



**NEAR EAST UNIVERSITY**

**Faculty of Engineering**

**Department of Computer Engineering**

**CONCRETE DELIVERY PROCESS & COST  
ANALYSIS PROGRAM USING VISUAL BASIC  
LANGUAGE**

**Graduating Project  
COM 400**

**Student: Mücahit Cihat BOZKURT(20010604)**

**Supervisor : Mr. Ümit İlhan**



**Nicosia 2008**



**NEAR EAST UNIVERSITY**

**Faculty of Engineering**

**Department of Computer Engineering**

**CONCRETE DELIVERY PROCESS & COST  
ANALYSIS PROGRAM USING VISUAL BASIC  
LANGUAGE**

**Graduating Project  
COM 400**

**Student: Mücahit Cihat BOZKURT(20010604)**

**Supervisor : Mr. Ümit İlhan**

**Nicosia 2008**

## ACKNOWLEDGEMENTS

*"firstly I would like to thank my supervisor Mr. Umid Ilhan for his greatness in his way of treatment with students, He never make us feeling shy to ask any question whenever we faced an obstacle or difficulty not just during this project but during our whole life in Near East University.*

*We have not to forget other Instructors, Especially Mr. Okan Donangil, And Mr. Kaan Uyar, whenever I cross any problem morally or educationally they helped me, I thank them very much with my all for his advices and guiding.*

*Secondly I would like to give my best regards to my family, especially I would thank to my father for all his support*

*Thirdly I have not to forget my dear friends Turgay Aksoy, Ercan Kasapoğlu and especially Mr. Haluk Yerli they helped me in my project, either by giving their thoughts or with working with me.*

*Finally I would like to thank all my friends who aid me during my university life, to resist and study."*

## ABSTRACT

Concrete Delivery Process & Cost Analysis Program is written to a private company with their wish so this program is not a useful program in general. By using this program the company can control all their concrete transportation and cost analysis, time and m<sup>3</sup>, kilometer and liter in details ... etc. of their vehicles.

Visual basic programming language and Access Data Base is used in this program, the program is easy in use, everyone can use it easily without getting difficulty. The program shows you all the time, transportation and invoice records, you can make invoice calculation between two dates, calculate average time periods ... etc.

Also when you add new cost or vehicle id to your parameters you can make your calculation updated costs and parameters

Many forms are used in this project, the main form is the MDIForm in which the menu takes place, the program records everything, we can query all the information's at any time, the user can add records or parameter to vehicle records and parameters from the vehicle record page and add parameters page, can include data's to the data base, with this program.

So using this kind of programs make easy to delivery process and cost analysis and can take statistics of the past time and at present time and can see if the company has too much or less costs and depends on that the manager can take precautions to make profit.



## TABLE OF CONTENTS

<b>ACKNOWLEDGEMENT</b>	<b>i</b>
<b>ABSTRACT</b>	<b>ii</b>
<b>TABLE OF CONTENTS</b>	<b>iii</b>
<b>LIST OF FIGURES</b>	<b>vi</b>
<b>INTRODUCTION</b>	<b>viii</b>
<b>CHAPTER I: VISUAL BASIC</b>	<b>01</b>
1.1. What is Visual Basic?	01
1.2. The Advantages and Disadvantages of Visual Basic	01
1.2.1. The Advantages of Visual Basic	01
1.2.2. The Disadvantages of Visual Basic	02
1.3. Significant Language Features	02
1.4. Areas of Application	03
1.5. The Development Environment	04
1.6. The Project Explore Window	05
1.6.1. Properties Window	05
1.6.2. The Default Layout	05
1.6.3. Understanding The Tool Box	06
1.6.4. Opening an existing Visual Basic Project	06
1.6.5. Opening a new Visual Basic File & Inserting Source code	07
1.6.6. Running and Viewing the project in Detail	07
1.7. Understanding Events	09
1.7.1. What an Event is?	09
1.7.2. Click Event	10
1.7.3. Writing a Code	10
1.7.4. Using the Event GotFocus event	10
1.7.5. CheckBoxes	11
1.7.6. Option Buttons	11
1.7.7. ListBoxes	12
1.7.7.1. Add to a Listbox	13
1.7.7.2. Removing from Listboxes & Advanced Options	14
1.7.8. MsgBoxes	14
1.7.9. Opening & Retrieving Information From files	16
1.7.10. Storing Information to a file	17
1.7.11. Printing Text to the printer	17
1.8. Control Arrays	18
1.9. Brief Information to the usages of Access databases	18
1.10. Database Object	19
1.10.1. RecordSet Object	19
1.10.2. Accessing Records	20
1.10.3. Searching the RecordSet	21
1.10.4. Updating The database	21
1.10.5. Deleting And Adding records	21
1.10.5.1. Deleting Records	21
1.10.5.2. Adding Records	22
1.11. Managing Data Types	22
1.12. Introduction to VB Functions	24

1.13. Database Application	25
<b>CHAPTER II: DATABASE STRUCTURE</b>	<b>26</b>
2.1. Microsoft Access Description	26
2.2. Creating a database without using the Database Wizard	26
2.3. Tables	27
2.4. Primary Key	27
2.5. Tables Structure in Database	29
2.6. Microsoft Access Database	30
2.6.1. Microsoft Access Database Fundamentals	30
2.6.2. Creating Table	31
2.7. Introducing Database	31
2.8. Database Keys	32
2.9. Working With SQL	33
2.9.1. Data Manipulation Language	33
2.9.1.1. Insert	34
2.9.1.2. Select	34
 <b>CHAPTER III: CONCRETE DELIVERY PROCESS &amp; COST ANALYSIS PROGRAM USING VISUAL BASIC LANGUAGE</b>	 <b>35</b>
3.1. Login Page	35
3.2. Main Page	35
3.3. Menu Bar	36
3.3.1. Record Menu	36
3.3.2. Reports Menu	36
3.3.3. Invoice Menu	37
3.3.4. Parameters Menu	37
3.3.4.1. Unit Price Menu	37
3.3.4.2. Fuel Price Menu	38
3.3.4.3. Vehicle Menu	38
3.3.4.4. Personal Menu	38
3.3.5. User Menu	39
3.3.6. About Menu	39
3.4. Add New Record Page	40
3.5. Edit Record Page	41
3.6. Time Report Page	42
3.7. Transportation Report Page	43
3.8. Invoice Page	44
3.9. Add New Unit Price Page	45
3.10. Edit Unit Price Page	45
3.11. Add New Fuel Liter Price Page	46
3.12. Edit Fuel Liter Price Page	46
3.13. Add New Vehicle Page	47
3.14. Edit Vehicle ID Page	47
3.15. Add New Personal Page	48
3.16. Edit Personal Page	49
3.17. Add User Page	49
3.18. Edit User Page	50
3.19. About Page	51

CONCLUSION	52
REFERENCES	53
APPENDICES	54

52
53
54

## LIST OF FIGURES & TABLES

Figure 1.1: An Empty Form	02
Figure 1.2: A Simple Program	03
Figure 1.3: Development Environment	04
Figure 1.4: Properties Window	05
Figure 1.5: Project of Program in Detail	07
Figure 1.6: Form & Load	09
Figure 1.7: Click Event Form	10
Figure 1.8: CheckBox Example	11
Figure 1.9: Background Changing Example	12
Figure 1.10: A list on Form	13
Figure 1.10.1: Adding to List	13
Figure 1.11: Showing Message	16
Figure 2.1: Creating A database file	27
Figure 2.2: Company table as an Example in Access	27
Figure 2.3: Sample Table	30
Figure 3.1: Login Page	35
Figure 3.2: Main Page	35
Figure 3.3: Menu Bar	36
Figure 3.4: Record Menu	36
Figure 3.5: Report Menu	36
Figure 3.6: Invoice Menu	37
Figure 3.7: Parameters Menu	37
Figure 3.7.1: Unit Price Menu	37
Figure 3.7.2: Fuel Price Menu	38
Figure 3.7.3: Vehicle Menu	38
Figure 3.7.4: Personal Menu	38
Figure 3.8: User Menu	39
Figure 3.9: About Menu	39
Figure 3.10: Add New Record Page	40
Figure 3.11: Edit Record Page	41
Figure 3.12: Time Report Page	42
Figure 3.13: Transportation Report Page	43
Figure 3.14: Invoice Page	44
Figure 3.15: Add New Unit Price Page	45
Figure 3.16: Edit Unit Price Page	45
Figure 3.17: Add Fuel Price Page	46
Figure 3.18: Edit Fuel Price Page	46
Figure 3.19: Add Vehicle Page	47
Figure 3.20: Edit Vehicle Page	47
Figure 3.21: Add Personal Page	48
Figure 3.22: Edit Personal Page	49
Figure 3.23: Add New User Page	49
Figure 3.24: Edit User Page	50
Figure 3.25: About Page	51
 Table 1.1: Numeric Data Types	 22
Table 1.2: Non-Numeric Data Types	23



Table 2.1: ParVID Database Table	29
Table 2.2: ParUP Database Table	29
Table 2.3: ParFuel Database Table	29

# INTRODUCTION

Concrete Delivery Process & Cost Analysis Program is a useful program for a Concrete Transporter company, by the way of this program we can arrange everything of the concrete transporting, if we use the program in a correct way nothing will be confused during any operation that take place in this sector for present time and future time, also we can investigate past time operations too.

By the way of this program we can calculate the invoice of the transportation, see our costs, investigate what happened daily... etc.

In the first Chapter is described Visual basic, it's properties, components and some examples, we used Visual Basic Program in my project, because we find it easy and we like it's coding system. Visual basic for applications delivers a competitive advantage for ISV seeking to provide full customization and integration capabilities to customer.

In the Second Chapter is described Database system, we used Microsoft Access Database system in my program with Visual Basic, Advantages of using access is to provide exactly the same options for the problems we write as it does for the problems we selected from a data base. Secondly, the process of writing or selecting problems is almost independent of page layout dictions. Also you can see more details about access and SQL.

Third Chapter is Concrete Delivery Process & Cost Analysis Program, explained above, it is the most important Chapter of this project, because I made that program by myself alone by the help of some friends also through investigations from internet.

## **CHAPTER I: VISUAL BASIC**

### **1.1. What Is Visual Basic?**

**VISUAL BASIC** is a high level programming language evolved from the earlier DOS version called BASIC. BASIC means **B**eginners' **A**ll-purpose **S**ymbolic **I**nstruction **C**ode. It is a fairly easy programming language to learn. The codes look a bit like English Language. Different software companies produced different version of BASIC, such as Microsoft QBASIC, QUICKBASIC, GWBASIC, IBM BASICA and so on. Programmers have undergone a major change in many years of programming various machines. For example what could be created in minutes with Visual Basic could take days in other languages such: as "C" or "Pascal". Visual Basic provides many interesting sets of tools to aid you in building exciting applications. Visual Basic provides these tools to make your life far more easier because all the real hard code is already written for you.

With controls like these you can create many applications which use certain parts of windows. For example, one of the controls could be a button, which we have demonstrated in the "Hello World" program below. First create the control on the screen, then write the code which would be executed once the control button is pressed. With this sort of operation in mind, simple programs would take very little code. Why do it like the poor old "C" programmer who would have to write code to even display a window on the screen, when Visual Basic already has this part written for you.

Even though people tend to say Visual Basic's compiler is far behind the compilers of Pascal and C, it has earned itself the status of a professional programming language, and has almost freed BASIC of the reputation of a children's language. Overall you would class Visual Basic as a Graphics User Interface(GUI). Because as you draw, you write for the program. This must always be remembered in any kind of creation of a Visual Basic program. All in all, VB is the preferred language of many future programmers. If you want to start programming Windows, and don't know how to start, give Visual Basic a shot.

### **1.2. The advantages & disadvantages of Visual Basic.**

#### **1.2.1. The advantages of Visual Basic:**

- 1) It's simple language. Things that may be difficult to program with other language, can be done in Visual Basic very easily.
- 2) Because Visual Basic is so popular, There are many good resources (Books, Web sites, News groups and more) that can help you to learn the language. You can find the answers to your programming problems much more easily than other programming languages.
- 3) You can find many tools (Sharewares and Freewares) on the internet that will spare you some programming time.



For example, if you want to ping a user over the internet in your program, Instead of writing the ping function yourself, you can download a control that does it, and use it in your program.

Compare to other languages, Visual Basic have the widest variety of tools that you can download from the internet and use them in your programs.

### 1.2.2. The disadvantages of Visual Basic:

- 1) Visual Basic is powerful language, but it's not suit for programming really sophisticated games.
- 2) It's much more slower than other langauges.

### 1.3. Significant Language Features.

Visual Basic is not only a programming language, but also a complete graphical development environment. This environment allows users with little programming experience to **quickly** develop useful Microsoft Windows applications which have the ability to use OLE ( Object Linking and Embedding ) objects, such as an Excel spreadsheet. Visual Basic also has the ability to develop programs that can be used as a front end application to a database system, serving as the user interface which collects user input and displays formatted output in a more appealing and useful form than many SQL versions are capable of.

Visual Basic's main selling point is the ease with which it allows the user to create nice looking, graphical programs with little coding by the programmer, unlike many other languages that may take hundreds of lines of programmer keyed code. As the programmer works in the graphical environment, much of the program code is automatically generated by the Visual Basic program. In order to understand how this happens it is necessary to understand the major concepts, objects and tools used by Visual Basic. The main object in Visual Basic is called a **form**. When you open a new project, you will start with a clear form that looks similar to this :

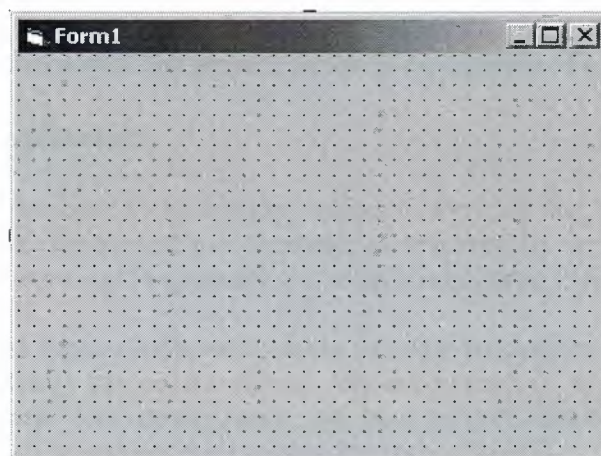


Figure 1.1.An Empty Form

This form will eventually be incorporated into our program as a window. To this form we add **controls**. Controls are things like text boxes, check boxes and command buttons...etc.

We can add **events** to our controls. Events are responses to actions performed on controls. For example, in the "Hello world" program sample on this page, when you click on the command button on our form the event that is triggered is the output of the message "Hello world" to the screen. Code must be written to create an event. we can do this in Visual Basic's **code window**.

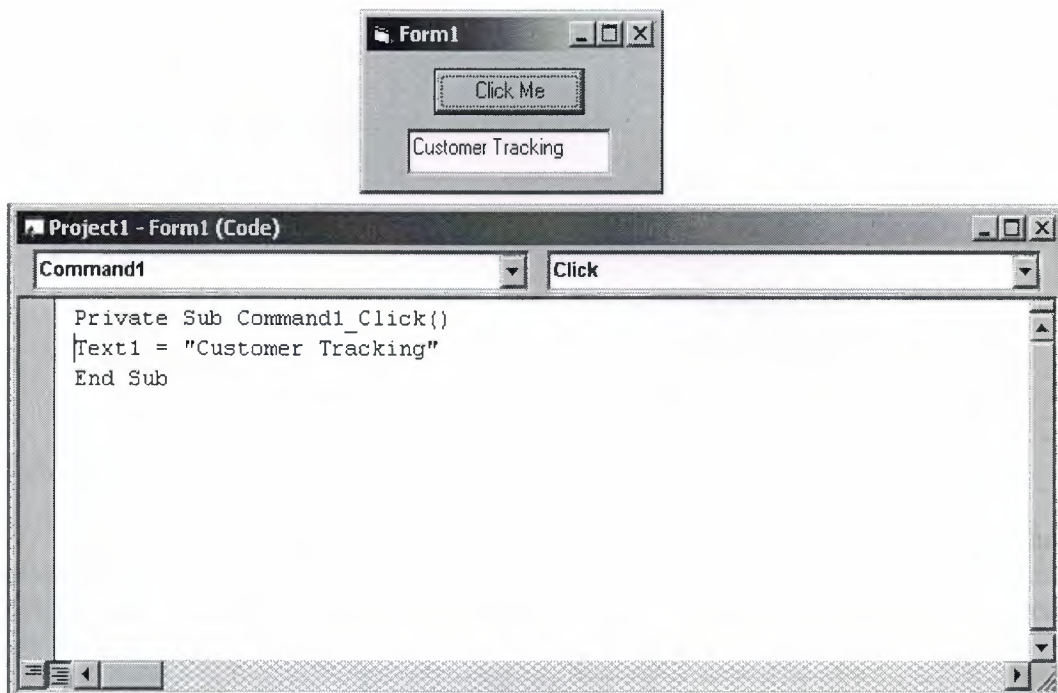


Figure 1.2. A Simple Program

Once the code box is open, you select the object to create an event for and the triggering action ( such as a certain mouse action ) from the drop down menus in the code box. You can open a code box for a particular form by choosing it from the **project window** and selecting the **View Code** button. The project window contains a list of objects associated with that project. Below is an example of a project window :

#### 1.4. Areas of Application.

The term "*Personal Programming*" refers to the idea that, wherever we work, whatever we do, we can expand our computer's usefulness by writing applications to use in our own job. Personal Programming is what Visual Basic is all about.

Using Visual Basic's tools, we quickly translate an abstract idea into a program design, we can actually see on the screen. VB encourages us to experiment, revise, correct, and network your design until the new project meets our requirements. However , most of all, it inspires our imagination and creativity.



Visual Basic is ideal for developing applications that run in the new Windows 95 operating system.

**VB presents a 3-step approach for creating programs:**

1. Design the appearance of application.
2. Assign property settings to the objects of program.
3. Write the code to direct specific tasks at runtime.

**Visual Basic can be used in a number of different areas, for example:**

- Education
- Research
- Medecine
- Business
- Commerce
- Marketing and Sales
- Accounting
- Consulting
- Law
- Science

## 1.5. The Development Environment.

Learning the ins and outs of the Development Environment before we learn visual basic is somewhat like learning for a test, we must know where all the functions belong and what their purpose is. First we will start with labelling the development environment.

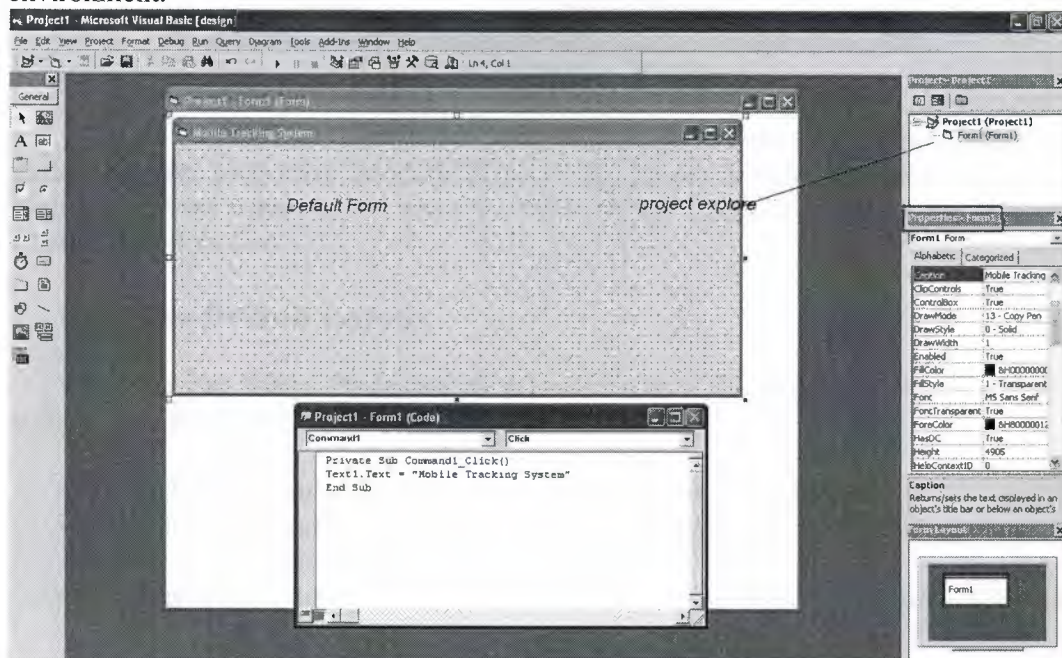


Figure 1.3. Development Environment

The above diagram shows the development environment with all the important points labelled. Many of Visual basic functions work similar to Microsoft word eg the **Tool Bar** and the tool box is similar to other products on the market which work off a single click then drag the width of the object required. The **Tool Box** contains the control we placed on the form window. All of the controls that appear on the **Tool Box** controls on the above picture never runs out of controls as soon as we place one on the form another awaits us on the **Tool Box** ready to be placed as needed.

## 1.6. The project explorer window.

The Project explorer window gives us a tree-structured view of all the files inserted into the application. We can expand these and collapse branches of the views to get more or less detail (**Project explorer**). The project explorer window displays forms, modules or other separators which are supported by the visual basic like class'es and Advanced Modules. If we want to select a form on its own simply double click on the project explorer window for a more detailed look. And it will display it where the **Default form** is located.

### 1.6.1.Properties Window.

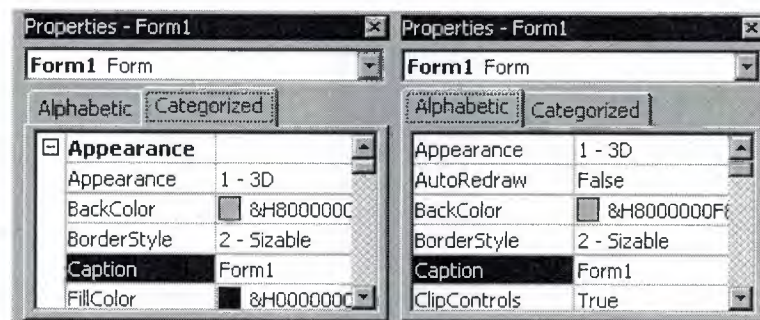


Figure 1.4. Properties Window

Some programmers prefer the Categorized view of the properties window. By defaulting, the properties window displays its properties alphabetically (with the exception of the name value) when we click on the categorized button the window changes to left picture.

### 1.6.2.The Default Layout.

When we start Visual Basic, we are provided with a VB project. A VB project is a collection of the following modules and files.

- The **global module**( *that contains declaration and procedures*)
- The **form module**(*that contains the graphic elements of the VB application along with the instruction* )
- The **general module** (*that generally contains general-purpose instructions not pertaining to anything graphic on-screen*)
- The **class module**(*that contains the defining characteristics of a class, including its properties and methods*)



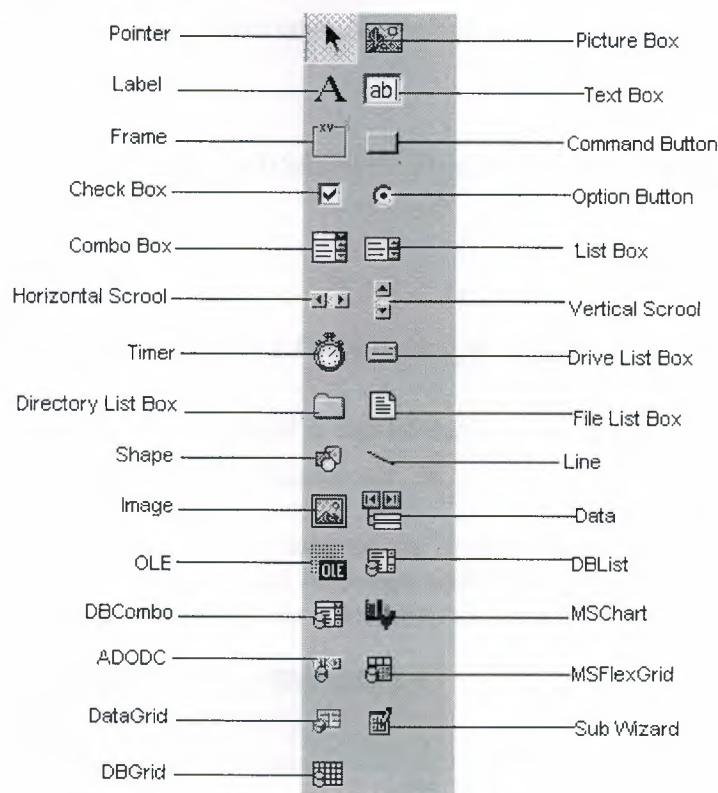
- The **resource files** (that allows you to collect all of the texts and bitmaps for an application in one place)

**On start up, Visual Basic will display the following windows :**

- The **Blank Form** window
- The **Project** window
- The **Properties** window

It also includes a **Toolbox** that consists of all the controls essential for developing a VB Application. Controls are tools such as boxes, buttons, labels and other objects drawn on a form to get **input** or **display output**. They also add visual appeal.

### 1.6.3. Understanding the tool box.



When we click on different controls the Properties Window changes slightly this is due to different controls having different functions. Therefore more options are needed, for example if we had a picture then we want to show an image. But if we wanted to open an internet connection we would have to fill in the remote host and other such settings. When we use the command () we will find that a new set of properties come up the following will provide a description and a property.

### 1.6.4. Opening an existing Visual Basic project.

Microsoft have included some freebies with visual basic to show its capabilities and functions. Dismantling or modifying these sample projects is a good way to

understand what is happening at runtime. These files can be located at our default directory /SAMPLES/

To Open these projects choose 'Open Project' from the 'File' menu. Then Double click on the samples folder to open the directory then Double click on any project to load it.

### 1.6.5. Opening a New Visual Basic File & Inserting Source code.

When we open the program we choose 'New Project' from the 'File' menu. We use the blank form1 to design a simple interface for an estate agents database, have some textboxes for names and other details. We insert some controls and make it look professional. Textboxes can be used to store there name and other details, we make sure that we put a picture box in for a picture of the house.




Later we insert the following source code for application.

#### Private Sub

```
Form_Load()Picture1.Picture=LoadPicture("C:\ProgramFiles\VB\Graphics\Icons\Misc\MISC42.ICO")
```

#### End Sub

### 1.6.6. Running and viewing the project in detail.

Once an application is loaded it can be run by click on the  icon from the toolbar, to pause press  and to terminate use .

Once a project is loaded, the name of the form(s) that it contains is displayed in the project window. To view a form in design mode, we select the form required by clicking with the mouse to highlight its name, then clicking on the view form button.

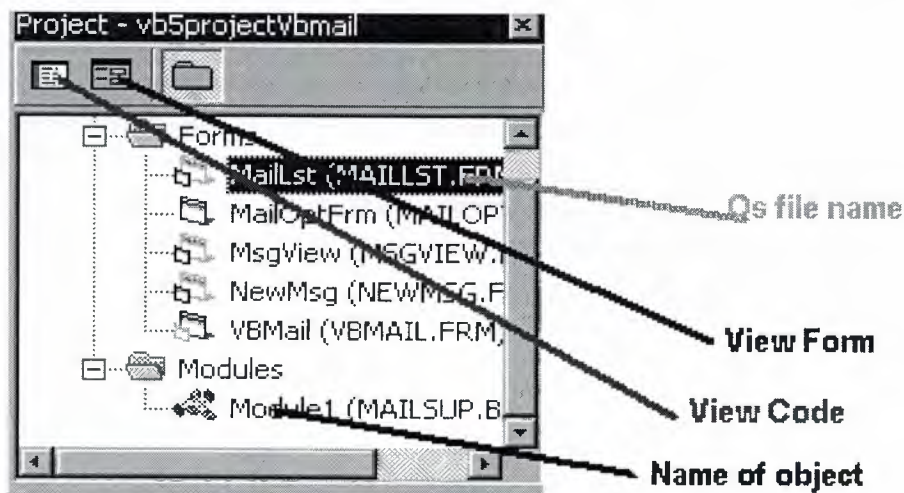


Figure 1.5. Project of the Program in Detail



## Button Properties for reference

, Command Button &  labels properties


Property	Description
<b>Name</b>	The name of the object so you can call it at runtime
<b>BackColor</b>	This specifies the command button's background color. Click the <i>BackColor's</i> palette down arrow to see a list of common Windows control colours, you must change this to the style property from 0 - standard to 1 - graphical
<b>Cancel</b>	Determines whether the command button gets a Click event if the user presses escape
<b>Caption</b>	Holds the text that appears on the command button.
<b>Default</b>	Determines if the command button responds to an enter keypress even if another control has the focus
<b>Enable</b>	Determines whether the command button is active. Often, you'll change the enable property at runtime with code to prevent the user pressing the button
<b>Font</b>	Produces a Font dialog box in which you can set the caption's font name, style and size.
<b>Height</b>	Positions the height of the object - can be used for down
<b>Left</b>	Positions the left control - can be used for right
<b>MousePointer</b>	If selected to an icon can change the picture of the mouse pointer over that object
<b>Picture</b>	Holds the name of an icon graphic image so that it appears as a picture instead of a <b>Button</b> for this option to work the graphical tag must be set to 1
<b>Style</b>	This determines if the <b>Command Button</b> appears as a standard windows dialog box or a graphical image
<b>Tab index</b>	Specifies the order of the command button in tab order
<b>Tab Stop</b>	Whether the object can be tabbed to ( this can be used in labels which have no other function )
<b>Tool Tip Text</b>	If the mouse is held over the object a brief description can be displayed (for example hold your mouse over one of the above pictures to see this happening)
<b>Visible</b>	If you want the user to see the button/label select true other wise just press false
<b>Width</b>	Show the width of the object



## 1.7. Understanding Events.

- Know what an Event is.
- Determine what Events a control can have
- Write code for one or more Events.
- Using optionbuttons to produce an event
- Using checkboxes to produce an event
- Grouping controls using a frame
- Make a simple alteration to the interface, such as changing background colour, at run time.
- Creating a listbox.
- Remove and Add listboxs functions.
- Creating Combo Boxes
- What the different types of combo boxes are.

### 1.7.1. What an event is?

The 'look' of a Visual Basic application is determined by what controls are used, but the 'feel' is determined by the events. An event is something which can happen to a control. For example, a user can click on a button, change a text box, or resize a form. As explained in Creating a Visual Basic Application, writing a program is made up of three events: 1) select suitable controls, 2) set the properties, and 3) write the code. It is at the code writing stage when it becomes important to choose appropriate events for each control. To do this double click on the control the event will be used for, or click on the  icon in the project window (usually top right of screen). A code window should now be displayed similar to the one shown below.

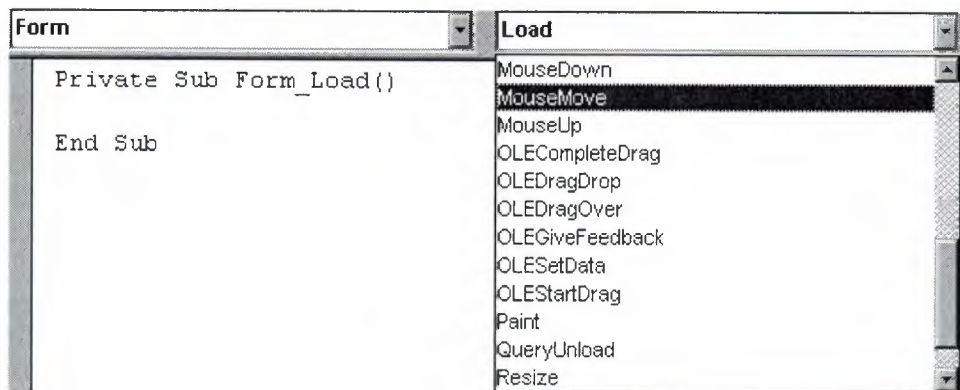


Figure 1.6. Form & Load

The left hand dropdown box provides a list of all controls used by the current form, the form itself, and a special section called General Declarations. The corresponding dropdown box on the right displays a list of all events applicable to the current control (as specified by the left hand dropdown box). Events displayed in bold signify that code has already been written for them, unbold events are unused. To demonstrate that different events can play a significant role in determining the feel of an application, a small example program will be written to add two numbers together and display the answer. The first solution to this problem will use the click event of a command button, while the second will the change event of two text boxes.

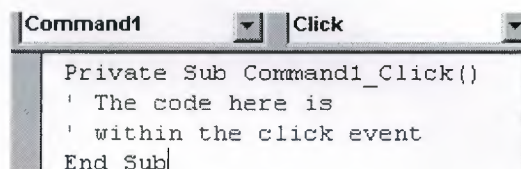
### 1.7.2. Click Event.

Before any events can be coded it is necessary to design the interface from suitable controls. As shown in the screen shot below use: 2 text boxes to enter the numbers, a label for the '+' sign, a command button for the '=' sign, and another label for the answer.



Figure 1.7. Click Event Form

Making the click event is very simple just select the button with the mouse and double click visual basic will generate



You can see on the top right there is a 'click' dropdown list this is known as a event handler.

### 1.7.3. Writing a Code


In the first example the user enter two numbers and then click on the equals button to produce an answer. However, the program can be changed so that the answer will be calculated every time either of the two numbers are changed without requiring an equals button.

```
Private Sub txtNumber1_Change()  
label2.Caption = Str$(Val(text1.Text) + Val(text2.Text))  
End Sub
```

```
Private Sub txtNumber2_Change()  
label2.Caption = Str$(Val(text1.Text) + Val(text2.Text))  
End Sub
```

When we run the program, we enter two numbers and observe what happens. Each time a digit changes the answer is recalculated.

### 1.7.4. Using the event GotFocus event.

So far only one event has been used per control, however this does not have to be the case! Add a StatusBar control to the bottom of the form, bring up the code window using , select the first text box (txtNumber1) from the left hand dropdown box, and then select the **GotFocus** event from the right hand dropdown box. Now some basic instructions can be written in the status bar so that when the cursor is in the text box (the text box has focus) the status bar reads "Enter the first number". After

completing this change to the second text box and using the same GotFocus event change the statusbar text to "Enter a second number". The code to set the status bar can be seen below.

### 1.7. 5. CheckBoxes.

Option bars are used quite often in the windows environment as they can only have two outputs 0 and 1 these get used to process the form. In this example it will be used to change the some text from normal to bold or to italic.

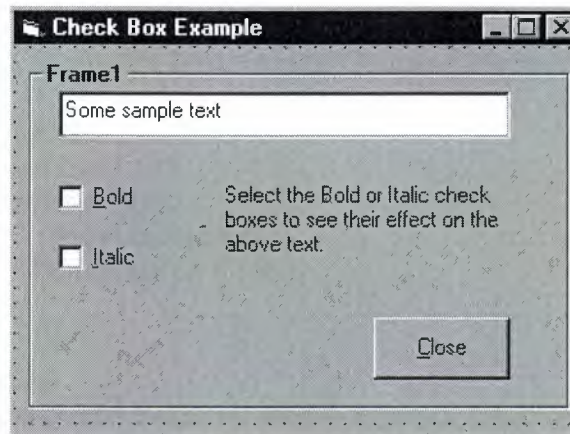


Figure 1.8. Check Box Example

```
Private Sub chkBold_Click()  
If chkBold.Value = 1 Then ' If checked.  
txtDisplay.FontBold = True  
Else ' If not checked.  
txtDisplay.FontBold = False  
End If  
End Sub
```

```
Private Sub chkItalic_Click()  
If chkItalic.Value = 1 Then ' If checked.  
txtDisplay.FontItalic = True  
Else ' If not checked.  
txtDisplay.FontItalic = False  
End If  
End Sub
```

### 1.7.6.Option Buttons.

#### Changing the background colour

Changing the background colour gets used mostly by freeware, or the type of programs we generate for use with banners or adverts, anyway it might come in useful sometime. This example shows an ordinary picture of a smiley face then I put in some nice background colours to make it stand out more.







Figure 1.9. Back Ground Changing Example


```
Private Sub Form_Load()
BackColor = QBColor(Rnd * 15)
ForeColor = QBColor(Rnd * 10)
Picture1.BackColor = QBColor(Rnd * 15)
Picture1.ForeColor = QBColor(Rnd * 10)
End Sub
```

#### 1.7.7. List boxes.

List boxes and combo boxes are used to supply a list of options to the user. The toolbox icons representing these two controls are  for list box and  for combo box.

A list box is used when the user is to be presented with a fixed set of selections (i.e. a choice must be made only from the items displayed, there is no possibility of the user typing in an alternative).

Examples might be offering a list of the days in a week, the available modules in an elective catalogue, the holiday destinations available from a particular airport or the types of treatment offered by a beauty salon.

To create a list box, double click on the toolbox icon . Drag the resulting box into place on the form and size it according to the data it will hold. The left hand picture below shows a list box that has been created and sized on Form1. In the middle is the code that is required to load a selection of cars into the list. The data has been included in the procedure Sub Form\_Load so that it appears when Form1 is loaded. Finally, the picture on the right shows what appears on the form when the application is run. Notice that vertical scroll bars are added automatically if the list box is not deep enough to display all the choices.

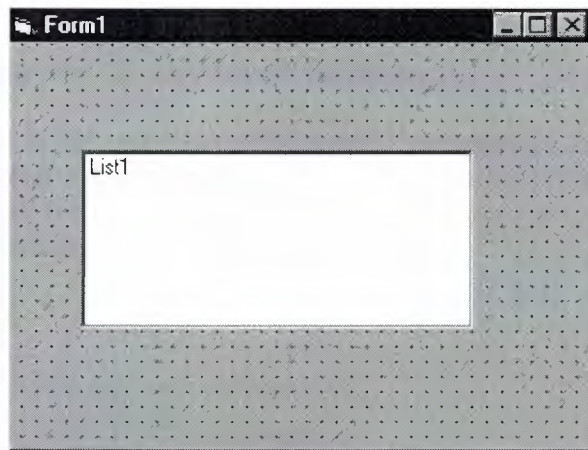


Figure 1.10. A list On Form

If we however add the following source code to this project

#### 1.7.7.1. Add to a Lisbox.

```
Private Sub Form_Load()  
List1.AddItem "Monday"  
List1.AddItem "Tuesday"  
List1.AddItem "Wednesday"  
List1.AddItem "Thursday"  
List1.AddItem "Friday"  
List1.AddItem "Saturday"  
List1.AddItem "Sunday"  
End Sub
```

The source code should look something like this :

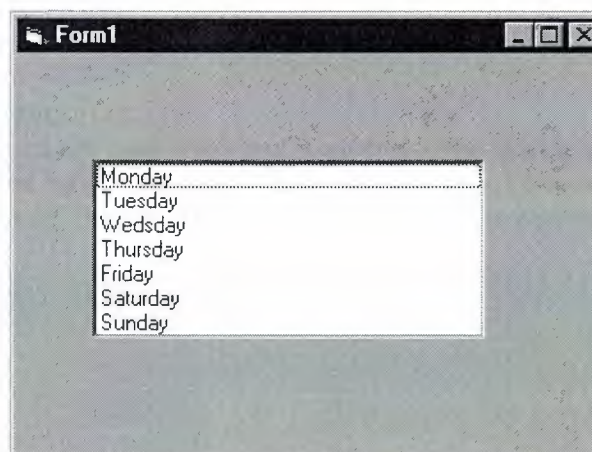


Figure 1.10.1. Adding To List



### 1.7.7.2.Removing & Advanced Options.

They appear in the order they were typed if you changed the properties window by changing sort order = true then they will go into alphabetical order. List items can be added and deleted all the list is counted as the order they are in so for example if you wanted to delete "Tuesday" you would type

#### **list1.RemoveItem 1**

And to add to the list in a particular order just add

#### **list1.additem "My Day", 5**

This will be added after Saturday.

And finally to clear the listbox type

**List1.clear** This will completely clear the contents of the listbox.

The property ListCount stores the number of items in a list, so list1.ListCount can be used to determine the number of items in list box list1.

The property ListIndex gives the index of the currently selected list item. So the statement list1.RemoveItem List1.ListIndex removes the currently highlighted list item.

Adding an item can be accomplished very neatly using an input dialog box. Try this:

#### **list1.AddItem InputBox("Enter a day", "Add a Day")**

This will open a message box and prompt you for a new day to enter this will then be added to the list index at the bottom unless you specify where it should go.

### 1.7.8. MsgBoxes.

Message boxes are used when you want to ask the user a question or display an error message(s) and advise the user. There are six types of message boxes here are their functions and what they do. Here is the listing of all the possible msgbox events

The Buttons displayed in a message here Button Layout	Value	Short Description
vbOKOnly	0	Displays the OK button.
vbOKCancel	1	Displays the ok and cancel button.
vbAbortRetryIgnore	2	Displays the Abort , Retry , Ignore
vbYesNoCancel	3	Displays Yes , No and Cancel button
vbYesNo	4	Displays the Yes / No button

vbRetryCancel	5	Displays the retry and Cancel buttons.
---------------	---	--

**The Icons displayed in the message box are here**

Icon on message	Value	Short Description
vbCritical	16	Displays critical message icon
vbQuestion	32	Displays question icon
vbExclamation	48	Displays exclamation icon
vbInformation	64	Displays information icon

The Default button displayed in a message form Default Button	Value	Short Description
vbDefaultButton1	0	Button 1 is default
vbDefaultButton2	256	Button 2 is default
vbDefaultButton3	512	Button 3 is default

Msgbox Return Value	Value	Short Description
vbOk	1	The User Clicked OK
vbCancel	2	The User Clicked Cancel
vbAbort	3	The User Clicked Abort
vbRetry	4	The User Clicked Retry
vbIgnore	5	The User Clicked Ignore
VbYes	6	The User Clicked Yes
VbNo	7	The User Clicked No

The syntax for use of the message box in Msgbox "TEXT", VALUE, "TITLE"  
If you want to use two or more tables just add the values together. Therefore to print a message box to say "The Device was not Found!" OK & Explanation :

Source code 1

**Private Sub Form\_Load()**

MsgBox "The Device was not Found!", 48, "Header"

**End Sub**

Source code 2

```
Private Sub Form_Load()
```

```
MsgBox "The Device was not found!", vbExclamation, "Header"
```

```
End Sub
```

we should get the picture shown below whatever source code you used.

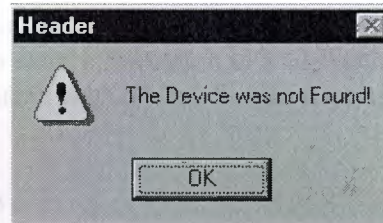


Figure 1.11. Showing message

This is a basic msgbox which in this case has not been processed in any way. The following Source code displays a msgbox that ask you for specific text. For example lets make a password program out of this message box.

```
Private Sub Form_Load()
```

```
lngBefore = Timer
```

```
Do
```

```
strAns = InputBox("What is the password Password is Example", "Password Required")
```

```
Loop Until Val(strAns) = Example
```

```
lngAfter = Timer
```

```
msgbox "Correct Password", vbInformation
```

```
End Sub
```

Once you copy and paste the source code you should get prompted for a password in a different type of msgbox as it includes text. From looking at this example you should be able to see how the loop function works to generate and then check the value. All of the Return Values work in the same way to return an object input.

### 1.7.9.Opening & Retriving information from files.

When applications are loaded they normal get some setting out of the registry or file this section will show you how to retrieve 1 string out of a file.

```
Private Sub Form_Load()
```

```
Dim F As Integer, password As String
```

```
F = FreeFile
```

```
Open App.Path & "\password.txt" For Input As F
```

```
Input #F, password
```



Close #F

**End Sub**

As you can see from this source code the password is previously declared as a string. After this is done be sure to close the file otherwise next time you want to store or read the file the computer will think it is being used by another application and windows will not let you do anything with it. So as you can see it is **Very Important** to close the file

#### **1.7.10. Storing Information to a file**

FTP programs often store information to a file such as a username and password or host information in the same way. This following example will put some information into the file.

**Private Sub Form\_Load()**

```
Dim F As Integer, pass As String
F = FreeFile
save = txtNew
Open App.Path & "\password.txt" For Output As F
Write #F, Text1.text
Close #F
```

**End Sub**

Although this is a bit obvious I think I should include it just incase I think differently to other people.

#### **1.7.11. Printing text to the printer.**

This is rather easy to do and it gets used in notepad etc...

**Private Sub Form\_Load()**

```
Printer.Print " The printer will print this text "
Printer.Print ""
Printer.Print " It will leave a line here"
Printer.Print " It should add the contence of text1.text here : " & Text1.Text & " As you
can see it works"
Printer.Print ""
Printer.EndDoc "This will tell the printer it has finished."
```

**End Sub**

Everything that appears in position (A) will get printed by the default printer printer.print " A ". The printer enddoc is used to tell the printer the job is finished if this is not added the printer can not estimate how near it will be until it has finish and when it has finished it will think it has'nt so be sure to include this to prevent confusion.

## **1.8. Control Arrays.**

A control array is a list of controls with the same name. Therefore, instead of using five command buttons with separate five names, you can place a command button control array on the form, and that control array holds five command buttons. The control array can have a single name, and you will distinguish the control from each other with a subscript. One of the best reasons to use control array from that first control, all the elements in the control array take on the same property values. You then can change those properties that need to be changed without having to set every property for each control individually. Control arrays have a lot in common with data arrays. A control array has one array, and you distinguish all the array's controls from each other with the zero-based subscript. ( The index property holds the controls subscript number ). All of the control elements must be the same data type. As soon as you place a control on a form that has the same name as an existing control, Visual Basic makes sure you that you want to begin a control array by issuing the warning message to show that the control is already in use. This is used as a built in safety so that you do not over right an existing control by putting it some where else on the same form. If you answer the warning box with a no button, Visual Basic uses a default control name for the placed control.

All event procedures that use control from a control array require a special argument value passed to them that the determines which control is being worked on. For example if your application contains a single control command button named cmdtotal the click () event begins and ends as follows

If however you create a control array named the same name as before (cmdtotal) it will end up like this

## **1.9. Brief introduction to the usages of Access data bases.**

What I think is the most compelling thing about Visual Basic is it's easy way of accessing and modifying databases. This is what I think you should learn next; you will find many applications for this knowledge. I almost never make a program without using a database for data storage.

There are many ways to work with databases in Visual Basic, and I would think you have at least glanced at the Data control. I will not even mention the Data control further in this text, since it is so easy to use and too limited to be interesting for a professional developer. (Ok, there are some exceptions to this.)

What I will teach you to use in this text is DAO (Data Access Objects). You will get familiar with opening a database and retrieving/adding/deleting/updating records from tables. I will only use an Access Database (\*.mdb) in my examples, since this is the most used DBMS (DataBase Management System) for smaller applications made in Visual Basic. We will at the end of this lesson have made a simple, yet functional, phone book application.



## 1.10. Database Object.

The first thing we must do in our application is to open a database where our tables are stored. we need to declare a variable to hold our database in order to do this. This is done with:

```
Dim dbMyDB As Database
```

This gives us a variable/object that can hold a reference to our database. To open a simple Access database named "MyDatabase.mdb", do this:

```
Set dbMyDB = OpenDatabase("MyDatabase.mdb")
```

So, now we have opened a database. This won't give us any data. What we need to do is open a table in the database. We're not limited to open a single table; sometimes we have two or more tables that are related to each other and linked together with foreign keys, and there are ways to handle this to.

### 1.10.1. RecordSet Object.

Visual Basic uses an object called RecordSet to hold our table. To declare such an object and to open the table, we do this:

```
Dim rsMyRS As RecordSet
```

```
Set rsMyRS = dbMyDB.OpenRecordSet("MyTable", dbOpenDynaset)
```

Well, I declared a RecordSet object and used the Database object's OpenRecordSet method to open a table of type Dynaset. We can open a RecordSet in several modes. VB's online help file explains the different modes and what they are for. The Dynaset mode is the mode I use mostly. It gives us a RecordSet that we can add, delete and modify records in.

### 1.10.2. Accessing records.

Now that we have opened a table (referred to as RecordSet from now on) we want to access the records in it. The RecordSet object allows us to move in it by using the methods MoveFirst, MoveNext, MovePrevious, MoveLast (among others). I will use some of these to fill up a list box with the records of our RecordSet.

To get this example to work, we make a database (with Access) called "MyDatabase.mdb" with the table "MyTable" in it. This table should have the fields "ID" of type "Counter" that you set to be the primary key, the field "Name" of type Text and a field "Phone" of type Text. We add some records to it. We put a list box on a form and call it "lstRecords".

```
Dim dbMyDB As Database
Dim rsMyRS As RecordSet
```

#### **Private Sub Form\_Load()**

```
Set dbMyDB = OpenDatabase("MyDatabase.mdb")
Set rsMyRS = dbMyDB.OpenRecordSet("MyTable", dbOpenDynaset)
```

```
If Not rsMyRS.EOF Then rsMyRS.MoveFirst
Do While Not rsMyRS.EOF
    lstRecords.AddItem rsMyRS!Name
    lstRecords.ItemData(lstRecords.NewIndex) = rsMyRS!ID
    rsMyRS.MoveNext
Loop
```

#### **End Sub**

This will make the list box fill up with our records when the form loads. I have introduced some new concepts with this example. We have all ready covered the first part where we open the table. The line that says `If Not rsMyRS.EOF Then rsMyRS.MoveFirst` tells the program to move to the first record in case there are any records at all. The `EOF` is a Boolean property that is true if the current record is the last. It is also true if there are no records in the RecordSet.

Then we make the program add the "Name" field of all records to the list box by adding the current records field "Name" and moving to the next record. You ask for a field of a RecordSet by putting a `!` between the name of the RecordSet object and the name of the field. The while loop checks to see if there are more records to add.

### **1.10.3. Searching the RecordSet.**

I put the value of the field "ID" in the list box's ItemData property. I did this so that we would know the primary key for all the records in order to search for a record.

If we put a text box somewhere on the form and call it "txtPhone". Then copy the following code to the project.

#### **Private Sub lstRecords\_Click()**

```
rsMyRS.FindFirst "ID=" & Str(lstRecords.ItemData(lstRecords.ListIndex))
txtPhone.Text = rsMyRS!Phone
```

#### **End Sub**

This will display the phone number of the selected person when clicking in the list box. It uses the FindFirst method of the RecordSet object. This takes a string parameter that is like what is after WHERE in a SQL expression. We state the field that we want to search in (here "ID"), then the evaluation criteria (here "=") and last the value to search for.

So what we did was to search for the record with the "ID" field value that was the same as the ItemData property of the selected item in the list box. Then we show the value of the "Phone" field in the text box.

#### 1.10.4. Updating the Database.

We want to be able to update some value of some field when doing database programming. This is done with `Edit` and `Update`. We will try to change the value of the "Phone" field by editing the text in the text box and clicking a button.

If we put a command button on the form and name it "cmdUpdate". Then copy the following code to the project.

```
Private Sub cmdUpdate_Click()
```

```
rsMyRS.Edit  
rsMyRS!Phone = txtPhone.Text  
rsMyRS.Update
```

```
End Sub
```

This changes the value of the "Phone" field of our current record. Imagine the current record being a set of boxes, with a field in each box. The `Edit` method takes the lid off all of the boxes and `Update` puts them back on. When we write `rsMyRS!Phone = txtPhone.Text` we replace the content of the "Phone" box with the content in the text box.

#### 1.10.5. Deleting and Adding records.

##### 1.10.5.1. Deleting.

Deleting records couldn't be simpler. To delete the current record we just invoke the `Delete` method of the RecordSet object. We will put this feature in our little project. If we make one more command button named "cmdDelete" and the following code will do the work of deleting our currently selected person.

```
Private Sub cmdDelete_Click()
```

```
rsMyRS.Delete  
lstRecords.RemoveItem lstRecords.ListIndex
```

```
End Sub
```

I won't even bother to explain that in greater detail =). The first statement deletes the record and the second removes the list box entry.



### 1.10.5.2. Adding.

Adding records is much like updateing, except that we use AddNew instead of Edit. Let's add one more command button to our application.

#### Private Sub cmdNew\_Click()

```
rsMyRS.AddNew
rsMyRS!Name = "A New Person"
lstRecords.AddItem rsMyRS!Name
lstRecords.ItemData(lstRecords.NewIndex) = rsMyRS!ID
rsMyRS!Phone = "Person's Phone Number"
rsMyRS.Update
```

#### End Sub

### 1.11. Managing Data Types.

There are many types of data we come across in our daily life. For example, we need to handle data such as names, addresses, money, date, stock quotes, statistics and etc everyday. Similarly in Visual Basic, we are also going to deal with these kinds of data. However, to be more systematic, VB divides data into different types.

#### Numeric Data

Numeric data are data that consists of numbers, which can be computed mathematically with various standard operators such as add, minus, multiply, divide and so on. In Visual Basic, the numeric data are divided into 7 types, they are summarized in Table1.1

Type	Storage	Range of Values
Byte	1 byte	0 to 255
Integer	2 bytes	-32,768 to 32,767
Long	4 bytes	-2,147,483,648 to 2,147,483,648
Single	4 bytes	-3.402823E+38 to -1.401298E-45 for negative values 1.401298E-45 to 3.402823E+38 for positive values.
Double	8 bytes	-1.79769313486232e+308 to -4.94065645841247E-324 for negative values 4.94065645841247E-324 to 1.79769313486232e+308 for positive values.

Table 1.1: Numeric Data Types

#### Non-numeric Data Types

The nonnumeric data types are summarized in Table1.2

Data Type	Storage	Range
String(fixed length)	Length of string	1 to 65,400 characters
String(variable length)	Length + 10 bytes	0 to 2 billion characters
Date	8 bytes	January 1, 100 to December 31, 9999
Boolean	2 bytes	True or False

**Table 1.2: Nonnumeric Data Types**

## Declaring Variables

In Visual Basic, one needs to declare the variables before using them by assigning names and data types. They are normally declared in the general section of the code's windows using the **Dim** statement.

The format is as follows:

`Dim variableName as DataType`

### Example 1.3

```
Dim password As String
Dim yourName As String
Dim firstnum As Integer
Dim secondnum As Integer
Dim total As Integer
Dim doDate As Date
```

You may also combine them in one line, separating each variable with a comma, as follows: `Dim password As String, yourName As String, firstnum As Integer,.....`

### If.....Then.....Else Statements with Operators

To effectively control the VB program flow, we shall use If...Then...Else statement together with the conditional operators and logical operators. The general format for the if...then...else statement is

```
If conditions Then
VB expressions
Else
VB expressions
End If
```

### Select Case

If you have a lot of conditional statements, using If..Then..Else could be very messy. For multiple conditional statements, it is better to use Select Case. The format is:

### Select Case expression

```
Case value1
    Block of one or more VB statements
Case value2
    Block of one or more VB Statements
End Select
```

### Looping

Visual Basic allows a procedure to be repeated as many times as long as the processor could support. This is generally called looping .

#### 1. Do

Block of one or more VB statements

**Loop While condition**

#### 2. Do

Block of one or more VB statements

**Loop Until condition**

#### 3. For....Next Loop

For counter=startNumber to endNumber (Step increment)

One or more VB statements

**Next**

## 1.12. Introduction to VB Functions.

### Creating Functions

The general format of a function is as follows:

```
Public Function functionName (Arg As dataType,.....) As dataType
or
Private Function functionName (Arg As dataType,.....) As dataType
```

\*Public indicates that the function is applicable to the whole program and Private indicates that the function is only applicable to a certain module or procedure.

### Declaring Array

We could use Public or Dim statement to declare an array just as the way we declare a single variable. The Public statement declares an array that can be used throughout an application while the Dim statement declare an array that could be used only in a local procedure. The general format to declare an array is as follow:



**Dim arrayName(subs) as dataType**

where subs indicates the last subscript in the array.

**Example 13.1**

**Dim FindName(10) as String**

**1.13.Database Application.**

Finally, I want to put end point in that part, with telling about database application. Of course, All topics is not restricted what I emphasize in that part.

Visual basic allows us to manage databases created with different database program such as MS Access, Dbase, Paradox and etc. In this lesson, we are not dealing with how to create database files but we will see how we can access database files in the VB environment

Database File

Database File: This is a file that contains data and the structure of the data. It is a file that is used to store data and the structure of the data. It is a file that is used to store data and the structure of the data.

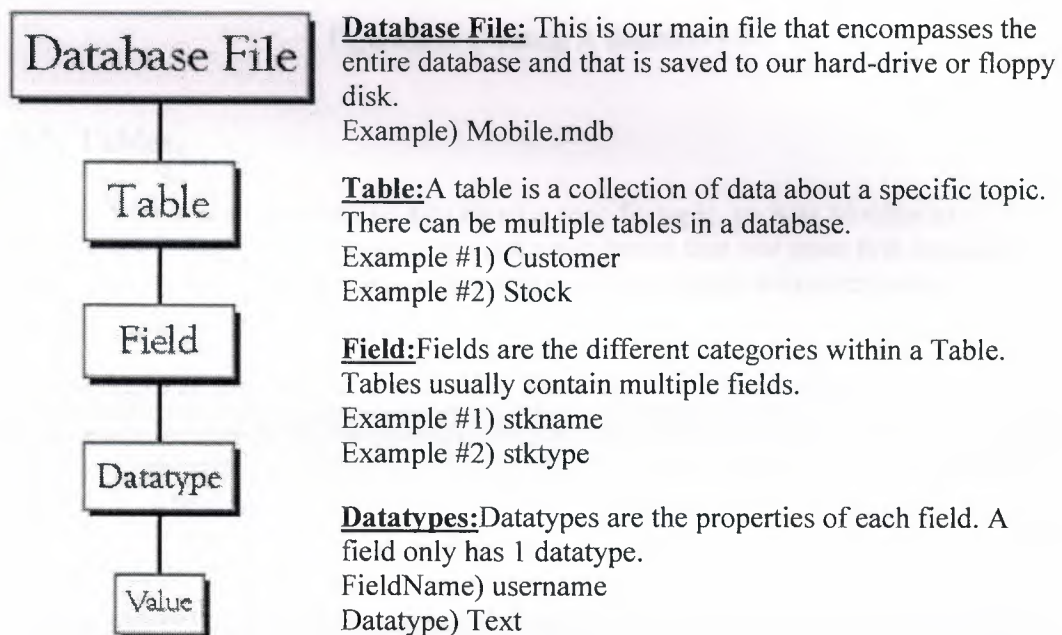
A database is a collection of data about a particular subject. It is a collection of data that is organized in a way that makes it easy to find and use.

## CHAPTER II: DATABASE STRUCTURE

### 2.1. Microsoft Access Description.

Microsoft Access is a powerful program to create and manage our databases. It has many built in features to assist us in constructing and viewing our information. Access is much more involved and is a more genuine database application than other programs such as Microsoft Works.

First of all we need to understand how Microsoft Access breaks down a database. Some keywords involved in this process are: *Database File, Table, Record, Field, Data-type*. Here is the Hierarchy that Microsoft Access uses in breaking down a



database.

### 2.2. Creating a database without using the Database Wizard.

1. When Microsoft Access first starts up, a dialog box is automatically displayed with options to create a new database or open an existing one. If this dialog box is displayed, we click **Blank Access Database**, and then click **OK**.

If we have already opened a database or closed the dialog box that displays when Microsoft Access starts up, we click **New Database** on the toolbar, and then double-click the **Blank Database** icon on the **General** tab.

2. We specify a name and location for the database and we click **Create**. (Below is the screen that shows up following this step)

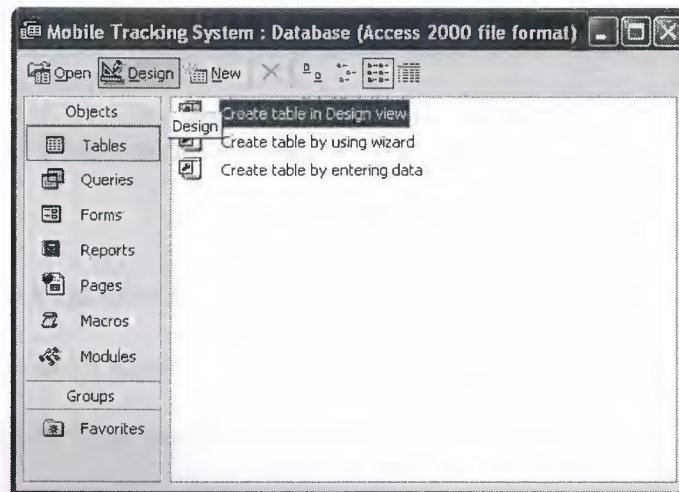


Figure 2.1. Creating A database File

### 2.3. Tables.

A table is a collection of data about a specific topic, such as Mobiles or Customers. Using a separate table for each topic means that you store that data only once, which makes your database more efficient, and reduces data-entry errors.

Tables organize data into columns (called **fields**) and rows (called **records**).

company : Table									
	id	cname	tel	fax	branch	address	email	web	adddate
	1	AKSOTECH LTD	03922281758	03922281758	1	Yzb. Tekin Yure	info@aksotech.	www.aksotech.r	12/18/2006
	5	RSB							12/18/2006
	6	CWS							12/18/2006
*	(AutoNumber)								

Figure 2.2. Company Table As an Example in Access

### 2.4. Primary Key.

- One or more fields (columns) whose value or values uniquely identify each record in a table. A primary key does not allow Null values and must always have a unique value. A primary key is used to relate a table to foreign keys in other tables.



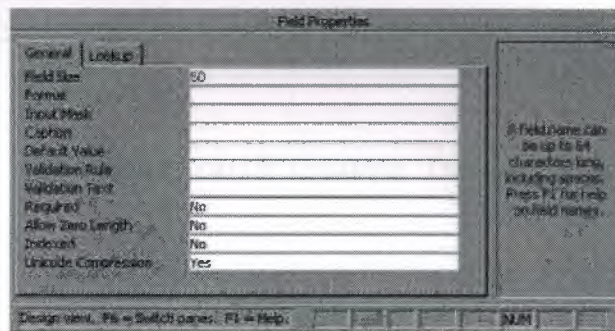
- We do not have to define a primary key, but it's usually a good idea. If you don't define a primary key, Microsoft Access asks us if we would like to create one when we save the table.
- For our tutorial, we make the **Soc Sec #** field the primary key, meaning that *every* one has a social security number and no 2 are the same.

## Advanced Table Features w/Microsoft Access

- **Assigning a field a specific set of characters**

Example: Customer Name needs maximum 20 character.

1. Switch to Design View
2. Select the field you want to alter
3. At the bottom select the General Tab



4. Select **Field Size**
5. Enter the number of characters you want this field to have

- **Formatting a field to look a specific way**

Example) Formatting Phone Number w/ Area Code (xxx) xxx-xxxx

1. Switch to Design View
  2. Select the field you want to format
  3. At the bottom select the General Tab
  4. Select **Input Mask Box** and click on the ... button at the right.
- **Selecting a value from a dropdown box with a set of values that you assign to it. This saves you from typing it in each time**

Example) Choosing a city that is either Auburn, Bay City, Flint, Midland, or Saginaw

1. Switch to Design View
2. Select the field you want to alter (City)

3. At the bottom select the Lookup Tab
4. In the **Display Control** box, select **Combo Box**
5. Under **Row Source Type**, select **Value List**

## 2.5. Tables Structure In Database.

As I mentioned before I used seven tables in Access Database. All the tables are given below.

ParVID			
Field Name	Type	Size	Key
dat	Date/Time		
veid	Text		

**Table 2.1. ParVID Database Table**

ParUP			
Field Name	Type	Size	Key
dat	Date/Time		
dis1	Text		

**Table 2.2. ParUP Database Table**

ParFuel			
Field Name	Type	Size	Key
dat	Date/Time		
ltp	Text		

**Table 2.3. ParFuel Database Table**

## 2.6. Microsoft Access Database.

### 2.6.1. Microsoft Access Database Fundamentals.

Are you overwhelmed by the large quantities of data that need to be tracked in your organization? Perhaps you're currently using a paper filing system, text documents or a spreadsheet to keep track of your critical information. If you're searching for a more flexible data management system, a database might be just the salvation you're looking for.

What is a database? Quite simply, it's an organized collection of data. A database management system (DBMS) such as Access, FileMaker Pro, Oracle or SQL Server provides you with the software tools you need to organize that data in a flexible manner. It includes facilities to add, modify or delete data from the database, ask questions (or queries) about the data stored in the database and produce reports summarizing selected contents.

Microsoft Access provides users with one of the simplest and most flexible DBMS solutions on the market today. Regular users of Microsoft products will enjoy the familiar Windows “look and feel” as well as the tight integration with other Microsoft Office family products. An abundance of wizards lessen the complexity of administrative tasks and the ever-present Microsoft Office Helper (you know... the paper clip!) is available for those who care to use it. Before purchasing Access, be sure that your system meets Microsoft’s minimum system requirements. To further our discussion, let’s first examine three of the major components of Access that most database users will encounter – tables, queries, forms. Once we’ve completed that we’ll look at the added benefits of reports, web integration and SQL Server integration:

Payment.DB			
Field Name	Type	Size	Key
Cuscode	Text	50	
Date	Date/Time	Short Date	
Total	Currency	Standart, Decimal: 2	

**Figure2.3. Sample Table**

The table above contains the employee information for our organization characteristics like name, date of birth and title. Examine the construction of the table and you'll find that each column of the table corresponds to a specific employee characteristic (or **attribute** in database terms). Each row corresponds to one particular employee and contains his or her information. That's all there is to it! If it helps, think of each one of these tables as a spreadsheet-style listing of information.

In the previous section, we learned how tables allow us to create the framework for storing information in a database. Obviously, a database that only stored information would be useless -- we need methods to retrieve information as well. If you simply want to recall the information stored in a table, Microsoft Access allows you to open the table and scroll through the records contained within it. However, the real power of a database lies in its capabilities to answer more complex requests, or **queries**. Access queries provide the capability to combine data from multiple tables and place specific conditions on the data retrieved.

### **2.6.2. Creating Table.**

Many techniques allow you to create a database, the fastest of which consists of using one of the provided examples. Microsoft Access 97 shipped with 22 sample databases while Microsoft Access 2000 ships with 10. Furthermore, the 97 version allowed to provide sample data into the database. This is not available with the 2000 release. The databases that ship with Microsoft Access can help you in two main ways: they provide a fast means of creating a database and you can learn from their structure. To create a database using one of the samples, there is a little detail to follow depending on whether you had launched the program already or not. If Microsoft Access is not running, you can start it. When the first dialog box comes up, you can click the second radio button: Access Database Wizard, Pages



The New dialog box displays two property pages labeled General and Databases. If you want to create a database based on one of the samples, you can click the Databases property page. A list of the sample databases appears. You can then choose one and click OK.

When creating a database using one of the samples, depending on the sample you selected, the Database Wizard will display a few objects and suggest some fields for your database. Some fields are already associated with the objects and some other fields can be added. You can examine them, then add some fields you think are important for your database. You will also have the option of selecting a design layout. Some of the sample databases have been configured to require information about the company you are creating the database for.

## **2.7. Introducing Database.**

Databases are designed to offer an organized mechanism for storing, managing and retrieving information. They do so through the use of tables. If you're familiar with spreadsheets like Microsoft Excel, you're probably already accustomed to storing data in tabular form. It's not much of a stretch to make the leap from spreadsheets to databases.

Just like Excel tables, database tables consist of columns and rows. Each column contains a different type of attribute and each row corresponds to a single record. For example, imagine that we were building a database table that contained names and telephone numbers. We'd probably set up columns named "FirstName", "LastName" and "TelephoneNumber." Then we'd simply start adding rows underneath those columns that contained the data we're planning to store.

At this point, you're probably asking yourself an obvious question if a database is so much like a spreadsheet, why can't I just use a spreadsheet? Databases are actually much more powerful than spreadsheets in the way you're able to manipulate data. Here are just a few of the actions that you can perform on a database that would be difficult if not impossible to perform on a spreadsheet:

- Retrieve all records that match certain criteria
- Update records in bulk
- Cross-reference records in different tables
- Perform complex aggregate calculations

As we walk through this tutorial, you'll learn how you can use databases to achieve each of these objectives. Page 2 of this lesson provides you with an overview of how database keys can be used to uniquely identify records and form relationships between tables. Page 3 describes how the Structured Query Language allows you to interact with your database. On page 4, we examine the different types of databases available on the market today.

## **2.8. Database Keys.**

On the previous page of this article, you learned how databases use tables to organize data. As you probably recall, each table consists of a number of rows, each of which corresponds to a single database record. So, how do databases keep all of these records straight? It's through the use of **keys**.

The first type of key we'll discuss is the **primary key**. Every database table should have one or more columns designated as the primary key. The value this key holds should be unique for each record in the database. For example, assume we have a table called Employees that contains personnel information for every employee in our firm. We'd need to select an appropriate primary key that would uniquely identify each employee. Your first thought might be to use the employee's name.

This wouldn't work out very well because it's conceivable that you'd hire two employees with the same name. A better choice might be to use a unique employee ID number that you assign to each employee when they're hired. Some organizations choose to use Social Security Numbers (or similar government identifiers) for this task because each employee already has one and they're guaranteed to be unique. However, the use of Social Security Numbers for this purpose is highly controversial due to privacy concerns.

Once you decide upon a primary key and inform the database of this decision, it will enforce the uniqueness of the key. If you try to insert a record into a table with a primary key that duplicates an existing record, the insert will fail.

Most databases are also capable of generating their own primary keys. Microsoft Access, for example, may be configured to use the AutoNumber data type to assign a unique ID to each record in the table. While effective, this is a bad design practice because it leaves you with a meaningless value in each record in the table. Why not use that space to store something useful?

The other type of key that we'll discuss in this course is the **foreign key**. These keys are used to create relationships between tables. Natural relationships exist between tables in most database structures. Returning to our employees database, let's imagine that we wanted to add a table containing departmental information to the database. This new table might be called Departments and would contain a large amount of information about the department as a whole. We'd also want to include information about the employees in the department, but it would be redundant to have the same information in two tables (Employees and Departments). Instead, we can create a **relationship** between the two tables.

Let's assume that the Departments table uses the Department Name column as the primary key. To create a relationship between the two tables, we add a new column to the Employees table called Department. We then fill in the name of the department to which each employee belongs. We also inform the database that the Department column in the Employees table is a **foreign key** that references the Departments table. The database will then enforce *referential integrity* by ensuring that all of the values in the Departments column of the Employees table have corresponding entries in the Departments table.



Note that there is no uniqueness constraint for a foreign key. We may (and most likely do!) have more than one employee belonging to a single department. Similarly, there's no requirement that an entry in the Departments table have *any* corresponding entry in the Employees table. It is possible that we'd have a department with no employees.

## **2.9. Working with SQL.**

SQL (pronounced "ess-que-el") stands for structured Query Language. SQL is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, Ingress, etc. although most database systems use SQL, most of them also have their own additional proprietary extensions that are usually only used on their system. However, the standard SQL commands such as "select", "insert", "delete", "create" and "drop" can be used to accomplish almost everything that one needs to do with a database.

### **2.9.1. Data Manipulation Language.**

The Data Manipulation Language (DML) is used to retrieve, insert and modify database information. These commands will be used by all database users during the routine operation of the database. Let's take a brief look at the basic DML commands:

The Data Manipulation Language (DML) is used to retrieve, insert and modify database information. These commands will be used by all database users during the routine operation of the database. Let's take a brief look at the basic DML commands:

#### **2.9.1.1.INSERT.**

The INSERT command in SQL is used to add records to an existing table. Returning to the personal\_info example from the previous section, let's imagine that our HR department needs to add a new employee to their database. They could use a command similar to the one shown below:

```
INSERT INTO department  
VALUES('c01','computer','engineering',5,6)
```

Note that there are four values specified for the record.

These correspond to the table attributes in the order they were defined: first\_name, last\_name, employee\_id, and salary.

#### **2.9.1.2.SELECT.**

The SELECT command is the most commonly used command in SQL. It allows database users to retrieve the specific information they desire from an operational database. Let's take a look at a few examples, again using the personal\_info table from our employees database.



The command shown below retrieves all of the information contained within the personal\_info table. Note that the asterisk is used as a wildcard in SQL. This literally means "Select everything from the personal\_info table."

```
SELECT *  
FROM student
```

Alternatively, users may want to limit the attributes that are retrieved from the database. For example, the Human Resources department may require a list of the last names of all employees in the company. The following SQL command would retrieve only that information:

```
SELECT stname  
FROM student WHERE stno=20020872
```

## CHAPTER III : CONCRETE DELIVERY PROCESS& COST ANALYSIS PROGRAM USING VISUAL BASIC LANGUAGE

### 3.1. Login Page.

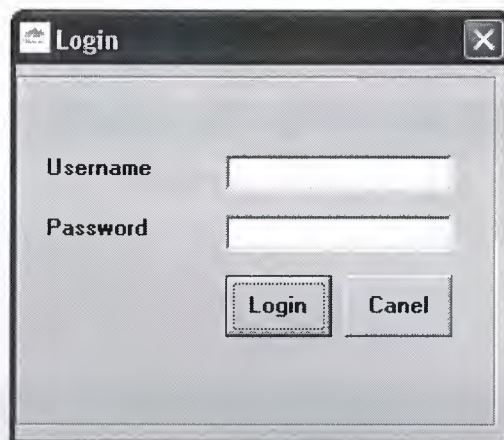


Figure 3.1 Login Page

When we run our program login page window appears and this window contains Username and Password.

### 3.2. Main Page.



**Figure 3.2 Main Page**

When we run our program main page window appears and this window contains menu bar, Time and Date and at the background a truck picture.

### **3.3. Menu Bar.**



**Figure 3.3 Menu Bar**

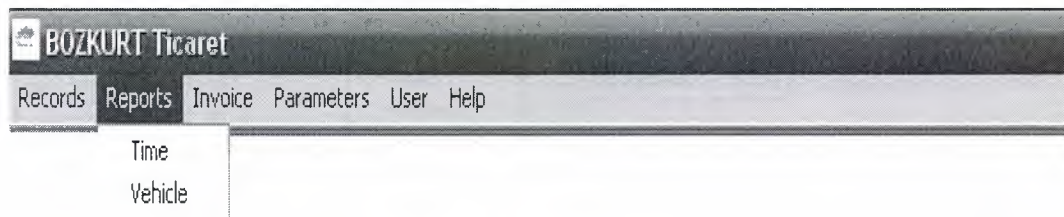
#### **3.3.1. Record Menu.**



**Figure 3.4 Record Menu**

This menu contains Add Record menu and List Record menu. When we press to the Add Record menu the program opens the record adding page. When we press to the List Record menu the program opens the record search and editing page.

#### **3.3.2. Reports Menu.**



**Figure 3.5 Reports Menu**

This menu contains Time Reports menu and Vehicle Reports menu. If we press to the Time Reports the program opens the calculation of times page, if we press to the Vehicle Report menu the program will open the vehicle reports page.



### 3.3.3. Invoice Menu.

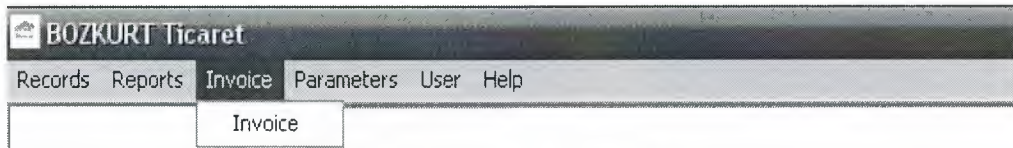


Figure 3.6 Invoice Menu

This menu contains Invoice menu. When the Invoice is pressed the calculation of Invoice page is opening.

### 3.3.4. Parameters Menu.

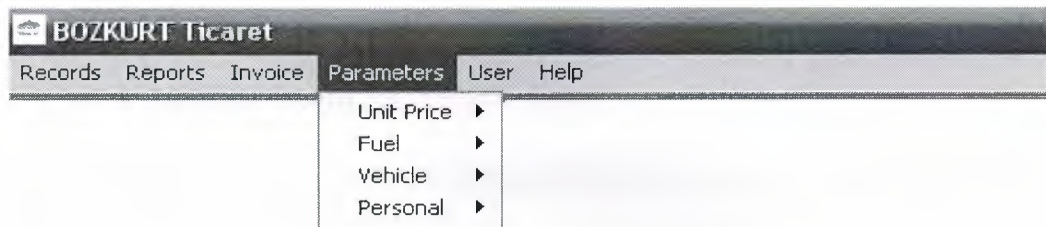


Figure 3.7 Parameters Menu

This menu contains Unit Price menu, Fuel menu, Vehicle menu, Personal menu when we press to the Unit Price menu the menu to adding and listing the Unit Price is opening, when we press to the Fuel menu the adding and listing the Fuel Price menu is opening, adding and listing the Vehicle menu is opening, adding and listing the Personal menu is opening will be opened.

#### 3.3.4.1. Unit Price Menu.

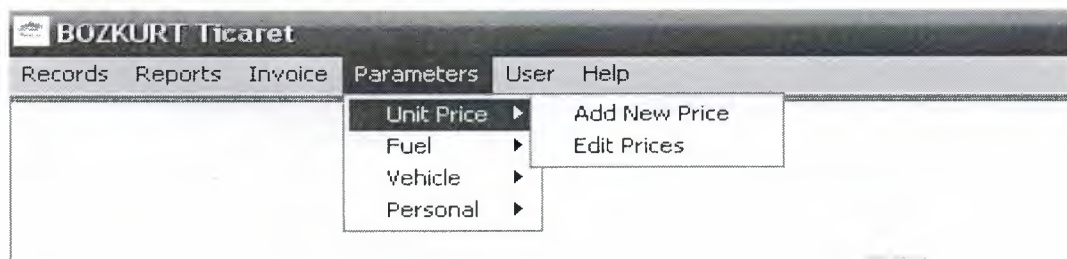
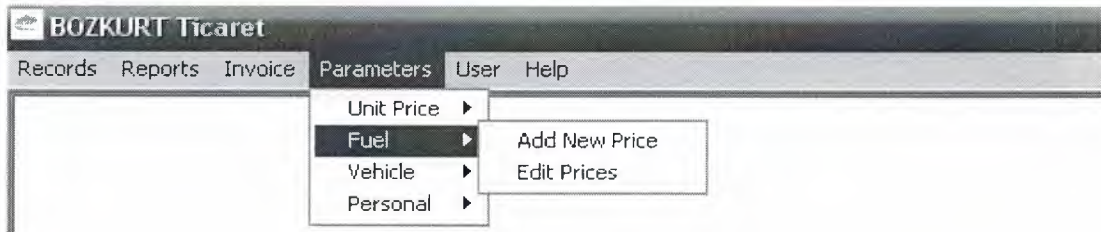


Figure 3.7.1 Unit Price Menu

This menu contains Adding and Listing Unit Price menu. When the Adding Unit Price is pressed add new unit price for km. page is opening, if we press to the Listing Unit Price menu the program will open the listing and editing unit price page.

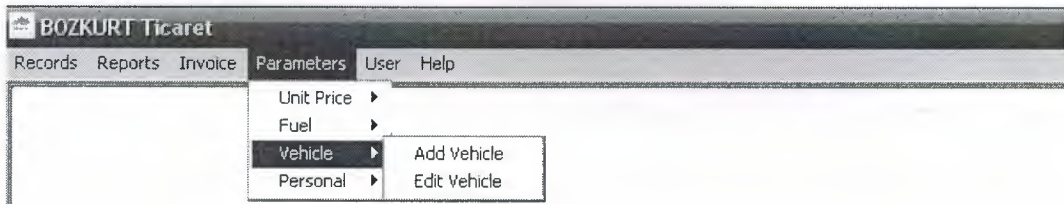
### 3.3.4.2. Fuel Price Menu.



**Figure 3.7.2 Unit Price Menu**

This menu contains Adding and Listing Fuel Price menu. When the Adding Fuel Price is pressed add new fuel price for lt. page is opening, if we press to the Listing Fuel Price menu the program will open the listing and editing fuel price page.

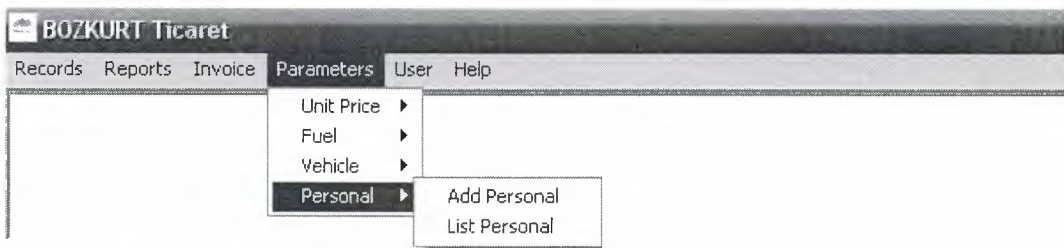
### 3.3.4.3. Vehicle Menu.



**Figure 3.7.3 Vehicle Menu**

This menu contains Adding and Listing Vehicle ID menu. When the Adding Vehicle ID is pressed add new vehicle id page is opening, if we press to the Listing Vehicle ID menu the program will open the listing and editing vehicle id page.

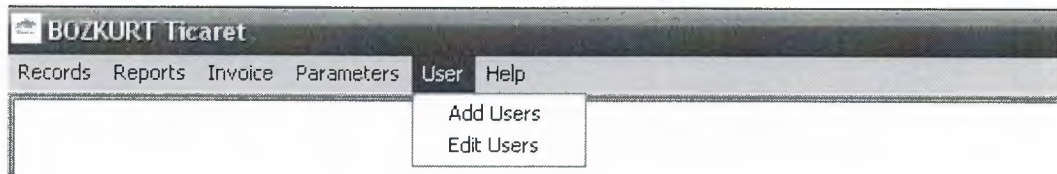
### 3.3.4.4. Personal Menu.



**Figure 3.7.4 Personal Menu**

This menu contains Adding and Listing Personal menu. When the Adding Personal is pressed add new personal page is opening, if we press to the Listing Personal menu the program will open the listing and editing personal page.

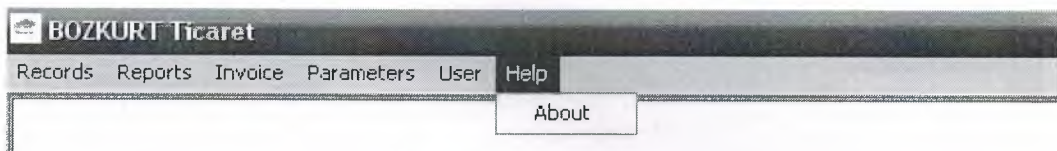
### 3.3.5. User Menu.



**Figure 3.8 User Menu**

This menu contains Add User menu and Listing User menu. When we press to the Add User menu the program runs user adding page and when we press to the Listing User menu the program runs the listing and editing page of user.

### 3.3.6. About Menu.



**Figure 3.9 About Menu**

This menu contains the information about the programmer of the program.



### 3.4. Add New Record Page.

The screenshot shows a software window titled "Vehicle Records" with a close button (X) in the top right corner. The form is divided into two main sections. The top section contains fields for: Reference No., Date, Vehicle ID (with a dropdown arrow), Personnel Name (with a dropdown arrow), Beginning Time of Work, Ending Time of Work, Beginning Km. of Vehicle, Ending Km. of Vehicle, Beginning Time of Engine, Ending Time of Engine, and Fuel. A "Clear" button is located at the bottom right of this section. The bottom section contains fields for: Lap No., Waybill No., Customer Name, Region, Filling Time of Concrete, Leaving Time from Central, Arriving Time to Yard, Leaving Time from Yard, Arriving Time to Central, Distance, Concrete Type, Volume, and Casting Place of Concrete. At the bottom of the window, there are two buttons: "Add Record" and "Cancel".

Reference No :	<input type="text"/>	Beginning Km. of Vehicle :	<input type="text"/>
Date :	<input type="text"/>	Ending Km. of Vehicle :	<input type="text"/>
Vehicle ID :	<input type="text"/>	Beginning Time of Engine :	<input type="text"/>
Personel Name :	<input type="text"/>	Ending Time of Engine :	<input type="text"/>
Beginning Time of Work :	<input type="text"/>	Fuel :	<input type="text"/>
Ending Time of Work :	<input type="text"/>		

Clear

Lap No :	<input type="text"/>	Leaving Time from Yard :	<input type="text"/>
Waybill No :	<input type="text"/>	Arriving Time to Central :	<input type="text"/>
Customer Name :	<input type="text"/>	Distance :	<input type="text"/>
Region :	<input type="text"/>	Concrete Type :	<input type="text"/>
Filling Time of Concrete :	<input type="text"/>	Volume :	<input type="text"/>
Leaving Time from Central :	<input type="text"/>	Casting Place of Concrete :	<input type="text"/>
Arriving Time to Yard :	<input type="text"/>		

Add Record Cancel

**Figure 3.10 Add New Record Page**

In this page the user can add a new vehicle records to the database.

### 3.5. Edit Record Page.

List of Records

Date  Vehicle ID  Lap No

Reference No	Date	Vehicle ID	Lap No	Waybill No	Customer Name	Region	Concrete Type	Casting Place
1	02.01.2008	34 AR 9602	1	6	Bumen Shefik	BLK	C30	P
2	02.01.2008	34 AR 9603	2	5	Bumen Shefik	BLK	C30	P
3	02.01.2008	34 AR 9602	1	4	Bumen Shefik	BLK	C30	P
4	03.01.2008	34 AR 9603	1	6	Bumen Shefik	BLK	C30	P
5	04.01.2008	34 AR 9602	1	9	Bumen Shefik	BLK	C30	P
6	06.01.2008	34 AR 9603	1	11	Bumen Shefik	BLK	C30	P
7	07.01.2008	34 AR 9602	1	12	Bumen Shefik	BLK	C30	P

Reference No  B. KM of Vehicle  Waybill No  Leaving Time Yard

Date  E. KM of Vehicle  Customer Name  Arriving Time Central

Vehicle ID  B. Time of Engine  Region  Distance

Personal Name  E. Time of Engine  Filling Time of Con.  Concrete Type

B. Time of Work  Fuel  Leaving Time Central  Volume

E. Time of Work  Lap No  Arriving Time Yard  Casting Place

**Figure 3.11 Edit Record Page**

In this page the user can filter a record from database, can edit or delete a record from database.



3.6. Time Report Page.

Date

Vehicle ID

Date	Vehicle ID	Filling Concrete Period	Arriving Yard Period	Discharge Concrete Period	Arriving Central Period	Total Lap Time
02.01.2008	34 AR 9602	00:15	00:45	00:15	00:45	02:00
02.01.2008	34 AR 9603	01:00	01:00	00:30	00:30	03:00
02.01.2008	34 AR 9602	01:00	01:00	01:00	01:00	04:00
03.01.2008	34 AR 9603	00:15	00:45	00:15	00:45	02:00
04.01.2008	34 AR 9602	00:15	00:45	00:15	00:45	02:00
06.01.2008	34 AR 9603	00:15	00:45	00:15	00:45	02:00
07.01.2008	34 AR 9602	00:15	00:45	00:15	00:45	02:00

Beginning Date

Ending Date

Vehicle ID

Report Time

Cancel

Total Lap

Average Concrete Filling Period

Average Arriving Yard Period

Average Discharge Concrete Period

Average Arriving Central Period

Average Lap Time

Figure 3.12 Time Report Page

In this page the user can calculate the average time of filling concrete, arriving yard, discharge concrete, arriving central and total lap time periods getting data from the database.



### 3.7. Transportation Report Page.

Transportation Report

Date:  Vehicle ID:

Date	Vehicle ID	KM	M <sup>3</sup>	Lt.
02.01.2008	34 AR 9602	15	12	125
02.01.2008	34 AR 9603	150	12	250
02.01.2008	34 AR 9602	50	12	250
03.01.2008	34 AR 9603	15	9	125
04.01.2008	34 AR 9602	50	10	125
06.01.2008	34 AR 9603	50	10	125
07.01.2008	34 AR 9602	50	10	125

Search Report

Beginning Date:

Ending Date:

Vehicle ID:

Total

Total M<sup>3</sup>:

Total KM:

Total Lt.:

Rates

Lt / M<sup>3</sup>:

Lt / KM:

KM / M<sup>3</sup>:

Figure 3.13 Transportation Report Page

In this page the user can calculate total m<sup>3</sup>, litre, km., litre/ m<sup>3</sup>, km/ m<sup>3</sup>, km/ m<sup>3</sup> Between two date from the database.

### 3.8. Invoice Page.

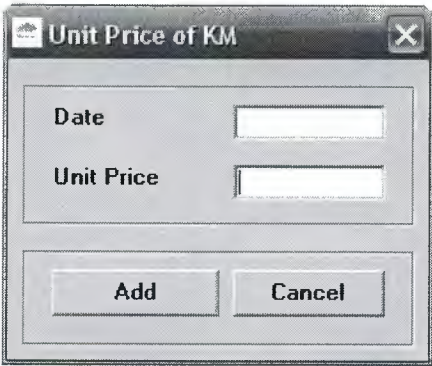
The screenshot shows a software window titled "Invoice Module". At the top, there are two input fields: "Date" and "Vehicle ID". Below these is a table with four columns: "Date", "Vehicle ID", "Region", and "M³". The table contains seven rows of data. Below the table is a horizontal scrollbar. At the bottom of the window, there are two main sections. The left section contains three input fields: "Beginning Date", "Ending Date", and "Vehicle ID", followed by two buttons: "Calculate Invoice" and "Cancel". The right section contains five input fields: "Total M³", "Unit Price", "SUM", "TAX" (with a percentage symbol), and "G. TOTAL".

Date	Vehicle ID	Region	M³
02.01.2008	34 AR 9602	BLK	12
02.01.2008	34 AR 9603	BLK	12
02.01.2008	34 AR 9602	BLK	12
03.01.2008	34 AR 9603	BLK	9
04.01.2008	34 AR 9602	BLK	10
06.01.2008	34 AR 9603	BLK	10
07.01.2008	34 AR 9602	BLK	10

**Figure 3.14 Invoice Page**

In this page the user can calculate the between two date Invoice sum from the database.

**3.9. Add New Unit Price Page.**

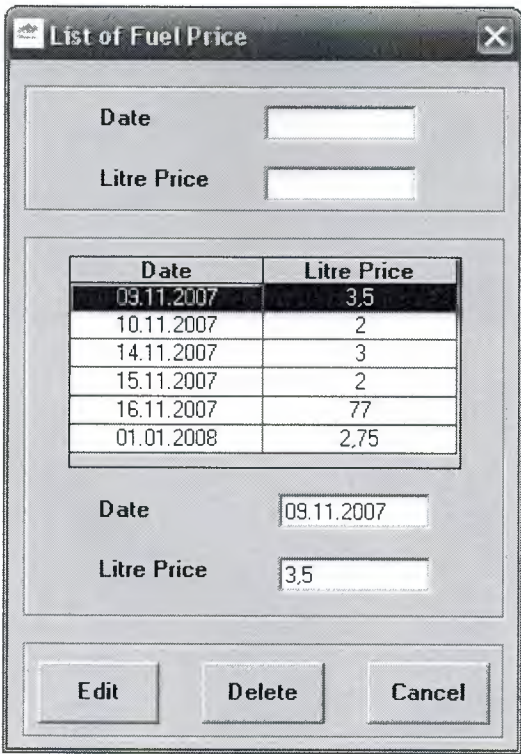


A dialog box titled "Unit Price of KM" with a close button (X) in the top right corner. It contains two input fields: "Date" and "Unit Price". At the bottom, there are two buttons: "Add" and "Cancel".

**Figure 3.15 Add New Unit Price Page**

In this page of the program the user can add new unit price for kilometer to the database.

**3.10. Edit Unit Price Page.**



A dialog box titled "List of Fuel Price" with a close button (X) in the top right corner. It contains two input fields: "Date" and "Litre Price". Below these is a table with two columns: "Date" and "Litre Price". The table has six rows of data. At the bottom, there are three buttons: "Edit", "Delete", and "Cancel".

Date	Litre Price
09.11.2007	3,5
10.11.2007	2
14.11.2007	3
15.11.2007	2
16.11.2007	77
01.01.2008	2,75

**Figure 3.16 Edit Unit Price Page**

In this part of the program the user can list, edit and delete the unit price for kilometer to the database.



**3.11. Add New Fuel Price Page.**

The 'Fuel' dialog box contains two input fields: 'Date :' and 'Litre Price'. Below these fields are two buttons: 'Add' and 'Cancel'.

**Figure 3.17 Add New Unit Price Page**

In this page of the program the user can add new fuel price for litre to the database.

**3.12. Edit Fuel Price Page.**

The 'List of Fuel Price' dialog box features a table with two columns: 'Date' and 'Litre Price'. Above the table are input fields for 'Date' and 'Litre Price'. Below the table are input fields for 'Date' (containing '09.11.2007') and 'Litre Price' (containing '3,5'). At the bottom are three buttons: 'Edit', 'Delete', and 'Cancel'.

Date	Litre Price
09.11.2007	3,5
10.11.2007	2
14.11.2007	3
15.11.2007	2
16.11.2007	77
01.01.2008	2,75

**Figure 3.18 Edit Fuel Price Page**

In this part of the program the user can list, edit and delete the fuel price for litre to the database.

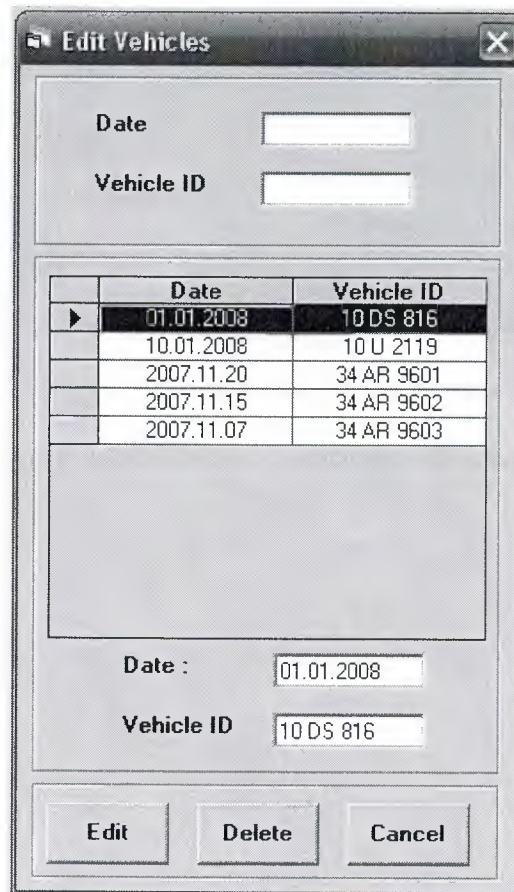
### 3.13. Add New Vehicle ID Page.



**Figure 3.19 Add New Vehicle ID Page**

In this page of the program the user can add new vehicle id to the database.

### 3.14. Edit Vehicle ID Page.



	Date	Vehicle ID
▶	01.01.2008	10 DS 816
	10.01.2008	10 U 2119
	2007.11.20	34 AR 9601
	2007.11.15	34 AR 9602
	2007.11.07	34 AR 9603

**Figure 3.20 Edit Vehicle ID Page**

In this part of the program the user can list, edit and delete the vehicle id to the database.

### 3.15. Add New Personal Page.



**Personal Record Page**

Personel No :

R.T. Identity No :  Identity Serial No :

Name :  Surname :

Birth Place :  Birth Date :

Mother Name :  Father Name :

Address :

County :

Phone :

Mobile :

Driving Licience No :  Social Insurance No :

Driving Licience Class  Begining Date of Work

Blood Group :

**Figure 3.21 Add New Personal Page**

In this page of the program the user can add new personal to the database.

### **3.16. Edit Personal Page.**



The 'List of Personal' window contains a table with the following data:

Personal No	Rep. of Türkiye Identity No	Personal Name	Social Insurance No
1	249373	Mücahit Bozkurt	212321
2	10000	Burak	44444

Below the table is a large empty area. To the right of the table is a form for editing a record. The form includes the following fields:

- Personal No: 1
- Address: Balıkesir
- R.T Identity No: 249373
- County: Merkez
- Identity Serial No: 252763
- Phone: 227 29 23
- Name: Mücahit Bozkurt
- Mobile: 536 707 00 81
- Surname: Bozkurt
- Driving Licence No: 138937
- Birth Place: Balıkesir
- Driving Licence Class: B
- Birth Date: 02.12.1983
- Blood Group: A RH(+)
- Mother Name: Nebbe
- Social Insurance no: 212321
- Father Name: Yakup
- Beginning Date of Work: 01.01.2000

At the bottom right of the form are three buttons: 'Edit', 'Delete', and 'Cancel'.

**Figure 3.22 Edit Personal Page**

In this part of the program the user can list, edit and delete the personal to the database.

### 3.17. Add New User Page.

The 'Users Manage' window contains the following fields and buttons:

- Username: [Text Field]
- Password: [Text Field]
- Re-Type Password: [Text Field]
- Add User: [Button]
- Cancel: [Button]

**Figure 3.23 Add New User Page**

This part of the program is used to add add new user to the data base.

### 3.18. Edit User Page.

The screenshot shows a Windows-style dialog box titled "ListUser". Inside the dialog, there is a section for listing users with a table. The table has two columns: "Username" and "Password". The first row of the table contains the values "Sebahat" and "s". Below the table, there are two input fields: "Username" containing "Sebahat" and "Password" containing "s". At the bottom of the dialog, there are three buttons: "Edit User", "Delete User", and "Cancel".

Username	Password
Sebahat	s

**Figure 3.24 Edit User Page**

In this part of the program the user can list, edit and delete the username or password to the database.

### 3.19. About Page.



**Figure 3.25 About Page**

This part of the program is used to give information about the programmer.



## CONCLUSION

Through building of this project we learn many things, because until now we were just learning programs literary or we were doing small programs, but by the way of this program we investigated a lot in internet, we read many books, and we tried many examples in order to improve my knowledge about Visual Basic to make my program a useful thing.

There are many advantages and disadvantages of using Visual Basic in programming, the one of advantages of using VB it is easy in use, and one of the disadvantages of using Visual Basic it is to slow than the some of the other programming languages and it does not have so much properties.

The Program can be updated in the future, we can add other properties to it, and we can use it through internet by adding extra codes, while doing the project wonderful imaginations come to my mind but if we applied them we couldn't be able to finish this project on time, but we'll do my imaginations project in the future if God Wishes.

My Project program is a tracking concrete transportation program, we used Concrete Delivery Process & Cost Analysis, because this program wanted from a concrete transporter company. And this program is designed for a private company so that this program is not useful in general purpose.

## REFERENCES

1. Ihsan Karagülle and Zeydin Pala, Microsoft Visual Basic 6.0 Pro, Türkmen Printing House, Istanbul ,2001.
2. Yakup Varol and Hasan Koca, VB KodBank 1.7, E-Book, World Wide Web [www.vbkodbank.com](http://www.vbkodbank.com) ,2001.
3. A guide research for writing program. Retrieved December 15, 2006 from the World Wide Web "<http://www.programlama.com>
4. A guide research for writing program. Retrieved January 3, 2007 from the World Wide Web "[http:// www.devarticles.com](http://www.devarticles.com)

## APPENDICES

### Entrance.frm

```
Private Sub Command1_Click()
```

```
Data7.RecordSource = ("select * from userx where USERNAME = '" & Text1 & "' and  
PASSWORD = '" & Text2 & "'")
```

```
Data7.Refresh
```

```
If Data7.Recordset.RecordCount < 1 Then
```

```
MsgBox "Wrong Username or Password, Please Try Again", vbCritical, "Username or  
Password Mistake!"
```

```
Text1.SetFocus
```

```
Exit Sub
```

```
Else
```

```
End If
```

```
If Text1.Text = "" Or Text2.Text = "" Then
```

```
MsgBox "Wrong Username or Password, Please Try Again", vbCritical, "Username or  
Password Mistake!"
```

```
Text1.SetFocus
```

```
Exit Sub
```

```
Else
```

```
Unload Entrance
```

```
Load Main
```

```
Main.Show
```

```
End If
```

```
End Sub
```

---

```
Private Sub Command2_Click()
```

```
Unload Me
```

```
End Sub
```

---

```
Private Sub Form_Load()
```

```
Move (Screen.Width - Width) / 2, (Screen.Height - Height) / 2
```

```
Data7.DatabaseName = App.Path & "\btp.mdb"
```

```
Text1.Text = ""
```

```
Text2.Text = ""
```

```
End Sub
```



## MDIForm1.frm

```
Private Sub Label10_Click()  
ParUP.Show  
End Sub
```

---

```
Private Sub Label11_Click()  
ListUP.Show  
End Sub
```

---

```
Private Sub label13_Click()  
ParFuel.Show  
End Sub
```

---

```
Private Sub label14_Click()  
ListFuel.Show  
End Sub
```

---

```
Private Sub label16_Click()  
ParVID.Show  
End Sub
```

---

```
Private Sub label17_Click()  
ListVID.Show  
End Sub
```

---

```
Private Sub label2_Click()  
RecVeh.Show  
End Sub
```

---

```
Private Sub label20_Click()  
ParUsers.Show  
End Sub
```

---

```
Private Sub label21_Click()  
ListUser.Show  
End Sub
```

---

```
Private Sub label23_Click()  
About.Show  
End Sub
```

---

```
Private Sub Label24_Click()  
ListReco.Show  
End Sub
```

---

```
Private Sub label25_Click()  
ParPers.Show  
End Sub
```

```
Private Sub label26_Click()  
ListPers.Show  
End Sub
```

---

```
Private Sub label4_Click()  
RepVehTime.Show  
End Sub
```

---

```
Private Sub Label5_Click()  
RepVehTran.Show  
End Sub
```

---

```
Private Sub Label7_Click()  
Invoice.Show  
End Sub
```

---

```
Private Sub Timer1_Timer()  
Text1 = Date  
Text2 = Time  
End Sub
```

#### **Main.frm**

```
Private Sub Timer1_Timer()  
Text1 = Date  
Text2 = Time  
End Sub
```

#### **RecVeh.frm**

```
Private Sub Command1_Click()  
Unload Me  
End Sub
```

---

```
Private Sub Command10_Click()  
Text1.Text = ""  
Text2.Text = ""  
Text3.Text = ""  
Text4.Text = ""  
Text5.Text = ""  
Text6.Text = ""  
Text7.Text = ""  
Text8.Text = ""  
Text9.Text = ""
```

```
End Sub
```

---

```
Private Sub Command3_Click()
```

```

If Text1.Text <> "" Or Text2.Text <> "" Or Text14.Text <> "" Then
Text23 = CSng((CDate(Text15) - CDate(Text14)) * 24)
Text23 = surebul(Text23)
Text24 = CSng((CDate(Text16) - CDate(Text15)) * 24)
Text24 = surebul(Text24)
Text25 = CSng((CDate(Text17) - CDate(Text16)) * 24)
Text25 = surebul(Text25)
Text26 = CSng((CDate(Text18) - CDate(Text17)) * 24)
Text26 = surebul(Text26)
a = (CDate(Text23) + CDate(Text24) + CDate(Text25) + CDate(Text26))
Text27 = a
If Text1.Text <> "" And Text19.Text > (Val(Text6.Text) - Val(Text5.Text)) And
Text23.Text <> "" And Text24.Text <> "" And Text25.Text <> "" And Text26.Text <>
"" And Text4.Text <> "" Then
MsgBox " Distance Can Not Be Bigger Then Total Distance "
Else
Data1.Recordset.AddNew
Data1.Recordset.refno = Text1.Text
Data1.Recordset.dat = Text2.Text
Data1.Recordset.btw = Text3.Text
Data1.Recordset.etw = Text4.Text
Data1.Recordset.bkv = Text5.Text
Data1.Recordset.ekv = Text6.Text
Data1.Recordset.bte = Text7.Text
Data1.Recordset.ete = Text8.Text
Data1.Recordset.fuel = Text9.Text
Data1.Recordset.lapno = Text10.Text
Data1.Recordset.wbn = Text11.Text
Data1.Recordset.cuna = Text12.Text
Data1.Recordset.rgn = Text13.Text
Data1.Recordset.ftc = Text14.Text
Data1.Recordset.fcp = Text23.Text
Data1.Recordset.ltc = Text15.Text
Data1.Recordset.ayp = Text24.Text
Data1.Recordset.aty = Text16.Text
Data1.Recordset.dcp = Text25.Text
Data1.Recordset.lty = Text17.Text
Data1.Recordset.acp = Text26.Text
Data1.Recordset.atc = Text18.Text
Data1.Recordset.tlt = Text27.Text
Data1.Recordset.dis = Text19.Text
Data1.Recordset.ct = Text20.Text
Data1.Recordset.vol = Text21.Text
Data1.Recordset.cpc = Text22.Text
Data1.Recordset.veid = DBCombo1.Text
Data1.Recordset.pena = DBCombo2.Text

Data1.Recordset.Update
Data1.Refresh
Text10.Text = ""

```



```
Text11.Text = ""
Text12.Text = ""
Text13.Text = ""
Text14.Text = ""
Text15.Text = ""
Text16.Text = ""
Text17.Text = ""
Text18.Text = ""
Text19.Text = ""
Text20.Text = ""
Text21.Text = ""
Text22.Text = ""
```

```
Text2.SetFocus
```

```
End If
Else
```

```
MsgBox "You can not leave the Referance No, Date or Filling Concrete Time empty.",
48, "Caution Message"
```

```
    If Text2.Text = "" Then
        Text1.SetFocus
```

```
    End If
End If
```

```
End Sub
```

---

```
Private Sub Form_Load()
Data1.DatabaseName = App.Path & "\btp.mdb"
Data2.DatabaseName = App.Path & "\btp.mdb"
Data3.DatabaseName = App.Path & "\btp.mdb"
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
Text7.Text = ""
Text8.Text = ""
Text9.Text = ""
Text10.Text = ""
Text11.Text = ""
Text12.Text = ""
Text13.Text = ""
Text14.Text = ""
Text15.Text = ""
Text16.Text = ""
Text17.Text = ""
```

```
Text18.Text = ""
Text19.Text = ""
Text20.Text = ""
Text21.Text = ""
Text22.Text = ""
```

```
End Sub
```

---

```
Public Function surebul(sure As Single) As String
Dim saniye As Single
Dim saatdeg As Single
Dim dakikadeg As Single
Dim saniyedeg As Single
```

```
saniye = sure * 3600
```

```
If saniye >= 60 Then
dakikadeg = saniye \ 60
saniyedeg = saniye Mod 60
```

```
If dakikadeg >= 60 Then
```

```
saatdeg = dakikadeg \ 60
dakikadeg = dakikadeg Mod 60
If dakikadeg < 10 And saniyedeg < 10 Then
surebul = saatdeg & ":0" & dakikadeg & ":0" & saniyedeg
ElseIf dakikadeg < 10 And saniyedeg > 10 Then
surebul = saatdeg & ":0" & dakikadeg & ":" & saniyedeg
ElseIf dakikadeg > 10 And saniyedeg < 10 Then
surebul = saatdeg & ":" & dakikadeg & ":0" & saniyedeg
ElseIf dakikadeg > 10 And saniyedeg > 10 Then
surebul = saatdeg & ":" & dakikadeg & ":" & saniyedeg
End If
Else
surebul = "00" & ":" & dakikadeg & ":" & saniyedeg
End If
Else
surebul = "00:" & "00:" & saniye
End If
```

```
End Function
```

```

Private Sub Command1_Click()
Text28 = CSng((CDate(Text20) - CDate(Text19)) * 24)
Text28 = surebul(Text28)
Text29 = CSng((CDate(Text21) - CDate(Text20)) * 24)
Text29 = surebul(Text29)
Text30 = CSng((CDate(Text22) - CDate(Text21)) * 24)
Text30 = surebul(Text30)
Text31 = CSng((CDate(Text23) - CDate(Text22)) * 24)
Text31 = surebul(Text31)
a = (CDate(Text28) + CDate(Text29) + CDate(Text30) + CDate(Text31))
Text32 = a
If Data1.Recordset.RecordCount = 0 Then
Exit Sub
End If
If Text4.Text <> "" And Text5.Text <> "" Then

```

```

Data1.Recordset.Edit
Data1.Recordset.refno = Text4.Text
Data1.Recordset.dat = Text5.Text
Data1.Recordset.veid = Text6.Text
Data1.Recordset.pena = Text7.Text
Data1.Recordset.btw = Text8.Text
Data1.Recordset.etw = Text9.Text
Data1.Recordset.bkv = Text10.Text
Data1.Recordset.ekv = Text11.Text
Data1.Recordset.bte = Text12.Text
Data1.Recordset.ete = Text13.Text
Data1.Recordset.fuel = Text14.Text
Data1.Recordset.lapno = Text15.Text
Data1.Recordset.wbn = Text16.Text
Data1.Recordset.cuna = Text17.Text
Data1.Recordset.rgn = Text18.Text
Data1.Recordset.ftc = Text19.Text
Data1.Recordset.fcp = Text28.Text
Data1.Recordset.ltc = Text20.Text
Data1.Recordset.ayp = Text29.Text
Data1.Recordset.aty = Text21.Text
Data1.Recordset.dcp = Text30.Text
Data1.Recordset.lty = Text22.Text
Data1.Recordset.acp = Text31.Text
Data1.Recordset.atc = Text23.Text
Data1.Recordset.tlt = Text32.Text
Data1.Recordset.dis = Text24.Text
Data1.Recordset.ct = Text25.Text
Data1.Recordset.vol = Text26.Text
Data1.Recordset.cpc = Text27.Text

```

```

Data1.Recordset.Update

```



```
Data1.Refresh
End If
End Sub
```

---

```
Private Sub Command3_Click()
Unload Me
End Sub
```

---

```
Private Sub Command5_Click()
If Data1.Recordset.RecordCount = 0 Then Exit Sub
answer = MsgBox("Do you want to DELETE this record ?", vbQuestion + vbYesNo +
vbDefaultButton2, "Record Deleting")
If answer = vbYes Then Data1.Recordset.Delete
If answer = vbNo Then Exit Sub
Data1.Refresh
End Sub
```

---

```
Private Sub Form_Load()
Data1.DatabaseName = App.Path & "\btp.mdb"
End Sub
```

---

```
Private Sub Text1_Change()
squ = "SELECT * From vehrec WHERE dat LIKE '"
squ = squ & Text1 & "' order by dat ASC;"
Data1.RecordSource = squ
Data1.Refresh
End Sub
```

---

```
Private Sub Text2_Change()
squ = "SELECT * From vehrec WHERE veid LIKE '"
squ = squ & Text2 & "' order by dat ASC;"
Data1.RecordSource = squ
Data1.Refresh
End Sub
```

---

```
Private Sub Text3_Change()
squ = "SELECT * From vehrec WHERE lapno LIKE '"
squ = squ & Text3 & "' order by dat ASC;"
Data1.RecordSource = squ
Data1.Refresh
End Sub
```

---

```
Private Sub DBGrid1_RowColChange>LastRow As Variant, ByVal LastCol As Integer)
If Data1.Recordset.RecordCount = 0 Then
Text4 = ""
Text5 = ""
Text6 = ""
Text7 = ""
Text8 = ""
```

Text9 = ""  
Text10 = ""  
Text11 = ""  
Text12 = ""  
Text13 = ""  
Text14 = ""  
Text15 = ""  
Text16 = ""  
Text17 = ""  
Text18 = ""  
Text19 = ""  
Text20 = ""  
Text21 = ""  
Text22 = ""  
Text23 = ""  
Text24 = ""  
Text25 = ""  
Text26 = ""  
Text27 = ""  
Exit Sub  
Else

Text4 = Data1.Recordset.refno & ""  
Text5 = Data1.Recordset.dat & ""  
Text6 = Data1.Recordset.veid & ""  
Text7 = Data1.Recordset.pena & ""  
Text8 = Data1.Recordset.btw & ""  
Text9 = Data1.Recordset.etw & ""  
Text10 = Data1.Recordset.bkv & ""  
Text11 = Data1.Recordset.ekv & ""  
Text12 = Data1.Recordset.bte & ""  
Text13 = Data1.Recordset.ete & ""  
Text14 = Data1.Recordset.fuel & ""  
Text15 = Data1.Recordset.lapno & ""  
Text16 = Data1.Recordset.wbn & ""  
Text17 = Data1.Recordset.cuna & ""  
Text18 = Data1.Recordset.rgn & ""  
Text19 = Data1.Recordset.ftc & ""  
Text28 = Data1.Recordset.fcp & ""  
Text20 = Data1.Recordset.ltc & ""  
Text29 = Data1.Recordset.ayp & ""  
Text21 = Data1.Recordset.aty & ""  
Text30 = Data1.Recordset.dcp & ""  
Text22 = Data1.Recordset.lty & ""  
Text31 = Data1.Recordset.acp & ""  
Text23 = Data1.Recordset.atc & ""  
Text32 = Data1.Recordset.tlt & ""  
Text24 = Data1.Recordset.dis & ""  
Text25 = Data1.Recordset.ct & ""

```
Text26 = Data1.Recordset.vol & ""  
Text27 = Data1.Recordset.cpc & ""
```

```
End If
```

```
End Sub
```

---

```
Public Function surebul(sure As Single) As String  
Dim saniye As Single  
Dim saatdeg As Single  
Dim dakikadeg As Single  
Dim saniyedeg As Single
```

```
saniye = sure * 3600
```

```
If saniye >= 60 Then  
dakikadeg = saniye \ 60  
saniyedeg = saniye Mod 60
```

```
If dakikadeg >= 60 Then
```

```
saatdeg = dakikadeg \ 60  
dakikadeg = dakikadeg Mod 60  
If dakikadeg < 10 And saniyedeg < 10 Then  
surebul = saatdeg & ":0" & dakikadeg & ":0" & saniyedeg  
ElseIf dakikadeg < 10 And saniyedeg > 10 Then  
surebul = saatdeg & ":0" & dakikadeg & ":" & saniyedeg  
ElseIf dakikadeg > 10 And saniyedeg < 10 Then  
surebul = saatdeg & ":" & dakikadeg & ":0" & saniyedeg  
ElseIf dakikadeg > 10 And saniyedeg > 10 Then  
surebul = saatdeg & ":" & dakikadeg & ":" & saniyedeg  
End If  
Else  
surebul = "00" & ":" & dakikadeg & ":" & saniyedeg  
End If  
Else  
surebul = "00:" & "00:" & saniye  
End If
```

```
End Function
```

### **RepVehTime.frm**

```
Private Sub Command1_Click()
```

```
If Text10.Text <> "" And Text11.Text <> "" And Text12.Text <> "" Then  
Data1.RecordSource = "select * from vehrec where dat>=" & Text10.Text & "*" and  
dat<=" & Text11.Text & "*" and veid like " & Text12.Text & ""  
Data1.Refresh
```



```
Dim X, Y
X = 0
Y = 0
```

```
Data1.Recordset.MoveFirst
Do Until Data1.Recordset.EOF
Y = Data1.Recordset.RecordCount
Data1.Recordset.MoveNext
Loop
Text8 = Y
```

```
Dim a, b
```

```
a = 0
b = 0
```

```
Data1.Recordset.MoveFirst
Do Until Data1.Recordset.EOF
b = Data1.Recordset![fcp]
a = ((CDate(a) + CDate(b)))
Data1.Recordset.MoveNext
Loop
```

```
Text3 = (((Val(Left(a, 2)) * 60) + (Val(Mid(a, 4, 2)))) / Y)
Text3 = surebul(Text3)
```

```
Dim c, d
```

```
c = 0
d = 0
```

```
Data1.Recordset.MoveFirst
Do Until Data1.Recordset.EOF
d = Data1.Recordset![ayp]
c = ((CDate(c) + CDate(d)))
Data1.Recordset.MoveNext
Loop
```

```
Text4 = (((Val(Left(c, 2)) * 60) + (Val(Mid(c, 4, 2)))) / Y)
Text4 = surebul(Text4)
```

```
Dim e, f
```

```
e = 0
f = 0
```

```
Data1.Recordset.MoveFirst
Do Until Data1.Recordset.EOF
f = Data1.Recordset![dcp]
e = ((CDate(e) + CDate(f)))
```

```

        Data1.Recordset.MoveNext
    Loop

    Text5 = (((Val(Left(e, 2)) * 60) + (Val(Mid(e, 4, 2)))) / Y)
    Text5 = surebul(Text5)

Dim g, h

    g = 0
    h = 0

    Data1.Recordset.MoveFirst
    Do Until Data1.Recordset.EOF
        h = Data1.Recordset![acp]
        g = ((CDate(g) + CDate(h)))
        Data1.Recordset.MoveNext
    Loop

    Text6 = (((Val(Left(g, 2)) * 60) + (Val(Mid(g, 4, 2)))) / Y)
    Text6 = surebul(Text6)

Dim i, j

    i = 0
    j = 0

    Data1.Recordset.MoveFirst
    Do Until Data1.Recordset.EOF
        j = Data1.Recordset![tlt]
        i = ((CDate(i) + CDate(j)))
        Data1.Recordset.MoveNext
    Loop

    Text7 = (((Val(Left(i, 2)) * 60) + (Val(Mid(i, 4, 2)))) / Y)
    Text7 = surebul(Text7)

Else
    MsgBox "You can not leave the Begining and Ending Dates or Vehicle ID empty.",
48, "Caution Message"
    If Text2.Text = "" Then
        Text1.SetFocus
    End If

End If
End Sub



---


Private Sub Command2_Click()
    Unload Me
End Sub


---



```

```
Private Sub Form_Load()
```

```
Data1.DatabaseName = App.Path & "\btp.mdb"
```

```
Text1.Text = ""
```

```
Text2.Text = ""
```

```
Text3.Text = ""
```

```
Text4.Text = ""
```

```
Text5.Text = ""
```

```
Text6.Text = ""
```

```
Text7.Text = ""
```

```
Text8.Text = ""
```

```
Text10.Text = ""
```

```
Text11.Text = ""
```

```
Text12.Text = ""
```

```
End Sub
```

---

```
Private Sub Text1_Change()
```

```
squ = "SELECT * From vehrec WHERE dat LIKE '"
```

```
squ = squ & Text1 & "' order by dat ASC;"
```

```
Data1.RecordSource = squ
```

```
Data1.Refresh
```

```
End Sub
```

---

```
Private Sub Text2_Change()
```

```
squ = "SELECT * From vehrec WHERE veid LIKE '"
```

```
squ = squ & Text2 & "' order by dat ASC;"
```

```
Data1.RecordSource = squ
```

```
Data1.Refresh
```

```
End Sub
```

---

```
Public Function surebul(sure As Single) As String
```

```
Dim saniye As Single
```

```
Dim saatdeg As Single
```

```
Dim dakikadeg As Single
```

```
Dim saniyedeg As Single
```

```
saniye = sure * 60
```

```
If saniye >= 60 Then
```

```
dakikadeg = saniye \ 60
```

```
saniyedeg = saniye Mod 60
```

```
If dakikadeg >= 60 Then
```

```
saatdeg = dakikadeg \ 60
```

```
dakikadeg = dakikadeg Mod 60
```

```
If dakikadeg < 10 And saniyedeg < 10 Then
```

```
surebul = saatdeg & ":0" & dakikadeg & "
```



```

ElseIf dakikadeg < 10 And saniyedeg > 10 Then
surebul = saatdeg & ":0" & dakikadeg & ""
ElseIf dakikadeg > 10 And saniyedeg < 10 Then
surebul = saatdeg & ":" & dakikadeg & ""
ElseIf dakikadeg > 10 And saniyedeg > 10 Then
surebul = saatdeg & ":" & dakikadeg & ""
End If
Else
surebul = "00" & ":" & dakikadeg & ""
End If
Else
surebul = "00:" & "00:" & saniye
End If

End Function

```

### **RepVehTran.frm**

```

Private Sub Command1_Click()

On Error Resume Next

If Text9.Text <> "" And Text10.Text <> "" And Text11.Text <> "" Then

Dim p, q
Data1.Recordset![dat] = p
p = Text9
Data1.Recordset![dat] = q
q = Text10
If p > q Or p = q Then
MsgBox " Begining Date can not be bigger then or equal Ending Date",
vbCritical, "Date Error!"
Exit Sub
Else
If Data1.Recordset.RecordCount = 0 And Text9.Text = "" Or Text10.Text = ""
Or Text11.Text = "" Then
MsgBox ("There is not any record at database")
Exit Sub
Else
Data1.RecordSource = "select * from vehrec where dat>='" &
Text9.Text & "'" and dat<='" & Text10.Text & "'" and veid like '" &
Text11.Text & "'"
Data1.Refresh
If Data1.Recordset.RecordCount <> 0 Then

Dim a, b
a = 0
b = 0
Data1.Recordset.MoveFirst

```

```

        Do Until Data1.Recordset.EOF
            b = Data1.Recordset![vol]
            a = Val(a) + Val(b)
            Data1.Recordset.MoveNext
        Loop
        Text3 = a

Dim c, d
    c = 0
    d = 0
    Data1.Recordset.MoveFirst
    Do Until Data1.Recordset.EOF
        d = Data1.Recordset![dis]
        c = Val(c) + Val(d)
        Data1.Recordset.MoveNext
    Loop
    Text4 = c

Dim e, f
    e = 0
    f = 0
    Data1.Recordset.MoveFirst
    Do Until Data1.Recordset.EOF
        f = Data1.Recordset![fuel]
        e = Val(e) + Val(f)
        Data1.Recordset.MoveNext
    Loop
    Text5 = e

    Text6 = CDBl(Val(e) / Val(a))
    Text7 = CDBl(Val(e) / Val(c))
    Text8 = CDBl(Val(c) / Val(a))
Else
    MsgBox "There is not any record in database"
Exit Sub
End If
End If
End If
Else
    MsgBox "You can not leave the Begining and Ending Dates or Vehicle ID empty.",
48, "Caution Message"
    If Text2.Text = "" Then
        Text1.SetFocus
    End If
End If
End Sub

```

---

```
Private Sub Command2_Click()
Unload Me
End Sub
```

---

```
Private Sub Form_Load()
```

```
Data1.DatabaseName = App.Path & "\btp.mdb"
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
Text7.Text = ""
Text8.Text = ""
Text9.Text = ""
Text10.Text = ""
Text11.Text = ""
```

```
End Sub
```

---

```
Private Sub Text1_Change()
```

```
squ = "SELECT * From vehrec WHERE dat LIKE '"
squ = squ & Text1 & "' order by dat ASC;"
Data1.RecordSource = squ
Data1.Refresh
```

```
End Sub
```

---

```
Private Sub Text2_Change()
```

```
squ = "SELECT * From vehrec WHERE veid LIKE '"
squ = squ & Text2 & "' order by dat ASC;"
Data1.RecordSource = squ
Data1.Refresh
```

```
End Sub
```

#### **Incoice.frm**

```
Private Sub Command1_Click()
On Error Resume Next
If Text9.Text <> "" And Text10.Text <> "" And Text11.Text <> "" Then
    Dim p, q
    Data1.Recordset![dat] = p
    p = Text9
```



```

Data1.Recordset![dat] = q
q = Text10

If p > q Or p = q Then
    MsgBox "Beginning Date can not be bigger then or equal Ending Date", vbCritical,
    "Date Error!"
    Exit Sub
Else

    If Data1.Recordset.RecordCount = 0 Then
        MsgBox ("There is not any record at database")
        Exit Sub
    Else

        Data1.RecordSource = "select * from vehrec where dat between" & Text9.Text
        & "*" and " & Text10.Text & "*" and veid like " & Text11 & ""
        Data1.Refresh

        Dim a, b, c, d, e
        a = 0
        b = 0
        Data1.Recordset.MoveFirst
        Do Until Data1.Recordset.EOF
            b = Data1.Recordset![vol]
            a = Val(a) + Val(b)
            Data1.Recordset.MoveNext
        Loop
        Text4 = a
        Data5.Recordset.MoveLast
        Text5 = Data5.Recordset![dis1]
        Text6 = Val(Text4) * Val(Text5)
        Text7 = (Val(Text6) * Val(Text2)) / 100
        c = CDbl(Text6)
        d = CDbl(Text7)
        e = c + d
        Text8 = e

    End If
End If
Else
    MsgBox "You can not leave the Beginning and Ending Date and Vehicle ID empty.",
    48, "Caution Message"
    If Text2.Text = "" Then
        Text1.SetFocus
    End If
End If
End Sub

```

---

```
Private Sub Command2_Click()
```

```
Unload Me
```

```
End Sub
```

---

```
Private Sub Form_Load()
```

```
Data1.DatabaseName = App.Path & "\btp.mdb"
```

```
Data5.DatabaseName = App.Path & "\btp.mdb"
```

```
Text1.Text = ""
```

```
Text2.Text = ""
```

```
Text3.Text = ""
```

```
Text4.Text = ""
```

```
Text5.Text = ""
```

```
Text6.Text = ""
```

```
Text7.Text = ""
```

```
Text8.Text = ""
```

```
Text9.Text = ""
```

```
Text10.Text = ""
```

```
Text11.Text = ""
```

```
End Sub
```

---

```
Private Sub Text1_Change()
```

```
squ = "SELECT * From VehRec WHERE dat LIKE "
```

```
squ = squ & Text1 & "*" order by dat ASC;"
```

```
Data1.RecordSource = squ
```

```
Data1.Refresh
```

```
End Sub
```

---

```
Private Sub Text3_Change()
```

```
squ = "SELECT * From VehRec WHERE veid LIKE "
```

```
squ = squ & Text3 & "*" order by dat ASC;"
```

```
Data1.RecordSource = squ
```

```
Data1.Refresh
```

```
End Sub
```

### ParUP.frm

```
Private Sub Command1_Click()
```

```
If Text1.Text <> "" Or Text2.Text <> "" Then  
    Data5(0).Recordset.AddNew
```

```
        Data5(0).Recordset.dat = Text1.Text  
        Data5(0).Recordset.dis1 = Text2.Text
```

```
    Data5(0).Recordset.Update  
    Data5(0).Refresh
```

```
    Text1.Text = ""  
    Text2.Text = ""  
    Text1.SetFocus
```

```
Else
```

```
    MsgBox "You can not leave the Date or Unit Price empty.", 48, "Caution Message"
```

```
    If Text2.Text = "" Then
```

```
        Text1.SetFocus
```

```
    End If
```

```
    End If
```

```
End Sub
```

---

```
Private Sub Command2_Click()
```

```
Unload Me
```

```
End Sub
```

---

```
Private Sub Form_Load()
```

```
Data5(0).DatabaseName = App.Path & "\btp.mdb"
```

```
Text1.Text = ""
```

```
Text2.Text = ""
```

```
End Sub
```

### ListUP.frm

```
Private Sub Command1_Click()
```

```
If Data5.Recordset.RecordCount = 0 Then
```

```
    Exit Sub
```

```
End If
```

```
If Text3.Text <> "" And Text4.Text <> "" Then
```

```
    Data5.Recordset.Edit
```

```
    dt = Left(Text3.Text, 2) + Mid(Text3.Text, 3, 4) + Right(Text3.Text, 4)
```

```
    Data5.Recordset.dat = dt
```

```
    Data5.Recordset.dis1 = Text4.Text
```



```

        Data5.Recordset.Update
        Data5.Refresh
    Else
        MsgBox "You can not leave the Date and Unit Price No empty.", 48, "Caution Message"
        If Text3.Text = "" Then
            Text3.SetFocus
        End If
        If Text4.Text = "" Then
            Text4.SetFocus
        End If
    End If
End Sub

```

---

```

Private Sub Command2_Click()

    If Data5.Recordset.RecordCount = 0 Then Exit Sub
    answer = MsgBox("Do you want to DELETE this record ?", vbQuestion + vbYesNo + vbDefaultButton2, "Record Deleting")
    If answer = vbYes Then Data5.Recordset.Delete
    If answer = vbNo Then Exit Sub
    Data5.Refresh

End Sub

```

---

```

Private Sub Command3_Click()

    Unload Me

End Sub

```

---

```

Private Sub DBGrid1_RowColChange(LastRow As Variant, ByVal LastCol As Integer)

    If Data5.Recordset.RecordCount = 0 Then
        Text3 = ""
        Text4 = ""
    End If
    Text3 = Data5.Recordset.dat & ""
    Text4 = Data5.Recordset.dis1 & ""

End Sub

```

---

```

Private Sub Form_Load()

    Data5.DatabaseName = App.Path & "\btp.mdb"
    squ = "SELECT * From ParUP WHERE dat LIKE ""
    squ = squ & Text1 & ""*'' order by dat ASC;"
    Data5.RecordSource = squ
    Data5.Refresh

```

End Sub

---

Private Sub Text1\_Change()

```
squ = "SELECT * From ParUP WHERE dat LIKE ""  
squ = squ & Text1 & "*" order by dat ASC;"  
Data5.RecordSource = squ  
Data5.Refresh
```

End Sub

---

Private Sub Text2\_Change()

```
squ = "SELECT * From ParUP WHERE dis1 LIKE ""  
squ = squ & Text2 & "*" order by dat ASC;"  
Data5.RecordSource = squ  
Data5.Refresh
```

End Sub

#### **ParFuel.frm**

Private Sub Command1\_Click()

If Text1.Text <> "" Or Text2.Text <> "" Then

```
Data4.Recordset.AddNew  
Data4.Recordset.dat = Text1.Text  
Data4.Recordset.ltp = Text2.Text  
Data4.Recordset.Update  
Data4.Refresh  
Text1.Text = ""  
Text2.Text = ""  
Text1.SetFocus
```

Else

MsgBox "You can not leave the Date or Litre Price empty.", 48, "Caution Message"

If Text2.Text = "" Then

Text1.SetFocus

End If

End If

End Sub

---

Private Sub Command2\_Click()

Unload Me

End Sub

---

```
Private Sub Form_Load()
```

```
Data4.DatabaseName = App.Path & "\btp.mdb"
```

```
Text1.Text = ""
```

```
Text2.Text = ""
```

```
End Sub
```

#### **ListFuel.frm**

```
Private Sub Command1_Click()
```

```
If Data4.Recordset.RecordCount = 0 Then
```

```
Exit Sub
```

```
End If
```

```
If Text3.Text <> "" And Text4.Text <> "" Then
```

```
Data4.Recordset.Edit
```

```
dt = Right(Text3.Text, 4) + Mid(Text3.Text, 3, 4) + Left(Text3.Text, 2)
```

```
Data4.Recordset.dat = dt
```

```
Data4.Recordset.ltp = Text4.Text
```

```
Data4.Recordset.Update
```

```
Data4.Refresh
```

```
Else
```

```
MsgBox "You can not leave the Date or Litre Price empty.", 48, "Caution Message"
```

```
If Text3.Text = "" Then
```

```
Text3.SetFocus
```

```
End If
```

```
If Text4.Text = "" Then
```

```
Text4.SetFocus
```

```
End If
```

```
End If
```

```
End Sub
```

---

```
Private Sub Command2_Click()
```

```
If Data4.Recordset.RecordCount = 0 Then Exit Sub
```

```
answer = MsgBox("Do you want to DELETE this record ?", vbQuestion + vbYesNo +  
vbDefaultButton2, "Record Deleting")
```

```
If answer = vbYes Then Data4.Recordset.Delete
```

```
If answer = vbNo Then Exit Sub
```

```
Data4.Refresh
```

```
End Sub
```

---



```
Private Sub Command3_Click()
```

```
Unload Me
```

```
End Sub
```

---

```
Private Sub DBGrid1_RowColChange(LastRow As Variant, ByVal LastCol As Integer)
```

```
If Data4.Recordset.RecordCount = 0 Then
```

```
Text3 = ""
```

```
Text4 = ""
```

```
End If
```

```
Text3 = Data4.Recordset.dat & ""
```

```
Text4 = Data4.Recordset.ltp & ""
```

```
End Sub
```

---

```
Private Sub Form_Load()
```

```
Data4.DatabaseName = App.Path & "\btp.mdb"
```

```
squ = "SELECT * From ParFuel WHERE dat LIKE ""
```

```
squ = squ & Text1 & "*" order by dat ASC;"
```

```
Data4.RecordSource = squ
```

```
Data4.Refresh
```

```
End Sub
```

---

```
Private Sub Text1_Change()
```

```
squ = "SELECT * From ParFuel WHERE dat LIKE ""
```

```
squ = squ & Text1 & "*" order by dat ASC;"
```

```
Data4.RecordSource = squ
```

```
Data4.Refresh
```

```
End Sub
```

---

```
Private Sub Text2_Change()
```

```
squ = "SELECT * From ParFuel WHERE veid LIKE ""
```

```
squ = squ & Text2 & "*" order by ltp ASC;"
```

```
Data4.RecordSource = squ
```

```
Data4.Refresh
```

```
End Sub
```

### ParVID.frm

```
Private Sub Command1_Click()
```

```
    If Text2.Text <> "" Then
```

```
        Data2.Recordset.AddNew
```

```
        Data2.Recordset.dat = Text1.Text
```

```
        Data2.Recordset.veid = Text2.Text
```

```
        Data2.Recordset.Update
```

```
        Text1.Text = ""
```

```
        Text2.Text = ""
```

```
        Text1.SetFocus
```

```
    Else
```

```
        MsgBox "You can not leave the Date or Vehicle ID empty.", 48, "Caution Message"
```

```
        If Text2.Text = "" Then
```

```
            Text1.SetFocus
```

```
        End If
```

```
    End If
```

```
End Sub
```

---

```
Private Sub Command2_Click()
```

```
    Unload Me
```

```
End Sub
```

---

```
Private Sub Form_Load()
```

```
    Data2.DatabaseName = App.Path & "\btp.mdb"
```

```
    Text1.Text = ""
```

```
    Text2.Text = ""
```

```
End Sub
```

### ListVID.frm

```
Private Sub Command1_Click()
```

```
    If Data2.Recordset.RecordCount = 0 Then
```

```
        Exit Sub
```

```
    End If
```

```
    If Text3.Text <> "" And Text4.Text <> "" Then
```

```
        Data2.Recordset.Edit
```

```
        dt = Right(Text3.Text, 4) + Mid(Text3.Text, 3, 4) + Left(Text3.Text, 2)
```

```

Data2.Recordset.dat = dt

Data2.Recordset.veid = Text4.Text
Data2.Recordset.Update
Data2.Refresh
Else
    MsgBox "You can not leave the Name, Company and Invoice No empty.", 48,
    "Caution Message"
    If Text3.Text = "" Then
        Text3.SetFocus
    End If
    If Text4.Text = "" Then
        Text4.SetFocus
    End If

End If
End Sub

```

---

```

Private Sub Command4_Click()

If Data2.Recordset.RecordCount = 0 Then Exit Sub
answer = MsgBox("Do you want to DELETE this record ?", vbQuestion + vbYesNo +
vbDefaultButton2, "Record Deleting")
If answer = vbYes Then Data2.Recordset.Delete
If answer = vbNo Then Exit Sub
Data2.Refresh

End Sub

```

---

```

Private Sub Command5_Click()

Unload Me

End Sub

```

---

```

Private Sub DBGrid1_RowColChange(LastRow As Variant, ByVal LastCol As Integer)

If Data2.Recordset.RecordCount = 0 Then
    Text3 = ""
    Text4 = ""
End If
Text3 = Data2.Recordset.dat & ""
Text4 = Data2.Recordset.veid & ""

End Sub

```

---

```

Private Sub Form_Load()

Data2.DatabaseName = App.Path & "\btp.mdb"
squ = "SELECT * From ParVID WHERE dat LIKE ""

```



```
squ = squ & Text1 & "*" order by dat ASC;"
Data2.RecordSource = squ
Data2.Refresh
Text1.Text = ""
Text2.Text = ""
```

End Sub

---

Private Sub Text1\_Change()

```
squ = "SELECT * From ParVID WHERE dat LIKE '"
squ = squ & Text1 & "*" order by dat ASC;"
Data2.RecordSource = squ
Data2.Refresh
```

End Sub

---

Private Sub Text2\_Change()

```
squ = "SELECT * From ParVID WHERE veid LIKE '"
squ = squ & Text2 & "*" order by veid ASC;"
Data2.RecordSource = squ
Data2.Refresh
```

End Sub

### **ParPers.frm**

Private Sub Command1\_Click()

```
On Error Resume Next
If Text1.Text <> "" Then
```

```
Data3.Recordset.AddNew
Data3.Recordset.perno = Text1.Text
Data3.Recordset.trin = Text2.Text
Data3.Recordset.idsn = Text3.Text
Data3.Recordset.nme = Text4.Text
Data3.Recordset.sna = Text5.Text
Data3.Recordset.bipl = Text6.Text
Data3.Recordset.bidy = Text7.Text
Data3.Recordset.mona = Text8.Text
Data3.Recordset.fana = Text9.Text
Data3.Recordset.adrs = Text10.Text
Data3.Recordset.cty = Text11.Text
Data3.Recordset.phn = Text12.Text
Data3.Recordset.mbl = Text13.Text
Data3.Recordset.dln = Text14.Text
Data3.Recordset.dlc = Text15.Text
```

```
Data3.Recordset.bg = Text16.Text
Data3.Recordset.sino = Text17.Text
Data3.Recordset.bdw = Text18.Text
```

```
Data3.Recordset.Update
Data3.Refresh
```

```
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
Text7.Text = ""
Text8.Text = ""
Text9.Text = ""
Text10.Text = ""
Text11.Text = ""
Text12.Text = ""
Text13.Text = ""
Text14.Text = ""
Text15.Text = ""
Text16.Text = ""
Text17.Text = ""
Text18.Text = ""
Text19.Text = ""
```

```
Text2.SetFocus
```

```
Else
```

```
MsgBox "You can not leave the Personal No or other text fields empty.", 48, "Caution  
Message"
```

```
If Text2.Text = "" Then
```

```
Text1.SetFocus
```

```
End If
```

```
End If
```

```
End Sub
```

---

```
Private Sub Command5_Click()
```

```
Unload Me
```

```
End Sub
```

---

```
Private Sub Form_Load()
```

```
Data3.DatabaseName = App.Path & "\btp.mdb"
```

```
Text1.Text = ""
```

```
Text2.Text = ""
```

```
Text3.Text = ""
```

```
Text4.Text = ""
```

```
Text5.Text = ""
```

```

Text6.Text = ""
Text7.Text = ""
Text8.Text = ""
Text9.Text = ""
Text10.Text = ""
Text11.Text = ""
Text12.Text = ""
Text13.Text = ""
Text14.Text = ""
Text15.Text = ""
Text16.Text = ""
Text17.Text = ""
Text18.Text = ""

```

End Sub

#### **ListPers.frm**

```

Private Sub Command1_Click()
If Data3.Recordset.RecordCount = 0 Then
Exit Sub
End If
If Text4.Text <> "" And Text5.Text <> "" Then

```

```

Data3.Recordset.Edit
Data3.Recordset.perno = Text3.Text
Data3.Recordset.trin = Text4.Text
Data3.Recordset.idsn = Text5.Text
Data3.Recordset.nme = Text6.Text
Data3.Recordset.sna = Text7.Text
Data3.Recordset.bipl = Text8.Text
Data3.Recordset.bidy = Text9.Text
Data3.Recordset.mona = Text10.Text
Data3.Recordset.fana = Text11.Text
Data3.Recordset.adrs = Text12.Text
Data3.Recordset.cty = Text13.Text
Data3.Recordset.phn = Text14.Text
Data3.Recordset.mbl = Text15.Text
Data3.Recordset.dln = Text16.Text
Data3.Recordset.dlc = Text17.Text
Data3.Recordset.bg = Text18.Text
Data3.Recordset.sino = Text19.Text
Data3.Recordset.bdw = Text20.Text
Data3.Recordset.Update
Data3.Refresh

```

End If



End Sub

---

Private Sub Command2\_Click()

If Data3.Recordset.RecordCount = 0 Then Exit Sub  
answer = MsgBox("Do you want to DELETE this record ?", vbQuestion + vbYesNo +  
vbDefaultButton2, "Record Deleting")  
If answer = vbYes Then Data3.Recordset.Delete  
If answer = vbNo Then Exit Sub  
Data1.Refresh

End Sub

---

Private Sub Command3\_Click()

Unload Me

End Sub

---

Private Sub Form\_Load()

Data3.DatabaseName = App.Path & "\btp.mdb"

Text3.Text = ""  
Text4.Text = ""  
Text5.Text = ""  
Text6.Text = ""  
Text7.Text = ""  
Text8.Text = ""  
Text9.Text = ""  
Text10.Text = ""  
Text11.Text = ""  
Text12.Text = ""  
Text13.Text = ""  
Text14.Text = ""  
Text15.Text = ""  
Text16.Text = ""  
Text17.Text = ""  
Text18.Text = ""  
Text19.Text = ""  
Text20.Text = ""

End Sub

---

```
Private Sub Text1_Change()
```

```
squ = "SELECT * From parpers WHERE perno LIKE ""  
squ = squ & Text1 & "*" order by dat ASC;"  
Data3.RecordSource = squ  
Data3.Refresh
```

```
End Sub
```

---

```
Private Sub Text2_Change()
```

```
squ = "SELECT * From parpers WHERE nme LIKE ""  
squ = squ & Text2 & "*" order by dat ASC;"  
Data3.RecordSource = squ  
Data3.Refresh
```

```
End Sub
```

```
Private Sub DBGrid1_RowColChange(LastRow As Variant, ByVal LastCol As Integer)
```

```
If Data3.Recordset.RecordCount = 0 Then
```

```
Text3 = ""
```

```
Text4 = ""
```

```
Text5 = ""
```

```
Text6 = ""
```

```
Text7 = ""
```

```
Text8 = ""
```

```
Text9 = ""
```

```
Text10 = ""
```

```
Text11 = ""
```

```
Text12 = ""
```

```
Text13 = ""
```

```
Text14 = ""
```

```
Text15 = ""
```

```
Text16 = ""
```

```
Text17 = ""
```

```
Text18 = ""
```

```
Text19 = ""
```

```
Text20 = ""
```

```
Exit Sub
```

```
Else
```

```
Text3.Text = Data3.Recordset.perno & ""
```

```
Text4.Text = Data3.Recordset.trin & ""
```

```
Text5.Text = Data3.Recordset.idsn & ""
```

```
Text6.Text = Data3.Recordset.nme & ""
```

```
Text7.Text = Data3.Recordset.sna & ""
```

```
Text8.Text = Data3.Recordset.bipl & ""
```

```
Text9.Text = Data3.Recordset.bidy & ""
```

```

Text10.Text = Data3.Recordset.mona & ""
Text11.Text = Data3.Recordset.fana & ""
Text12.Text = Data3.Recordset.adrs & ""
Text13.Text = Data3.Recordset.cty & ""
Text14.Text = Data3.Recordset.phn & ""
Text15.Text = Data3.Recordset.mbl & ""
Text16.Text = Data3.Recordset.dln & ""
Text17.Text = Data3.Recordset.dlc & ""
Text18.Text = Data3.Recordset.bg & ""
Text19.Text = Data3.Recordset.sino & ""
Text20.Text = Data3.Recordset.bdw & "
End If
End Sub

```

#### **ParUser.frm**

```

Private Sub Command1_Click()

If Text1.Text <> "" And Text2.Text <> "" And Text3.Text <> "" Then

If Text3.Text = Text2.Text Then
    Data7.Recordset.AddNew
    Data7.Recordset.UserName = Text1.Text
    Data7.Recordset.Password = Text2.Text
    Data7.Recordset.Update
    Data7.Refresh
    Text1.Text = ""
    Text2.Text = ""
    Text3.Text = ""
    Text1.SetFocus
Else
    MsgBox "Your Password is Wrong, Please Check Your Password", 48, "Caution
Message"
    If Text2.Text = "" Then
        Text1.SetFocus
    End If
End If
Else
    MsgBox "You can not leave the Username or Password text fields empty.", 48,
"Caution Message"
    If Text2.Text = "" Then
        Text1.SetFocus
    End If
End If

End Sub

```

---

```

Private Sub Command2_Click()

Unload Me

```



End Sub

---

Private Sub Form\_Load()

Data7.DatabaseName = App.Path & "\btp.mdb"

Text1.Text = ""

Text2.Text = ""

Text3.Text = ""

End Sub

#### **ListUser.frm**

Private Sub Command1\_Click()

Unload Me

End Sub

---

Private Sub Command2\_Click()

If Data7.Recordset.RecordCount = 0 Then

Exit Sub

End If

If Text4.Text <> "" And Text5.Text <> "" Then

Data7.Recordset.Edit

Data7.Recordset.UserName = Text4.Text

Data7.Recordset.Password = Text5.Text

Data7.Recordset.Update

Data7.Refresh

Else

MsgBox "You can not leave the Username and Password No empty.", 48, "Caution Message"

If Text4.Text = "" Then

Text4.SetFocus

End If

If Text5.Text = "" Then

Text5.SetFocus

End If

End If

End Sub

---

Private Sub Command3\_Click()

If Data7.Recordset.RecordCount = 0 Then Exit Sub

answer = MsgBox("Do you want to DELETE this record ?", vbQuestion + vbYesNo + vbDefaultButton2, "Record Deleting")

If answer = vbYes Then Data7.Recordset.Delete

If answer = vbNo Then Exit Sub

Data7.Refresh

End Sub

---

Private Sub DBGrid1\_RowColChange(LastRow As Variant, ByVal LastCol As Integer)

If Data7.Recordset.RecordCount = 0 Then

Text4 = ""

Text5 = ""

End If

Text4 = Data7.Recordset.UserName & ""

Text5 = Data7.Recordset.Password & ""

End Sub

---

Private Sub Form\_Load()

Data7.DatabaseName = App.Path & "\btp.mdb"

End Sub

---

Private Sub Text1\_Change()

squ = "SELECT \* From VehRec WHERE dat LIKE "

squ = squ & Text1 & "\*" order by dat ASC;"

Data7.RecordSource = squ

Data7.Refresh

End Sub

### **About.frm**

Private Sub Command1\_Click()

Unload Me

End Sub