'creating a reference to Form2
   Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs)Handles Button1.Click
        F2.Show()
    End Sub
End Class
```

NOTES

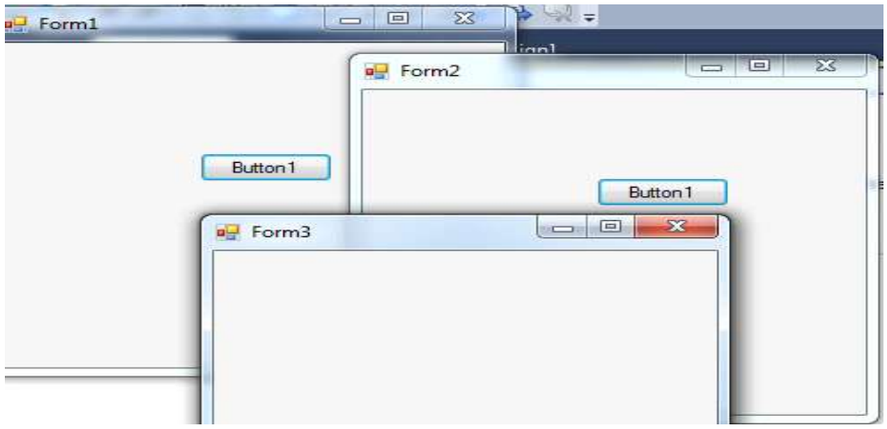
```
Public Class Form1

    Dim F2 As New Form2
    'creating a reference to Form2
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        F2.Show()
    End Sub
End Class
```

After that, Double click on Button1 on Form2 and place the code given below in the click event of the button.

```
Public Class Form2
     Dim F3 As New Form3
    'creating a reference to Form3
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        F3.Show()
    End Sub
End Class
```

```
Public Class Form2

    Dim F3 As New Form3
    'creating a reference to Form3
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        F3.Show()
    End Sub
End Class
```

The output of the above code is given below:



VB.NET Dialog Box

A dialog box is a temporary window that accepts user response with the help of keyboard or mouse to open, save a file, for alert messages, print, color etc. VB.NET dialog box is used to create interaction between the user and the application. The dialog box appears in a form when the program needs to interact with users, like an alert message, when an error occurs, when the program requires immediate

NOTES

action, acknowledgment from the user etc. VB.NET Dialog box inherits the CommonDialog class and overrides the RunDialog**()** method of the base class which is used to create the PrintDialogbox, Font Dialog box, OpenFileDialog box, Color. When the dialog box calls its ShowDialog**()** method, the RunDialog**()** method is automatically called in a window form.

There are various **ShowDialog() method** in the Windows Form.

- **OK:** It returns a DialogResult.OK, when the user clicks the OK button of the Dialog box.
- **Ignore:** It is used when a user clicks on the Ignore button to return the DialogResult.Ignore.
- **Abort:** It is used when a user clicks on the Abort button to return the DialogResult.Abort value.
- **Cancel:** It returns DialogResult.Cancel, when a user clicks on the Cancel button of the Dialog Box.
- **No:** It returns DialogResult.No, when a user clicks on the No button of the Dialog box.
- **None:** It is used to return nothing when the user clicks on the None button, and the dialog box is continued running.
- **Yes:** It returns DialogResult.Yes, when a user clicks the Yes button of the dialog box.
- **Retry:** It returns a DialogResult.Retry, when a user clicks on the Dialog Box Retry button.

There are various types of commonly used dialog box controls in the VB.NET that are given below:

1. **Color Dialog Box:** It allows the user to select a color from the predefined colors or specify the custom colors.
2. **OpenFile Dialog Box:** It allows the users to select a file to open and allows the selection of multiple files.
3. **Print Dialog Box:** It allows the user to print documents by selecting the printer and setting of the page printed through the Windows application.
4. **Font DialogBox:** It allows the user to select the font size, font, style and color to be applied to the current text selection.

Consider an example given below:

Dialog.vb

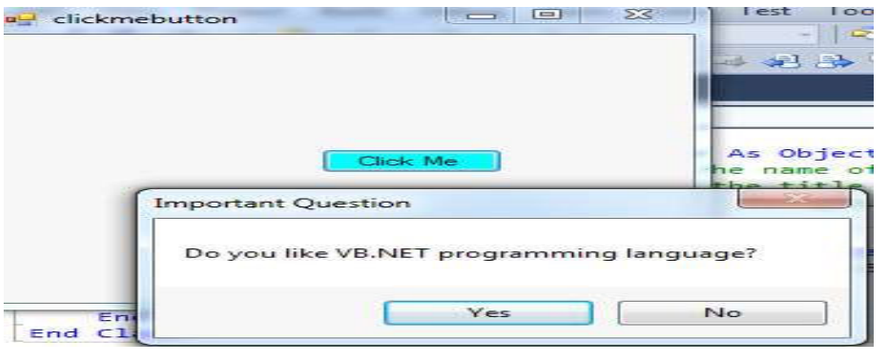
```
Public Class Dialog
 Private Sub Dialog_Load(sender As Object, e As E
ventArgs) Handles MyBase.Load        Button1.Text = "Click Me" 'Set the name of button
       Me.Text = "clickmebutton" ' Set the title name for the Windows Form
       Button1.BackColor = Color.Aqua ' Background color of the button
```

```
    End Sub
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
Dim result1 As DialogResult = MessageBox.Show("Do you like
VB.NET programming language?", "Important Question", MessageBoxButtons.YesNo)
End Sub
End Class
```

NOTES

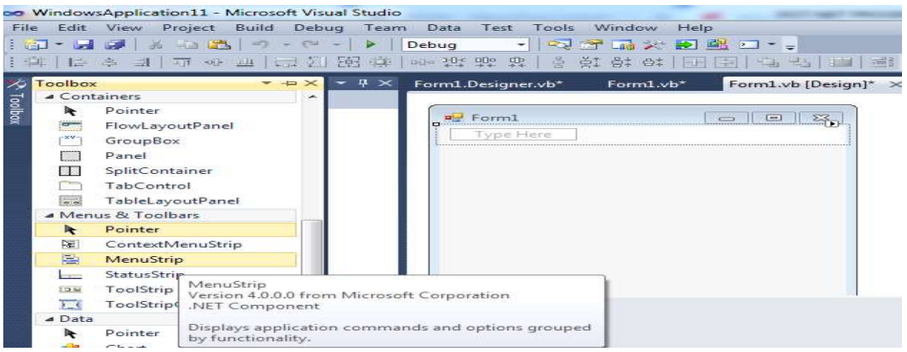
```
Form1.vb | Form1.vb [Design]
(General) | (Declarations)
Public Class Dialog
    Private Sub Dialog_Load(ByVal sender As Object, ByVal e As EventArgs) Handles MyBase.Load
        Button1.Text = "Click Me" 'Set the name of button
        Me.Text = "clickmebutton" ' Set the title name for the Windows Form
        Button1.BackColor = Color.Aqua  ' Background color of the button
    End Sub

    Private Sub Button1_Click(ByVal sender As Object, ByVal e As EventArgs) Handles Button1.Click
        Dim result1 As DialogResult = MessageBox.Show("Do you like VB.NET programming language?",
    "Important Question",
    MessageBoxButtons.YesNo)
    End Sub
End Class
```

After clicking on Click Me button, the output produced is shown below:



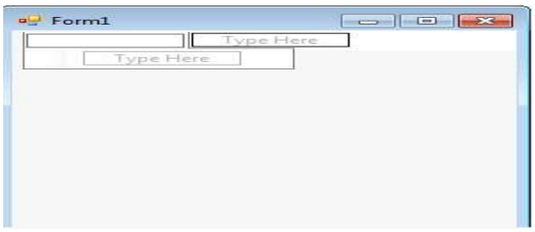
VB.NET Menu Control

A menu is located on the menu bar. It consists of a list of various commands. Menus are made of MenuItem objects that represent individual parts of a menu. MainMenu is the container for the Menu structure of the form. By using the MainMenu control, you can create a main menu object on your form. The figure given below shows the dragging of Menustrip Object to the Form.

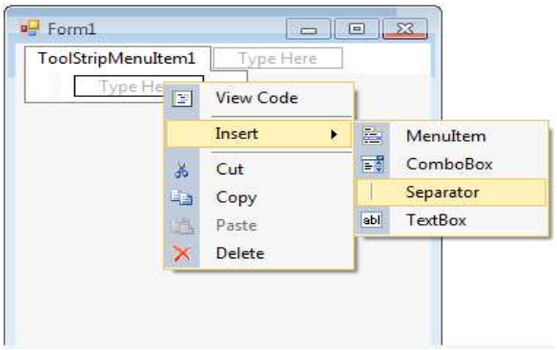


NOTES

After dragging, Menustrip control on the form, you can create menu items by typing a value into the "Type Here" box on the menubar as shown below.

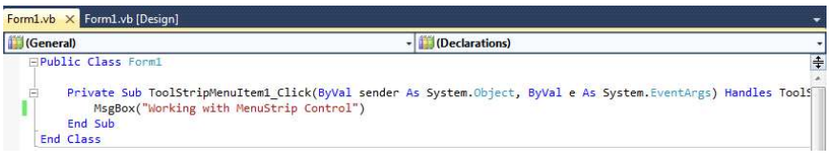


To create a separator bar, just right click on menu and go to insert'!Separator.

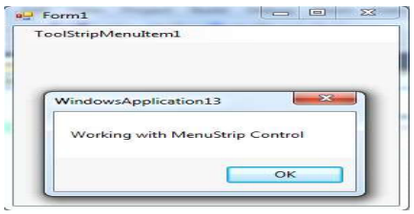


After doing the above steps, double click on each menu item and write the code. When clicking a menu item, the program shows a messagebox as shown below.

```
Public Class Form1
Private Sub ToolStripMenuItem1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ToolStripMenuItem1.Click
        MsgBox("Working with MenuStrip Control")
    End Sub
End Class
```



The output of the above code is shown below:



Application with Data Controls

1. Write a program to demonstrate the implementation of various data controls in ASP.NET using VB.

NOTES

Step 1: Create a ASP.NET based web project using following steps:

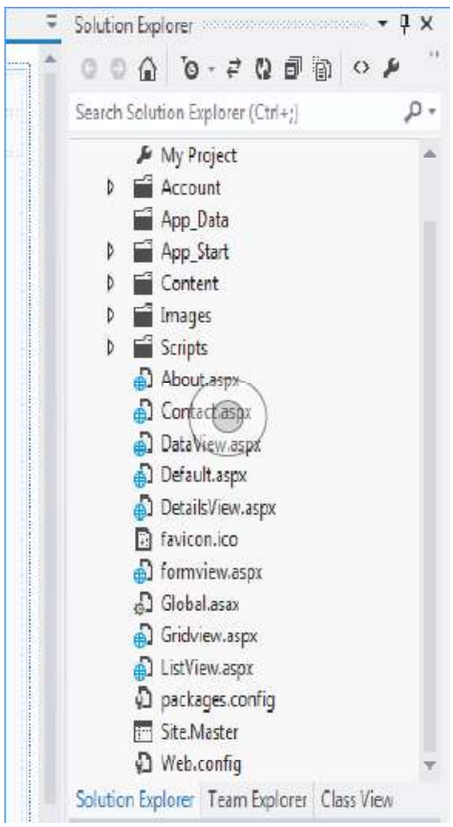
File→New→Project→ASP→NET Empty Web

Application→DataControls→Ok.

Step 2: Right click on data controls in Solution Explorer, Add Windows Forms, name the form as:

- DataView.aspx
- formView.aspx
- GridView.aspx
- ListView.aspx

As shown in Solution Explorer below:



Step 3: Open each data control form to implement their functionality as shown below:

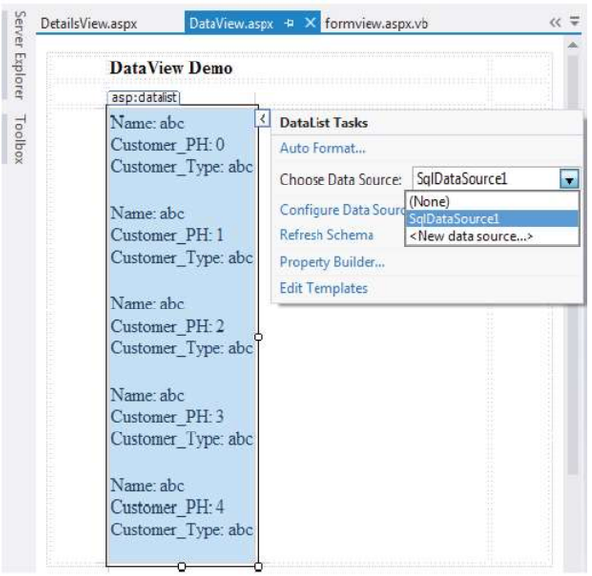
a. DataView.aspx

NOTES

Step I: From Menu, Click on Table and then click Insert Table as per the requirements:

Step II: From ToolBox select and insert into any of the table cell or anywhere on the form the DataView Control.sb

Step III: Click onr arrow sign appears after you click over DataView Control in DataView.aspx and click on choose data source to link with this DataView Control as shown below:



Step IV: Select datasource from the database you have created using (say) SQLServer. Follow the steps as and when prompted to fullfil connection with desired data source in database.

Step V: Here in this case the database selected contains five rows with three columns as shown in figure above:

Step VI: Before you build the project you need to specify the server to host the project. In order to does that click on project from Menu bar, click DataControl properties option.

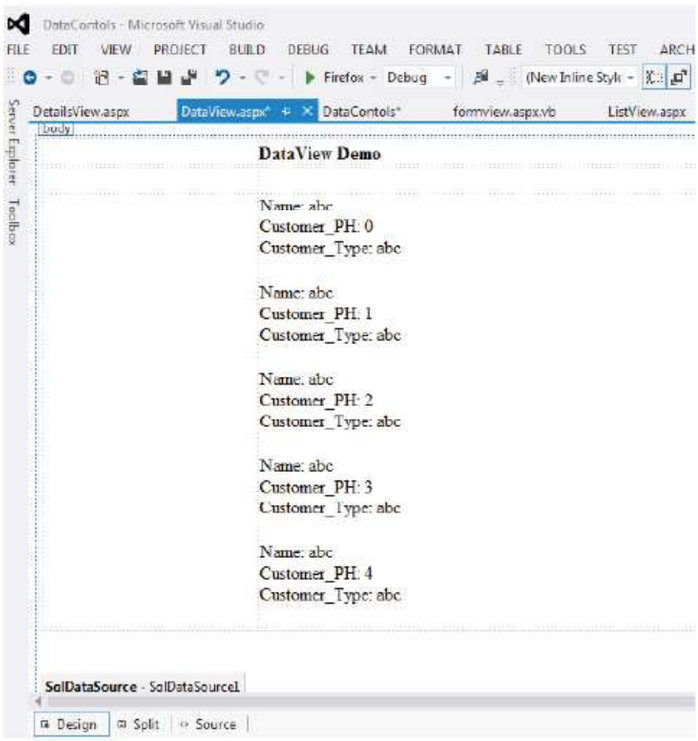
Step VII: Click on Web from the options displayed on the left side of the form displayed. Go to Start Action tab, choose specific page to start your project to run. From Servers tab select "Use Visiual Studio Development Server", check "Auto-Assign Port".

Step VIII: Build the project.

Step IX: Specify the browser to display the outcome of the project.

Step X: If project builds without errors the resultant display will be loaded into a browser specified by the programmer say FireFox in this case.

The Design of DataView.aspx will look like the figure shown below:



DataView.aspx source code will look like as given below:

```
<style type="text/css">
    .auto-style1 {
        width: 100%;
        height: 378px;
    }
    .auto-style2 {
        width: 251px;
    }
    .auto-style3 {
        width: 86px;
    }
    .auto-style4 {
        width: 67px;
    }
</style>
<p>
    <table class="auto-style1">
        <tr>
```

NOTES

NOTES

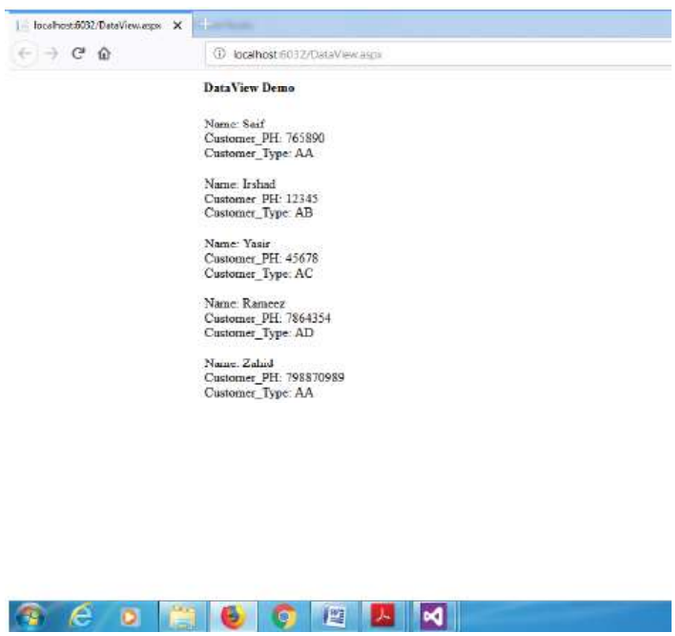
```
            <td class="auto-style4"> </td>
            <td class="auto-style2"><strong>DataView Demo</
strong></td>
            <td class="auto-style3"> </td>
        </tr>
        <tr>
            <td class="auto-style4"> </td>
            <td class="auto-style2"> </td>
            <td class="auto-style3"> </td>
        </tr>
        <tr>
            <td class="auto-style4"> </td>
            <td class="auto-style2"><asp:datalist
runat="server" DataSourceID="SqlDataSource1">
                <ItemTemplate>
                    Name:
                    <asp:Label ID="NameLabel" runat="server"
Text='<%# Eval("Name") %>' />
                    <br />
                    Customer_PH:
                    <asp:Label ID="Customer_PHLabel"
runat="server" Text='<%# Eval("Customer_PH") %>' />
                    <br />
                    Customer_Type:
                    <asp:Label ID="Customer_TypeLabel"
runat="server" Text='<%# Eval("Customer_Type") %>' />
                    <br />
<br />
                </ItemTemplate>
                </asp:datalist>
            </td>
            <td class="auto-style3"> </td>
        </tr>
    </table>
    <br />
</p>
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%$
ConnectionStrings:CustomerDetailConnectionString %>"
SelectCommand="SELECT * FROM [Cust_Det]"></
asp:SqlDataSource>
<%@ Page Language="vb" AutoEventWireup="false"
CodeBehind="DataView.aspx.vb"
```

```
Inherits="DataContols.DataView" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

    </div>
    </form>
</body>
</html>
```

DataView.aspx.vb

```
'Program to implement DataView Control in ASP.NET
Public Class DataView
    Inherits System.Web.UI.Page
    Protected Sub Page_Load(ByVal sender As Object, ByVal e
As System.EventArgs) Handles Me.Load
    End Sub
End Class
```

Step 4: After successful build and start the output window obtained is shown in figure below:



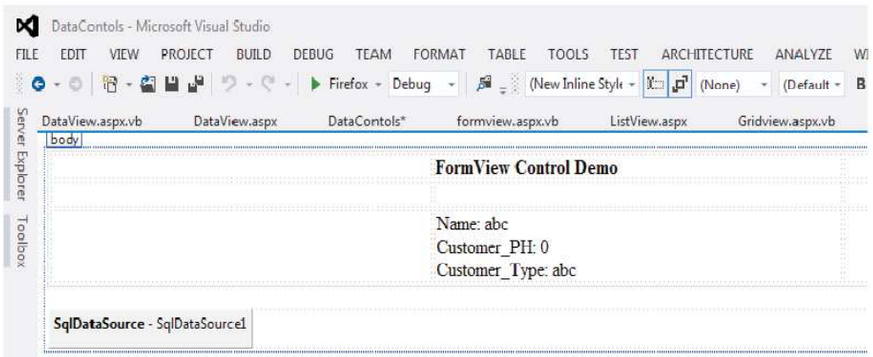
Similarly the other DataControls can be used to implement their functionality in your ASP.NET web project.

2. Write a program to use FormView data control.

NOTES

Step 1: Follow similar steps as discussed above for DataView Control. However, instead of DataView Control you need to Use FormView Control from toolbox.

Step 2: Choose data source for FormView Control and design the windows from "formview.aspx" as shown below:



Source code of formview.aspx is given below:

formview.aspx

```
<%@ Page Language="vb" AutoEventWireup="false"
CodeBehind="formview.aspx.vb"
Inherits="DataContols.formview" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title></title>
<style type="text/css">
    .auto-style1 {
        width: 100%;
    }
    .auto-style2 {
        width: 368px;
    }
</style>
    </head>
<body>
    <form id="form1" runat="server">
        <div>
<table class="auto-style1">
    <tr>
```

```
      <td> </td>
      <td class="auto-style2"><strong>FormView Control
Demo</strong></td>
      <td> </td>
   </tr>
   <tr>
      <td> </td>
      <td class="auto-style2"> </td>
      <td> </td>
   </tr>
   <tr>
      <td> </td>
      <td class="auto-style2">
         <asp:FormView ID="FormView1" runat="server"
DataSourceID="SqlDataSource1" Height="78px">
            <EditItemTemplate>
               Name:
               <asp:TextBox ID="NameTextBox"
runat="server" Text='<%# Bind("Name") %>' />
               <br />
               Customer_PH:
               <asp:TextBox ID="Customer_PHTextBox"
runat="server" Text='<%# Bind("Customer_PH") %>' />
               <br />
               Customer_Type:
               <asp:TextBox ID="Customer_TypeTextBox"
runat="server" Text='<%# Bind("Customer_Type") %>' />
               <br />
               <asp:LinkButton ID="UpdateButton"
runat="server" CausesValidation="True" CommandName="Update"
Text="Update" />
                <asp:LinkButton
ID="UpdateCancelButton"       runat="server"
CausesValidation="False" CommandName="Cancel" Text="Cancel"
/>
            </EditItemTemplate>
            <InsertItemTemplate>
               Name:
               <asp:TextBox    ID="NameTextBox"
runat="server" Text='<%# Bind("Name") %>' />
               <br />
               Customer_PH:
               <asp:TextBox ID="Customer_PHTextBox"
```

NOTES

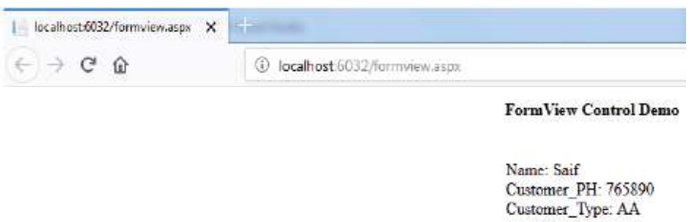
NOTES

```
runat="server" Text='<%# Bind("Customer_PH") %>' />
                <br />
                Customer_Type:
                <asp:TextBox ID="Customer_TypeTextBox"
runat="server" Text='<%# Bind("Customer_Type") %>' />
                <br />
                <asp:LinkButton ID="InsertButton"
runat="server" CausesValidation="True" CommandName="Insert"
Text="Insert" />
                 <asp:LinkButton
ID="InsertCancelButton"       runat="server"
CausesValidation="False" CommandName="Cancel" Text="Cancel"
/>
            </InsertItemTemplate>
            <ItemTemplate>
                Name:
                <asp:Label ID="NameLabel" runat="server"
Text='<%# Bind("Name") %>' />
                <br />
                Customer_PH:
                <asp:Label ID="Customer_PHLabel"
runat="server" Text='<%# Bind("Customer_PH") %>' />
                <br />
                Customer_Type:
                <asp:Label ID="Customer_TypeLabel"
runat="server" Text='<%# Bind("Customer_Type") %>' />
                <br />
            </ItemTemplate>
        </asp:FormView>
        <asp:SqlDataSource ID="SqlDataSource1"
runat="server"         ConnectionString="<%$
ConnectionStrings:CustomerDetailConnectionString %>"
SelectCommand="SELECT * FROM [Cust_Det]"></
asp:SqlDataSource>
      </td>
      <td> </td>
   </tr>
</table>
</div>
      </form>
   </body>
</html>
```

Code behind formview.aspx that is formview.aspx.vb is given below:

```
'Implementation of FormView DataControl in asp.net using
VB
Public Class formview
    Inherits System.Web.UI.Page
    Protected Sub Page_Load(ByVal sender As Object, ByVal
e As System.EventArgs) Handles Me.Load
    End Sub
    Protected Sub SqlDataSource1_Selecting(sender As
Object, e As SqlDataSourceSelectingEventArgs) Handles
SqlDataSource1.Selecting
    End Sub
End Class
```

Step 3: Build and run the project. The output generated is shown in figure below:

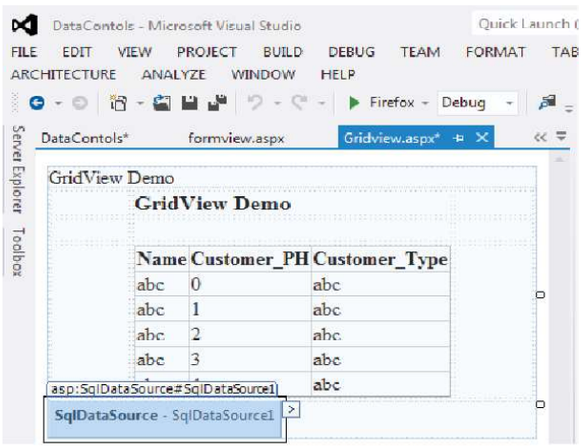


Note: FormView data control displays only a single row retrieved from the linked data source into the browser window as shown above.

3. Write a program to use GridView data control.

Step 1: Follow similar steps as discussed above for DataView Control. However, instead of DataView Control you need to Use GridViewControl from toolbox.

Step 2: Choose data source for to be linked with GridView Control and design the windows from "Gridview.aspx" as shown below:



NOTES

Source code of Gridview.aspx is given below:

```
'Gridview.aspx
<%@ Page Language="vb" AutoEventWireup="false"
C o d e B e h i n d = " G r i d v i e w . a s p x . v b "
Inherits="DataContols.Gridview" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <style type="text/css">
        .auto-style1 {
            width: 100%;
        }
        .auto-style2 {
            width: 206px;
        }
    </style>
</head>
<body>
    <form id="form1" runat="server">
    <div >

        <asp:Label ID="Label1"  runat="server"
Text="GridView Demo" ></asp:Label>

        <br />
        <table class="auto-style1">
            <tr>
                <td> </td>
                <td class="auto-style2"><strong ><h3>GridView
Demo</h3>
                    </strong></td>
                <td> </td>
            </tr>
            <tr>
                <td> </td>
                <td class="auto-style2"> </td>
                <td> </td>
            </tr>
            <tr>
                <td> </td>
```

```
            <td class="auto-style2">
                <asp:GridView    ID="GridView1"
runat="server"  AutoGenerateColumns="False"
DataSourceID="SqlDataSource1">
                 <Columns>
                     <asp:BoundField DataField="Name"
HeaderText="Name" SortExpression="Name" />
                     <asp:BoundField
DataField="Customer_PH" HeaderText="Customer_PH"
SortExpression="Customer_PH" />
                     <asp:BoundField
DataField="Customer_Type" HeaderText="Customer_Type"
SortExpression="Customer_Type" />
                 </Columns>
                </asp:GridView>
            </td>
            <td> </td>
          </tr>
       </table>
       <asp:SqlDataSource ID="SqlDataSource1"
runat="server"        ConnectionString="<%$
ConnectionStrings:CustomerDetailConnectionString %>"
SelectCommand="SELECT * FROM [Cust_Det]"></
asp:SqlDataSource>


    </div>
    </form>
</body>
</html>
```

Code behind Gridview.aspx that is Gridview.aspx.vb is given below:

```
'Gridview.aspx.vb
'Implementation of GridView DataControl in asp.net using
VB
Public Class Gridview
    Inherits System.Web.UI.Page
    Protected Sub Page_Load(ByVal sender As Object, ByVal
e As System.EventArgs) Handles Me.Load
    End Sub
    Protected Sub SqlDataSource1_Selecting(sender As
Object, e As SqlDataSourceSelectingEventArgs) Handles
SqlDataSource1.Selecting
    End Sub
    Protected Sub GridView1_SelectedIndexChanged(sender
```

NOTES

```
As Object, e As EventArgs) Handles
GridView1.SelectedIndexChanged
    End Sub
End Class
```

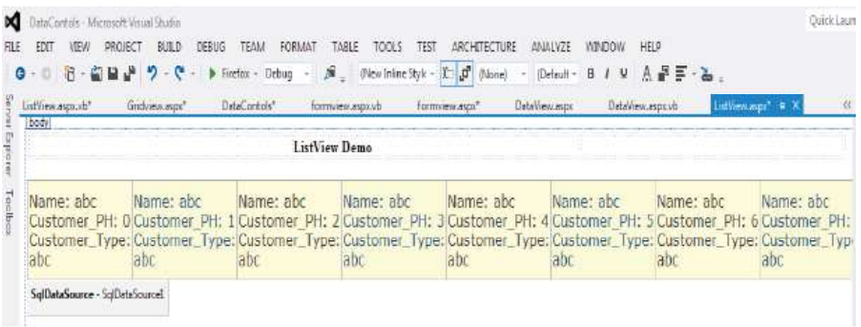
NOTES

Step 3: Build and run the project. The output generated is shown in figure given below:

4. Write a program to use ListView data control.

Step 1: Follow similar steps as discussed above for DataView Control. However, instead of DataView Control you need to Use ListViewControl from toolbox.

Step 2: Choose data source for to be linked with ListView Control and design the windows from "ListView.aspx" as shown below:



Source code of ListView.aspx is given below:

```
'ListView.aspx
<%@ Page Language="vb" AutoEventWireup="false"
CodeBehind="ListView.aspx.vb"
Inherits="DataContols.ListView" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<style type="text/css">
    .auto-style1 {
```

NOTES

```
        width: 100%;
    }
    .auto-style2 {
        width: 374px;
    }
</style>
<p>
    <table class="auto-style1">
        <tr>
            <td> </td>
            <td class="auto-style2"><strong>ListView Demo</
strong></td>
            <td> </td>
        </tr>
    </table>
</p>
<asp:listview runat="server" DataSourceID="SqlDataSource1"
OnSelectedIndexChanged="Unnamed1_SelectedIndexChanged">
    <AlternatingItemTemplate>
        <td runat="server" style="background-color:
#FAFAD2;color: #284775;">Name:
            <asp:Label ID="NameLabel" runat="server"
Text='<%# Eval("Name") %>' />
            <br />
            Customer_PH:
            <asp:Label ID="Customer_PHLabel" runat="server"
Text='<%# Eval("Customer_PH") %>' />
            <br />
            Customer_Type:
            <asp:Label ID="Customer_TypeLabel"
runat="server" Text='<%# Eval("Customer_Type") %>' />
            <br />
        </td>
    </AlternatingItemTemplate>
    <EditItemTemplate>
        <td runat="server" style="background-color:
#FFCC66;color: #000080;">Name:
            <asp:TextBox ID="NameTextBox" runat="server"
Text='<%# Bind("Name") %>' />
            <br />
            Customer_PH:
            <asp:TextBox ID="Customer_PHTextBox"
```

NOTES

```
runat="server" Text='<%# Bind("Customer_PH") %>' />
         <br />
         Customer_Type:
         <asp:TextBox  ID="Customer_TypeTextBox"
runat="server" Text='<%# Bind("Customer_Type") %>' />
         <br />
         <asp:Button ID="UpdateButton" runat="server"
CommandName="Update" Text="Update" />
         <asp:Button ID="CancelButton" runat="server"
CommandName="Cancel" Text="Cancel" />
      </td>
   </EditItemTemplate>
   <EmptyDataTemplate>
      <table style="background-color: #FFFFFF;border-
collapse: collapse;border-color: #999999;border-
style:none;border-width:1px;">
         <tr>
            <td>No data was returned.</td>
         </tr>
      </table>
   </EmptyDataTemplate>
   <InsertItemTemplate>
      <td runat="server" style="">Name:
         <asp:TextBox ID="NameTextBox" runat="server"
Text='<%# Bind("Name") %>' />
         <br />Customer_PH:
         <asp:TextBox  ID="Customer_PHTextBox"
runat="server" Text='<%# Bind("Customer_PH") %>' />
         <br />Customer_Type:
         <asp:TextBox  ID="Customer_TypeTextBox"
runat="server" Text='<%# Bind("Customer_Type") %>' />
         <br />
         <asp:Button ID="InsertButton" runat="server"
CommandName="Insert" Text="Insert" />
         <asp:Button ID="CancelButton" runat="server"
CommandName="Cancel" Text="Clear" />
      </td>
   </InsertItemTemplate>
   <ItemTemplate>
      <td  runat="server"  style="background-color:
#FFFBD6;color: #333333;">Name:
         <asp:Label  ID="NameLabel"  runat="server"
Text='<%# Eval("Name") %>' />
```

```
         <br />
         Customer_PH:
         <asp:Label ID="Customer_PHLabel" runat="server"
Text='<%# Eval("Customer_PH") %>' />
         <br />
         Customer_Type:
         <asp:Label   ID="Customer_TypeLabel"
runat="server" Text='<%# Eval("Customer_Type") %>' />
         <br />
      </td>
   </ItemTemplate>
   <LayoutTemplate>
      <table runat="server" border="1" style="background-
color: #FFFFFF;border-collapse: collapse;border-color:
#999999;border-style:none;border-width:1px;font-family:
Verdana, Arial, Helvetica, sans-serif;">
         <tr    id="itemPlaceholderContainer"
runat="server">
            <td id="itemPlaceholder" runat="server"></
td>
         </tr>
      </table>
      <div style="text-align: center;background-color:
#FFCC66;font-family: Verdana, Arial, Helvetica, sans-
serif;color: #333333;">
      </div>
   </LayoutTemplate>
   <SelectedItemTemplate>
      <td  runat="server"  style="background-color:
#FFCC66;font-weight: bold;color: #000080;">Name:
         <asp:Label  ID="NameLabel"  runat="server"
Text='<%# Eval("Name") %>' />
         <br />
         Customer_PH:
         <asp:Label ID="Customer_PHLabel" runat="server"
Text='<%# Eval("Customer_PH") %>' />
         <br />
         Customer_Type:
         <asp:Label   ID="Customer_TypeLabel"
runat="server" Text='<%# Eval("Customer_Type") %>' />
         <br />
      </td>
   </SelectedItemTemplate>
```

NOTES

NOTES

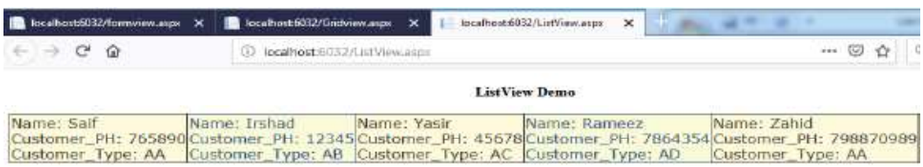
```
</asp:listview>
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%$
ConnectionStrings:CustomerDetailConnectionString %>"
SelectCommand="SELECT * FROM [Cust_Det]"></
asp:SqlDataSource>
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

    </div>
    </form>
</body>
</html>
```

Code behind ListView.aspx that is ListView.aspx.vb is given below:

ListView.aspx.vb

```
'Program to demonestrate the use of ListView DataContol
in asp.net
Public Class ListView
    Inherits System.Web.UI.Page
    Protected Sub Page_Load(ByVal sender As Object, ByVal
e As System.EventArgs) Handles Me.Load
    End Sub
    Protected Sub Unnamed1_SelectedIndexChanged(sender As
Object, e As EventArgs)
    End Sub
End Class
```

Step 3: Build and run the project. The output generated is shown in figure given below:



Common Dialog Controls

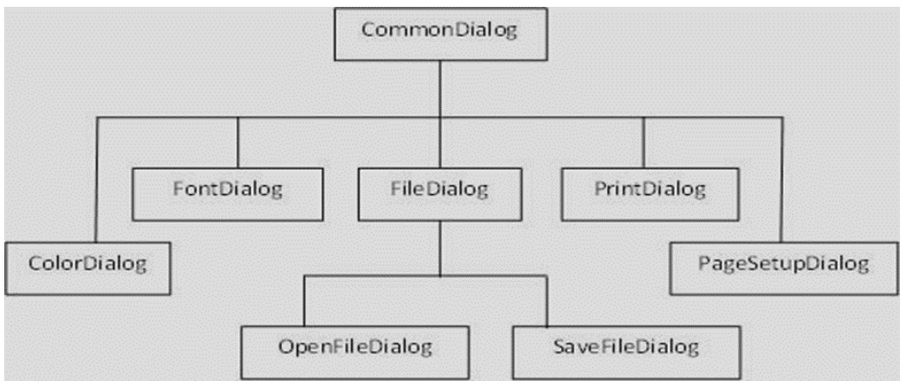
There are various built-in dialog boxes which can be used in Windows forms. These dialog controls are used for various tasks like opening files, saving files,

providing choices for colors, printing a page, page setup, fonts etc. All of these dialog box control classes is inherited from the CommonDialog class and override the *RunDialog()* function of the base class to create the specific dialog box. The RunDialog() function is automatically invoked when a user of a dialog box calls its *ShowDialog()* function.The ShowDialog method is used to display all dialog box controls at run-time. It returns a value of the type of DialogResult enumeration. The values of DialogResult are given below:

NOTES

- **Yes** – when user clicks a Yes button, returns DialogResult.Yes.
- **Abort** – when user clicks an Abort button, returns DialogResult.Abort value.
- **Cancel** – when user clicks a Cancel button, returns DialogResult.Cancel.
- **Ignore** – when user clicks an Ignore button, returns DialogResult.Ignore.
- **No** – when user clicks a No button, returns DialogResult.No.
- **OK** – when user clicks an OK button, returns DialogResult.OK.
- **Retry** – when user clicks a Retry button, returns DialogResult.Retry.
- **None** " returns nothing and the dialog box continues running.

The following diagram shows the inheritance in common dialog class.



All these classes have subsequent controls that could be added from the toolbox during design time. You can include relevant functionality of these classes either by instantiating the class programmatically or by using relevant controls to your application.

When you drag the control onto the form or double click any of the dialog controls in the toolbox, it shows in the component tray at the bottom of the Windows Forms Designer.form. Following are the commonly used dialog box controls.

- *SaveFileDialog*: It allows the user to specify the name of the file to save data.
- *OpenFileDialog*: It allows the user to select a file to open.

NOTES

- *ColorDialog*: It represents a common dialog box that displays available colors along with controls that enable the user to define custom colors.
- *FontDialog*: It prompts the user to choose a font from among those installed on the local computer. It lets the user select color, font size and font size.
- *PrintDialog*: It lets the user to print documents by selecting a printer and choosing which sections of the document to print.

BLOCK 3

NOTES

This block will cover the following topics:

1. Work with drag and drop event, inbuilt functions, mathematical and string functions.
2. Understand the ADO.NET data architecture
3. Create ActiveX controls
4. Active Data Objects (ADO) and OLE DB

Drag and Drop Event

Basically in drag and drop event, it is a pointing device gesture in which the user selects a virtual object by "grabbing" it and dragging it to a different location or onto another virtual object.

Consider an example of drag and drop operation. For this, just create a VB.NET windows application, and then design a form with drag and drop and control & event procedure. To enable drag & drop for text, first you have to place two textboxes and set allow drop property of a second text box to true and after that write the code as given below:

```
Private MouseIsDown As Boolean = False 'variable declaration
Private Sub TextBox1_MouseDown(ByVal sender As Object,
ByVal e As _
System.Windows.Forms.MouseEventArgs)  Handles
TextBox1.MouseDown
'Set a flag to show that the mouse is down.
MouseIsDown = True
End Sub

Private Sub TextBox1_MouseMove(ByVal sender As Object,
ByVal e As _
System.Windows.Forms.MouseEventArgs)  Handles
TextBox1.MouseMove
If MouseIsDown Then
'Initiate dragging.
TextBox1.DoDragDrop(TextBox1.Text,DragDropEffects.Copy)
End If
MouseIsDown = False
End Sub

Private Sub TextBox2_DragEnter(ByVal sender As Object,
ByVal e As _
```

NOTES

```
System.Windows.Forms.DragEventArgs)  Handles
TextBox2.DragEnter
'Check the format of the data being dropped.
If (e.Data.GetDataPresent(DataFormats.Text)) Then
'Display the copy cursor.
e.Effect = DragDropEffects.Copy
Else
'Display the no-drop cursor.
e.Effect = DragDropEffects.None
End If
End Sub

Private Sub TextBox2_DragDrop(ByVal sender As Object,
ByVal e As _
System.Windows.Forms.DragEventArgs)  Handles
TextBox2.DragDrop
'Paste the text.
TextBox2.Text = e.Data.GetData(DataFormats.Text)
End Sub
```

From the above code, it can be seen that the DoDragDrop method is called in the MouseMove event and the MouseDown event is used to set a flag, which shows that the mouse is down. In the MouseMove event, the MouseIsDown flag is set to False. You can handle the drag in the MouseDown event also. Dring this every time a user clicks the control, and then no-drag cursor would be displayed.

The GetDataPresent method checks the format of the data being dragged in case of DragEnter event. In our case it is text, so the Effect property is set to Copy, which in turn displays the copy cursor. The GetData method is used to retrieve the text from the DataObject. In case of DragDrop event it also assigns it to the target TextBox.

The example code given below draggs a different type of data and provides support for both cutting and copying. For these just add two picturebox controls and write the code given below:

```
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As _
System.EventArgs) Handles MyBase.Load
'Enable dropping.
PictureBox2.AllowDrop = True
End Sub

Private Sub PictureBox1_MouseDown(ByVal sender As Object,
ByVal e As _
```

```
System.Windows.Forms.MouseEventArgs) Handles
PictureBox1.MouseDown
If Not PictureBox1.Image Is Nothing Then
'Set a flag to show that the mouse is down.
m_MouseIsDown = True
End If
End Sub

Private Sub PictureBox1_MouseMove(ByVal sender As Object,
ByVal e As _
System.Windows.Forms.MouseEventArgs) Handles
PictureBox1.MouseMove
If m_MouseIsDown Then
'Initiate dragging and allow either copy or move.
PictureBox1.DoDragDrop(PictureBox1.Image,
DragDropEffects.Copy Or _
DragDropEffects.Move)
End If
m_MouseIsDown = False
End Sub

Private Sub PictureBox2_DragEnter(ByVal sender As Object,
ByVal e As _
System.Windows.Forms.DragEventArgs) Handles
PictureBox2.DragEnter
If e.Data.GetDataPresent(DataFormats.Bitmap) Then
'Check for the CTRL key.
If e.KeyState = 9 Then
e.Effect = DragDropEffects.Copy
Else
e.Effect = DragDropEffects.Move
End If
Else
e.Effect = DragDropEffects.None
End if
End sub

Private Sub PictureBox2_DragDrop(ByVal sender As Object,
ByVal e As _
System.Windows.Forms.DragEventArgs) Handles
PictureBox2.DragDrop
```

NOTES

NOTES

```
'Assign the image to the PictureBox.
PictureBox2.Image = e.Data.GetData(DataFormats.Bitmap)
'If the CTRL key is not pressed, delete the source picture.
If Not e.KeyState = 8 Then
PictureBox1.Image = Nothing
End If
End Sub
```

The AllowDrop property for the second PictureBox control is set in the Form1_Load event. In both the DragEnter and DragDrop events, the code checks to see if the CTRL key is pressed to determine whether to copy or move the picture.

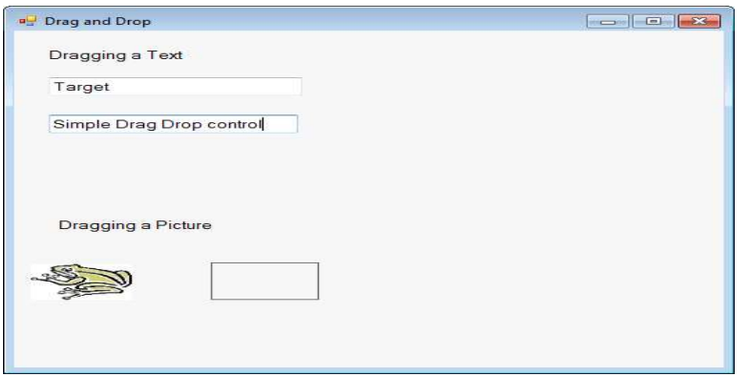


Fig. 1 *Control before being dragged to a target*

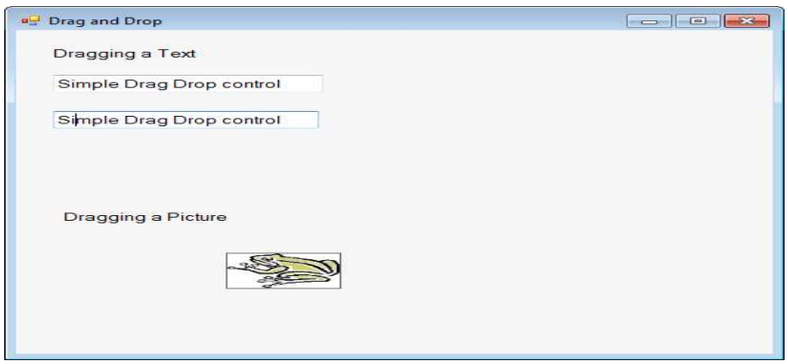


Fig. 2 *Control after being dragged to a target*

VB.NET Inbuilt Functions

Built-in functions are used for manipulating text as well as for carrying out mathematical operations. These are used to format data in user-defined and standard styles. Basically, there are two types of functions: the MsgBox() function and the InputBox() function.

1. MsgBox ( ) Function

NOTES

The MsgBox is used to generate a pop-up message box which prompts the user to click on a command button. For example:

```
yourMsg=MsgBox(Prompt,Style Value, Title)
```

`Prompt` will display the message in the message box. The `Style` Value is used to find what type of command buttons appear on the message box. `Title` argument will display the title of the message board.

Table 1 *Style Values*

Style Value	Named Constant	Buttons Displayed
0	vbOkOnly	Ok button
1	vbOkCancel	Ok and Cancel buttons
2	vbAbortRetryIgnore	Abort, Retry and Ignore buttons.
3	vbYesNoCancel	Yes, No and Cancel buttons
4	vbYesNo	Yes and No buttons
5	vbRetryCancel	Retry and Cancel buttons

In the second argument, we can use named constant in place of integers to make the programs more readable. For example:

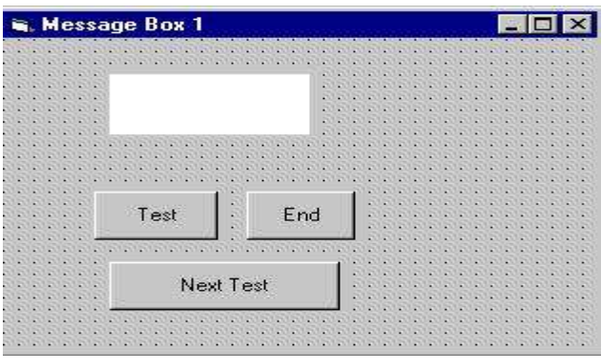
```
yourMsg=MsgBox( "Click OK to Proceed", 1, "Startup Menu")
and
yourMsg=Msg("Click OK to Proceed". vbOkCancel,"Startup
Menu")
```

Both the codes given above are same. The table below shows the value, named constant and buttons.

Value	Named Constant	Button Clicked
1	vbOk	Ok button
2	vbCancel>	Cancel button
3	vbAbort	Abort button
4	vbRetry	Retry button
5	vbIgnore	Ignore button
6	vbYes	Yes button
7	vbNo	No button

NOTES

The example below shows the interface which is to be drawn with a label and three command buttons.



Write the following code for test button.

```
Private Sub Test_Click()
Dim testmsg As Integer
testmsg = MsgBox("Click to test", 1, "Test message")
If testmsg = 1 Then
Display.Caption = "Testing Successful"
Else
Display.Caption = "Testing fail"
End If
End Sub
```

After clicking the test button, the message shown below will appears.

After clicking Ok button, the message "Testing successful" will be displayed and after clicking on the cancel button, "Testing fail" will be displayed. There are various types of icons that can be displayed.

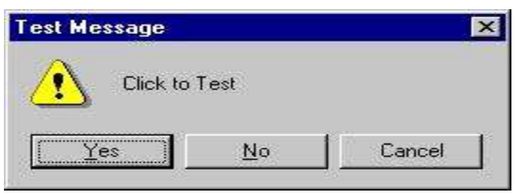
NOTES

Value	Named Constant	Icon
16	vbCritical	
3	vbQuestion	
48	vbExclamation	
64	vbInformation	

Consider the example given below:

```
Private Sub test2_Click()
Dim testMsg2 As
Integer testMsg2 = MsgBox("Click to Test", vbYesNoCancel
+ vbExclamation, "TestMessage")
If testMsg2 = 6 Then
display2.Caption ="Testing successful"
ElseIf testMsg2 = 7 Then
display2.Caption = "Are you sure?"
Else display2.Caption ="Testing fail"
End If
End Sub
```

Output:



2. The InputBox( ) Function

An InputBox( ) function display a message box where the user can enter a value or a message in the form of text. For example:

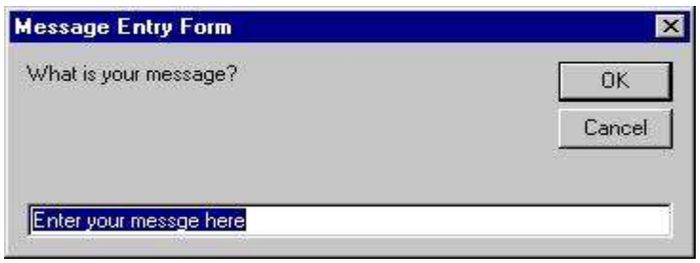
```
myMessage=InputBox(Prompt, Title, default_text, x-
position, y-position)
```

`myMessage` is data type which is declared as string. Here, the message input by the users is `default-text` displays the default text that appears in the input field where users can use it as his intended input. `Title` dislays the title of the Input Box. `Prompt` is the message displayed normally as a question asked. x-position and y-position is the position or the coordinate of the input box. Consider an example given below:

NOTES

```
Private Sub OK_Click()
Dim userMsg As String
userMsg = InputBox("What is your message?", "Message Entry
Form", "Enter your messge here", 500, 700)
If userMsg <>"" Then
message.Caption = userMsg
Else
message.Caption = "No Message"
End If
End Sub
```

After clicking the OK button, the message will be displayed and after clicking the cancel button, "No message" will be displayed.



Mathematical Functions

In VB.NET, math functions are stored in System.Math namespace. The namespace is used to import Math functions. The functions built into Math class can be applied to calculate square roots, logarithm values, trigonometry etc. Consider an example given below:

```
Imports System.Console
Imports System.Math
Module Module1
    Sub Main()
        WriteLine("Sine 60 is" & " " & Sin(60))
        'display Sine60 value
      WriteLine("Square root of 72 is " & " " & Sqrt(72))
        'displays square root of 72
        WriteLine("Log value of 14 is" & " " & Log(14))
        'displays the logarithm value of 14
        Read()
    End Sub
End Module
```

```
Module1.vb
(General)
Imports System.Console
Imports System.Math
Module Module1
    Sub Main()
        WriteLine("Sine 60 is" & " " & Sin(60))
        'display Sine60 value
        WriteLine("Square root of 72 is " & " " & Sqrt(72))
        'displays square root of 72
        WriteLine("Log value of 14 is" & " " & Log(14))
        'displays the logarithm value of 14
        Read()
    End Sub
End Module
```

The output from above code is given below.

```
file:///c:/users/imsit09/documents/visual studio 2010/prc
Sine 60 is -0.304810621102217
Square root of 72 is  8.48528137423857
Log value of 14 is 2.63905732961526
```

String Functions

String functions are mainly used to edit and manipute the string. Following are the string functions in VB.

Methods	Description
Asc, AscW	This method will return an integer value that represents a character code corresponding to a character.
Chr, ChrW	It will return the character associated to a character code.
Filter	This method returns a zero-based array having a subset of a string array on the basis of specified filter criteria.
Format	This method will return a string formatted according to instructions contained in a format string expression.
FormatCurrency	It will return an expression formatted as a currency value using the currency symbol defined in the system control panel.
FormatDateTime	It will return a string expression showing date/time value.
FormatNumber	It will return an expression in a number format.
FormatPercent	It will return an expression in percentage followed by % character.
InStr	This method will return an integer that specifies the start position of the first occurrence of one string in another.
InStrRev	This method will return the position of the first occurrence of one string within another, starting from the right side of the string.
Join	It will return a string created by concatenating a number of substrings.
LCase	Converts a string or character to lowercase.
Left	This method will return a number of characters in a string from the left.

NOTES

Len	It will return an integer containing the number of characters in a string.
LSet	This method will return a left-aligned string containing the specified string adjusted to the specified length.
LTrim	It will return a string containing a copy of a specified string having no spaces.
Mid	This method returns a string containing a specified number of characters from mid.
Replace	This method replaces a substring with another with a specific number of times.
Right	It will return a number of characters from the right side of a string.
Space	It will return a string containing a given number of spaces.
StrComp	It will return -1, 0, or 1, based on the result of comparison.
StrConv	Converts a string as specified.
StrDup	It will return a string that contains repeated character a number of times.
StrReverse	This method returns a string in which the character order of a specified string is reversed.
Trim	Returns a copy of string having no spaces.
UCase	Converts a string to uppercase.

ActiveX controls

ActiveX controls are objects or COM components that can be used in a web page or other application that is already programmed by someone else. **ActiveX controls** developed for **Visual Basic** 6.0 and earlier versions can be used to add features to the toolbox of Visual Studio. You can add ActiveX controls to the toolbox using the following steps.

1. Click **Choose Toolbox Items** on the **Tools** menu. **Choose Toolbox** dialog box will appears.
2. Now, click the **COM Components** tab.
3. You have to select the check box next to ActiveX control and click **OK**.

The new control appears with the other tools in the **Toolbox**.

Database Access Objects (DAO)

It is an abstract pattern that provides interface to some types of database. DAO provides some specific data operations without exposing details of the database by mapping application calls to the persistence layer. The data needs by the application is separated in terms of domain-specific objects and data types from how these needs can be satisfied with a specific DBMS, database schema, etc.

NOTES

Database object properties

Some of the properties of database objects are:

1. It is the relatively simple and rigorous separation between two important parts of an application that can but should not know anything of each other.
2. It can be expected to evolve frequently and independently.
3. Changing business logic can rely on the same DAO interface, while changes to persistence logic do not affect DAO clients as long as the interface remains correctly implemented.
4. All details of storage are hidden from the rest of the application.
5. It acts as an intermediary between the application and the database.
6. They move data back and forth between objects and database records.

ADO.NET

ADO is a Microsoft technology that stands for ActiveX Data Objects. It is automatically installed with Microsoft IIS. It provides an interface to access data in a database. There are various applications that require data access while working with applications. It makes the application to interact with a database. There are various applications which have different requirements for database access. For example: VB .NET uses ADO.NET (Active X Data Object) as its data access and manipulation protocol which also enables us to work with data on the internet.

ADO.NET Data Architecture

Data Access in ADO.NET is based on two components i.e. DataSet and Data Provider**.**

1. **DataSet**: The dataset is a disconnected and in-memory representation of data. It is a local copy of the relevant portions of the database. When the use of the DataSet is completed, then changes can be made back to the central database for updating. The DataSet is persisted in memory and the data in it can be updated and manipulated independent of the database. The data in DataSet can be loaded from any valid data source like Microsoft SQL server database, an Oracle database or from a Microsoft Access database.
2. **Data Provider:** When the use of the DataSet is completed, then changes can be made back to the central database. The Data Provider is responsible for providing and maintaining the connection to the database. Data Provider is a set of related components that work together to provide data in an efficient and performance driven manner. The .NET framework currently comes with two DataProviders i.e. the SQL Data Provider which is designed only to work with OleDb DataProvider or Microsoft's SQL Server which allows us to connect to other types of databases like Access and Oracle.

NOTES

Each DataProvider consists of the following component classes:

1. The Connection object provides a connection to the database.
2. The Command object is used to execute a command.
3. The DataReader object provides a forward-only, read only, connected recordset.
4. The DataAdapter object populates a disconnected DataSet with data and performs update.

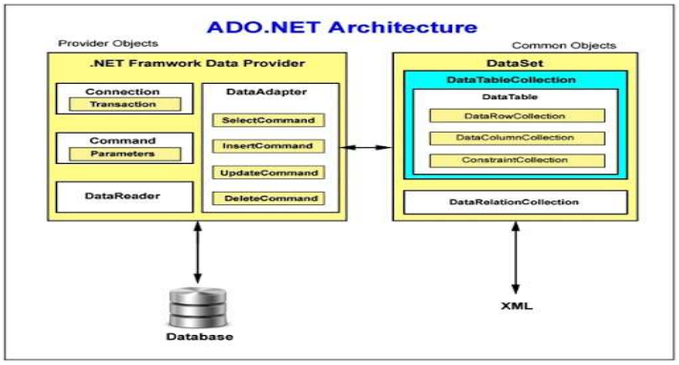


Fig. 3 *ADO.NET Architecture*

Component classes that make up the data providers are as follows:

1) The Connection Object

The Connection object creates the connection to the database. Microsoft VB.NET provides two types of connection classes: the SqlConnection object, which is designed specifically to connect to Microsoft SQL Server and the OleDbConnection object, which can provide connections to a wide range of database types like Microsoft Access and Oracle. The Connection object contains all of the information required to open a connection to the database.

2) The Command Object

The Command object is represented by two corresponding classes: SqlCommand and OleDbCommand. The Command objects are used to execute the commands to a database across a data connection. These can be used to execute stored procedures on the database, SQL commands, or return complete tables directly. Command objects provide three methods that are used to execute commands on the database.

1. **ExecuteScalar:** Returns a single value from a database query.
2. **ExecuteNonQuery:** Executes commands that have no return values such as INSERT,

   UPDATE or DELETE.
3. **ExecuteReader:** Returns a result set by way of a DataReader object.

NOTES

3) The DataReader Object

The DataReader object provides a read-only, forward-only connected stream recordset from a database. It cannot be directly instantiated. Instead, The OleDbCommand.ExecuteReader method returns an OleDbDataReader object. The DataReader is returned as the result of the Command object's ExecuteReader method. The SqlCommand.ExecuteReader method returns a SqlDataReader object. The DataReader can provide rows of data directly to application logic when you don't require keeping the data cached in memory because only one row is in memory at a time. It provides the lowest overhead in terms of system performance but requires the exclusive use of an open Connection object for the lifetime of the DataReader.

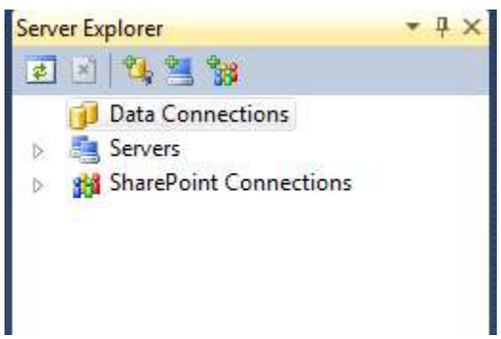
4) The DataAdapter Object

The DataAdapter is the class at the core of ADO.NET's disconnected data access. The DataAdapter is used either to fill a DataSet or DataTable with data from the database with its Fill method. After the memory-resident data has been manipulated, the DataAdapter can commit the changes to the database by calling the Update method. The DataAdapter provides four properties that represent database commands.

1. SelectCommand
2. DeleteCommand
3. InsertCommand
4. UpdateCommand

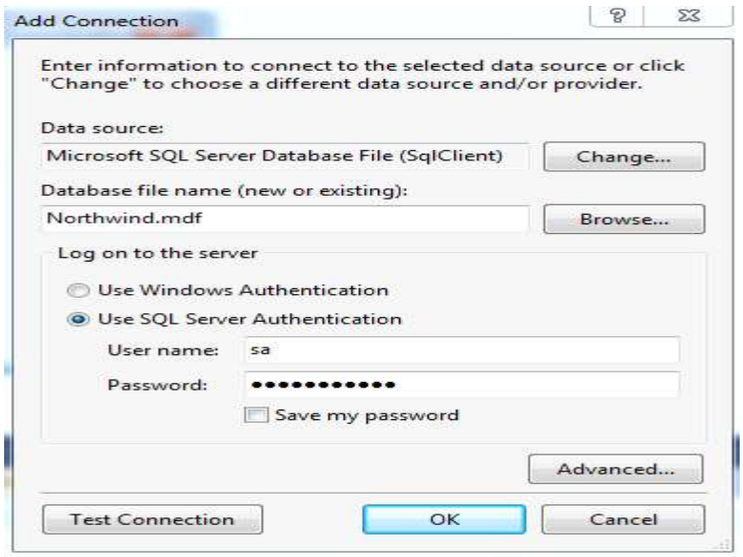
Data Access with Server Explorer

VB allows us to work with database in two ways, visually and code. In VB, server explorer allows us to work with connections across different data sources visually. The window that is displayed is the Server Explorer lets us create and examine data connections. Server Explorer can be viewed by selecting ViewàServer Explorer from the main menu or by pressing Ctrl+Alt+S on the keyboard as shown below.

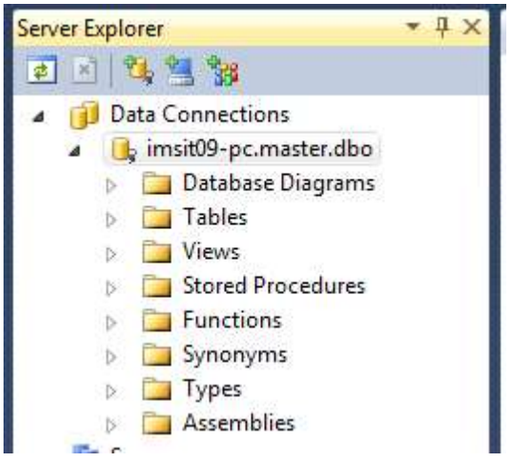


NOTES

In order to work with the Server Explorer, we will work with SQL Server, the default provider for .NET. We will be displaying data from Customers table in sample Northwind database in SQL Server. For this, we need to establish a connection to this database. You need to just right-click on the data connections icon in Server Explorer and select Add Connection that opens the Data Link Properties dialog which allows you to enter the name of the server you want to work along with login name and password.



To work with a database which is already on the server, you have to select the option "select the database on the server". Now, select Northwind database from the list. After that, click on the Test Connection tab to test the connection. If the connection is successful, the message "Test Connection Succeeded" is displayed. When connection to the database is set, click OK and close the Data Link Properties or add connection. When, you expand the connection node that is ("+" sign), it displays the Tables, Views and Stored Procedures in that Northwind sample database. Expanding the Tables node will display all the tables available in the database.



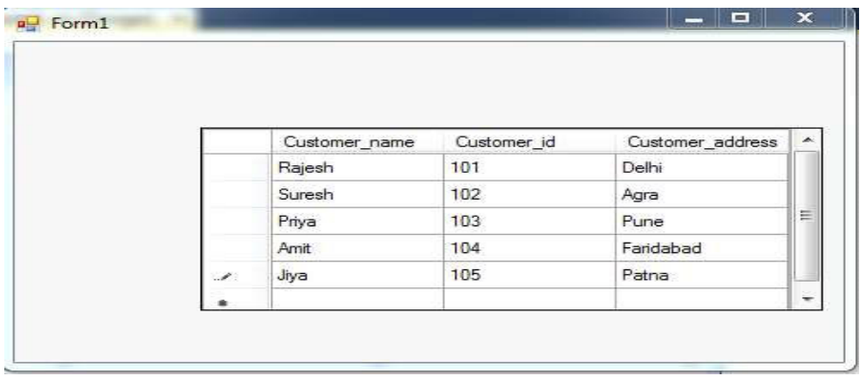
NOTES

In this example given below, we will work with Customers table to display its data. Now drag Customers table onto the form from the Server Explorer. Doing that creates SQLConnection1 and SQLDataAdapter1 objects which are the data connection and data adapter objects used to work with data. They are displayed on the component tray. Now, we need to generate the dataset that holds data from the data adapter. To do that select Data'!Generate DataSet from the main menu or right click on SQLDataAdapter1 object and select generate DataSet menu. Dataset dialogbox will open.

Once the dialogbox is displayed, select the radio button with New option to create a new dataset. Make sure Customers table is checked and click OK. Clicking OK adds a dataset to the component tray. After that, drag a DataGrid from toolbox. We will display Customers table in this data grid. Set the data grid's DataSource property to DataSet and its DataMember property to Customers. Next, we need to fill the dataset with data from the data adapter. The code is given below:

```
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs)_
Handles MyBase.Load
DataSet.Clear()
SqlDataAdapter1.Fill(DataSet)
'filling the dataset with the dataadapter's fill method
End Sub
```

The output of the above code is given below:

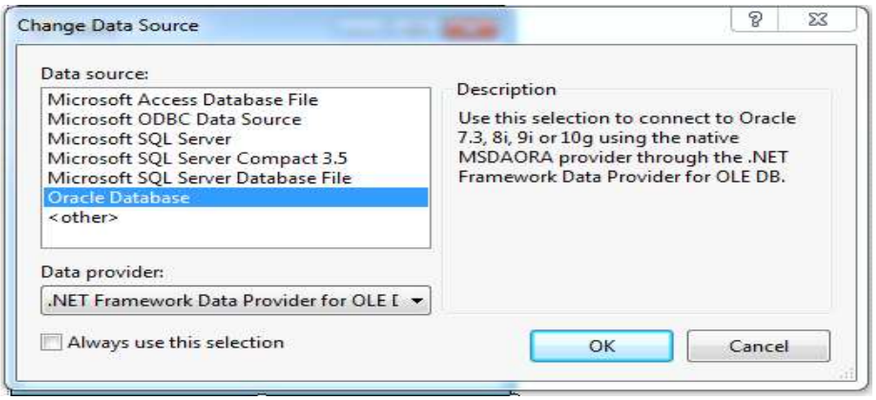


Once the application is executed, Customers table is displayed in the data grid. That is one of the simplest ways of displaying data using the Server Explorer window.

Microsoft Access and Oracle Database

On working with Oracle, you need to select Microsoft OLE DB provider for Oracle from the provider tab in the DataLink dialog. The process is similar in working with Oracle or MS Access but with some minor changes. You need to enter the appropriate Username and password.

NOTES


Change Data Source
Data source:
Microsoft Access Database File
Microsoft ODBC Data Source
Microsoft SQL Server
Microsoft SQL Server Compact 3.5
Microsoft SQL Server Database File
Oracle Database
<other>
Description
Use this selection to connect to Oracle 7.3, 8i, 9i or 10g using the native MSDAORA provider through the .NET Framework Data Provider for OLE DB.
Data provider:
.NET Framework Data Provider for OLE D
Always use this selection
OK
Cancel

BLOCK 4

NOTES

This block will cover the following topics:

1. Using DataReaders and SQL Server
2. Retrieving, inserting, updating and deleting the records using OleDB provider and MS access.

Using DataReaders and SQL Server

In this section, will work with ADO.NET objects in code to create connections and read data using the data reader. The namespace that requires to be imported while working with SQL Connections is System.Data.SqlClient. Here, we will check that how to connect by using our own connection objects. We also check how to use the command object.

Working with SQL Server

The classes in SQL server are discussed below:

a) **The SqlConnection Class:** This class allows the connection to SQL server data source. We will use OleDB connection object, when working with databases instead of SQL Server. The performance of Sqlconnections is 70% faster than OleDB connections.

b) **The SqlCommand Class:** This class represents a SQL statement or stored procedure for use in a database with SQL Server.

c) **The SqlDataAdapter Class:** This class represents a bridge between SQL server database and dataset. It includes the Select, Insert, Update and Delete commands for loading and updating the data.

d) **The SqlDataReader Class:** The SqlDataReader class creates a data reader to be used with SQL Server.

DataReaders

A DataReader is a lightweight object which provides forward-only, read-only data in a very efficient and fast way. Data access with DataReader is read-only, if we cannot make any changes (update) to data and forward-only, which means we cannot go back to the previous record which was accessed. A DataReader requires the use of an active connection for the entire time. We can instantiate a DataReader by making a call to a Command object's ExecuteReader command. When the DataReader is first returned, it is positioned before the first record of the result set. To make the first record available, we need to call the Read method. If a record is available, then Read method moves the DataReader to next record and returns True. If a record is not available the Read method returns False.

NOTES

Program 1: To retrieve data using Select command (to display data from Discounts table in Pubs sample database).

```
Imports System.Data.SqlClient
Public Class Form1 Inherits System.Windows.Forms.Form
Dim myConnection As SqlConnection
Dim myCommand As SqlCommand
Dim dr As New SqlDataReader()
'declaring the objects
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As
System.EventArgs)_
Handles MyBase.Load
myConnection = New
SqlConnection("server=localhost;uid=sa;pwd=;database=pubs")
'establishing connection. you need to provide password
for sql server
Try
myConnection.Open()
'opening the connection
myCommand = New SqlCommand("Select * from discounts",
myConnection)
'executing the command and assigning it to connection
dr = myCommand.ExecuteReader()
While dr.Read()
'reading from the datareader
MessageBox.Show("discounttype" & dr(0).ToString())
MessageBox.Show("stor_id" & dr(1).ToString())
MessageBox.Show("lowqty" & dr(2).ToString())
MessageBox.Show("highqty" & dr(3).ToString())
MessageBox.Show("discount" & dr(4).ToString())
'displaying the data from the table
End While
dr.Close()
myConnection.Close()
Catch e As Exception
End Try
End Sub
End Class
```

The above code displays records from discounts table in MessageBoxes.

Retrieving records with a Console Application

NOTES

```
Imports System.Data.SqlClient
Imports System.Console
Module Module1
Dim myConnection As SqlConnection
Dim myCommand As SqlCommand
Dim dr As SqlDataReader
Sub Main()
Try
myConnection = New
SqlConnection("server=localhost;uid=sa;pwd=;database=pubs")
'you need to provide password for sql server
myConnection.Open()
myCommand = New SqlCommand("Select * from discounts",
myConnection)
dr = myCommand.ExecuteReader
Do
While dr.Read()
WriteLine(dr(0))
WriteLine(dr(1))
WriteLine(dr(2))
WriteLine(dr(3))
WriteLine(dr(4))
' writing to console
End While
Loop While dr.NextResult()
Catch
End Try
dr.Close()
myConnection.Close()
End Sub
End Module
```

Inserting a Record

Program 2: To insert a record into the Jobs table in Pubs sample database.

```
Imports System.Data.SqlClient
Public Class Form2 Inherits System.Windows.Forms.Form
Dim myConnection As SqlConnection
Dim myCommand As SqlCommand
Dim ra as Integer
'integer holds the number of records inserted
```

NOTES

```
Private Sub Form2_Load(ByVal sender As System.Object,
ByVal e_
As System.EventArgs) Handles MyBase.Load
End Sub
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e_
As System.EventArgs) Handles Button1.Click
myConnection = New
SqlConnection("server=localhost;uid=sa;pwd=;database=pubs")
'you need to provide password for sql server
myConnection.Open()
myCommand = New SqlCommand("Insert into Jobs values 12,'IT
Manager',100,300,_
myConnection)
ra=myCommand.ExecuteNonQuery()
MessageBox.Show("New Row Inserted" & ra)
myConnection.Close()
End Sub
End Class
```

Deleting a Record

Program 3: For deleting a record, we will use Authors table in Pubs sample database to work with this code. Drag a button onto the form and place the following code.

```
Imports System.Data.SqlClient
Public Class Form3 Inherits System.Windows.Forms.Form
Dim myConnection As SqlConnection
Dim myCommand As SqlCommand
Dim ra as Integer
Private Sub Form3_Load(ByVal sender As System.Object,
ByVal e_
As System.EventArgs) Handles MyBase.Load
End Sub
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e_
As System.EventArgs) Handles Button1.Click
myConnection = New
SqlConnection("server=localhost;uid=sa;pwd=;database=pubs")
'you need to provide password for sql server
myConnection.Open()
myCommand = New SqlCommand("Delete from Authors where
```

```
city='Oakland'",_
myConnection)
'since no value is returned we use ExecuteNonQuery
ra=myCommand.ExecuteNonQuery()
MessageBox.Show("Records affected" & ra)
myConnection.Close()
End Sub
End Class
```

NOTES

Updating a Record

Program 4: For updating a record, we will update a row in Authors table. Drag a button onto the form and place the following code.

```
Imports System.Data.SqlClient
Public Class Form4 Inherits System.Windows.Forms.Form
Dim myConnection As SqlConnection
Dim myCommand As SqlCommand
Dim ra as Integer
Private Sub Form4_Load(ByVal sender As System.Object,
ByVal e_
As System.EventArgs) Handles MyBase.Load
End Sub
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e_
As System.EventArgs) Handles Button1.Click
myConnection = New
SqlConnection("server=localhost;uid=sa;pwd=;database=pubs")
'you need to provide password for sql server
myConnection.Open()
myCommand = New SqlCommand("Update Authors Set
city='Oakland'

'San where city=_
Jose' ",myConnection)
ra=myCommand.ExecuteNonQuery()
MessageBox.Show("Records affected" & ra)
myConnection.Close()
End Sub
End Class
```

NOTES

Using OleDb Provider

The classes of the OleDb provider with which we work are as follows:

1. **The OleDbConnection Class:** The OleDbConnection class allows a connection to OleDb data source. OleDbconnections are used to connect to most databases.
2. **The OleDbCommand Class:** The OleDbCommand class shows a SQL statement or stored procedure which is to be executed in a database by an OLEDB provider.
3. **The OleDbDataAdapter Class:** The OleDbDataAdapter class represents as an intermediate between OleDb data source and datasets. We use the Select, Insert, Delete and Update commands for loading and updating the data.
4. **The OleDbDataReader Class:** The OleDbDataReader class creates a datareader for use with an OleDb data provider. The data is read as forward-only stream which means that data is read sequentially, one row after another not allowing you to choose a row you want or going backwards. It is used to read a row of data from the database.

Program 5: To retrieve the records. In the code below, we are working with Emp table in Oracle.

```
Imports System.Data.OleDB
Public Class Form1 Inherits System.Windows.Forms.Form
Dim myConnection As OleDbConnection
Dim myCommand As OleDbCommand
Dim dr As New OleDbDataReader()
'declaration
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As
System.EventArgs)_
Handles MyBase.Load
myConnection = New OleDbConnection_
("Provider=MSDAORA.1;UserID=scott;password=tiger;
database=ora")
'MSDORA is the provider when working with Oracle



Try
myConnection.Open()
'opening the connection
myCommand = New OleDbCommand("Select * from emp",
```

```
myConnection)
'executing the command and assigning it to connection
dr = myCommand.ExecuteReader()
While dr.Read()
'reading from the datareader
MessageBox.Show("EmpNo" & dr(0))
MessageBox.Show("EName" & dr(1))
MessageBox.Show("Job" & dr(2))
MessageBox.Show("Mgr" & dr(3))
MessageBox.Show("HireDate" & dr(4))
'displaying data from the table
End While
dr.Close()
myConnection.Close()
Catch e As Exception
End Try
End Sub
AND CLASS
```

The above code displays first 5 columns from the Emp table in Oracle.

Inserting a Record

Program 6: Drag a Button from the toolbox onto the Form. When this Button is clicked the values specified in code will be inserted into the Emp table.

```
Imports System.Data.OleDb
Public Class Form2 Inherits System.Windows.Forms.Form
Dim myConnection As OleDbConnection
Dim myCommand As OleDbCommand
Dim ra as Integer
'integer holds the number of records inserted
Private Sub Form2_Load(ByVal sender As System.Object,
ByVal e As_
System.EventArgs) Handles MyBase.Load
End Sub
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As _
System.EventArgs) Handles Button1.Click
myConnection = New
OleDbConnection(""Provider=MSDAORA.1;User_
ID=scott;password=tiger;database=ora"
)
Try
```

NOTES

```
myConnection.Open()
myCommand = New OleDbCommand("Insert into emp values
12,'Ben','Salesman',300
12-10-2001,3000,500,10 ", myConnection)
'emp table has 8 columns. You can work only with the
columns you want
ra=myCommand.ExecuteNonQuery()
MessageBox.Show("Records Inserted" & ra)
myConnection.Close()
Catch
End Try
End Sub
End Class
Deleting Records
Drag a Button on a new form and paste the following code.
Imports System.Data.OleDb
Public Class Form3 Inherits System.Windows.Forms.Form
Dim myConnection As OleDbConnection
Dim myCommand As OleDbCommand
Dim ra as Integer
Private Sub Form3_Load(ByVal sender As System.Object,
ByVal e As_
System.EventArgs) Handles MyBase.Load
End Sub
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e_
As System.EventArgs) Handles Button1.Click
Try
myConnection.Open()ID=scott;password=tiger;database=ora")

myCommand = myConnection = New
OleDbConnection(""Provider=MSDAORA.1;User_
New OleDbCommand("Delete from emp where
DeptNo=790220",_
myConnection)
ra=myCommand.ExecuteNonQuery()
MessageBox.Show("Records Deleted" & ra)
myConnection.Close()
Catch
End Try
End Sub
End Class
```

Updating a Record

Program 7: Drag a Button on a new form and paste the following code.

```
Imports System.Data.OleDb
Public Class Form4 Inherits System.Windows.Forms.Form
Dim myConnection As OleDbConnection
Dim myCommand As OleDbCommand
Dim ra as Integer
Private Sub Form4_Load(ByVal sender As System.Object,
ByVal e As_
System.EventArgs) Handles MyBase.Load
End Sub
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e_
As System.EventArgs) Handles Button1.Click
Try
myConnection = New
OleDbConnection(“”Provider=MSDAORA.1;User_
ID=scott;password=tiger;database=ora”)
myConnection.Open()
myCommand = New OleDbCommand(“Update emp Set DeptNo=65
where DeptNo=793410",_ myConnection)
ra=myCommand.ExecuteNonQuery()
MessageBox.Show(“Records Updated” & ra)
myConnection.Close()
Catch
End Try
End Sub
End Class
```

NOTES

Data Access using MSAccess

Program 8: In this program, create a database named Emp in Microsoft Access in the C drive of your computer. In the Emp database, create a table, Table1 with EmpNo, EName and Department as columns, insert some values in the table and close it. Drag three TextBoxes and a Button. The following code will assume that TextBox1 is for EmpNo, TextBox2 is for EName and TextBox3 is for Department. Our intention is to retrieve data from Table1 in the Emp Database and display the values in these TextBoxes without binding, when the Button is clicked.

```
Imports System.Data.OleDb
Public Class Form1 Inherits System.Windows.Forms.Form
Dim cn As OleDbConnection
Dim cmd As OleDbCommand
```

NOTES

```
Dim dr As OleDbDataReader
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e as _
System.EventArgs) Handles MyBase.Load
End Sub
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As_
System.EventArgs) Handles Button1.Click
Try
cn                          =                          New
OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;_
Data Source=C:\emp.mdb;")
'provider to be used when working with access database
cn.Open()
cmd = New OleDbCommand("select * from table1", cn)
dr = cmd.ExecuteReader
While dr.Read()
TextBox1.Text = dr(0)
TextBox2.Text = dr(1)
TextBox3.Text = dr(2)
' loading data into TextBoxes by column index
End While
Catch
End Try
dr.Close()
cn.Close()
End Sub
End Class
```

When you run the code and click the Button, records from Table1 of the Emp database will be displayed in the TextBoxes.

Retrieving a Record

Program 9: Write a code for retrieving records with a Console Application.

```
Imports System.Data.OleDb
Imports System.Console
Module Module1
Dim cn As OleDbConnection
Dim cmd As OleDbCommand
Dim dr As OleDbDataReader
Sub Main()
Try
```

```
cn = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\emp.mdb;_
Persist Security Info=False")
cn.Open()
cmd = New OleDbCommand("select * from table1", cn)
dr = cmd.ExecuteReader
While dr.Read()
WriteLine(dr(0))
WriteLine(dr(1))
'writing to console
End While
Catch
End Try WriteLine(dr(2))

dr.Close()
cn.Close()
End Sub
End Module
```

NOTES

Code for Inserting a Record

```
Imports System.Data.OleDb
Public Class Form2 Inherits System.Windows.Forms.Form
Dim cn As OleDbConnection
Dim cmd As OleDbCommand
Dim dr As OleDbDataReader
Dim icount As Integer
Dim str As String
Private Sub Form2_Load(ByVal sender As System.Object,
ByVal e As_
System.EventArgs) Handles MyBase.Load
End Sub
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As_
System.EventArgs) Handles Button2.Click
Try
cn = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\emp.mdb;")
cn.Open()
str = "insert into table1 values(" & CInt(TextBox1.Text)
& ",'" &
```

NOTES

```
TextBox2.Text & "'," &_
TextBox3.Text & "')"
'string stores the command and CInt is used to convert
number to string
cmd = New OleDbCommand(str, cn)
icount = cmd.ExecuteNonQuery
MessageBox.Show(icount)
'displays number of records inserted
Catch
End Try
cn.Close()
End Sub
End Class
```

BLOCK 5

This block will cover the development of following simple applications:

1. Library Information System
2. Students Marksheet Processing
3. Telephone Directory Maintenance
4. Gas Booking and Delivering
5. Electricity Bill Processing
6. Bank Transaction
7. Pay Roll Processing
8. Personal Information System
9. Question Database and Conducting Quiz
10. Personal Diary

NOTES

1. Library Information System

Add Books:

```
Public Class AddBooks
    Public NameFrm, NameTo As String
       Private Sub Button9_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button9.Click
        Me.Close()
    End Sub


      Private Sub AddBooks_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Call generateyear()
        Call disablethem()
        Call readData()
       Call GroupID_Combo()
    End Sub
   Sub GroupID_Combo()
        Try
          If objcon.State = ConnectionState.Closed Then
objcon.Open()
           com = New OleDb.OleDbCommand("Select GroupID
from GroupD", objcon)
            dr = com.ExecuteReader
```

NOTES

```
            While dr.Read
               ComboBox1.Items.Add(dr.Item(0))
            End While
            dr.Close()
            objcon.Close()
        Catch ex As Exception

        End Try
    End Sub
    Sub generateyear()
        Dim YearNow As Integer
                                          YearNow     =
Int(My.Computer.Clock.LocalTime.Year.ToString)
        Dim a, b, c As Integer
        a = YearNow - 5
        b = YearNow
        For c = a To b
           ComboBox2.Items.Add(c)
        Next
    End Sub

     Private Sub ComboBox1_LostFocus(ByVal sender As
Object, ByVal e As System.EventArgs) Handles
ComboBox1.LostFocus
       ComboBox1.Text = ComboBox1.Text.ToUpper()
    End Sub



      Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
       ComboBox3.Text = "Available"
       Call enablethem()
    End Sub

   Private Sub TextBox2_LostFocus(ByVal sender As Object,
ByVal e As System.EventArgs) Handles TextBox2.LostFocus
       NameFrm = TextBox2.Text
       Call Sentence()
       TextBox2.Text = NameTo
    End Sub
```

```
Sub disablethem()
    'TextBox1.Enabled = False
    TextBox2.Enabled = False
    TextBox3.Enabled = False
    ComboBox1.Enabled = False
    TextBox4.Enabled = False
    TextBox5.Enabled = False
    TextBox6.Enabled = False
    ComboBox2.Enabled = False
    ComboBox3.Enabled = False
End Sub
Sub enablethem()
    TextBox1.Enabled = True
    TextBox2.Enabled = True
    TextBox3.Enabled = True
    ComboBox1.Enabled = True
    TextBox4.Enabled = True
    TextBox5.Enabled = True
    TextBox6.Enabled = True
    ComboBox2.Enabled = True
    ComboBox3.Enabled = True
    TextBox1.Clear()
    TextBox2.Clear()
    TextBox3.Clear()
    TextBox4.Clear()
    TextBox5.Clear()
    TextBox6.Clear()
    ComboBox1.Text = ""
    ComboBox2.Text = ""
    ComboBox3.Text = ""
End Sub

Sub Sentence()
    Dim a, b As Integer
    a = NameFrm.Length
    NameTo = ""
    For b = 0 To a - 1
        If b = 0 Then
            If Char.IsLower(NameFrm(0)) Then
                NameTo = Char.ToUpper(NameFrm(0))
```

NOTES

NOTES

```
                Else
                    NameTo = NameFrm(0)
                End If
            Else
                If NameFrm(b - 1) = " " Then
                                NameTo = NameTo +
Char.ToUpper(NameFrm(b))
                Else
                    NameTo = NameTo + NameFrm(b)
                End If
            End If
        Next
    End Sub

   Private Sub TextBox3_LostFocus(ByVal sender As Object,
ByVal e As System.EventArgs) Handles TextBox3.LostFocus
        NameFrm = TextBox3.Text
        Call Sentence()
        TextBox3.Text = NameTo
    End Sub

     Private Sub TextBox3_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
TextBox3.TextChanged


    End Sub

   Private Sub TextBox4_LostFocus(ByVal sender As Object,
ByVal e As System.EventArgs) Handles TextBox4.LostFocus
        NameFrm = TextBox4.Text
        Call Sentence()
        TextBox4.Text = NameTo
    End Sub

     Private Sub TextBox4_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
TextBox4.TextChanged


    End Sub

   Private Sub TextBox5_LostFocus(ByVal sender As Object,
ByVal e As System.EventArgs) Handles TextBox5.LostFocus
```

```
        NameFrm = TextBox5.Text
        Call Sentence()
        TextBox5.Text = NameTo
    End Sub

     Private Sub TextBox5_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
TextBox5.TextChanged


    End Sub


      Private Sub Button2_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button2.Click
       If TextBox1.Text = "" Then
          MsgBox("Please enter the Book ID!", 0, "")
       Else
           Try
             If objcon.State = ConnectionState.Closed
Then objcon.Open()
            com = New OleDb.OleDbCommand("INSERT INTO
Books VALUES('" & TextBox1.Text & "','" & ComboBox1.Text
& "','" & TextBox2.Text & "','" & TextBox3.Text & "','" &
TextBox4.Text & "','" & ComboBox2.Text & "','" &
TextBox5.Text & "','" & TextBox6.Text & "','" &
ComboBox3.Text & "')", objcon)
              com.ExecuteNonQuery()
              Call readData()
            MsgBox("Saved successfully", 0, "SUCCESS")
              objcon.Close()
           Catch ex As Exception
              MsgBox(ex.Message, 0, "")
           End Try
       End If
    End Sub
    Sub readData()
       ListView1.Clear()
            ListView1.Columns.Add("BOOK ID", 90,
HorizontalAlignment.Center)
            ListView1.Columns.Add("GROUP ID", 90,
HorizontalAlignment.Center)
           ListView1.Columns.Add("BOOK NAME", 310,
HorizontalAlignment.Center)
```

NOTES

NOTES

```
            ListView1.Columns.Add("PUBLISHER",  90,
HorizontalAlignment.Center)
            ListView1.Columns.Add("AUTHOR",  90,
HorizontalAlignment.Center)
        ListView1.Columns.Add("PUBLISHING YEAR", 130,
HorizontalAlignment.Center)
            ListView1.Columns.Add("EDITION",  90,
HorizontalAlignment.Center)
            ListView1.Columns.Add("PRICE",  90,
HorizontalAlignment.Center)
            ListView1.Columns.Add("STATUS",  90,
HorizontalAlignment.Center)
       ListView1.View = View.Details
        Try


           If (objcon.State = ConnectionState.Closed)
Then objcon.Open()
          com = New OleDb.OleDbCommand("SELECT * FROM
Books ", objcon)
           dr = com.ExecuteReader
           While dr.Read()
             Call adddatatolistview(ListView1, dr(0),
dr(1), dr(2), dr(3), dr(4), dr(5), dr(6), dr(7), dr(8))
           End While
           dr.Close()
           objcon.Close()
        Catch
                      'MsgBox("Please  Refresh",
MsgBoxStyle.Information, "")
        End Try
    End Sub
   Public Sub adddatatolistview(ByVal lvw As ListView,
ByVal BookID As String, ByVal GroupID As String, ByVal
BookName As String, ByVal Publisher As String, ByVal
Author As String, ByVal PubYear As String, ByVal edi As
String, ByVal pric As String, ByVal st As String)
       Dim lv As New ListViewItem
       lvw.Items.Add(lv)
       lv.Text = BookID
       lv.SubItems.Add(GroupID)
       lv.SubItems.Add(BookName)
       lv.SubItems.Add(Publisher)
       lv.SubItems.Add(Author)
```

NOTES

```
        lv.SubItems.Add(PubYear)
        lv.SubItems.Add(edi)
        lv.SubItems.Add(pric)
        lv.SubItems.Add(st)
    End Sub

      Private Sub Button8_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button8.Click
        Try
          If objcon.State = ConnectionState.Closed Then
objcon.Open()

            If MessageBox.Show("Do you really want to
delete?", "ARE YOU SURE", MessageBoxButtons.YesNo) =
Windows.Forms.DialogResult.Yes Then
             com = New OleDb.OleDbCommand("DELETE FROM
Books WHERE BookID='" & TextBox1.Text & "'", objcon)
              com.ExecuteNonQuery()
              objcon.Close()
                   MsgBox("Deleted successfully", 0,
"SUCCESS")
           End If
        Catch ex As Exception

        End Try
    End Sub
    Sub fill_list()
      com = New OleDb.OleDbCommand("Select * from Books",
objcon)
       Dim dr As OleDb.OleDbDataReader
       dr = com.ExecuteReader
       dr.Read()
       While (dr.NextResult)

        End While
    End Sub

      Private Sub GroupBox1_Enter(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
GroupBox1.Enter
```

NOTES

```
    End Sub

    Private Sub TextBox1_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
TextBox1.TextChanged
        Dim i As Integer
       ListView1.SelectedItems.Clear()
        TextBox1.Focus()
        Try
            If Me.TextBox1.Text = "" Then
                TextBox2.Text = ""
            Else
                For i = 0 To ListView1.Items.Count - 1
                                    If  TextBox1.Text  =
ListView1.Items(i).SubItems(0).Text Then
                                        ComboBox1.Text  =
ListView1.Items(i).SubItems(1).Text
                                         TextBox2.Text  =
ListView1.Items(i).SubItems(2).Text
                                         TextBox3.Text  =
ListView1.Items(i).SubItems(3).Text
                                         TextBox4.Text  =
ListView1.Items(i).SubItems(4).Text
                                        ComboBox2.Text  =
ListView1.Items(i).SubItems(5).Text
                                         TextBox5.Text  =
ListView1.Items(i).SubItems(6).Text
                                         TextBox6.Text  =
ListView1.Items(i).SubItems(7).Text
                                        ComboBox3.Text  =
ListView1.Items(i).SubItems(8).Text
                            ListView1.Items(i).Selected =
True
                        Exit For
                    End If
                Next
            End If
        Catch

        End Try
    End Sub

    Private Sub ListView1_SelectedIndexChanged(ByVal
```

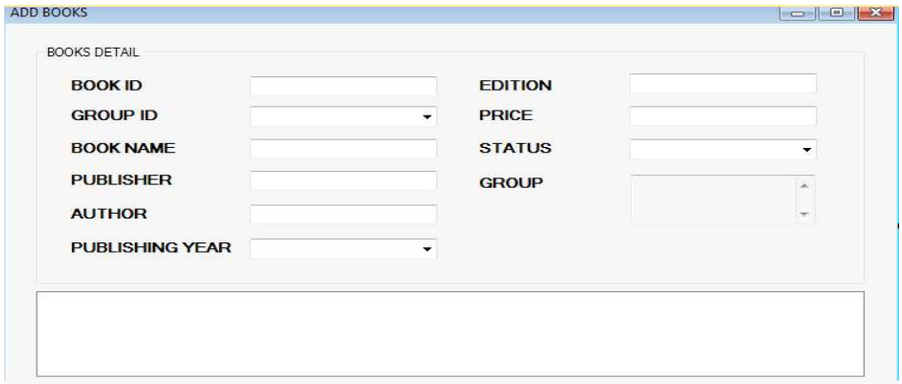
```
sender As System.Object, ByVal e As System.EventArgs)
Handles ListView1.SelectedIndexChanged
        Dim i As Integer
        For i = 0 To ListView1.Items.Count - 1
            If ListView1.Items(i).Selected = True Then
                                        TextBox1.Text  =
ListView1.Items(i).SubItems(0).Text
                TextBox7.Clear()
                 Exit For
             End If
        Next
        ListView1.Focus()
        ListView1.FullRowSelect = True
    End Sub

      Private Sub Button6_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button6.Click
        Try
            Dim i As Integer
            For i = 0 To ListView1.Items.Count - 1
                 If ListView1.Items(i).Selected = True
Then
                   TextBox1.Text = ListView1.Items(i +
1).SubItems(0).Text
                    Exit For
                End If
             Next
            ListView1.Focus()
           ListView1.FullRowSelect = True
        Catch ex As Exception

        End Try
    End Sub

     Private Sub ComboBox1_SelectedIndexChanged(ByVal
sender As System.Object, ByVal e As System.EventArgs)
Handles ComboBox1.SelectedIndexChanged
       Call GroupNameCom()
    End Sub

   Sub GroupNameCom()
```

NOTES

NOTES

```
        Try
          If objcon.State = ConnectionState.Closed Then
objcon.Open()
            com = New OleDb.OleDbCommand("Select * from
GroupD", objcon)
            dr = com.ExecuteReader
            While dr.Read
                If dr.Item(0) = ComboBox1.Text Then
                    TextBox7.Text = dr.Item(1)
                End If

            End While
            dr.Close()
            objcon.Close()
        Catch ex As Exception

        End Try
    End Sub

    Private Sub ComboBox1_TextUpdate(ByVal sender As
Object, ByVal e As System.EventArgs) Handles
ComboBox1.TextUpdate
        Call GroupNameCom()
    End Sub

      Private Sub Button5_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button5.Click
        Try
            Dim i As Integer
            For i = 0 To ListView1.Items.Count - 1
                If ListView1.Items(i).Selected = True
Then
                    TextBox1.Text = ListView1.Items(i -
1).SubItems(0).Text
                    Exit For
                End If
            Next
            ListView1.Focus()
            ListView1.FullRowSelect = True
        Catch ex As Exception
```

```
        End Try
    End Sub
End Class
```



NOTES

Book Details

```
Public Class BookDetail
    Dim sel As Integer
     Private Sub ComboBox1_SelectedIndexChanged(ByVal
sender As System.Object, ByVal e As System.EventArgs)
Handles ComboBox1.SelectedIndexChanged
       Label1.Text = ComboBox1.Text
       Label1.Visible = True
       If Label1.Text = "STATUS" Then
           ComboBox2.Enabled = True
           ComboBox2.Visible = True
           TextBox1.Visible = False

       Else
           ComboBox2.Enabled = False
           ComboBox2.Visible = False
           TextBox1.Visible = True

       End If
       Call forselect()
    End Sub
    Sub forselect()
       If ComboBox1.Text = "BOOK ID" Then
           sel = 1
       ElseIf ComboBox1.Text = "BOOK NAME" Then
           sel = 2
```

NOTES

```
        ElseIf ComboBox1.Text = "AUTHOR" Then
            sel = 3
        ElseIf ComboBox1.Text = "STATUS" Then
            sel = 8
        End If
    End Sub

    Private Sub BookDetail_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        ComboBox2.Visible = False
        TextBox1.Visible = False
        Label1.Visible = False
        Call readData()
    End Sub
    Sub readData()
        ListView1.Clear()
            ListView1.Columns.Add("BOOK ID", 90,
HorizontalAlignment.Center)
            ListView1.Columns.Add("GROUP ID", 90,
HorizontalAlignment.Center)
            ListView1.Columns.Add("BOOK NAME", 310,
HorizontalAlignment.Center)
            ListView1.Columns.Add("PUBLISHER", 90,
HorizontalAlignment.Center)
            ListView1.Columns.Add("AUTHOR", 90,
HorizontalAlignment.Center)
        ListView1.Columns.Add("PUBLISHING YEAR", 130,
HorizontalAlignment.Center)
            ListView1.Columns.Add("EDITION", 90,
HorizontalAlignment.Center)
            ListView1.Columns.Add("PRICE", 90,
HorizontalAlignment.Center)
            ListView1.Columns.Add("STATUS", 90,
HorizontalAlignment.Center)
        ListView1.View = View.Details
        sel = 5
        'Call whenclick()
    End Sub
    Sub whenclick()
        Try

            While dr.Read()
```

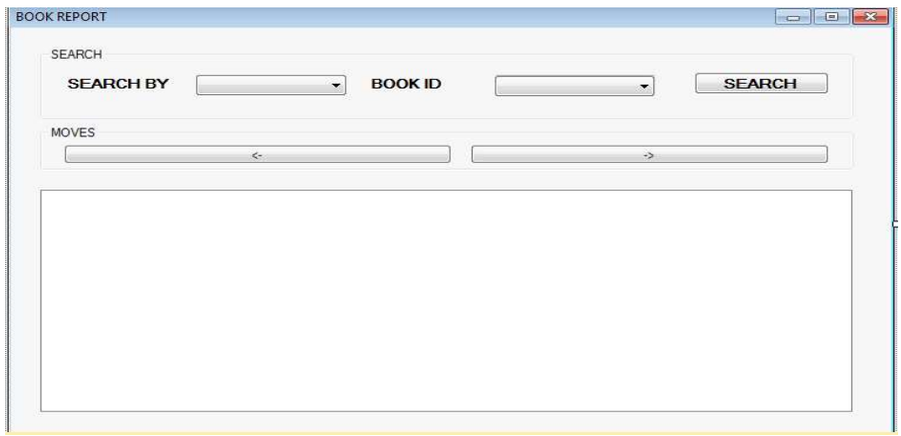
```
            Call adddatatolistview(ListView1, dr(0),
dr(1), dr(2), dr(3), dr(4), dr(5), dr(6), dr(7), dr(8))
          End While
          dr.Close()
          objcon.Close()
       Catch
                    'MsgBox("Please Refresh",
MsgBoxStyle.Information, "")
       End Try
   End Sub
   Public Sub adddatatolistview(ByVal lvw As ListView,
ByVal BookID As String, ByVal GroupID As String, ByVal
BookName As String, ByVal publisher As String, ByVal
author As String, ByVal pubyear As String, ByVal edi As
String, ByVal pric As String, ByVal status As String)
       Dim lv As New ListViewItem
       lvw.Items.Add(lv)
       lv.Text = BookID
       lv.SubItems.Add(GroupID)
       lv.SubItems.Add(BookName)
       lv.SubItems.Add(publisher)
       lv.SubItems.Add(author)
       lv.SubItems.Add(pubyear)
       lv.SubItems.Add(edi)
       lv.SubItems.Add(pric)
       lv.SubItems.Add(status)
   End Sub

     Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
       If objcon.State = ConnectionState.Closed Then
objcon.Open()
       Select Case (sel)
          Case 1
             com = New OleDb.OleDbCommand("select *
from Books where BookID='" & TextBox1.Text & "'", objcon)
             dr = com.ExecuteReader
          Case 2
             com = New OleDb.OleDbCommand("select *
from Books where BookName='" & TextBox1.Text & "'", objcon)
             dr = com.ExecuteReader
          Case 3
```

NOTES

NOTES

```
                com = New OleDb.OleDbCommand("select *
from Books where Author='" & TextBox1.Text & "'", objcon)
                dr = com.ExecuteReader
            Case 5
                com = New OleDb.OleDbCommand("select *
from Books", objcon)
                dr = com.ExecuteReader
            Case 8
                com = New OleDb.OleDbCommand("select *
from Books where Status='" & ComboBox2.Text & "'", objcon)
                dr = com.ExecuteReader
        End Select
        Call readData()
        Call whenclick()
        objcon.Close()
    End Sub

     Private Sub ListView1_SelectedIndexChanged(ByVal
sender As System.Object, ByVal e As System.EventArgs)
Handles ListView1.SelectedIndexChanged

    End Sub

      Private Sub Button6_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button6.Click
        Try
            Dim i As Integer
            For i = 0 To ListView1.Items.Count - 1
                If ListView1.Items(i).Selected = True
Then
                    TextBox1.Text = ListView1.Items(i +
1).SubItems(0).Text
                    Exit For
                End If
            Next
            ListView1.Focus()
            ListView1.FullRowSelect = True
        Catch ex As Exception

        End Try
    End Sub
```

```
      Private Sub Button5_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button5.Click
        Try
            Dim i As Integer
            For i = 0 To ListView1.Items.Count - 1
                 If ListView1.Items(i).Selected = True
Then
                   TextBox1.Text = ListView1.Items(i -
1).SubItems(0).Text
                    Exit For
                End If
            Next
            ListView1.Focus()
            ListView1.FullRowSelect = True
        Catch ex As Exception

        End Try
    End Sub
End Class
```



Issue Book

```
Public Class IssueBook


      Private Sub Button9_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button9.Click
        Me.Close()
    End Sub
```

NOTES

NOTES

```
    Private Sub PictureBox1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)

End Sub

    Private Sub IssueBook_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
    Call Retrive_C()
    Call BookID_Combo()
    Call readData()
End Sub
Sub Retrive_C()
    Try
      If objcon.State = ConnectionState.Closed Then
objcon.Open()
      com = New OleDb.OleDbCommand("Select CID from
Customer", objcon)
       dr = com.ExecuteReader
       While dr.Read
          ComboBox5.Items.Add(dr.Item(0))
       End While
       dr.Close()
       objcon.Close()
    Catch ex As Exception

    End Try
End Sub
Sub BookID_Combo()
    Try
      If objcon.State = ConnectionState.Closed Then
objcon.Open()
       com = New OleDb.OleDbCommand("Select BookID
from Books WHERE status='Available'", objcon)
       dr = com.ExecuteReader
       While dr.Read
          ComboBox1.Items.Add(dr.Item(0))
       End While
       dr.Close()
       objcon.Close()
    Catch ex As Exception
```

```
        End Try
    End Sub
    Sub readData()
        ListView1.Clear()
             ListView1.Columns.Add("BOOK ID", 90,
HorizontalAlignment.Center)
            ListView1.Columns.Add("GROUP ID", 90,
HorizontalAlignment.Center)
           ListView1.Columns.Add("BOOK NAME", 310,
HorizontalAlignment.Center)
            ListView1.Columns.Add("PUBLISHER", 90,
HorizontalAlignment.Center)
             ListView1.Columns.Add("AUTHOR", 90,
HorizontalAlignment.Center)
        ListView1.Columns.Add("PUBLISHING YEAR", 130,
HorizontalAlignment.Center)
             ListView1.Columns.Add("EDITION", 90,
HorizontalAlignment.Center)
             ListView1.Columns.Add("PRICE", 90,
HorizontalAlignment.Center)
             ListView1.Columns.Add("STATUS", 90,
HorizontalAlignment.Center)
       ListView1.GridLines = True
       ListView1.View = View.Details
        Try

           If (objcon.State = ConnectionState.Closed)
Then objcon.Open()
          com = New OleDb.OleDbCommand("SELECT * FROM
Books WHERE status='Available'", objcon)
           dr = com.ExecuteReader
           While dr.Read()
              Call adddatatolistview(ListView1, dr(0),
dr(1), dr(2), dr(3), dr(4), dr(5), dr(6), dr(7), dr(8))
            End While
            dr.Close()
            objcon.Close()
        Catch
                    'MsgBox("Please Refresh",
MsgBoxStyle.Information, "")
        End Try
    End Sub
   Public Sub adddatatolistview(ByVal lvw As ListView,
```

NOTES

NOTES

```
ByVal BookID As String, ByVal GroupID As String, ByVal
BookName As String, ByVal Publisher As String, ByVal
Author As String, ByVal PubYear As String, ByVal edi As
String, ByVal pric As String, ByVal st As String)
        Dim lv As New ListViewItem
        lvw.Items.Add(lv)
        lv.Text = BookID
       lv.SubItems.Add(GroupID)
       lv.SubItems.Add(BookName)
       lv.SubItems.Add(Publisher)
       lv.SubItems.Add(Author)
       lv.SubItems.Add(PubYear)
       lv.SubItems.Add(edi)
       lv.SubItems.Add(pric)
       lv.SubItems.Add(st)
    End Sub
    Sub Retrive()
        objcon.Open()
      com = New OleDb.OleDbCommand("SELECT * FROM Books",
objcon)
       com.ExecuteNonQuery()
       dr = com.ExecuteReader
       dr.Read()
       While (dr.NextResult)
           ComboBox1.Items.Add(dr(1))
       End While
       objcon.Close()
    End Sub


      Private Sub Button2_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button2.Click
        Try
          If objcon.State = ConnectionState.Closed Then
objcon.Open()
           com = New OleDb.OleDbCommand("UPDATE Books
SET status='Rented' WHERE BookID='" & ComboBox1.Text &
"'", objcon)
           com.ExecuteNonQuery()
           objcon.Close()
           Call readData()
         If objcon.State = ConnectionState.Closed Then
objcon.Open()
```

```
            com = New OleDb.OleDbCommand("INSERT INTO
Issue VALUES('" & ComboBox1.Text & "','" & ComboBox2.Text
& "','" & TextBox2.Text & "','" & ComboBox5.Text & "','"
& TextBox1.Text & "','" & DateTimePicker1.Text & "','" &
DateTimePicker2.Text & "')", objcon)
            com.ExecuteNonQuery()
            MsgBox("Book has been Issued!", 0, "")
            Call readData()
            objcon.Close()
        Catch ex As Exception
            MsgBox(ex.Message, 0, "")
        End Try
    End Sub

    Private Sub ComboBox1_SelectedIndexChanged(ByVal
sender As System.Object, ByVal e As System.EventArgs)
Handles ComboBox1.SelectedIndexChanged
        Dim i As Integer
        ListView1.SelectedItems.Clear()
        TextBox1.Focus()
        Try
            If Me.ComboBox1.Text = "" Then
                TextBox2.Text = ""
            Else
                For i = 0 To ListView1.Items.Count - 1
                                    If  ComboBox1.Text  =
ListView1.Items(i).SubItems(0).Text Then
                                        ComboBox2.Text  =
ListView1.Items(i).SubItems(1).Text
                                        TextBox2.Text  =
ListView1.Items(i).SubItems(2).Text
                                        TextBox3.Text  =
ListView1.Items(i).SubItems(3).Text
                                        TextBox4.Text  =
ListView1.Items(i).SubItems(4).Text
                                        ComboBox3.Text  =
ListView1.Items(i).SubItems(5).Text
                                        TextBox5.Text  =
ListView1.Items(i).SubItems(6).Text
                                        TextBox6.Text  =
ListView1.Items(i).SubItems(7).Text
                                        ComboBox4.Text  =
ListView1.Items(i).SubItems(8).Text
```

NOTES

NOTES

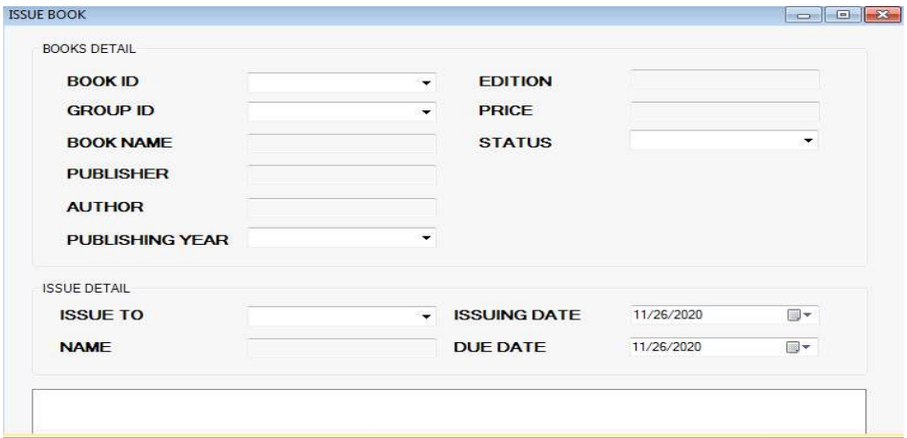
```
                        ListView1.Items(i).Selected =
True
                        Exit For
                    End If
                Next
            End If
        Catch

        End Try
    End Sub

      Private Sub Button8_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button8.Click
        Try
            If ComboBox1.Text = "" Then
                MsgBox("Please mention the BookID", 0,
"")
            Else
              If objcon.State = ConnectionState.Closed
Then
                  com = New OleDb.OleDbCommand("delete
from Issue where BookID='" & ComboBox1.Text & "'", objcon)
                    If MsgBox("Do you really want to
delete?", MsgBoxStyle.YesNo, "Are you sure?") =
Windows.Forms.DialogResult.Yes Then
                      com.ExecuteNonQuery()
                    End If
                    objcon.Close()
                End If
            End If
        Catch ex As Exception

        End Try
    End Sub

     Private Sub ListView1_SelectedIndexChanged(ByVal
sender As System.Object, ByVal e As System.EventArgs)
Handles ListView1.SelectedIndexChanged
        Dim i As Integer
        For i = 0 To ListView1.Items.Count - 1
            If ListView1.Items(i).Selected = True Then
```

```
                                    ComboBox1.Text =
ListView1.Items(i).SubItems(0).Text
                Exit For
            End If
        Next
        ListView1.Focus()
        ListView1.FullRowSelect = True
    End Sub

     Private Sub ComboBox5_SelectedIndexChanged(ByVal
sender As System.Object, ByVal e As System.EventArgs)
Handles ComboBox5.SelectedIndexChanged
        Try
          If objcon.State = ConnectionState.Closed Then
objcon.Open()
         com = New OleDb.OleDbCommand("Select CID,CName
from Customer", objcon)
           dr = com.ExecuteReader
           While dr.Read
               If dr.Item(0) = ComboBox5.Text Then
                   TextBox1.Text = dr.Item(1)
                End If

           End While
           dr.Close()
           objcon.Close()
        Catch ex As Exception

        End Try
    End Sub

      Private Sub Button6_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button6.Click
        Try
           Dim i As Integer
           For i = 0 To ListView1.Items.Count - 1
                If ListView1.Items(i).Selected = True
Then
                   TextBox1.Text = ListView1.Items(i +
1).SubItems(0).Text
                   Exit For
```

NOTES

NOTES

```
                End If
            Next
            ListView1.Focus()
            ListView1.FullRowSelect = True
        Catch ex As Exception

        End Try
    End Sub

    Private Sub Button5_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button5.Click
        Try
            Dim i As Integer
            For i = 0 To ListView1.Items.Count - 1
                If ListView1.Items(i).Selected = True
Then
                    TextBox1.Text = ListView1.Items(i -
1).SubItems(0).Text
                    Exit For
                End If
            Next
            ListView1.Focus()
            ListView1.FullRowSelect = True
        Catch ex As Exception

        End Try
    End Sub
End Class
```



Return Book

NOTES

```
Public Class ReturnBook

      Private Sub Button9_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button9.Click
       Me.Close()
    End Sub

      Private Sub Button2_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button2.Click
       If ComboBox1.Text = "" Then
          MsgBox("Please mention the Book ID", 0, "")
       Else
           Try
              If objcon.State = ConnectionState.Closed
Then objcon.Open()
            com = New OleDb.OleDbCommand("UPDATE Books
SET status='Available' WHERE BookID='" & ComboBox1.Text
& "'", objcon)
              com.ExecuteNonQuery()
              objcon.Close()
              Call readData()
             If objcon.State = ConnectionState.Closed
Then objcon.Open()
            com = New OleDb.OleDbCommand("INSERT INTO
Returns VALUES('" & ComboBox1.Text & "','" & ComboBox2.Text
& "','" & TextBox2.Text & "','" & ComboBox5.Text & "','"
& TextBox1.Text & "','" & TextBox3.Text & "','" &
TextBox7.Text & "','" & DateTimePicker2.Text & "','" &
TextBox6.Text & "')", objcon)
              com.ExecuteNonQuery()
             MsgBox("Book has been returned!", 0, "")
              objcon.Close()
           Catch ex As Exception
              MsgBox(ex.Message, 0, "")
           End Try
       End If
    End Sub

      Private Sub Button8_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
```

NOTES

```
Button8.Click
       If ComboBox1.Text = "" Then
          MsgBox("Please mention a Book ID", 0, "")
        Else


            Try
              If objcon.State = ConnectionState.Closed
Then objcon.Open()
             com = New OleDb.OleDbCommand("DELETE FROM
Returns WHERE BookID='" & ComboBox1.Text & "'", objcon)
              com.ExecuteNonQuery()
              MsgBox("Deleted Success!", 0, "")
               Call ClearThem()
               objcon.Close()
            Catch ex As Exception


            End Try
        End If
    End Sub
    Sub ClearThem()
        ComboBox1.TabIndex = ""
        ComboBox2.Text = ""
        TextBox2.Text = ""
        TextBox3.Text = ""
        TextBox6.Text = ""
        ComboBox5.Text = ""
        TextBox1.Text = ""
        TextBox7.Text = ""
       DateTimePicker2.Refresh()
    End Sub


      Private Sub ReturnBook_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
       Call BookID_Combo()
       Call readData()


    End Sub
    Sub BookID_Combo()
        Try
          If objcon.State = ConnectionState.Closed Then
```

```
objcon.Open()
          com = New OleDb.OleDbCommand("Select BookID
from Books WHERE status='Rented'", objcon)
          dr = com.ExecuteReader
          While dr.Read
             ComboBox1.Items.Add(dr.Item(0))
          End While
          dr.Close()
          objcon.Close()
       Catch ex As Exception

       End Try
   End Sub
   Sub readData()
       ListView1.Clear()
            ListView1.Columns.Add("BOOK  ID",  90,
HorizontalAlignment.Center)
            ListView1.Columns.Add("GROUP  ID",  90,
HorizontalAlignment.Center)
           ListView1.Columns.Add("BOOK  NAME",  310,
HorizontalAlignment.Center)
            ListView1.Columns.Add("PUBLISHER",  90,
HorizontalAlignment.Center)
             ListView1.Columns.Add("AUTHOR",  90,
HorizontalAlignment.Center)
        ListView1.Columns.Add("PUBLISHING YEAR", 130,
HorizontalAlignment.Center)
             ListView1.Columns.Add("EDITION",  90,
HorizontalAlignment.Center)
             ListView1.Columns.Add("PRICE",  90,
HorizontalAlignment.Center)
             ListView1.Columns.Add("STATUS",  90,
HorizontalAlignment.Center)
      ListView1.View = View.Details
       Try

          If (objcon.State = ConnectionState.Closed)
Then objcon.Open()
          com = New OleDb.OleDbCommand("SELECT * FROM
Books WHERE status='Rented'", objcon)
          dr = com.ExecuteReader
          While dr.Read()
             Call adddatatolistview(ListView1, dr(0),
```

NOTES

NOTES

```
dr(1), dr(2), dr(3), dr(4), dr(5), dr(6), dr(7), dr(8))
            End While
            dr.Close()
            objcon.Close()
        Catch
                        'MsgBox("Please Refresh",
MsgBoxStyle.Information, "")
        End Try
    End Sub
   Public Sub adddatatolistview(ByVal lvw As ListView,
ByVal BookID As String, ByVal GroupID As String, ByVal
BookName As String, ByVal Publisher As String, ByVal
Author As String, ByVal PubYear As String, ByVal edi As
String, ByVal pric As String, ByVal st As String)
        Dim lv As New ListViewItem
        lvw.Items.Add(lv)
        lv.Text = BookID
        lv.SubItems.Add(GroupID)
        lv.SubItems.Add(BookName)
        lv.SubItems.Add(Publisher)
        lv.SubItems.Add(Author)
        lv.SubItems.Add(PubYear)
        lv.SubItems.Add(edi)
        lv.SubItems.Add(pric)
        lv.SubItems.Add(st)
    End Sub

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

     Private Sub ListView1_SelectedIndexChanged(ByVal
sender As System.Object, ByVal e As System.EventArgs)
Handles ListView1.SelectedIndexChanged
        Dim i As Integer
        For i = 0 To ListView1.Items.Count - 1
            If ListView1.Items(i).Selected = True Then
                                ComboBox1.Text =
ListView1.Items(i).SubItems(0).Text
                Exit For
```

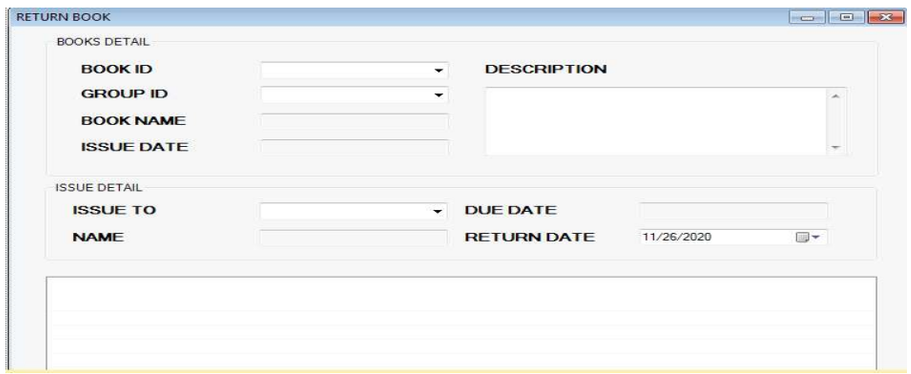
NOTES

```
            End If
        Next
        ListView1.Focus()
        ListView1.FullRowSelect = True
    End Sub

     Private Sub ComboBox1_SelectedIndexChanged(ByVal
sender As System.Object, ByVal e As System.EventArgs)
Handles ComboBox1.SelectedIndexChanged
        Dim i As Integer
       ListView1.SelectedItems.Clear()
        TextBox1.Focus()
        Try
            If Me.ComboBox1.Text = "" Then
                TextBox2.Text = ""
            Else
                For i = 0 To ListView1.Items.Count - 1
                                    If  ComboBox1.Text  =
ListView1.Items(i).SubItems(0).Text Then
                                        ComboBox2.Text  =
ListView1.Items(i).SubItems(1).Text
                                         TextBox2.Text  =
ListView1.Items(i).SubItems(2).Text
                            ListView1.Items(i).Selected =
True
                        Exit For
                    End If
                Next
            End If
        Catch

        End Try
        Call IssueDetail()
    End Sub
    Sub IssueDetail() '
        Try
          If objcon.State = ConnectionState.Closed Then
objcon.Open()
                  com = New OleDb.OleDbCommand("Select
IssueDate, IssueName, IssueTo, DueDate from Issue WHERE
BookID='" & ComboBox1.Text & "'", objcon)
            dr = com.ExecuteReader
```

NOTES

```
            While dr.Read
                ComboBox5.Text = dr.Item(2)
                TextBox1.Text = dr.Item(1)
                TextBox3.Text = dr.Item(0)
                TextBox7.Text = dr.Item(3)
            End While
            dr.Close()
            objcon.Close()
        Catch ex As Exception

        End Try
    End Sub

    Private Sub Button6_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button6.Click
        Try
            Dim i As Integer
            For i = 0 To ListView1.Items.Count - 1
                If ListView1.Items(i).Selected = True
Then
                    TextBox1.Text = ListView1.Items(i +
1).SubItems(0).Text
                    Exit For
                End If
            Next
            ListView1.Focus()
            ListView1.FullRowSelect = True
        Catch ex As Exception

        End Try
    End Sub

    Private Sub Button5_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button5.Click
        Try
            Dim i As Integer
            For i = 0 To ListView1.Items.Count - 1
                If ListView1.Items(i).Selected = True
Then
                    TextBox1.Text = ListView1.Items(i +
```

```
1).SubItems(0).Text
                    Exit For
                End If
            Next
            ListView1.Focus()
            ListView1.FullRowSelect = True
        Catch ex As Exception

        End Try
    End Sub
End Class
```



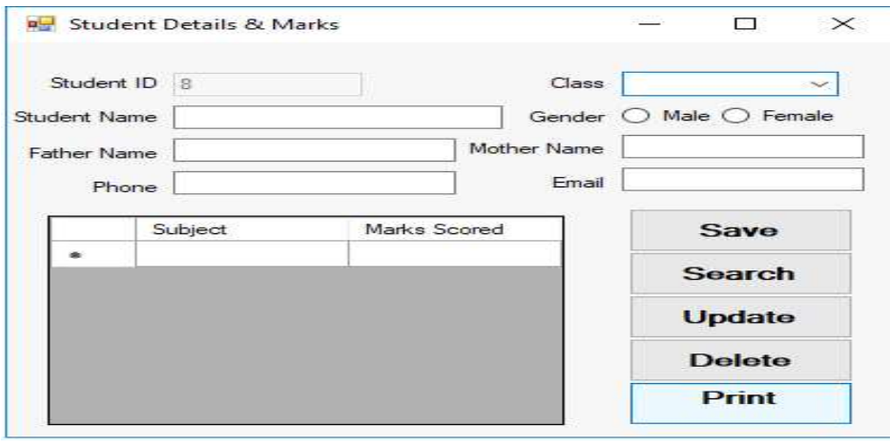
2. Students Marksheet Processing

```
Public conDB As New OleDb.OleDbConnection
   Public Sub connectDB()
       If conDB.State = ConnectionState.Closed Then
            conDB.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0; Data Source=" & Application.StartupPath & "\stuDB.accdb"
           conDB.Open()
       End If
   End Sub
   Function getNewID(tblName As String, fldName As String) As String
       Dim strVal, sql As String
       Dim cmd As OleDb.OleDbCommand
       connectDB()
       sql = "select max(" & fldName & ") from " & tblName
       cmd = New OleDb.OleDbCommand(sql, conDB)
       strVal = Convert.ToString(cmd.ExecuteScalar())
       If strVal = "" Then
```

NOTES

NOTES

```
            strVal = "1"
        Else
            strVal = Convert.ToString(CInt(strVal) + 1)
        End If
        Return strVal
    End Function
```



Button Click

```
Dim strSQL As String
 Dim gndr As String
 Dim i As Integer
 If rdbFemale.Checked = True Then
     gndr = "Female"
 Else
     gndr = "Male"
 End If
  strSQL = "insert into studentmaster values(" &
txtStuID.Text & ",'" & cboClass.Text & "','" &
txtStuName.Text & "','" & txtFName.Text & "','" &
txtMName.Text & "','" & gndr & "','" & txtPhone.Text &
"','" & txtEmail.Text & "')"
 cmd = New OleDb.OleDbCommand(strSQL, conDB)
 cmd.ExecuteNonQuery()
 For i = 0 To dgvMarks.RowCount - 2
       strSQL = "insert into studentmarks values(" &
txtStuID.Text & ",'" & dgvMarks.Item(0, i).Value & "'," &
dgvMarks.Item(1, i).Value & ")"
     cmd = New OleDb.OleDbCommand(strSQL, conDB)
     cmd.ExecuteNonQuery()
 Next
```

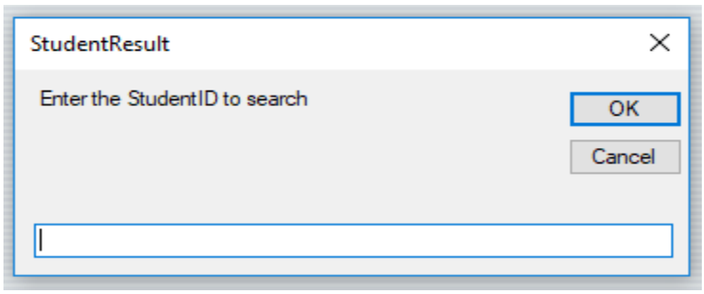
Search Button:

```
Dim sid, cnt As Integer
       Dim drl As OleDb.OleDbDataReader
       Dim cmdl As New OleDb.OleDbCommand
          sid = CInt(InputBox("Enter the StudentID to
search"))
      cmdl = New OleDbPress Ctrl+V to copy the following
code

Dim sid, cnt As Integer
       Dim drl As OleDb.OleDbDataReader
       Dim cmdl As New OleDb.OleDbCommand
          sid = CInt(InputBox("Enter the StudentID to
search"))
         cmdl = New OleDb.OleDbCommand("select * from
studentmaster where stuid=" & sid, conDB)
       drl = cmdl.ExecuteReader()
       If drl.Read() Then
           txtStuID.Text = drl.Item(0)
           cboClass.Text = drl.Item(1)
           txtStuName.Text = drl.Item(2)
           txtFName.Text = drl.Item(3)
           txtMName.Text = drl.Item(4)
           If drl.Item(5) = "Female" Then
               rdbFemale.Checked = True
           Else
               rdbMale.Checked = True
           End If
           txtPhone.Text = drl.Item(6)
           txtEmail.Text = drl.Item(7)
           drl.Close()
         cmdl = New OleDb.OleDbCommand("select subject,
marks from studentmarks where stuid=" & sid, conDB)
           drl = cmdl.ExecuteReader()
           dgvMarks.Rows.Clear()
           cnt = 0
           While drl.Read()
               dgvMarks.Rows.Add()
                       dgvMarks.Item(0, cnt).Value =
Convert.ToString(drl.Item(0))
                       dgvMarks.Item(1, cnt).Value =
```

NOTES

NOTES

```
Convert.ToString(drl.Item(1))
                cnt = cnt + 1
            End While
        Else
           MsgBox("No student with this ID")
        End If


.OleDbCommand("select * from studentmaster where stuid="
& sid, conDB)
       drl = cmdl.ExecuteReader()
       If drl.Read() Then
           txtStuID.Text = drl.Item(0)
           cboClass.Text = drl.Item(1)
           txtStuName.Text = drl.Item(2)
           txtFName.Text = drl.Item(3)
           txtMName.Text = drl.Item(4)
           If drl.Item(5) = "Female" Then
               rdbFemale.Checked = True
            Else
               rdbMale.Checked = True
            End If
           txtPhone.Text = drl.Item(6)
           txtEmail.Text = drl.Item(7)
           drl.Close()
         cmdl = New OleDb.OleDbCommand("select subject,
marks from studentmarks where stuid=" & sid, conDB)
           drl = cmdl.ExecuteReader()
           dgvMarks.Rows.Clear()
            cnt = 0
           While drl.Read()
               dgvMarks.Rows.Add()
                       dgvMarks.Item(0,  cnt).Value  =
Convert.ToString(drl.Item(0))
                       dgvMarks.Item(1,  cnt).Value  =
Convert.ToString(drl.Item(1))
                cnt = cnt + 1
            End While
        Else
           MsgBox("No student with this ID")
        End If
```



NOTES

Button Update:

```
Dim strSQL As String
       Dim gndr As String
       Dim i As Integer
       If rdbFemale.Checked = True Then
           gndr = "Female"
       Else
           gndr = "Male"
       End If
      strSQL = "update studentmaster set stuClass='" &
cboClass.Text & "', StuName='" & txtStuName.Text & "',
StuFname='" _
                    & txtFName.Text  & "',StuMName='" &
txtMName.Text & "',StuGender='" & gndr & "',StuPhone='" &
txtPhone.Text _
             & "',StuEmail='" & txtEmail.Text & "' where
StuID=" & CInt(txtStuID.Text)
      cmd = New OleDb.OleDbCommand(strSQL, conDB)
      cmd.ExecuteNonQuery()
       ' delete all records from marks table to add the
new marks and subjects
     strSQL = "delete * from studentmarks  where StuID="
& CInt(txtStuID.Text)
      cmd = New OleDb.OleDbCommand(strSQL, conDB)
      cmd.ExecuteNonQuery()
     ' Insert the new subjects and marks for the student
      For i = 0 To dgvMarks.RowCount - 2
         strSQL = "insert into studentmarks values(" &
txtStuID.Text & ",'" & dgvMarks.Item(0, i).Value & "'," &
dgvMarks.Item(1, i).Value & ")"
          cmd = New OleDb.OleDbCommand(strSQL, conDB)
          cmd.ExecuteNonQuery()
       Next
```

NOTES

Button Delete:

```
Dim strSQL As String
      ' delete the record of student from master table
     strSQL = "delete * from studentmaster  where StuID="
& CInt(txtStuID.Text)
      cmd = New OleDb.OleDbCommand(strSQL, conDB)
      cmd.ExecuteNonQuery()
      ' delete all records from marks table
     strSQL = "delete * from studentmarks  where StuID="
& CInt(txtStuID.Text)
      cmd = New OleDb.OleDbCommand(strSQL, conDB)
      cmd.ExecuteNonQuery()
```

Print Button:

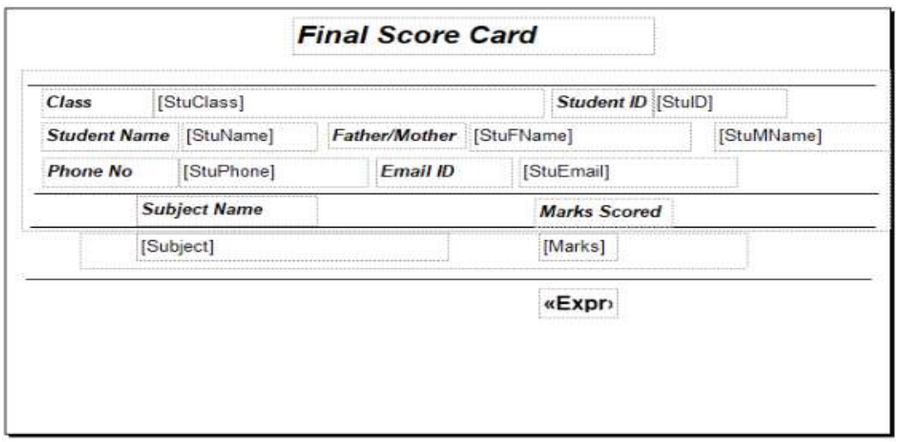
```
Dim frm As New Form2 ' creates an object of form containing
the reportviewer
 frm.Show()' displays the report
```

Report Viewer:

```
Private Sub Form2_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
       Dim dt1, dt2 As New DataTable
       Dim sid As Integer
       connectDB()
                                              sid     =
CInt(frmStuDetails.Controls("txtStuID").Text)
      Dim cmd1 As New OleDb.OleDbCommand("SELECT * from
StudentMarks where stuid=" & sid, conDB)
       cmd1.CommandTimeout = 4096
       Dim ta1 As New OleDb.OleDbDataAdapter(cmd1)
       ta1.Fill(dt1)
      Dim cmd2 As New OleDb.OleDbCommand("SELECT * from
StudentMaster where stuid=" & sid, conDB)
       cmd2.CommandTimeout = 4096
       Dim ta2 As New OleDb.OleDbDataAdapter(cmd2)
```

```
        ta2.Fill(dt2)
       With Me.ReportViewer1.LocalReport
           .DataSources.Clear()
                              .DataSources.Add(New
Microsoft.Reporting.WinForms.ReportDataSource("DataSet1",
dt1))
                              .DataSources.Add(New
Microsoft.Reporting.WinForms.ReportDataSource("DataSet2",
dt2))
        End With
       Me.ReportViewer1.RefreshReport()
    End Sub
```

NOTES



3. Telephone Directory Maintenance

```
Imports System.IO
Imports System.IO.Directory
Imports System.IO.DirectoryInfo
Imports System.IO.Path
Imports System.Environment
Imports System.IO.FileStream
Imports System.IO.File
Imports System.IO.FileInfo
Imports System.Data.SqlClient
Imports System.Data
Imports System.Data.OleDb

Public Class frmPonBuk

    Dim strPath As String
    Dim dsContact As New DataSet
```

NOTES

```
    Dim dsContactNam As New DataSet
    Dim daContact As New OleDbDataAdapter
    Dim daContactNam As New OleDbDataAdapter
    Dim sqlCommand As New OleDbCommand
    Dim strAction As String
    Dim strSQL As String
    Dim dt As New DataTable
    Dim dtContact As New DataTable
    Dim dtSearch As New DataTable
    Dim daSearch As New OleDbDataAdapter
    Dim dsSearch As New DataSet
    Dim drDSRow As DataRow
    Dim drNewRow As DataRow
    Dim cnPhoneBook As New OleDbConnection

   Private Sub frmPonBuk_KeyDown(ByVal sender As Object,
ByVal e As System.Windows.Forms.KeyEventArgs) Handles
Me.KeyDown
        'code for short cut key, note this will work if
you
        'set the form's keypreview property to true

        Select Case e.KeyCode
            Case Keys.F8
                If Me.cmdAdd.Enabled = True Then
                    Me.cmdAdd_Click(sender, e)
                End If
            Case Keys.F9
                If Me.cmdEdit.Enabled = True Then
                    Me.cmdEdit_Click(sender, e)
                End If
            Case Keys.F10
                If Me.cmdDelete.Enabled = True Then
                    Me.cmdDelete_Click(sender, e)
                End If
            Case Keys.F11
                If Me.cmdUpdate.Enabled = True Then
                    Me.cmdUpdate_Click(sender, e)
                End If
            Case Keys.F12
                If Me.cmdCancel.Enabled = True Then
```

```
                Me.cmdCancel_Click(sender, e)
            End If
        Case Keys.Enter
            SendKeys.Send("{TAB}")

    End Select
End Sub

    Private Sub frmPonBuk_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
    'Dim strPath As String
     'you can use this method in order to get your
database(Access) path
    'strPath = System.Environment.CurrentDirectory &
"\Data\PhoneBook.accdb"
          'cnPhoneBook.ConnectionString = "
Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" &
specialName & ";Persist Security Info=False;"

          cnPhoneBook.ConnectionString = "
Provider=Microsoft.ACE.OLEDB.12.0;Data Source=../Data/
PhoneBook.accdb;Persist Security Info=False;"
    strSQL = " SELECT [LastName]+', '+[FirstName]+'
'+[MiddleName] AS Name, TblContact.* FROM TblContact ORDER
BY [LastName]+', '+[FirstName]+' '+[MiddleName];"
   daContact.SelectCommand = New OleDbCommand(strSQL,
cnPhoneBook)
    daContact.Fill(dsContact, "TblContact")
    Me.dtContact = dsContact.Tables("TblContact")
     'binding controls to dataset
   Me.txtLstNam.DataBindings.Add("Text", dsContact,
"TblContact.LastName")
   Me.txtFstNam.DataBindings.Add("Text", dsContact,
"TblContact.FirstName")
   Me.txtMidNam.DataBindings.Add("Text", dsContact,
"TblContact.MiddleName")
   Me.txtHomAdr.DataBindings.Add("Text", dsContact,
"TblContact.HomeAdr")
   Me.txtBusAdr.DataBindings.Add("Text", dsContact,
"TblContact.BusAdr")
    Me.txtTelNo.DataBindings.Add("Text", dsContact,
```

NOTES

NOTES

```
"TblContact.TelNo")
      Me.txtMobNo.DataBindings.Add("Text", dsContact,
"TblContact.MobNo")
       Me.txtEml.DataBindings.Add("Text", dsContact,
"TblContact.EMail")

       'setting datagrid properties
       Me.dtgContact.DataSource = dsContact
       Me.dtgContact.DataMember = "TblContact"
       Me.dtgContact.Columns(0).HeaderText = "Name"
       Me.dtgContact.Columns(1).Visible = False
       Me.dtgContact.Columns(2).Visible = False
       Me.dtgContact.Columns(3).Visible = False
       Me.dtgContact.Columns(4).Visible = False
          Me.dtgContact.Columns(5).HeaderText = "Home
Address"
          Me.dtgContact.Columns(6).HeaderText = "Bus.
Address"
      Me.dtgContact.Columns(7).HeaderText = "Telephone"
       Me.dtgContact.Columns(8).HeaderText = "Mobile"
       Me.dtgContact.Columns(9).HeaderText = "E-Mail"

       'Used SQL statement for Combo box to display the
name of contact person
            strSQL = " SELECT TblContact.ContactID,
[LastName]+', '+[FirstName]+' '+[MiddleName] AS Name FROM
TblContact ORDER BY [LastName]+', '+[FirstName]+'
'+[MiddleName];"
             daContactNam.SelectCommand = New
OleDbCommand(strSQL, cnPhoneBook)
      daContactNam.Fill(dsContactNam, "TblContact")

       'datatable for combo box
       Me.dt = dsContactNam.Tables("TblContact")
       Me.cmbSearch.DataSource = dt
       Me.cmbSearch.DisplayMember = "Name"
       Me.cmbSearch.ValueMember = "ContactID"
       Me.cmbSearch.SelectedIndex = -1
            Me.txtRecPos.Text = "Contact Record " &
Me.BindingContext(dsContact, "TblContact").Position + 1
& " of : " & dsContact.Tables("TblContact").Rows.Count
       ' call procedure to lock the text field
       lockField()
```

```
        ' call procedure to disabled update
        UpdtOff()
    End Sub

      Private Sub cmdFstRec_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cmdFstRec.Click
                  Me.BindingContext(dsContact,
"TblContact").Position = 0
            Me.txtRecPos.Text = "Contact Record " &
Me.BindingContext(dsContact, "TblContact").Position + 1
& " of : " & dsContact.Tables("TblContact").Rows.Count
    End Sub

   Private Sub cmdPrv_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdPrv.Click
                  Me.BindingContext(dsContact,
"TblContact").Position = Me.BindingContext(dsContact,
"TblContact").Position - 1
            Me.txtRecPos.Text = "Contact Record " &
Me.BindingContext(dsContact, "TblContact").Position + 1
& " of : " & dsContact.Tables("TblContact").Rows.Count
    End Sub

      Private Sub cmdNext_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cmdNext.Click
                  Me.BindingContext(dsContact,
"TblContact").Position = Me.BindingContext(dsContact,
"TblContact").Position + 1
            Me.txtRecPos.Text = "Contact Record " &
Me.BindingContext(dsContact, "TblContact").Position + 1
& " of : " & dsContact.Tables("TblContact").Rows.Count
    End Sub

   Private Sub cmdLst_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdLst.Click
                  Me.BindingContext(dsContact,
"TblContact").Position = Me.BindingContext(dsContact,
"TblContact").Count - 1
            Me.txtRecPos.Text = "Contact Record " &
Me.BindingContext(dsContact, "TblContact").Position + 1
& " of : " & dsContact.Tables("TblContact").Rows.Count
    End Sub
   Private Sub UnlockField()
```

NOTES

NOTES

```
    Me.txtFstNam.ReadOnly = False
    Me.txtLstNam.ReadOnly = False
    Me.txtMidNam.ReadOnly = False
    Me.txtHomAdr.ReadOnly = False
    Me.txtBusAdr.ReadOnly = False
    Me.txtTelNo.ReadOnly = False
    Me.txtMobNo.ReadOnly = False
    Me.txtEml.ReadOnly = False

End Sub
Private Sub lockField()

    Me.txtFstNam.ReadOnly = True
    Me.txtLstNam.ReadOnly = True
    Me.txtMidNam.ReadOnly = True
    Me.txtHomAdr.ReadOnly = True
    Me.txtBusAdr.ReadOnly = True
    Me.txtTelNo.ReadOnly = True
    Me.txtMobNo.ReadOnly = True
    Me.txtEml.ReadOnly = True

End Sub
Private Sub UpdtOff()

    Me.cmdAdd.Enabled = True
    Me.cmdEdit.Enabled = True
    Me.cmdDelete.Enabled = True
    Me.cmdUpdate.Enabled = False
    Me.cmdCancel.Enabled = False

    Me.cmdAdd.BackColor = Color.Tan
    Me.cmdEdit.BackColor = Color.Tan
    Me.cmdDelete.BackColor = Color.Tan
    Me.cmdUpdate.BackColor = Color.Black
    Me.cmdCancel.BackColor = Color.Black
End Sub
Private Sub UpdtOn()

    Me.cmdAdd.Enabled = False
```

```
        Me.cmdEdit.Enabled = False
        Me.cmdDelete.Enabled = False
        Me.cmdUpdate.Enabled = True
        Me.cmdCancel.Enabled = True

        Me.cmdAdd.BackColor = Color.Black
        Me.cmdEdit.BackColor = Color.Black
        Me.cmdDelete.BackColor = Color.Black
        Me.cmdUpdate.BackColor = Color.Tan
        Me.cmdCancel.BackColor = Color.Tan

    End Sub

   Private Sub cmdAdd_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdAdd.Click
        strAction = "ADD"
        UpdtOn()
        UnlockField()
                  Me.BindingContext(dsContact,
"TblContact").AddNew()
        Me.txtLstNam.Focus()
    End Sub

      Private Sub cmdEdit_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cmdEdit.Click
        strAction = "EDIT"
        UpdtOn()
        UnlockField()
    End Sub

      Private Sub cmdDelete_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cmdDelete.Click
        Dim delCommand As New OleDbCommand
        Dim intPos As Integer
        Dim intContactID As Integer
        Dim strUsrRsp As String
             intPos = Me.BindingContext(dsContact,
"TblContact").Position
       intContactID = dtContact.Rows(intPos).Item(1)
        strUsrRsp = MsgBox("Do you want to delete this
```

NOTES

NOTES

```
record", MsgBoxStyle.YesNo + MsgBoxStyle.Question +
MsgBoxStyle.ApplicationModal, "Phone Book")
       If strUsrRsp = MsgBoxResult.Yes Then
            Try
                cnPhoneBook.Open()
                 strSQL = "Delete from TblContact where
(ContactID = " & intContactID & ")"

                  sqlCommand = New OleDbCommand(strSQL,
cnPhoneBook)

               sqlCommand.ExecuteNonQuery()

               cnPhoneBook.Close()
               dsContact.Clear()
              daContact.Fill(dsContact, "TblContact")
                   MsgBox("Record has been deleted",
MsgBoxStyle.OkOnly + MsgBoxStyle.Information +
MsgBoxStyle.ApplicationModal, "Phone Book")
           Catch ex As Exception
               MsgBox(Err.Description)
            End Try
        Else

        End If
       dsContactNam.Clear()
      daContactNam.Fill(dsContactNam, "TblContact")
       cmbSearch.SelectedIndex = -1
   End Sub

      Private Sub cmdUpdate_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cmdUpdate.Click
       Dim SubPos As Integer
       Dim intPos As Integer
       Dim intContactID As Integer

        Try
           Select Case strAction
                Case "ADD"
                          Me.BindingContext(dsContact,
"TblContact").EndCurrentEdit()
```

```
                cnPhoneBook.Open()
                  strSQL = "INSERT INTO TblContact
(LastName, FirstName, MiddleName, HomeAdr, BusAdr, TelNo,
MobNo, EMail) "
                  strSQL = strSQL & " VALUES ('" &
Me.txtLstNam.Text & "','" & Me.txtFstNam.Text & "','" &
Me.txtMidNam.Text & "','" & Me.txtHomAdr.Text & "','" &
Me.txtBusAdr.Text & "','" & Me.txtTelNo.Text & "','" &
Me.txtMobNo.Text & "','" & Me.txtEml.Text & "');"
              sqlCommand = New OleDbCommand(strSQL,
cnPhoneBook)

               sqlCommand.ExecuteNonQuery()

               cnPhoneBook.Close()
               dsContact.Clear()
                        daContact.Fill(dsContact,
"TblContact")

            Case "EDIT"
              intPos = Me.BindingContext(dsContact,
"TblContact").Position
                                    intContactID =
dtContact.Rows(intPos).Item(1)

                        Me.BindingContext(dsContact,
"TblContact").EndCurrentEdit()
                cnPhoneBook.Open()

                  strSQL = "UPDATE TblContact SET
LastName = '" & Me.txtLstNam.Text & "', FirstName = '" &
Me.txtFstNam.Text & "', MiddleName = '" & Me.txtMidNam.Text
& "', HomeAdr = '" & Me.txtHomAdr.Text & "', "
                  strSQL = strSQL & " BusAdr = '" &
Me.txtBusAdr.Text & "', TelNo = '" & Me.txtTelNo.Text &
"', MobNo = '" & Me.txtMobNo.Text & "', EMail = '" &
Me.txtEml.Text & "' WHERE (((TblContact.ContactID)=" &
intContactID & "));"
              sqlCommand = New OleDbCommand(strSQL,
cnPhoneBook)

               sqlCommand.ExecuteNonQuery()
               cnPhoneBook.Close()
              SubPos = Me.BindingContext(dsContact,
"TblContact").Position
```

NOTES

NOTES

```
                    dsContact.Clear()
                        daContact.Fill(dsContact,
"TblContact")
                    Me.BindingContext(dsContact,
"TblContact").Position = SubPos

            End Select
            UpdtOff()
            lockField()
        Catch ex As Exception
            MsgBox(strSQL)
        End Try
        dsContactNam.Clear()
       daContactNam.Fill(dsContactNam, "TblContact")
            Me.txtRecPos.Text = "Contact Record " &
Me.BindingContext(dsContact, "TblContact").Position + 1
& " of : " & dsContact.Tables("TblContact").Rows.Count
    End Sub

      Private Sub cmdCancel_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cmdCancel.Click

                Me.BindingContext(dsContact,
"TblContact").CancelCurrentEdit()
        UpdtOff()
        lockField()
            Me.txtRecPos.Text = "Contact Record " &
Me.BindingContext(dsContact, "TblContact").Position + 1
& " of : " & dsContact.Tables("TblContact").Rows.Count

    End Sub

      Private Sub cmdSearch_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cmdSearch.Click
        Dim ContactIDSrh As Integer
        Dim ColNum As Integer
        Dim RowNum As Integer
        Dim RecCount As Integer
        ColNum = 0
        RowNum = 0
        'Check Combo box if it has a value
```

NOTES

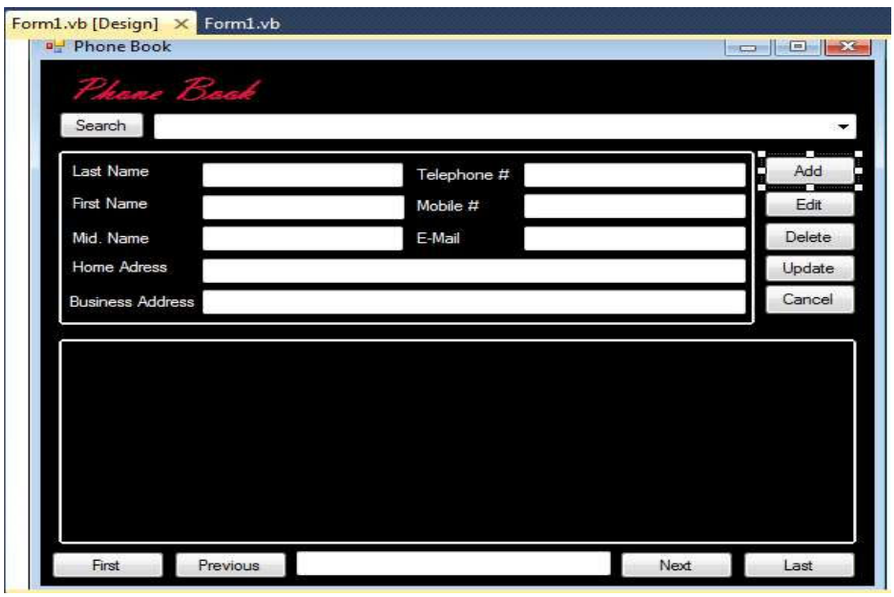
```
        If Me.cmbSearch.SelectedValue <> 0 Then
                RecCount = Me.BindingContext(dsContact,
"TblContact").Count
            ContactIDSrh = Me.cmbSearch.SelectedValue
             'move at first record
                      Me.BindingContext(dsContact,
"TblContact").Position = 0
           'loop until we find the desired Contact Person
                    Do While ContactIDSrh <>
dtContact.Rows(RowNum).Item(1)
                 If RowNum <> RecCount Then
                     'move record position
                           Me.BindingContext(dsContact,
"TblContact").Position = RowNum + 1
                     RowNum = RowNum + 1
                 Else
                     'exit loop if record found
                     Exit Do
                 End If
             Loop
         Else
            MsgBox("Please Select the Student name to be
searched")
         End If
    End Sub

     Private Sub txtLstNam_LostFocus(ByVal sender As
Object, ByVal e As System.EventArgs) Handles
txtLstNam.LostFocus
       'this will trigger if the txtLstNam has lost the
focus and during adding new or editting existing record
       If strAction = "ADD" Or strAction = "EDIT" Then
           'transform the string into proper case
          Me.txtLstNam.Text = StrConv(Me.txtLstNam.Text,
VbStrConv.ProperCase)
       End If
    End Sub

     Private Sub txtFstNam_LostFocus(ByVal sender As
Object, ByVal e As System.EventArgs) Handles
txtFstNam.LostFocus
       If strAction = "ADD" Or strAction = "EDIT" Then
          Me.txtFstNam.Text = StrConv(Me.txtFstNam.Text,
```

```
VbStrConv.ProperCase)
        End If
    End Sub
```

NOTES

```
     Private Sub txtMidNam_LostFocus(ByVal sender As
Object, ByVal e As System.EventArgs) Handles
txtMidNam.LostFocus
       If strAction = "ADD" Or strAction = "EDIT" Then
                              Me.txtMidNam.Text  =
StrConv(Me.txtMidNam.Text, VbStrConv.ProperCase)
        End If
    End Sub

     Private Sub txtHomAdr_LostFocus(ByVal sender As
Object, ByVal e As System.EventArgs) Handles
txtHomAdr.LostFocus
       If strAction = "ADD" Or strAction = "EDIT" Then
                              Me.txtHomAdr.Text  =
StrConv(Me.txtHomAdr.Text, VbStrConv.ProperCase)
        End If
    End Sub

     Private Sub txtBusAdr_LostFocus(ByVal sender As
Object, ByVal e As System.EventArgs) Handles
txtBusAdr.LostFocus
       If strAction = "ADD" Or strAction = "EDIT" Then
                              Me.txtBusAdr.Text  =
StrConv(Me.txtBusAdr.Text, VbStrConv.ProperCase)
        End If
    End Sub

   Private Sub txtTelNo_LostFocus(ByVal sender As Object,
ByVal e As System.EventArgs) Handles txtTelNo.LostFocus
       If strAction = "ADD" Or strAction = "EDIT" Then
           If Len(Me.txtTelNo.Text) = 7 Then
              Me.txtTelNo.Text = Mid(Me.txtTelNo.Text,
1, 3) & "-" & Mid(Me.txtTelNo.Text, 4, 2) & "-" &
Mid(Me.txtTelNo.Text, 6, 2)
            End If
        End If
    End Sub

     Private Sub dtgContact_CellClick(ByVal sender As
```

```
Object,            ByVal            e            As
System.Windows.Forms.DataGridViewCellEventArgs) Handles
dtgContact.CellClick
            Me.txtRecPos.Text = "Contact Record " &
Me.BindingContext(dsContact, "TblContact").Position + 1
& " of : " & dsContact.Tables("TblContact").Rows.Count
    End Sub

End Class
```



4. Gas Booking and Delivering

Main:

```
Private Sub Command1_Click() Handles Command1.Click
'#Const Compile_Command1_Click = True
#If Compile_Command1_Click Or CompileAll_Form1 Then
       Form2.Load()
       Form2.Show()
       Close()
#End If   ' Compile_Command1_Click
    End Sub

    Private Sub Command2_Click(ByVal sender As Object,
ByVal e As System.EventArgs) Handles Command2.Click
'#Const Compile_Command2_Click = True
#If Compile_Command2_Click Or CompileAll_Form1 Then
```

NOTES

```
        Form15.Load()
        Form15.Show()
        Close()
#End If   ' Compile_Command2_Click
    End Sub

    Private Sub Command3_Click(ByVal sender As Object,
ByVal e As System.EventArgs) Handles Command3.Click
'#Const Compile_Command3_Click = True
#If Compile_Command3_Click Or CompileAll_Form1 Then
      'b = InputBox("Enter Record No", "Find to Modify")
        Form6.Load()
        Form6.Show()
        Close()
#End If   ' Compile_Command3_Click
    End Sub

   Private Sub Command4_Click() Handles Command4.Click
'#Const Compile_Command4_Click = True
#If Compile_Command4_Click Or CompileAll_Form1 Then
        Form16.Load()
        Form16.Show()
        Close()
#End If   ' Compile_Command4_Click
    End Sub

    Private Sub Command5_Click(ByVal sender As Object,
ByVal e As System.EventArgs) Handles Command5.Click
'#Const Compile_Command5_Click = True
#If Compile_Command5_Click Or CompileAll_Form1 Then
        Form5.Load()
        Form5.Show()
        Close()
#End If   ' Compile_Command5_Click
    End Sub

    Private Sub Command6_Click(ByVal sender As Object,
ByVal e As System.EventArgs) Handles Command6.Click
'#Const Compile_Command6_Click = True
#If Compile_Command6_Click Or CompileAll_Form1 Then
        Form7.Load()
```

```
        Form7.Show()
        Close()
#End If   ' Compile_Command6_Click
    End Sub

   Private Sub Command7_Click() Handles Command7.Click
'#Const Compile_Command7_Click = True
#If Compile_Command7_Click Or CompileAll_Form1 Then
       Application.Exit()
#End If   ' Compile_Command7_Click
    End Sub

   Private Sub Command8_Click() Handles Command8.Click
'#Const Compile_Command8_Click = True
#If Compile_Command8_Click Or CompileAll_Form1 Then
        Form8.Load()
        Form8.Show()
        Close()
#End If   ' Compile_Command8_Click
    End Sub

    Private Sub Command9_Click(ByVal sender As Object,
ByVal e As System.EventArgs) Handles Command9.Click
'#Const Compile_Command9_Click = True
#If Compile_Command9_Click Or CompileAll_Form1 Then
        Form14.Load()
        Form14.Show()
        Close()
#End If   ' Compile_Command9_Click
    End Sub

   Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
'#Const Compile_Form_Load = True
#If Compile_Form_Load Or CompileAll_Form1 Then
        Timer1.Interval = 50
#End If   ' Compile_Form_Load
    End Sub

   Private Sub Timer1_Tick(ByVal sender As Object, ByVal
e As System.EventArgs) Handles Timer1.Tick
```

NOTES

NOTES

```
'#Const Compile_Timer1_Timer = True
#If Compile_Timer1_Timer Or CompileAll_Form1 Then
        l1.Top -= 60
        If l1.Top<=100 Then
            l1.Top = 13000
        End If

        L2.Top -= 60
        If L2.Top<=100 Then
            L2.Top = 13000
        End If

        L3.Top -= 60
        If L3.Top<=100 Then
            L3.Top = 13000
        End If

        L4.Top -= 60
        If L4.Top<=100 Then
            L4.Top = 13000
        End If

        L5.Top -= 60
        If L5.Top<=100 Then
            L5.Top = 13000
        End If

        L6.Top -= 60
        If L6.Top<=60 Then
            L6.Top = 13000
        End If

        l7.Top -= 60
        If l7.Top<=60 Then
            l7.Top = 13000
        End If

        l8.Top -= 60
        If l8.Top<=60 Then
            l8.Top = 13000
```

NOTES

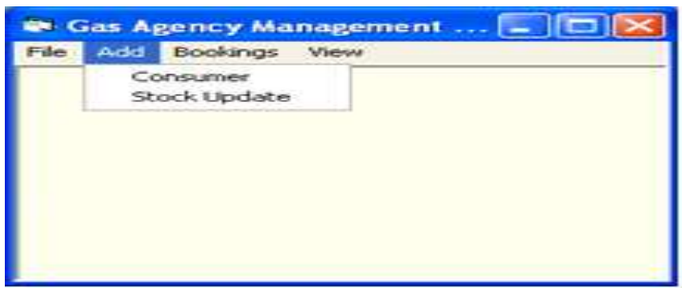
```
        End If

        l9.Top -= 60
        If l9.Top<=60 Then
            l9.Top = 13000
        End If
#End If   ' Compile_Timer1_Timer
    End Sub

   Private Sub Timer2_Tick(ByVal sender As Object, ByVal
e As System.EventArgs) Handles Timer2.Tick
'#Const Compile_Timer2_Timer = True
#If Compile_Timer2_Timer Or CompileAll_Form1 Then
                              Label4.ForeColor    =
ColorTranslator.FromOle(QBColor(Rnd()*15))
                              Label5.ForeColor    =
ColorTranslator.FromOle(QBColor(Rnd()*15))
#End If   ' Compile_Timer2_Timer
    End Sub

End Class
```

NOTES

File Menu:

Booking Menu:

Add Menu:



5. Electricity Bill Management Main form:

```
Private Sub Cmdexit_Click()
End
End Sub

Private Sub Cmd1_Click()
txtuser.Text = UCase(txtuser)
txtpass.Text = UCase(txtpass) '& LCase(txtpass)
If txtuser.Text = "ELECTRICITY" And txtpass = "KULKARNI"
Then
```

```
Main.Show
Me.Hide
Else
MsgBox ("Please try again")
txtuser.SetFocus
End If
End Sub

Private Sub Cmd2_Click()
End
End Sub

Customer Form:
Private Sub Cmdadd_Click()
Adodc1.Refresh
Adodc1.Recordset.AddNew
End Sub

Private Sub cmdclear_Click()
Adodc1.Refresh
cmbgn.Text = ""
txtnm.Text = ""
txtad.Text = ""
cmbec.Text = ""
cmbct.Text = ""
Txtpn.Text = ""
cmbpro.Text = ""
Txtdob.Text = ""
End Sub

Private Sub cmdsv_Click()
If cmbgn.Text = "" Or txtnm.Text = "" Or cmbec.Text = ""
Or cmbpro.Text = "" Or Txtdob.Text = "" Then
MsgBox "Please Fill Requireds Fields Then Save Your Record"
Else
Adodc1.Recordset.Fields(0) = cmbgn.Text
Adodc1.Recordset.Fields(1) = txtnm.Text
Adodc1.Recordset.Fields(2) = txtad.Text
Adodc1.Recordset.Fields(3) = cmbec.Text
Adodc1.Recordset.Fields(4) = cmbct.Text
```

NOTES

NOTES

```
Adodc1.Recordset.Fields(5) = Txtpn.Text
Adodc1.Recordset.Fields(6) = cmbpro.Text
Adodc1.Recordset.Fields(7) = Text1.Text 'lbldt.Caption
Adodc1.Recordset.Fields(8) = Txtdob.Text
Adodc1.Recordset.Save
Adodc1.Refresh
MsgBox "Record Save Successfully"

cmbgn.Text = ""
txtnm.Text = ""
txtad.Text = ""
cmbec.Text = ""
cmbct.Text = ""
Txtpn.Text = ""
cmbpro.Text = ""
Txtdob.Text = ""
End If
End Sub

Private Sub Command5_Click()
Unload Me
End Sub


Private Sub Form_Load()
'Adodc1.Refresh
cmbgn.Text = ""
txtnm.Text = ""
txtad.Text = ""
cmbec.Text = ""
cmbct.Text = ""
Txtdob.Text = ""
Txtpn.Text = ""
cmbpro.Text = ""
Text1.Text = Date

'FormatDateTime((DateTime.Day) & ("-") & (DateTime.Month)
& ("-") & (DateTime.Year))
'd & "/" & m & "/" & y
'lbldt.Caption = FormatDateTime(DateTime.Date, vbLongDate)
```

```
'vbGeneralDate
'DateTime.Date
End Sub
```

Bill:

```
Private Sub Cmbnm2_LostFocus()
'On Error Resume Next
'Adodc1.Refresh
'While Not Adodc1.Recordset.EOF = True
'If Adodc1.Recordset!Name = Cmbnm2.Text Then
'txtadd.Text = Adodc1.Recordset!Add 'ress
'Txtex.Text = Adodc1.Recordset!Exchange
'Txtpin.Text = Adodc1.Recordset!pincode
'Else
''
''Exit Do
'End If
'Loop
Adodc1.Refresh
Adodc2.Refresh
Do While Adodc1.Recordset.EOF = False
If Adodc1.Recordset!Name = Cmbnm2.Text Then
Txtadd.Text = Adodc1.Recordset!Add
Txtex.Text = Adodc1.Recordset!Exchange
Txtpin.Text = Adodc1.Recordset!pincode
Text1.Text = Adodc1.Recordset!plan
Exit Do
End If
'End If
```

NOTES

NOTES

```
Adodc1.Recordset.MoveNext
'Adodc2.Recordset.MoveNext
Loop

'Do While Adodc2.Recordset.EOF = False
'If Adodc2.Recordset!planname = Text1.Text Then
'Txtmcc.Text = Adodc2.Recordset!MonthlyCharges
'txtfc.Text = Adodc2.Recordset!free_calls
'Exit Do
'End If
'Adodc2.Recordset.MoveNext
'Loop


'Adodc2.Recordset.MoveNext

Txtn2.Text = Cmbnm2.Text
Txtn3.Text = Txtadd.Text
Txtn4.Text = txtcust.Text
Txtn5.Text = Txttel.Text
Txtn6.Text = Txtex.Text
Txtn7.Text = Txtpin.Text
Txtdb.Text = Txtfmc.Text
Txtdb1.Text = txtfc.Text
'Wend
End Sub

Private Sub Cmdadd_Click()
Adodc3.Refresh
Adodc3.Recordset.MoveNext
Adodc3.Recordset.AddNew
Cmdadd.Visible = False
cmdsv.Visible = True
End Sub

Private Sub cmdcalc_Click()
'Txtgmc.Text = Val(Txtcmr.Text) - Val(Txtomr.Text)
'Txtncc.Text = Val(Txtgmc.Text) - Val(txtfc.Text)
'If Txtncc.Text <= 0 Then
```

```
'Txtncc.Text = "0"
'Txtmcc.Text = Txtncc.Text
'Else
'Txtmcc.Text = Txtncc.Text
'End If
End Sub

Private Sub cmdsv_Click()
'Txtn2.Text = Cmbnm2.Text
'Txtn3.Text = txtadd.Text
'Txtn4.Text = txtcust.Text
'Txtn5.Text = txttel.Text
'Txtn6.Text = Txtex.Text
'Txtn7.Text = txtpin.Text
'Txtdb.Text = Txtfmc.Text
'Txtdb1.Text = txtfc.Text
Adodc3.Recordset.Fields(0) = Txtn2.Text
Adodc3.Recordset.Fields(1) = Txtn4.Text
Adodc3.Recordset.Fields(2) = Txtn5.Text
Adodc3.Recordset.Fields(3) = Txtn6.Text
Adodc3.Recordset.Fields(4) = Txtn7.Text
Adodc3.Recordset.Fields(5) = Txtn3.Text
Adodc3.Recordset.Fields(6) = Txtomr.Text
Adodc3.Recordset.Fields(7) = Txtcmr.Text
Adodc3.Recordset.Fields(8) = Txtgmc.Text
Adodc3.Recordset.Fields(9) = txtfc.Text
Adodc3.Recordset.Fields(10) = Txtncc.Text
Adodc3.Recordset.Fields(11) = Txtfmc.Text
Adodc3.Recordset.Fields(12) = Txtmcc.Text
'Adodc3.Recordset.Fields(13) = Txtdb.Text
Adodc3.Recordset.Fields(14) = Txttx.Text
'Adodc3.Recordset.Fields(15) = Txtdb1.Text
Adodc3.Recordset.Fields(18) = Txtapb.Text
Adodc3.Recordset.Fields(19) = Txtsfdp.Text
Adodc3.Recordset.Fields(20) = Txtapdd.Text

'Adodc1.Recordset.Save
'Adodc2.Recordset.Save
Adodc3.Recordset.Save
MsgBox "BILL SAVE Successfully"
```

NOTES

NOTES

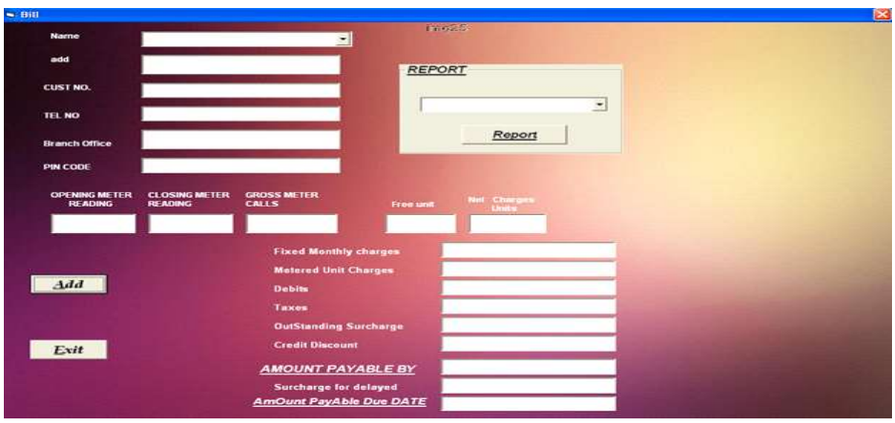
```
Adodc3.Refresh
While Adodc3.Recordset.EOF = False
Combo1.AddItem (Adodc3.Recordset!Name)
Adodc3.Recordset.MoveNext
Wend

'Val(Txtgmc.Text) = Val(Txtcmr.Text) - Val(Txtomr.Text)
'End
End Sub

Private Sub cmdx_Click()
Unload Me
End Sub

Private Sub Combo1_LostFocus()
'Text2.Text = Combo1.Text
'Adodc3.Refresh
'On Error Resume Next
'If DataEnvironment1.con1.State = 1 Then
DataEnvironment1.con1.Open
'DataEnvironment1.con1.Close
'DataEnvironment1.con1.Open
'DataEnvironment1.Bill_details (Text2.Text)
''DataReport3.Show
'BillReport.Show
End Sub



Private Sub Command1_Click()
Text2.Text = Combo1.Text
Adodc3.Refresh
On Error Resume Next
If DataEnvironment1.con1.State = 1 Then
DataEnvironment1.con1.Open
DataEnvironment1.con1.Close
DataEnvironment1.con1.Open
DataEnvironment1.Bill_details (Text2.Text)
'DataReport3.Show
BillReport.Show
End Sub
```

```
Private Sub Form_Load()
While Adodc1.Recordset.EOF = False
Cmbnm2.AddItem (Adodc1.Recordset!Name)
Adodc1.Recordset.MoveNext
Wend
txtfc.Text = ""
Txtfmc.Text = ""

'Label5.Caption = DateTime.Month(Date) & "/" &
DateTime.Year(Date)
Adodc3.Refresh
While Adodc3.Recordset.EOF = False
Combo1.AddItem (Adodc3.Recordset!Name)
Adodc3.Recordset.MoveNext
Wend

cmdsv.Visible = False
'Val(Txtgmc.Text) = Val(Txtcmr.Text) - Val(Txtomr.Text)
End Sub

Private Sub Frame1_DragDrop(Source As Control, X As Single,
Y As Single)
'BillReport.Show
End Sub

Private Sub Label5_Click()
End Sub

Private Sub Txtgmc_GotFocus()
Txtgmc.Text = Val(Txtcmr.Text) - Val(Txtomr.Text)
Txtncc.Text = Val(Txtgmc.Text) - Val(txtfc.Text)
If Txtncc.Text <= 0 Then
Txtncc.Text = "0"
Txtmcc.Text = Txtncc.Text
Else
Txtmcc.Text = Txtncc.Text
End If
Txttx.Text = (Val(Txtfmc.Text) + Val(Txtmcc.Text)) * 0.1023
Txttx.Text = Round(Txttx.Text)
Txtapb.Text = Val(Txttx.Text) + Val(Txtfmc.Text) +
```

NOTES

NOTES

```
Val(Txtmcc.Text)
If Val(Txtapb.Text) > 0 Then
Txtsfdp.Text = "10"
Txtapdd.Text = Val(Txtapb.Text) + Val(Txtsfdp.Text)
Else
MsgBox "Wrong Bill Amount"
End If
End Sub

Private Sub Txtomr_GotFocus()
Do While Adodc2.Recordset.EOF = False
If Adodc2.Recordset!planname = Text1.Text Then
'Txtmcc.Text = Adodc2.Recordset!MonthlyCharges
txtfc.Text = Adodc2.Recordset!free_calls
Exit Do
End If
Adodc2.Recordset.MoveNext
Loop
End Sub
```



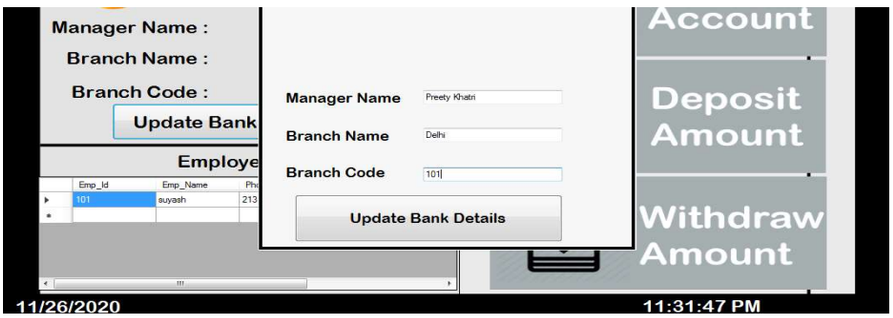
6. Bank Transaction System

Bank Details:

```
Public Class bankd
    Private Sub Label2_Click(sender As Object, e As
EventArgs)
    End Sub
```

NOTES

```
   Private Sub PictureBox1_Click(sender As Object, e As
EventArgs)
    End Sub

   Private Sub cls_Click(sender As Object, e As EventArgs)
Handles cls.Click
        Me.Close()
    End Sub

     Private Sub Button1_Click(sender As Object, e As
EventArgs) Handles Button1.Click
       home.managername.Text = TextBox1.Text
       home.brnamee.Text = TextBox2.Text
       home.Label6.Text = TextBox3.Text
       MsgBox("Bank Details Updated")
    End Sub

   Private Sub RectangleShape1_Click(sender As Object,
e As EventArgs) Handles RectangleShape1.Click
    End Sub
      Private Sub PictureBox1_Click_1(ByVal sender As
System.Object, ByVal e As System.EventArgs)
    End Sub
End Class
```



Deposit:

```
Public Class Deposit
   Private Sub cls_Click(sender As Object, e As EventArgs)
Handles cls.Click
        Me.Hide()
    End Sub
```

NOTES

```
    Private Sub Deposit_Load(sender As Object, e As
EventArgs) Handles MyBase.Load
        'TODO: This line of code loads data into the
'BankaccountsDataSet.baccounts' table. You can move, or
remove it, as needed.
            Me.BaccountsTableAdapter.Fill
(Me.BankaccountsDataSet.baccounts)
      dat.Text = Date.Now.ToString("MM/dd/yyyy")
       Timer1.Start()
   End Sub

    Private Sub Label3_Click(sender As Object, e As
EventArgs) Handles Label3.Click
   End Sub

  Private Sub dat_Click(sender As Object, e As EventArgs)
Handles dat.Click
   End Sub

    Private Sub Button1_Click(sender As Object, e As
EventArgs) Handles Button1.Click
       Me.Close()
   End Sub

    Private Sub Timer1_Tick(sender As Object, e As
EventArgs) Handles Timer1.Tick
      clock.Text = TimeOfDay
   End Sub

    Private Sub Button2_Click(sender As Object, e As
EventArgs) Handles Button2.Click
      Dim con As New OleDb.OleDbConnection
           con.ConnectionString = "PROVIDER =
Microsoft.Ace.OLEDB.12.0; Data Source =F:\Sem.4\extra vs
code\bankmanagementsystem\bankmanagementsystem\proje
ct\BankManageMentSystem\BankManage
MentSystem\bankaccounts.accdb"


       Dim SqlString As String = "update [baccounts]
set [Balance] = Balance+@TextBox2.Text where [Acc_Id] =
@TextBox1.Text"
```

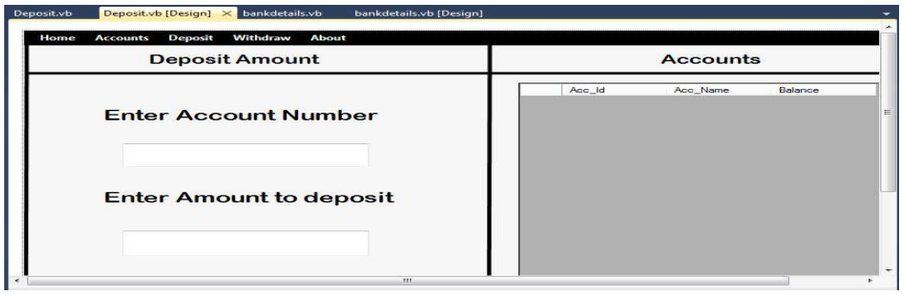
```
                        Using conn As New
OleDb.OleDbConnection(con.ConnectionString)
         Using cmd As New OleDb.OleDbCommand(SqlString,
con)
               cmd.CommandType = CommandType.Text
                 cmd.Parameters.AddWithValue("column",
TextBox2.Text)
                 cmd.Parameters.AddWithValue("column",
TextBox1.Text)
                con.Open()
              MsgBox("Amount Deposited Successfully")
                cmd.ExecuteNonQuery()
                Me.DataGridView1.Refresh()
                TextBox2.Text = ""
                TextBox1.Text = ""
            End Using
        End Using
    End Sub

    Private Sub HomeToolStripMenuItem_Click(sender As
Object, e As EventArgs) Handles HomeToolStripMenuItem.Click
        home.Show()
    End Sub

   Private Sub AccountsToolStripMenuItem_Click(sender As
Object,     e     As     EventArgs)     Handles
AccountsToolStripMenuItem.Click
    End Sub

   Private Sub AddAccountToolStripMenuItem_Click(sender
As    Object,    e    As    EventArgs)    Handles
AddAccountToolStripMenuItem.Click
        addaccount.Show()
    End Sub

                                  Private        Sub
UpdateAccountToolStripMenuItem_Click(sender As Object, e
As EventArgs) Handles UpdateAccountToolStripMenuItem.Click
        updateaccount.Show()
    End Sub

                                  Private        Sub
DeleteAccountToolStripMenuItem_Click(sender As Object, e
```

NOTES

NOTES

```
As EventArgs) Handles DeleteAccountToolStripMenuItem.Click
        deleteaccount.Show()
    End Sub

   Private Sub DepositToolStripMenuItem_Click(sender As
Object,     e     As     EventArgs)     Handles
DepositToolStripMenuItem.Click
        Me.Show()
    End Sub

   Private Sub WithdrawToolStripMenuItem_Click(sender As
Object,     e     As     EventArgs)     Handles
WithdrawToolStripMenuItem.Click
        Withdraw.Show()


    End Sub

                                    Private        Sub
RegisterProductToolStripMenuItem_Click(sender As Object,
e          As          EventArgs)          Handles
RegisterProductToolStripMenuItem.Click
        Register.Show()
    End Sub

   Private Sub CreditsToolStripMenuItem_Click(sender As
Object,     e     As     EventArgs)     Handles
CreditsToolStripMenuItem.Click
        about.Show()
    End Sub

    Private Sub HelpToolStripMenuItem_Click(sender As
Object, e As EventArgs) Handles HelpToolStripMenuItem.Click
        Help.Show()
    End Sub

   Private Sub AboutToolStripMenuItem_Click(sender As
Object,     e     As     EventArgs)     Handles
AboutToolStripMenuItem.Click
    End Sub
End Class
```

NOTES

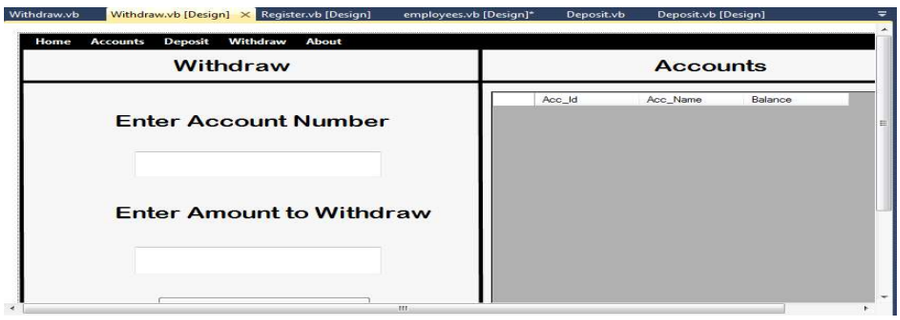


Withdraw:

```
Public Class Withdraw
   Private Sub AddAccountToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles AddAccountToolStripMenuItem.Click
    End Sub

   Private Sub cls_Click(sender As Object, e As EventArgs) Handles cls.Click
        Me.Hide()
    End Sub

     Private Sub Withdraw_Load(sender As Object, e As EventArgs) Handles MyBase.Load
         'TODO: This line of code loads data into the 'BankaccountsDataSet.baccounts' table. You can move, or remove it, as needed.
     Me.BaccountsTableAdapter.Fill(Me.BankaccountsDataSet.baccounts)
       dat.Text = Date.Now.ToString("MM/dd/yyyy")
        Timer1.Start()
    End Sub

   Private Sub dat_Click(sender As Object, e As EventArgs) Handles dat.Click
    End Sub

     Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
        clock.Text = TimeOfDay
    End Sub

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

NOTES

```
        Dim con As New OleDb.OleDbConnection
            con.ConnectionString = "PROVIDER =
Microsoft.Ace.OLEDB.12.0; Data Source =F:\Sem.4\extra vs
code\bankmanagementsystem\bankmanagementsystem\project\
BankManageMentSystem\BankManageMentSystem\bankaccounts.accdb"
        Dim SqlString As String = "update [baccounts]
set [Balance] = Balance-@TextBox2.Text where [Acc_Id] =
@TextBox1.Text"
                        Using conn As New
OleDb.OleDbConnection(con.ConnectionString)
         Using cmd As New OleDb.OleDbCommand(SqlString,
con)
            cmd.CommandType = CommandType.Text
                cmd.Parameters.AddWithValue("column",
TextBox2.Text)
                cmd.Parameters.AddWithValue("column",
TextBox1.Text)
               con.Open()
              MsgBox("Amount Withdrawn Successfully")
               cmd.ExecuteNonQuery()
               Me.DataGridView1.Refresh()
               TextBox2.Text = ""
               TextBox1.Text = ""
           End Using
        End Using
    End Sub
End Class
```



7. Payroll Processing

Login:

```
Imports System.Data.OleDb
Public Class frmloginA
```

```
      Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
                          Dim con As New
System.Data.OleDb.OleDbConnection("Provider =
Microsoft.jet.OleDB.4.0;Data Source = " &
Application.StartupPath & "\datastorage.mdb;")
       Dim cmd As OleDbCommand = New OleDbCommand( _
               "SELECT * FROM logininfo WHERE Username
= '" & _
                  TextBox1.Text & "' AND [Password] =
'" & txtPassword.Text & "' ", con)
        con.Open()
       Dim sdr As OleDbDataReader = cmd.ExecuteReader()
        If (sdr.Read() = True) Then
           MessageBox.Show("You are Now Logged In")
            frmMainA.Show()
            TextBox1.Focus()
            TextBox1.Clear()
            txtPassword.Clear()
            Me.Hide()
        Else
                MessageBox.Show("Invalid Username or
Password!")
        End If
    End Sub


      Private Sub Button2_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button2.Click
      If MsgBox("Do you want to switch user?", vbYesNo
+ vbQuestion) = vbYes Then
           Me.Hide()
           TextBox1.Clear()
           txtPassword.Clear()
           Frmchoose.Show()
        End If
    End Sub


   Private Sub txtUsername_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs)
    End Sub
```

NOTES

NOTES

```
   Private Sub CheckBox1_CheckedChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
CheckBox1.CheckedChanged
        If CheckBox1.Checked = True Then
            txtPassword.PasswordChar = ""
        Else
            txtPassword.PasswordChar = "•"
        End If
    End Sub

   Private Sub txtPassword_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs)


    End Sub


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


      Private Sub GroupBox1_Enter(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
GroupBox1.Enter
    End Sub


     Private Sub PictureBox1_Click_1(ByVal sender As
System.Object, ByVal e As System.EventArgs)
    End Sub
End Class
```

Form Main:

```
Imports System.IO
Public Class frmMainA
   Private Sub Timer1_Tick(ByVal sender As System.Object,
```

NOTES

```
ByVal e As System.EventArgs) Handles Timer1.Tick
        lblTime.Text = DateTime.Now.ToString("hh:mm:ss
tt")
        lblDate.Text = DateTime.Now.ToString("MMMM  dd
yyyy")
    End Sub

      Private Sub frmmainuser_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
       Label2.Text = frmloginA.TextBox1.Text
        Timer1.Start()
    End Sub

     Private Sub btnMaintenance_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnMaintenance.Click
        Try
           Dim fbd As New FolderBrowserDialog
           If fbd.ShowDialog() = vbOK Then
                    File.Copy("GenerallPayroll.accdb",
fbd.SelectedPath & "\GenerallPayroll.accdb")
               MsgBox("Done...")
            End If
        Catch ex As Exception
           MsgBox(ex.Message)
        End Try
    End Sub

      Private Sub btnMini_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnMini.Click
      Me.WindowState = FormWindowState.Minimized
    End Sub

      Private Sub btnLogout_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnLogout.Click
       btnLogout.BackColor = Color.White
       btnLogout.ForeColor = Color.Black
       If MsgBox("Do you want to switch user?", vbYesNo
+ vbQuestion) = vbYes Then
           Me.Hide()
```

```
            Frmchoose.Show()
        End If
    End Sub
```

NOTES

```
    Private Sub NotePadToolStripMenuItem_Click(ByVal
sender As System.Object, ByVal e As System.EventArgs)
Handles NotePadToolStripMenuItem.Click
        Try
         System.Diagnostics.Process.Start("Notepad.exe")
        Catch ex As Exception
                MessageBox.Show(ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub

   Private Sub CalculatorToolStripMenuItem_Click(ByVal
sender As System.Object, ByVal e As System.EventArgs)
Handles CalculatorToolStripMenuItem.Click
        Try
          System.Diagnostics.Process.Start("Calc.exe")
        Catch ex As Exception
                MessageBox.Show(ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub

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

    Private Sub btnCataloging_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnCataloging.Click
        frmregister.Show()
    End Sub

    Private Sub btnCirculation_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnCirculation.Click
        frmpayslip.Show()
    End Sub
```

```
    Private Sub AddStaffToolStripMenuItem_Click(ByVal
sender As System.Object, ByVal e As System.EventArgs)
Handles AddStaffToolStripMenuItem.Click
        frmaddstaff.Show()
    End Sub

   Private Sub RemoveStaffToolStripMenuItem_Click(ByVal
sender As System.Object, ByVal e As System.EventArgs)
Handles RemoveStaffToolStripMenuItem.Click
        frmremovestaff.Show()
    End Sub

   Private Sub ToolStripMenuItem1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ToolStripMenuItem1.Click
        About.Show()
    End Sub

    Private Sub EmployeeToolStripMenuItem_Click(ByVal
sender As System.Object, ByVal e As System.EventArgs)
    End Sub

   Private Sub SearchRecordsToolStripMenuItem_Click(ByVal
sender As System.Object, ByVal e As System.EventArgs)
    End Sub
End Class
```

Print Slip:

```
Public Class frmpayslip
                                    Private        Sub
GenPayFinalBindingNavigatorSaveItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
        Me.Validate()
       Me.GenPayFinalBindingSource.EndEdit()
```

NOTES

NOTES

```
        Me.TableAdapterManager.UpdateAll(Me.GenerallPayrollDataSet)
    End Sub

      Private Sub frmpayslip_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        'TODO: This line of code loads data into the
'GenerallPayrollDataSet.GenPayFinal' table. You can move,
or remove it, as needed.
    Me.GenPayFinalTableAdapter.Fill(Me.GenerallPayrollDataSet.GenPayFinal)
    End Sub

   Private Sub FacultyUnionLabel_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
    End Sub

     Private Sub TuitionLabel_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
    End Sub

      Private Sub Button5_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button5.Click
       Me.Validate()
      Me.GenPayFinalBindingSource.EndEdit()
     Me.TableAdapterManager.UpdateAll(Me.GenerallPayrollDataSet)
      MessageBox.Show("Successfully Added")
    End Sub

      Private Sub btnLogin_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
    End Sub

     Private Sub btnDeleteJHS_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnDeleteJHS.Click
       Try
          If PlantIDTextBox.Text = "" Then
             MessageBox.Show("Please select employee
id",     "Entry",     MessageBoxButtons.OK,
MessageBoxIcon.Warning)
              Exit Sub
           End If
```

```
            If PlantIDTextBox.Text.Count > 0 Then
                If MessageBox.Show("Do you really want
to delete the record?" & vbCrLf & "You can not restore the
record" & vbCrLf & "It will delete record permanently" &
vbCrLf & "related to selected employee", "Warning!!!",
MessageBoxButtons.YesNo,  MessageBoxIcon.Warning)  =
Windows.Forms.DialogResult.Yes Then
                GenPayFinalBindingSource.RemoveCurrent()
            Me.TableAdapterManager.UpdateAll(Me.GenerallPayrollDataSet)
                End If
            End If

        Catch ex As Exception
                MessageBox.Show(ex.Message,  "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub

      Private  Sub  Button2_Click(ByVal  sender  As
System.Object,  ByVal  e  As  System.EventArgs)  Handles
Button2.Click
        txtReceipt.Text = ""
       txtReceipt.AppendText("" + vbNewLine)
       txtReceipt.AppendText("" + vbNewLine)
       txtReceipt.AppendText("" + vbNewLine)
       txtReceipt.AppendText("" + vbNewLine)
       txtReceipt.AppendText("" + vbNewLine)
       txtReceipt.AppendText("" + vbNewLine)
       txtReceipt.AppendText("" + vbNewLine)
        txtReceipt.AppendText(vbTab + vbTab + vbTab +
vbTab + vbTab + vbTab & "PAY-SLIP" + vbNewLine)
       txtReceipt.AppendText("" + vbNewLine)
       txtReceipt.AppendText("" + vbNewLine)
       txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText("Plantilla Number: " + vbTab
& PlantIDTextBox.Text + vbTab + vbTab + vbTab + vbNewLine)
       txtReceipt.AppendText("Employee Name: " + vbTab
& EmployeeNameTextBox.Text + vbTab + vbTab + vbNewLine)
      txtReceipt.AppendText("Number: " + vbTab + vbTab
& NoTextBox.Text + vbNewLine)
       txtReceipt.AppendText("" + vbNewLine)
       txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText("Basic Salary: " + vbTab &
```

NOTES

NOTES

```
BasicTextBox.Text + vbNewLine)
      txtReceipt.AppendText("Pera: " + vbTab + vbTab &
PERATextBox.Text + vbNewLine)
      txtReceipt.AppendText("Gross Amount: " + vbTab &
GrossAmountTextBox.Text + vbNewLine)
       txtReceipt.AppendText("= = = = = = = = = = = = =
= = = = = = = = = = = = = = = = = = = = = = = = = = = = =
= = = = = = = = = = = = = " + vbNewLine)
       txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText(vbTab + vbTab & "Deductions"
+ vbNewLine)
       txtReceipt.AppendText("" + vbNewLine)
       txtReceipt.AppendText("W/ Tax: " + vbTab + vbTab
+ vbTab & WtaxTextBox.Text + vbNewLine)
      txtReceipt.AppendText("GSIS Premium: " + vbTab +
vbTab & GSISPremiumTextBox.Text + vbNewLine)
      txtReceipt.AppendText("GSIS Salary Loan: " + vbTab
& GSISSalaryLoanTextBox.Text + vbNewLine)
      txtReceipt.AppendText("GSIS EL: " + vbTab + vbTab
& GSISELTextBox.Text + vbNewLine)
       txtReceipt.AppendText("GSIS EMRGL: " + vbTab +
vbTab & GSISEMRGLTextBox.Text + vbNewLine)
      txtReceipt.AppendText("GSIS PL: " + vbTab + vbTab
& GSISPLTextBox.Text + vbNewLine)
      txtReceipt.AppendText("Pag-Ibig Premium: " + vbTab
& PagIbigPremTextBox.Text + vbNewLine)
       txtReceipt.AppendText("Pag-Ibig ML: " + vbTab +
vbTab & PagIbigMLTextBox.Text + vbNewLine)
       txtReceipt.AppendText("Pag-Ibig 2: " + vbTab +
vbTab & PagIbig2TextBox.Text + vbNewLine)
      txtReceipt.AppendText("Phil Health Premium:  " +
vbTab & PhilHealthPremiunTextBox.Text + vbNewLine)
      txtReceipt.AppendText("LEAP: " + vbTab + vbTab +
vbTab & LEAPTextBox.Text + vbNewLine)
       txtReceipt.AppendText("IGP: " + vbTab + vbTab +
vbTab & IGPTextBox.Text + vbNewLine)
       txtReceipt.AppendText("Faculty Union: " + vbTab
+ vbTab & FacultyUnionTextBox.Text + vbNewLine)
      txtReceipt.AppendText("Refund Disallow: " + vbTab
& RefundDisallowTextBox.Text + vbNewLine)
      txtReceipt.AppendText("Tuition: " + vbTab + vbTab
+ vbTab & TuitionTextBox.Text + vbNewLine)
       txtReceipt.AppendText("LBP Payment: " + vbTab +
vbTab & LBPPaymentTextBox.Text + vbNewLine)
      txtReceipt.AppendText("City Savings: " + vbTab +
```

```
vbTab & CitySavingsTextBox.Text + vbNewLine)
       txtReceipt.AppendText("" + vbNewLine)
      txtReceipt.AppendText("Total Deductions: " + vbTab
& TotalDeductionTextBox.Text + vbTab + vbTab & "Net Amount:
" + vbTab & NetAmountTextBox.Text + vbNewLine)
       txtReceipt.AppendText("= = = = = = = = = = = = =
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
= = = = = = = = = = = = = " + vbNewLine)
      txtReceipt.AppendText(vbTab & "Due Date: " + Today
& vbTab + vbTab + vbTab + vbTab + vbTab + vbTab & "Time:
" & TimeOfDay + vbNewLine)
       txtReceipt.AppendText("= = = = = = = = = = = = =
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
= = = = = = = = = = = = = " + vbNewLine)
       txtReceipt.AppendText("" + vbNewLine)
       txtReceipt.AppendText("" + vbNewLine)
       txtReceipt.AppendText("" + vbNewLine)
       txtReceipt.AppendText("" + vbNewLine)
        txtReceipt.AppendText(vbTab + "Recieve by:" +
vbNewLine)
        txtReceipt.AppendText(vbTab + vbTab + vbTab +
"_____"" + vbNewLine)
        txtReceipt.AppendText(vbTab + vbTab + vbTab +
EmployeeNameTextBox.Text + vbNewLine)
       txtReceipt.AppendText(vbTab + vbTab + vbTab + "
Employee" + vbNewLine)
       txtReceipt.AppendText("" + vbNewLine)
       txtReceipt.AppendText("" + vbNewLine)
       txtReceipt.AppendText("" + vbNewLine)
       txtReceipt.AppendText("= = = = = = = = = = = = =
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
= = = = = = = = = = = = = " + vbNewLine)
                        txtReceipt.AppendText("
Need Help?                Contact Us: 09096510899
" + vbNewLine)
       txtReceipt.AppendText("= = = = = = = = = = = = =
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
= = = = = = = = = = = = = " + vbNewLine)
        txtReceipt.AppendText(vbTab + vbTab + vbTab +
PictureBox1.Text + vbNewLine)
      PrintPreviewDialog1.ShowDialog()
   End Sub

  Private Sub PrintDocument1_PrintPage(ByVal sender As
System.Object,        ByVal        e        As
```

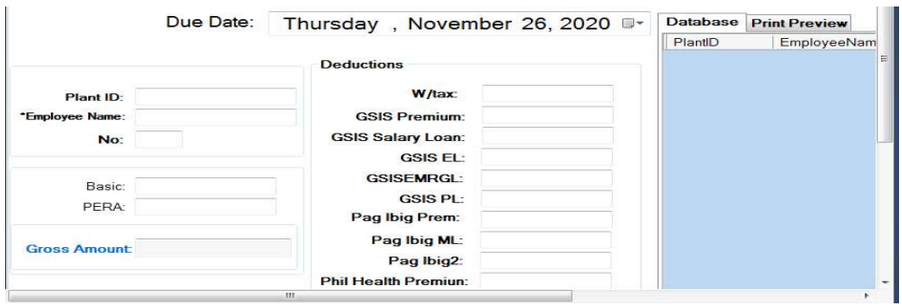
NOTES

NOTES

```
System.Drawing.Printing.PrintPageEventArgs) Handles
PrintDocument1.PrintPage
        e.Graphics.DrawString(txtReceipt.Text, Font,
Brushes.Black, 140, 140)
      e.Graphics.DrawImage(Me.PictureBox1.Image, 120,
130, PictureBox1.Width - 15, PictureBox1.Height - 25)
      e.Graphics.DrawImage(Me.PictureBox2.Image, 300,
130, PictureBox2.Width - 15, PictureBox2.Height - 25)
    End Sub
      Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
     TotalDeductionTextBox.Text = Val(WtaxTextBox.Text)
+        Val(GSISPremiumTextBox.Text)        +
Val(GSISSalaryLoanTextBox.Text) + Val(GSISELTextBox.Text)
+ Val(GSISEMRGLTextBox.Text) + Val(GSISPLTextBox.Text) +
Val(PagIbigPremTextBox.Text) + Val(PagIbigMLTextBox.Text)
+          Val(PagIbig2TextBox.Text)          +
Val(PhilHealthPremiunTextBox.Text) + Val(LEAPTextBox.Text)
+ Val(IGPTextBox.Text) + Val(FacultyUnionTextBox.Text) +
Val(RefundDisallowTextBox.Text) + Val(TuitionTextBox.Text)
+         Val(LBPPaymentTextBox.Text)         +
Val(CitySavingsTextBox.Text)
      GrossAmountTextBox.Text = Val(BasicTextBox.Text)
+ Val(PERATextBox.Text)
                      NetAmountTextBox.Text   =
Val(GrossAmountTextBox.Text)                  -
Val(TotalDeductionTextBox.Text)
                      NetAmountTextBox.Text   =
FormatCurrency(NetAmountTextBox.Text)
                TotalDeductionTextBox.Text  =
FormatCurrency(TotalDeductionTextBox.Text)
                    GrossAmountTextBox.Text   =
FormatCurrency(GrossAmountTextBox.Text)
      MessageBox.Show("Successfully Computed")
    End Sub

      Private Sub Button9_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button9.Click
     Me.TableAdapterManager.UpdateAll(Me.GenerallPayrollDataSet)
       Me.Close()
    End Sub

      Private Sub Button8_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
```

```
Button8.Click
      GenPayFinalBindingSource.MovePrevious()
    End Sub

      Private Sub Button7_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button7.Click
      GenPayFinalBindingSource.MoveNext()
    End Sub

    Private Sub TextBox14_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
TextBox14.TextChanged
      Me.GenPayFinalBindingSource.Filter = "PlantID LIKE
'" & TextBox14.Text & "%'"
    End Sub
End Class
```

NOTES



8. Personal Information System

Main:

```
Imports System.Data.OleDb
Public Class frmmain
    Dim Oledr As OleDbDataReader
    Dim Item As New ListViewItem()
    Dim ItemSearch As New ListViewItem
   Private Sub frmmain_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
       Call ListStudentColumns(lststudent)
        Call openconnection()
        Call Initialized()
        Call LoadListView()
        Call closeconnection()
```

NOTES

```
    End Sub
   Public Sub LoadListView()
       lststudent.Items.Clear()
       Call Initialized()
      Oledr = OleDa.SelectCommand.ExecuteReader()
       Do While Oledr.Read()
                                                Item    =
lststudent.Items.Add(Oledr("studentno").ToString())
        Item.SubItems.Add(Oledr("firstname").ToString())
        Item.SubItems.Add(Oledr("lastname").ToString())
         Item.SubItems.Add(Oledr("course").ToString())

       Loop
       Oledr.Close()
    End Sub

   Private Sub btnAdd_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnAdd.Click
       frmadd.ShowDialog()
    End Sub
   Private Function UpdateValidateStudent() As Boolean
       If lststudent.Items.Count = 0 Then
                           MsgBox("No  records.",
MsgBoxStyle.Information, "No Records")
           Return True
           Exit Function
       End If
      If lststudent.SelectedItems.Count > 1 Then
                   MsgBox("Double  click  the  record",
MsgBoxStyle.Information)
          lststudent.SelectedItems.Clear()
           Return True
           Exit Function
       End If
      If lststudent.SelectedItems.Count = 0 Then
          MsgBox("Please choose the record you want to
edit", MsgBoxStyle.Information)
           Return True
           Exit Function
       End If
    End Function
```

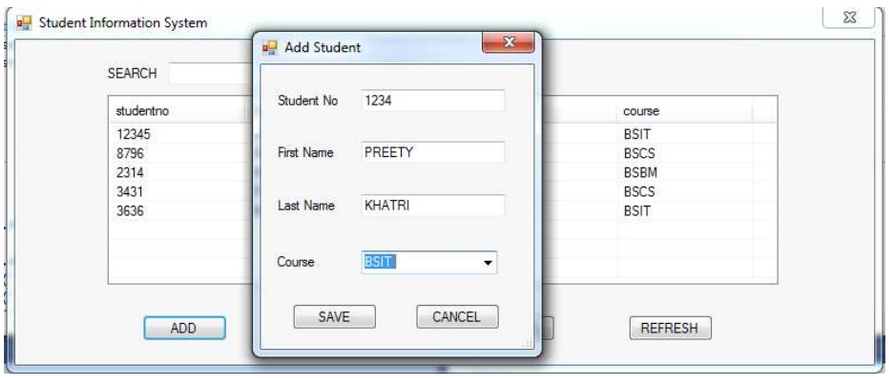
```
      Private Sub btnEdit_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnEdit.Click
       If UpdateValidateStudent() = True Then
            Return
        End If
       frmedit.ShowDialog()
    End Sub
   Private Function DeleteStudentValidate() As Boolean
       If lststudent.Items.Count = 0 Then
           MsgBox("No Records to delete")
            Return True
            Exit Function
        End If
       If lststudent.SelectedItems.Count = 0 Then
          MsgBox("Please choose the record you want to
delete.")
            Return True
            Exit Function
        End If
    End Function


      Private Sub btnDelete_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnDelete.Click
       If DeleteStudentValidate() = True Then
            Return
        End If


         If MsgBox("Do you really want to delete this
record?", MsgBoxStyle.YesNo + MsgBoxStyle.Question,
"Delete?") = MsgBoxResult.No Then
                    MsgBox("Delete Cancelled.",
MsgBoxStyle.Information)
          lststudent.SelectedItems.Clear()
            Exit Sub
        End If
             For Each Item As ListViewItem In
lststudent.SelectedItems
            Item.Remove()
          OleDa.DeleteCommand = New OleDbCommand()
```

NOTES

NOTES

```
            Call openconnection()
             OleDa.DeleteCommand.CommandText = "DELETE
FROM tblstudent WHERE studentno = @studentno"
          OleDa.DeleteCommand.Connection = OleCn
        OleDa.DeleteCommand.Parameters.Add("@studentno",
OleDbType.VarChar, 50, "studentno").Value =
Item.Text.ToString()
          OleDa.DeleteCommand.ExecuteNonQuery()
           Call LoadListView()
           Call closeconnection()
        Next
       MsgBox("Record Deleted")
       lststudent.SelectedItems.Clear()
    End Sub

      Private Sub btnRefresh_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnRefresh.Click
       Call openconnection()
       Call Initialized()
       Call LoadListView()
       Call closeconnection()
       txtSearch.Clear()
                MsgBox("Total Records = " &
lststudent.Items.Count, MsgBoxStyle.Information, "Record")
    End Sub
   Private Sub SearchStudent()
       lststudent.Items.Clear()
       Call Initialized()
      OleDa.SelectCommand.CommandText = "SELECT * FROM
tblstudent WHERE studentno Like '%%" &
txtSearch.Text.Trim.ToString() & "%%'"

       OleDa.SelectCommand.Connection = OleCn
       Oledr = OleDa.SelectCommand.ExecuteReader()
       Do While Oledr.Read()
                                      ItemSearch =
lststudent.Items.Add(Oledr("studentno").ToString())
        ItemSearch.SubItems.Add(Oledr("firstname").ToString())
        ItemSearch.SubItems.Add(Oledr("lastname").ToString())
        ItemSearch.SubItems.Add(Oledr("course").ToString())
```

```
        Loop
        Oledr.Close()
    End Sub

    Private Sub txtSearch_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
txtSearch.TextChanged
        OleDa.SelectCommand = New OleDbCommand()
       OleDa.SelectCommand.CommandText = "SELECT * FROM
tblstudent WHERE studentno Like '%%'"
        OleDa.SelectCommand.Connection = OleCn
        Call openconnection()
       OleDa.SelectCommand.ExecuteNonQuery()
        Call SearchStudent()
        Call closeconnection()
    End Sub
End Class
```



Add Information:

```
Imports System.Data.OleDb
Public Class frmadd
   Private Sub frmadd_FormClosing(ByVal sender As Object,
ByVal e As System.Windows.Forms.FormClosingEventArgs)
Handles Me.FormClosing
        Call cleartext()
        txtsn.Focus()
       frmmain.lststudent.SelectedItems.Clear()
    End Sub

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

NOTES

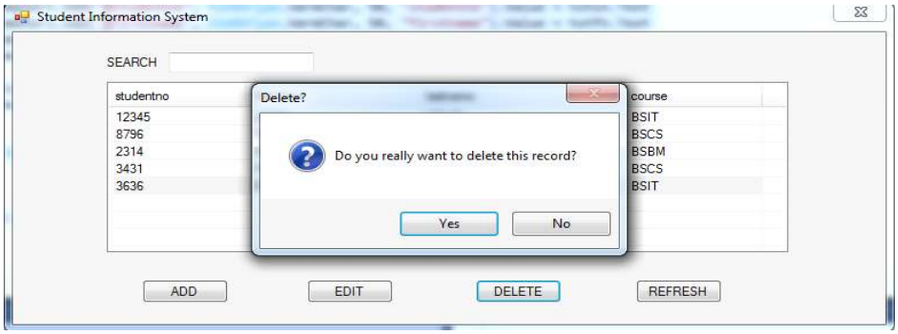
NOTES

```
End Sub
Private Sub cleartext()
    Me.txtsn.Clear()
    Me.txtfn.Clear()
    Me.txtln.Clear()
End Sub

  Private Sub btnCancel_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnCancel.Click
   Me.Close()
End Sub

  Private Sub btnSave_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnSave.Click
      If txtsn.Text = "" Or txtfn.Text = "" Or
cmbcourse.Text = "" Then
     MsgBox("Please don't leave blank textfields",
MsgBoxStyle.Information, "Missing data")
       Exit Sub
    End If
    Try
       Call openconnection()
       OleDa.InsertCommand = New OleDbCommand()
        OleDa.InsertCommand.CommandText = "INSERT
INTO tblstudent (studentno, firstname, lastname, course)"
& _
      "VALUES (@studentno , @firstname, @lastname,
@course)"
       OleDa.InsertCommand.Connection = OleCn
    OleDa.InsertCommand.Parameters.Add("@studentno",
OleDbType.VarWChar, 50, "studentno").Value = txtsn.Text
    OleDa.InsertCommand.Parameters.Add("@firstname",
OleDbType.VarWChar, 50, "firstname").Value = txtfn.Text
    OleDa.InsertCommand.Parameters.Add("@lastname",
OleDbType.VarWChar, 50, "lastname").Value = txtln.Text
    OleDa.InsertCommand.Parameters.Add("@course",
OleDbType.VarWChar, 50, "course").Value = cmbcourse.Text
       OleDa.InsertCommand.ExecuteNonQuery()
       Call frmmain.LoadListView()
       Call closeconnection()
                  MsgBox("Records  Saved",
```

```
MsgBoxStyle.Information, "Saved")
            Me.Close()
        Catch ex As Exception
             MsgBox("Cannot Save this record, Existing
Student Number", MsgBoxStyle.Information, "Error")
            Call closeconnection()
            txtsn.Focus()
            txtsn.SelectAll()
        End Try
    End Sub
End Class
```

NOTES

Delete Record:



Edit Record:

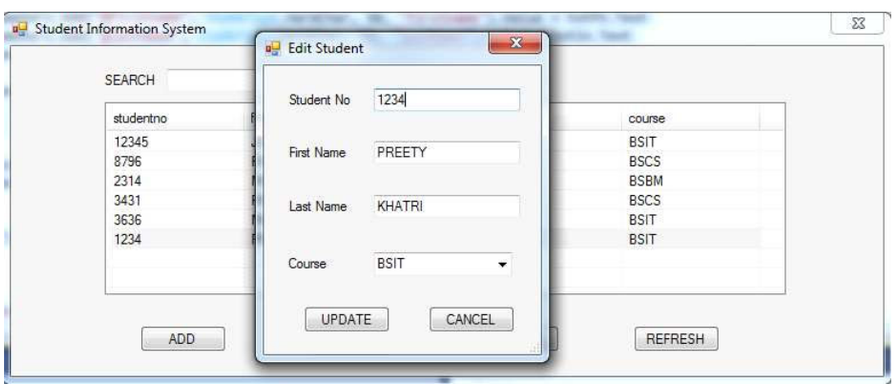
```
Imports System.Data.OleDb
Public Class frmedit
     Private Sub frmedit_FormClosing(ByVal sender As
Object,           ByVal           e           As
System.Windows.Forms.FormClosingEventArgs) Handles
Me.FormClosing
        Call cleartext()
        txtsn.Focus()
       frmmain.lststudent.SelectedItems.Clear()
    End Sub

   Private Sub frmedit_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        Call openconnection()
        Call Initialized()
                                    txtsn.Text     =
CStr(frmmain.lststudent.SelectedItems(0).Text)
        Call Fill()
        Call closeconnection()
```

NOTES

```
    End Sub
    Private Sub cleartext()
        Me.txtsn.Clear()
        Me.txtfn.Clear()
        Me.txtln.Clear()
    End Sub
    Private Sub Fill()
        Dim OleDr As OleDbDataReader
        OleDa.SelectCommand = New OleDbCommand()
       OleDa.SelectCommand.CommandText = "SELECT * From
tblstudent WHERE studentno = @studentno"
      OleDa.SelectCommand.Parameters.Add("@studentno",
OleDbType.VarWChar, 50, "studentno").Value = txtsn.Text
       OleDa.SelectCommand.Connection = OleCn
       OleDr = OleDa.SelectCommand.ExecuteReader()
        If OleDr.HasRows() Then
            OleDr.Read()
           txtsn.Text = OleDr("studentno").ToString()
           txtfn.Text = OleDr("firstname").ToString()
           txtln.Text = OleDr("lastname").ToString()
           cmbcourse.Text = OleDr("course").ToString()
        End If
        OleDr.Close()
    End Sub


      Private Sub btnCancel_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnCancel.Click
       Me.Close()
    End Sub


      Private Sub btnSave_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnSave.Click
      If txtsn.Text = "" Or txtfn.Text = "" Or txtln.Text
= "" Or cmbcourse.Text = "" Then
          MsgBox("Dont leave blank textfields")
           Exit Sub
        End If
        Try
           Call openconnection()
           OleDa.UpdateCommand = New OleDbCommand()
```

NOTES

```
            OleDa.UpdateCommand.CommandText = "UPDATE
tblstudent SET studentno = @studentno, firstname =
@firstname, lastname = @lastname, course = @course WHERE
studentno = ?"
          OleDa.UpdateCommand.Connection = OleCn
        OleDa.UpdateCommand.Parameters.Add("@studentno",
OleDbType.VarWChar, 50, "studentno").Value = txtsn.Text
        OleDa.UpdateCommand.Parameters.Add("@firstname",
OleDbType.VarWChar, 50, "firstname").Value = txtfn.Text
        OleDa.UpdateCommand.Parameters.Add("@lastName",
OleDbType.VarWChar, 50, "lastName").Value = txtln.Text
         OleDa.UpdateCommand.Parameters.Add("@Course",
OleDbType.VarWChar, 50, "Course").Value = cmbcourse.Text
              OleDa.UpdateCommand.Parameters.Add(New
System.Data.OleDb.OleDbParameter("EmpID",
System.Data.OleDb.OleDbType.VarWChar, 50, _
 System.Data.ParameterDirection.Input, False, CType(0,
Byte), CType(0, Byte), "studentno", _
  System.Data.DataRowVersion.Original, Nothing)).Value =
frmmain.lststudent.SelectedItems(0).Text
          OleDa.UpdateCommand.ExecuteNonQuery()
           Call frmmain.LoadListView()
           Call closeconnection()
           MsgBox("Records Updated")
            Me.Close()
       Catch ex As Exception
          MsgBox("Cannot Update StudentNo is present")
           Call closeconnection()
           txtsn.Focus()
           txtsn.SelectAll()
        End Try
    End Sub
End Class
```



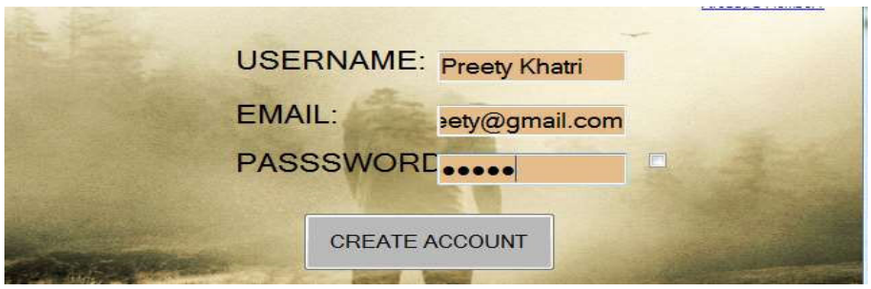
9. Question Database and Conducting Quiz

Register:

NOTES

```
Public Class Form2
     Private Sub Form2_Load(sender As Object, e As
EventArgs) Handles MyBase.Load
    End Sub

   Private Sub LinkLabel1_LinkClicked(sender As Object,
e As LinkLabelLinkClickedEventArgs)
       SIGN_IN.Show()
       Me.Close()
    End Sub

    Private Sub Button1_Click(sender As Object, e As
EventArgs)
       Home.Show()
       Me.Close()
    End Sub

    Private Sub Button2_Click(sender As Object, e As
EventArgs)
    End Sub

    Private Sub GroupBox1_Enter(sender As Object, e As
EventArgs)
    End Sub

    Private Sub Button3_Click(sender As Object, e As
EventArgs)
       quest6.Show()
    End Sub

    Private Sub Button1_Click_1(sender As Object, e As
EventArgs) Handles Button1.Click
       My.Settings.Username = username1.Text
       My.Settings.Password = password1.Text
       My.Settings.Save()
       MsgBox("Your Account Has Been Created")
       SIGN_IN.Show()
       Me.Close()
    End Sub
```

```
   Private Sub LinkLabel1_LinkClicked_1(sender As Object,
e  As  LinkLabelLinkClickedEventArgs)  Handles
LinkLabel1.LinkClicked
        SIGN_IN.Show()
        Me.Close()
    End Sub

    Private Sub Button2_Click_1(sender As Object, e As
EventArgs) Handles Button2.Click
        Form1.Show()
    End Sub

   Private Sub CheckBox1_CheckedChanged(sender As Object,
e As EventArgs) Handles CheckBox1.CheckedChanged

        If CheckBox1.Checked Then
           password1.UseSystemPasswordChar = False
        Else
           password1.UseSystemPasswordChar = True
        End If
    End Sub
End Class
```



Sign In:

```
Public Class SIGN_IN
    Private Sub Button1_Click(sender As Object, e As
EventArgs) Handles Button1.Click
       If username2.Text = My.Settings.Username And
          password2.Text = My.Settings.Password = True
Then
           Home.Show()
           Me.Close()
       Else
```

NOTES

NOTES

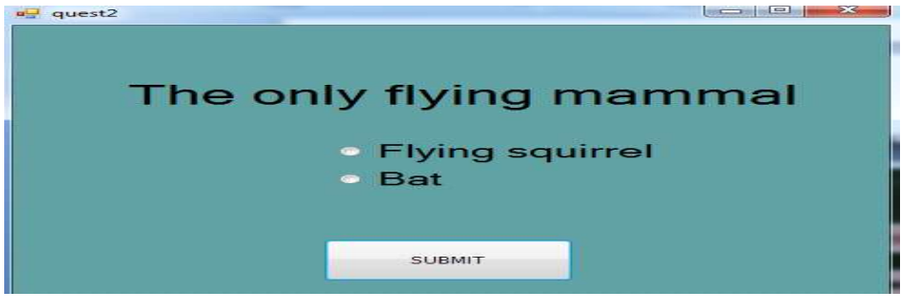
```
            MsgBox("Incorrect Username Or Password")
            username2.Clear()
            password2.Clear()
        End If
    End Sub

    Private Sub Button2_Click(sender As Object, e As
EventArgs) Handles Button2.Click
        Form1.Show()
        Me.Close()
    End Sub

    Private Sub Button3_Click(sender As Object, e As
EventArgs)
    End Sub

    Private Sub SIGN_IN_Load(sender As Object, e As
EventArgs) Handles MyBase.Load
    End Sub

   Private Sub CheckBox1_CheckedChanged(sender As Object,
e As EventArgs) Handles CheckBox1.CheckedChanged
        If CheckBox1.Checked Then
            password2.UseSystemPasswordChar = False
        Else
            password2.UseSystemPasswordChar = True
        End If
    End Sub
End Class
```

Question1:

```
Public Class quest2
    Private Sub Button2_Click(sender As Object, e As
EventArgs) Handles Button2.Click
```

NOTES

```
        Button2.Invalidate()
        If RadioButton3.Checked Then
            MsgBox("You are correct")
           quest8.LBLRIGHT.Text = quest8.LBLRIGHT.Text
+ 1
        Else
            MsgBox("You are wrong")
           quest8.LBLWRONG.Text = quest8.LBLWRONG.Text
+ 1
        End If
        Dim quest6 As New quest2
        Dim quest2 As New quest4
        quest4.Show()
        Me.Hide()
    End Sub

     Private Sub Label2_Click(sender As Object, e As
EventArgs) Handles Label2.Click
    End Sub

    Private Sub RadioButton4_CheckedChanged(sender As
Object, e As EventArgs) Handles RadioButton4.CheckedChanged
    End Sub

    Private Sub RadioButton3_CheckedChanged(sender As
Object, e As EventArgs) Handles RadioButton3.CheckedChanged
    End Sub
End Class
```



10. Personal Diary

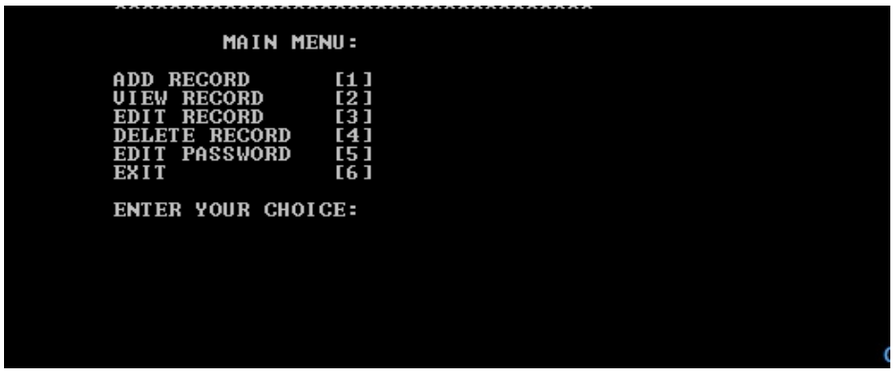
Main:

```
Class clsEntry
  Public Property dtDateOfentry As DateTime
```

NOTES

```
    Public Property strContent As String

   Public Sub New(ByVal dtDate As DateTime, _
         ByVal strText As String)
       dtDateOfentry = dtDate
      strContent = strText
    End Sub

   Public Overrides Function ToString() As String
       Return dtDateOfentry & " " & strContent
    End Function
 End Class

Module Module1
    Sub Main(ByVal args As String())
       Dim objDiary As clsDiary = New clsDiary()
       Dim cSelection As Char = "0"c
       While cSelection <> "4"c
          objDiary.Welcome()
         Console.WriteLine()
          Console.WriteLine("MAIN MENU")
         Console.WriteLine("1 - ADD RECORD")
         Console.WriteLine("2 - VIEW RECORD")
         Console.WriteLine("3 - EDIT RECORD")
         Console.WriteLine("4 - DELETE RECORD")
         Console.WriteLine("5 - EDIT PASSWORD")
         Console.WriteLine("6 - EXIT")
Console.WriteLine("ENTER YOUR CHOICE")

          cSelection = Console.ReadKey().KeyChar
         Console.WriteLine()
          Select Case cSelection
              Case "1"c
                 objDiary.Add()
              Case "2"c
                 objDiary.View()
              Case "3"c
                 objDiary.Edit()
Case "4"c
                 objDiary.Delete()
```

```
Case "5"c
              objDiary.Edit()
            Case "6"c
            Console.WriteLine("Press any key to exit.")
            Case Else
              Console.WriteLine("Error.")
         End Select
         Console.ReadKey()
      End While
   End Sub
End Module
```




```
Public Sub Add(ByVal dtDate As DateTime, ByVal strText _
        As String)
     lstEntries.Add(New clsEntry(dtDate, strText))
   End Sub

 Public Sub Delete(ByVal dtDate As DateTime)
    Dim lstResults As List(Of clsEntry) = Find(dtDate,
True)
     For Each Entry As clsEntry In lstResults
       lstEntries.Remove(Entry)
     Next
  End Sub

Public Function Find(ByVal dtDate As DateTime, ByVal
blnTime _
       As Boolean) As List(Of clsEntry)
     Dim lstResults As List(Of clsEntry) = New List(Of
clsEntry)()
      For Each Entry As clsEntry In lstEntries
```

NOTES

NOTES

```
        If ((blnTime) AndAlso (Entry.dtDateOfentry = _
           dtDate)) OrElse ((Not blnTime) AndAlso _
           (Entry.dtDateOfentry.Date = dtDate.Date))
         Then lstResults.Add(Entry)
       Next
       Return lstResults
    End Function

Class clsDiary
    Private dbData As clsDatabase
    Public Sub New()
       dbData = New clsDatabase()
    End Sub

   Private Function GetDate() As DateTime
             Dim dtDate As DateTime
       While Not DateTime.TryParse(Console.ReadLine(),
dtDate)
         Console.WriteLine("Error. Try again:")
       End While
       Return dtDate
    End Function

   Public Sub Print(ByVal dtDay As DateTime)
          Dim lstResults As List(Of clsEntry) =
dbData.Find(dtDay, _
         False)
       For Each Entry As clsEntry In lstResults
          Console.WriteLine(Entry)
       Next
    End Sub

   Public Sub Add()

      Dim dtDate As DateTime = GetDate()
       Console.WriteLine("Enter the entry text:")
       Dim strText As String = Console.ReadLine()
       dbData.Add(dtDate, strText)
    End Sub
```

```
   Public Sub Search()
       Dim dtDate As DateTime = GetDate()
           Dim lstResults As List(Of clsEntry) =
dbData.Find(dtDate, _
         False)
       If lstResults.Count() > 0 Then
          Console.WriteLine("Found:")
          For Each Entry As clsEntry In lstResults
             Console.WriteLine(Entry)
           Next
        Else
          Console.WriteLine("Nothing found.")
        End If
    End Sub

   Public Sub Delete()
       Dim dtDate As DateTime = GetDate()
       dbData.Delete(dtDate)
    End Sub

   Public Sub Welcome()
       Console.Clear()
     Console.WriteLine("ENTER DATE OF YOUR RECORD: [yyyy-
mm-dd]:", DateTime.Now))
           Console.WriteLine("ENTER TIME:")
Console.WriteLine("ENTER NAME:")
Console.WriteLine("ENTER PLACE:")
Console.WriteLine("ENTER DURATION:")
Console.WriteLine("NOTE:")
      Console.WriteLine("ADD ANOTHER RECORD…<Y/N>"
       Print(DateTime.Today)
       Console.WriteLine()
           Print(DateTime.Now.AddDays(1))
       Console.WriteLine()
    End Sub
 End Class
```

NOTES

NOTES

```
        ENTER DATE OF YOUR RECORD:[yyyy-mm-dd]:

        ENTER TIME:[hh:mm]:10:05
        ENTER NAME:Frank
        ENTER PLACE:Kathmandu
        ENTER DURATION:2hr
        NOTE:Office meeting
YOUR RECORD IS ADDED...

        ADD ANOTHER RECORD...<Y/N>
```