

NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

**ACCOUNT FEE STUDENT
SYSTEM**

**Graduation Project
COM – 400**

Student: Omar Yasin (20021859)

Supervisor: Assoc. Prof. Dr. Adil Amirjanov

Nicosia - 2007

ACKNOWLEDGEMENT

First of all I would like to thanks Allah {God} for guiding me through my study.

More over I want to pay special regards to my parents who are enduring these all expenses and supporting me in all events. I am nothing without their prayers. They also encouraged me in crises. I shall never forget their sacrifices for my education so that I can enjoy my successful life as they are expecting. They may get peaceful life in Heaven.

Also, I feel proud to pay my special regards to my project adviser "Assoc. Prof. Dr. Adil". He never disappointed me in any affair. He delivered me too much information and did his best of efforts to make me able to complete my project. He has Devine place in my heart and I am less than the half without his help. I am really thankful to my teacher.

The best of acknowledge, I want to honor those all persons who have supported me or helped me in my project. I also pay my special thanks to my all friends who have helped me in my project and gave me their precious time to complete my project. Also my especial thanks go to my friends, Fadi ageal, yosaf Al-arouri, Mosa Yasin, Rami Abdalah, Ibrahim Ismail, and Adana Abu Yosaf.

At the end I am again thankful to those all persons who helped me or even encouraged me to complete me, my project. My all efforts to complete this project might be fruitfully.

ABSTRACT

This project showed the useful uses in Database application in account fee student system, and for use.

In the project can add user will use this project and give user name and password for the user, in password have a particular authority in using this program by defining his actions and that related to his name and password, and also can delete the user and edit the information for the user if have any mastic in the information.

In the project can add student, and update the student only the money and also you can edit the information for student and you can see if he has debt or credit.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	i
ABSTRACT	ii
CONTENTS	iii
INTRODUCTION	1
1. Visual Basic	2
1.1 Overview	2
1.2 Creating A Project In Visual Basic	3
1.2.1 Designing The Tic-Tac-Toe Program	3
1.2.2 The Parts Of A Visual Basic Project	4
1.3 Coding In Visual Basic	7
1.3.1 PROGRAM DESIGN LANGUAGE	8
1.4 Coding To Get The Most From Visual Basic	9
2. Data Base	13
2.1 Overview	13
2.2 Relational Database	13
2.3 Changing Data Into Information	14
2.4 Access Database	14
2.4.1 Maintaining Access Databases	14
2.4.1.1 Repairing In Place	15
2.4.2 The Access User Interface	16
2.4.2.1 Navigating The Database View Window	17
2.5 SQL Database	19

3. Open Database Connection	21
3.1 Overview	21
3.2 Opening Database	21
3.3 Adding A Record To A Record Set	24
3.4 Editing A Record In A Record Set	25
3.5 Updating A Record In A Record Set	25
3.6 Moving To The First Record In A Record Set	26
3.7 Moving To The Last Record In A Record Set	27
3.8 Deleting A Record In A Record Set	28
3.9 Searching A Record Set	28
 4.Registration Student	 30
4.1 Over View	30
4.2 Security	31
4.3 Users	31
4.3.1 Add New Users	31
4.3.2 Edit Users	32
4.3.3 Delete Users	33
4.3.4 Viewing All Users	34
4.4 Student	35
4.4.1 Add New Student	35
4.4.2 Update Old Student	36
4.4.3 Edit Student Information	38
4.4.4 Delete Student	38
4.4.5 Debt And Payment For Student	39

CONCLUSION	40
REFERENCES	41
APPENDIX	42

INTRODUCTION

This project is talking about the registering of students in university by using the visual basic programming language and data base.

In my chapters I tried to describe how the visual basic working and how I used it in my program, how I connected it with the data base and how can the user use this program.

This project includes 4 chapters:

The first chapter describes the visual basic programming language with its coding and variable scope (including object variables) and procedure scope, how its working, how to create and design projects in visual basic.

The second chapter talks about the data base how to make it, active it and how to insert, search, delete and edit the information in it.

Chapter three describes the connection between data base and visual basic and how they are working together in the same program.

Chapter four represents my program in a diagrams how to use it and how to insert a new student, register old students and delete graduated students

CHAPTER 1

VISUAL BASIC

1.1 OVERVIEW

It's no secret that Visual Basic is the favorite programming environment of many programmers. When Visual Basic first appeared, it created a revolution in Windows programming, and that revolution continues to this day. Never before had Windows programming been so easy just build the program you want, right before your eyes, and then run it. Visual Basic introduced unheard-of ease to Windows programming and changed programming from a chore to something very fun.

We'll start with an overview of Visual Basic, taking a look at topics common to the material in the rest of the text. In this chapter, we'll create the foundation we'll rely on later as we take a look at the basics of Visual Basic, including how to create Visual Basic projects and seeing what is in such projects. We'll also get an overview of essential Visual Basic concepts like forms, controls, events, properties, methods, and so on. And we'll examine the structure of a Visual Basic program, taking a look at variables, variable scope, and modules. In other words, we're going to lay bare the anatomy of a Visual Basic program here.

Most Visual Basic programmers do not have formal programming training and have to learn a lot of this material the hard way. As programming has matured, programmers have learned more and more about what are called best practices the programming techniques that make robust, easily debugged programs. We'll take a look at those practices in this chapter, because they are becoming more and more essential for programmers in commercial environments these days, especially those programmers that work in teams. And we'll look at those practices from the viewpoint of programmers who program for a living; frequently there's a gap between the way best practices are taught by academics and how they are actually needed by programmers facing the prospect of writing a 20,000-line program as part of a team of programmers.

1.2 CREATING A PROJECT IN VISUAL BASIC

There are three different editions of Visual Basic:

- “The Learning Edition, which is the most basic edition. This edition allows you to write many different types of programs, but lacks a number of tools that the other editions have.
- “The Professional Edition, designed for professionals. This edition contains all that the Learning Edition contains and more, such as the capability to write ActiveX controls and documents.
- “The Enterprise Edition, which is the most complete Visual Basic edition. This edition is targeted towards professional programmers who may work in a team and includes additional tools such as Visual SourceSafe, a version-control system that coordinates team programming.

1.2.1 DESIGNING THE TIC-TAC-TOE PROGRAM

Using the Command Button tool in the Visual Basic toolbox, add a new command button to the main form in our program now, in the Properties window, change the Name property of this button from Command1 to Command in preparation for setting up a control array, and clear its Caption property so the button appears blank.

Next, add a second button to the form, and set its Name property to Command as well. When you do, Visual Basic opens a dialog box that states: `_You already have a control named _Command_. Do you want to set up a control array?_` Click Yes to create a control array, which means we will be able to refer to our controls using an index instead of simply by name.

Add a total of nine buttons to the main form in our program, arranged in a 3×3 grid similar to a standard tic-tac-toe game, give each of the buttons the name Command, and clear their captions. That completes the preliminary design now we are ready to write some code.

1.2.2 THE PARTS OF A VISUAL BASIC PROJECT

Projects can become quite advanced in Visual Basic, even containing subprojects of different types. From a programming point of view, however, standard Visual Basic projects usually contain just three types of items: global items, forms, and modules.

- **Forms**

Forms are familiar to all Visual Basic programmers, of course they are the templates you base windows on. Besides standard forms, Visual Basic also supports Multiple Document Interface (MDI) forms, as well as a whole number of predefined forms.

- **Modules**

Modules are collections of code and data that function something like objects in object-oriented programming (OOP), but without defining OOP characteristics like inheritance, polymorphism, and so on. The point behind modules is to enclose procedures and data in a way that hides them from the rest of the program.

We'll discuss the importance of doing this later in this chapter when we cover Visual Basic programming techniques and style; breaking a large program into smaller, self-contained modules can be invaluable for creating and maintaining code.

You can think of well-designed modules conceptually as programming objects; for example, you might have a module that handles screen display that includes a dozen internal (unseen by the rest of the program) procedures and one or two procedures accessible to the rest of the program. In this way, the rest of the program only has to deal with one or two procedures, not a dozen.

- **Global Items**

Global items are accessible to all modules and forms in a project, and you declare them with the Public keyword. However, Microsoft recommends that you keep the

number of global items to an absolute minimum and, in fact, suggests their use only when you need to communicate between forms.

One reason to avoid global variables is their accessibility from anywhere in the program; while you are working with a global variable in one part of a program, another part of the program might be busy changing that variable, giving you unpredictable results.

- **Project Scope**

An objects scope indicates how much visibility it has throughout the project in the procedure where it's declared, throughout a form or module, or global scope (which means it's accessible everywhere). There are two types of scope in Visual Basic projects:

- Variable scope (including object variables) and
- Procedure scope.

We'll take a look at both of them here as we continue our overview of Visual Basic projects and how the parts of those projects interact.

- **Variable Scope**

You declare variables in a number of ways. Most often, you use the Dim statement to declare a variable. If you do not specify the variable type when you use Dim, it creates a variant, which can operate as any variable type. You can specify the variable type using the as keyword like this:

Dim IntegerValue as Integer

Besides Dim, you can also use ReDim to redimension space for dynamic arrays, Private to restrict it to a module or form, Public to make it global that is, accessible to all modules or forms or Static to make sure its value does not change between procedure calls.

There are three levels of variable scope in Visual Basic: at the procedure level, at the form or module level, and at the global level schematically.

When you are designing your program, Microsoft suggests you limit your variables to the minimum possible scope in order to make things simpler and to avoid conflicts. Next, we'll take a look at the other type of scope: procedure scope.

- **Procedure Scope**

As with variables, you can restrict the scope of procedures, and you do that with the Private, Public, Friend, and Static keywords. The Private and Public keywords are the main keywords here; using them, you can specify if a subroutine or function is private to the module or form in which it is declared or public (that is, global) to all forms and modules. You use these keywords before the Sub or Function keywords like this:

```
Private Function Returns7 ()
```

```
Dim Retval
```

```
Retval = 7
```

```
Returns7 = Retval
```

```
End Function
```

You can also declare procedures as friend procedures with the Friend keyword.

Friend procedures are usually used in class modules (they are not available in standard modules, although you can declare them in forms) to declare that the procedure is available outside the class, but not outside the current project.

This restricts those functions from being called if the current project serves as an OLE automation server, for example.

Besides the earlier declarations, you can also declare procedures as Static, which means that the variables in the procedure do not change between procedure calls, and

that can be very useful in cases like this, where we support a counter variable that is incremented each time a function is called:

Static Function Counter ()

Dim CounterValue as Integer

CounterValue = CounterValue + 1

Counter = CounterValue

End Sub

1.3 CODING IN VISUAL BASIC

The full construction of a commercial program is usually a project that involves many clear and definite steps. There have been whole volumes written on this topic, which are usually only interesting if you are a software project manager (or write computer books and have to know the details so you can write about them!). Such books get pretty involved, encompassing ideas like module coupling and cohesion, bottom-up composition, incremental integration, and much more.

On the whole, however, one can break the software design process into steps like these (note that the explanation of each step is very flexible; there is no one-size-fits-all here):

- “Requirements analysis Identify the problem for the software to tackle.
- “Creating specifications Determine what exactly the software should do.
- “Overall design Break the overall project into parts, modules, and so on.
- “Detailed design the actual data structures, procedures, and so on.
- “Coding Go from PDL to code.
- “Debugging Solve design-time, compilation, and obvious errors.
- “Testing Try to break the software.
- “Maintenance React to user feedback and keep testing.

Each of these steps may have many subparts, of course. (For example, the maintenance part may take up as much time as the rest of the project taken together.)

As the design process continues, a model of what the program does evolves. You use this model to get a conceptual handle on the software (while keeping in mind that models are usually flawed at some level).

Keeping the model in mind, then, many programmers use a program design language to start the actual coding process.

1.3.1 PROGRAM DESIGN LANGUAGE

Everyone seems to think that programmers use flowcharts, but the reality is usually different (flowcharts are nice to show to nonprogrammers, though). One tool that commercial programmers do find useful is program design language (PDL). Although there are formal specifications for PDL, many programmers simply regard this step as writing out what a program does in English as a sort of pseudo-code.

For example, if we want to create a new function named `dblSqrt()` that returns a number's square root, we might write its PDL this way in English, where we break what the function does into steps:

Function dblSqrt ()

Check if the input parameter is negative

If the input parameter is negative, return -1

If the input parameter is positive, return its square root

End Function

When you actually write the code, the PDL can often become the comments in that code; for example, here's the completed function:

```
*****
```

```
' dblSqrt()
```

```
'Purpose: Returns the passed parameter's square root
```

```
'Inputs: dblParameter, the parameter whose square root we need
```

```
' Returns: The input value's square root
```

Function dblSqrt(dblParameter As Double) As Double

'Check if the input parameter is negative

If dblParameter < 0 Then

'If the input parameter is negative, return -1

dblSqrt = -1

Else

'If the input parameter is positive, return its square root

dblSqrt = Sqr(dblParameter)

End If

End Function

In this way, developing your program using PDL, where every line of PDL has one (and only one) specific task, can be very useful.

1.4 CODING TO GET THE MOST FROM VISUAL BASIC

In this section, we'll discuss some best practices coding for Visual Basic. All of these practices come from professional programmers, but of course whether you implement them or not is up to you. Here we go:

“Avoid `_magic numbers_` when you can. A magic number is a number (excluding 0 or 1) that `_s` hardwired right into your code like this:

Function blnCheckSize(dblParameter As Double) As Boolean

If dblParameter > 1024 Then

blnCheckSize = True

Else

blnCheckSize = False

End If

End Function

Here, 1024 is a magic number. It's better to declare such numbers as constants, especially if you have a number of them. When it's time to change your code, you just have to change the constant declaration in one place, not try to find all the magic numbers scattered around your code.

"Be modular. Putting code and data together into modules hides it from the rest of the program, makes it easier to debug, makes it easier to work with conceptually, and even makes load-time of procedures in the same module quicker. Being modular also called information-hiding (and encapsulation in true OOP) is the backbone of working with larger programs. Divide and conquer is the idea here.

"Program defensively. An example of programming defensively would be to check data passed to you in a procedure before using it. This can save a bug from propagating throughout your program and help pinpoint its source. Make no assumptions.

"Visual Basic procedures should have only one purpose, ideally. This is also an aid in larger programs when things start to get complex. Certainly if a procedure has two distinct tasks, consider breaking it up.

"Avoid deep nesting of conditionals or loops. Debugging deeply nested conditionals visually is very, very inefficient. If you need to, place some of the inner loops or conditionals in new procedures and call them. Three levels of nesting should be about the maximum.

"Use access procedures to protect sensitive data. (This is part of programming defensively.) Access procedures are also called Get/Set procedures, and they are called by the rest of the program when you want to work with sensitive data. If the rest of the program must call a Set() procedure to set that data, you can test to make sure that the new value is acceptable, providing a screen between that data and the rest of the program.

“Ideally, variables should always be defined with the smallest scope possible. Global variables can create enormously complex conditions. (In fact, Microsoft recommends that global variables should be used only when there is no other convenient way to share data between forms.)

“Do not pass global variables to procedures. If you pass global variables to procedures, the procedure you pass that variable to might give it one name (as a passed parameter) and also reference it as a global variable. This can lead to some serious bugs, because now the procedure has two different names for the variable.

“Use the operator when linking strings and the + operator when working with numerical values. This is per Microsoft’s recommendations.

“When you create a long string, use the underscore line-continuation character to create multiple lines of code. This is so you can read or debug the string easily. For example:

```
Dim Msg As String
```

```
Msg = "Well, there is a problem " _
```

```
&"with your program. I am not sure " _
```

```
&"what the problem is, but there is " _
```

```
&"definitely something wrong."
```

“Avoid using variants if you can. Although convenient, they waste not only memory but time. You may be surprised by this. Remember, however, that Visual Basic has to convert the data in a variant to the proper type when it learns what is required, and that conversion actually takes a great deal of time.

“Indent your code with four spaces per Microsoft is recommendations. Believe it or not, there have been serious studies undertaken here, and 2 to 4 spaces were found to be best. Be consistent.

“Finally, watch out for one big Visual Basic pitfall: misspelled variables. Because you do not have to declare a variable in Visual Basic to use it, you might end up

surprised when Visual Basic creates a new variable after you have misspelled a variable's name. For example, here is some perfectly legal code modified from our tic-tac-toe project that compiles and runs, but because of a misspelling `xNoww` for `xNow` it does not work at all:

```
Private Sub Command_Click(Index As Integer)
    If xNow Then
        Command(Index).Caption = "x"
    Else
        Command(Index).Caption = "o"
    End If
    xNoww = Not xNow
End Sub
```

Because Visual Basic treats `xNoww` as a legal variable, this kind of bug is very hard to find when debugging.

TIP: Because Visual Basic auto-declares variables, it's usually better to use variable names that say something (like `intCurrentIndex`) instead of ones that do not (like `intDD35A`) to avoid declaring a variable through misspelling its name. A better idea is to use `Option Explicit` to make sure all variables must be explicitly declared.

If you work in teams, use version control. There are several well-known utilities that help programmers work in teams, such as Microsoft's Visual SourceSafe. This utility, which is designed to work with programming environments like Visual Basic, restricts access to code so that two programmers do not end up modifying independent copies of the same file.

CHAPTER 2

DATA BASE

2.1 OVERVIEW

The purpose of a Database system such as Microsoft Access is to change data into information. Many people use those two terms interchangeably, but there is a world of difference between the two if you consider information as being the same as knowledge. Data is a collection of facts. Information is that data organized or presented in such a way as to be useful for decision making.

This shows actual voter registration data for a particular county shown in Access. It includes voters' names, addresses, registration information such as political party, and also the voting records for each person registered. It doesn't, of course, show who voters voted for (that's unavailable as data), but it does show whether and how the voters voted for each election cycle. A voter can vote by mail-in ballot, early voting, or at the polls.

2.2 RELATIONAL DATABASE

A relational Database, simply defined, is a Database that is made up of tables and columns that relate to one another. These relationships are based on a key value that is contained in a column. For example, you could have a table called Orders that contains all the information that is required to process an order, such as the order number, date the item was ordered, and the date the item was shipped. You could also have a table called Customers that contains all the data that pertains to customers, such as a name and address. These two tables could be related to each other.

The relational Database model was developed by E.F. Codd back in the early 1970s. He proposed that a Database should consist of data stored in columns and tables that could be related to each other. This kind of thinking was very different from the hierarchical file system that was used at the time. His thinking truly revolutionized the way Databases are created and used.

A relational Database is very intuitive. It mimics the way people think. People tend to group similar objects together and break down complex objects into simpler ones. Relational Databases are true to this nature. Because they mimic the way you think, they are easy to use and learn. In later days, you will discover how easy a relational Database is to design and learn.

Most modern Databases use a relational model to accomplish their tasks. SQL is no different. It truly conforms to the relational model. This further adds to the ease of use of SQL.

2.3 CHANGING DATA INTO INFORMATION

Now let's take that data and organize it into information. Suppose that in the 1998 congressional race the Democratic candidate lost by 3,216 votes. Using the Count() function built into Access, the Database user notes that Republican voters mailed in 5,423 more ballots than the Democratic voters. This is information. Using it, the Democrats can see that if their candidate had emphasized mail-in balloting more (perhaps by mailing out applications for such ballots), he might have won.

2.4 ACCESS DATABASE

2.4.1 MAINTAINING ACCESS DATABASES

You've probably heard it before—it's not *if* you'll lose data, but *when*.

Computers aren't infallible, power fails, and parts—especially disk drives—go bad. The only defense is to back up your data.

A full discussion of backup devices is beyond the intended scope of this book.

The two fundamental methods are some sort of network backup—either through your LAN or by subscription over the Internet—and backup to removable media such as tape, removable hard disk, or writable CDs. Pick a method and use it.

2.4.1.1 REPAIRING IN PLACE

Access files are somewhat more susceptible to corruption due to power failure than other non-Database programs, such as Word. Although you should be always able to restore your files from a backup, sometimes that isn't possible or desirable. For example, you might have done quite a bit of data entry after your last backup. In such a case, you would prefer not to restore from backup, which would require repeated data entry.

Access 2000 does have a facility that will repair many instances of damage to a Database. You can find this in the main Access menu under Tools, Database Utilities. Compact and repair are now one operation.

If you have a Database open and choose Tools, Database Utilities, Compact and Repair Access will operate on the open Database. If you have no Database open it will prompt you for a file to work on. This latter method is how you'd attempt a fix on a file you can't open.

Although Access will dynamically expand to accommodate additional Database objects and data, it won't dynamically compress itself as objects or data are removed. You must do this either programmatically or manually.

The repair and compact operation are as foolproof as Microsoft can make them, but nothing is truly 100% reliable. Unless you like playing with fire, back up your old Databases before doing the repair and compaction routine. If you do a compaction/repair on a non-open Database, Access will prompt you for a name for the output. This is a form of backup, but because it's not to an external source such as a CDR disk, it's not as foolproof as a real backup.

Of course, you must approach even such information using your own common sense or specific knowledge of the task at hand. Knowing that Republicans mailed in more ballots than Democrats did implies more mail-in votes for the Republican candidate, but that's inferred information. In reality, nobody can know whether this is the case, because all you really know is that those mail-in ballots came from

registered Republicans. The votes themselves might have been for the Democratic or Green candidate.

A world of information is possible from any proper dataset. Suppose the Democratic candidate won by those 3,216 votes, but you see that Republicans outvoted (in all modes) Democrats by 7,987 voters. This is rather irrefutable information that the Democrats fielded a candidate attractive to Republicans, or that the Republican candidate wasn't what Republicans wanted in a congressman, or both.

2.4.2 THE ACCESS USER INTERFACE

Access 2000 has a new user interface designed to be not only easier for the beginner to navigate through, but also to make the life of the Access expert simpler. The concept of the user interface stems from two metaphors, the bar and tab interface common to all Microsoft Office applications, plus the object collection concept from object-oriented programming. The original object metaphor is purely abstract, whereas the translation into Access user interface is concrete.

Like so many concepts in small computers, gaining familiarity with Access' interface is best done by a hands-on approach, so let's get started. Launch Access by choosing it from the Start, Programs menu entry. In some administrative (network) installs, Access will be part of a group under Programs, in which case you'll need to locate where Access is to launch it. For most people, Access will be an entry directly under Start, Programs. Upon launching, Access will offer you several choices.

If this is the first time you've launched Access, you won't have any entries in the list box at the bottom of the dialog box. From top to bottom the three post-launch options are

- Create a New Database Using a Blank Access Database—This will create a new container (explained in the following text), ready for you to populate with your Database objects.
- Create a New Database Using Access Database Wizards, Pages, and Projects—this will also create a new Database, but by use of a wizard or two to give you a quick start.

- Open an Existing File—this will allow you to choose from a list or browse for an existing Database to open.

In addition, you can click the Cancel button to open Access with no Database loaded. Because setup will register Microsoft Access 2000 with your operating system, you can also launch Access with a Database loaded by double-clicking on the Database (files with .mdb or .mde extensions) from the Explorer.

For this, a first tour of Access, locate the North wind sample Database supplied with Office. Highlight it, and then click Open or double-click on its entry in the list box. If you need to browse for it, highlight More Files and click OK.

That action will open up a standard File Open browsing dialog box.

2.4.2.1 NAVIGATING THE DATABASE VIEW WINDOW

After you've opened North wind. This is called the Database view and you'll become very familiar with its functions and features as you work with Access. This window contains all the objects in your Database and toolbars for manipulation of these objects, and provides starting points for working with a Database.

The new window on your screen is also an object in the Microsoft hierarchy of objects, a fact that you'll want to remember when you start working with objects in VBA or macro code. For now, though, we'll refer to this window as the Database View window for the object Northwind: Database, which appears in the title bar. This window is divided into three main parts:

- The toolbar with actions and view selections;
- The left pane, which lists the types, or *classes*, of available objects within all Access Databases, such as tables and forms;
- The right pane, which shows a listing of the individual objects within the selected class on the left pane.

One new feature here is the Group class option on the left pane, which you'll learn more about in the "Groups" section, later in this lesson. As you can see by clicking

through the various objects in the left pane, the North wind Database has several objects as a part of its application.

The new object, the one that says North wind: Database in its title bar, is the Database view or Database container, as it's sometimes called because it contains various objects that make up a Database system. It displays all the items or objects within your project collected by classification. The series of buttons on the left side of the container allows you to choose from different types of objects, such as tables or reports. Click on the Forms entry (or any entry other than the one currently selected) and the right pane will reflect all the Database objects so as to show the one class selected within the Database.

You'll see a set of icons telling you what actions on the toolbar you can perform on the objects listed in the Database view panes. The first eight entries, from left to right, allow you to do the following:

- Open—Launch an object in its native mode, such as for data entry. Access uses the term *view* for different object modes.
- Design—Launch an object in such a way as to allow you to edit its structure rather than its data.
- New—Create a new object of the type highlighted within the Object list.
- Delete—Delete the highlighted object. This functions only for objects created by a user or developer, not for the objects that appear in a new Database.
- Large icons—Display the Database objects in large icon view. Analogous to the same view in Windows Explorer or My Computer.
- Small icons—Display the Database objects in small icon view. Analogous to the same view in Windows Explorer or My Computer.
- List—Display the Database objects in list view. Analogous to the same view in Windows Explorer or My Computer.
- Details—Display the Database objects in Details view. Analogous to the same view in Windows Explorer or My Computer.

Right-clicking is alive and well in Microsoft Access 2000. Right-click on any true object (as opposed to an action within the Database view) and you'll see a context (or shortcut) menu containing all the actions within the Database view as well as a few more. True to its name, the context menu for each class of objects will vary.

2.5 SQL DATABASE

SQL is a full-featured relational Database management system. It is very stable and has proven itself over time. SQL has been in production for over 10 years.

SQL is a multithreaded server. *Multithreaded* means that every time someone establishes a connection with the server, the server program creates a thread or process to handle that client's requests. This makes for an extremely fast server. In effect, every client who connects to a SQL server gets his or her own thread.

SQL is also fully ANSI SQL92-compliant. It adheres to all the standards set forth by the American National Standards Institute. The developers at TcX take these standards seriously and have carefully adhered to them.

Note ANSI SQL92 is a set of standards for the Structured Query Language that was agreed on in 1992 by the American National Standards Institute.

Another valuable feature of SQL is its online help system. All commands for SQL are given at a command prompt. To see which arguments the commands take or what the utility or command does, all you have to do is type the command and include the -help or -? Switch. This will display a slew of information about the command.

Yet another feature of SQL is its portability it has been ported to almost every platform. This means that you don't have to change your main platform to take advantage of SQL. And if you do want to switch, there is probably a SQL port for your new platform.

SQL also has many different application programming interfaces (APIs). They include APIs for Perl, TCL, Python, C/C++, Java (JDBC), and ODBC. So no matter what your company's expertise is, SQL has a way for you to access it.

SQL is also very cheap. For an unlicensed, full version of SQL, the cost is nothing. To license your copy will currently cost you \$200. This is an incredible deal, considering what you are getting for your money. Database systems that provide half the features that SQL has can cost tens of thousands of dollars. SQL can do what they do better and for less.

CHAPTER 3

OPEN DATABASE CONNECTION

3.1 OVERVIEW

When Visual Basic first started working with Databases, it used the Microsoft Jet Database engine, which is what Microsoft Access uses. Using the Jet engine represented a considerable advance for Visual Basic, because now you could work with all kinds of data formats in the fields of a Database: text, numbers, integers, longs, singles, doubles, dates, binary values, OLE objects, currency values, Boolean values, and even memo objects (up to 1.2GB of text). The Jet engine also supports SQL, which Database programmers found attractive.

To support the Jet Database engine, Microsoft added the data control to Visual Basic, and you can use that control to open Jet Database (.mdb) files. Microsoft also added a set of Data Access Objects (DAO) to Visual Basic:

- DB Engine the Jet Database engine.
- Workspace an area can hold one or more Databases.
- Database a collection of tables.
- Table Def the definition of a table.
- Query Def the definition of a query.
- Record set the set of records that make up the result of a query.
- Field a column in a table.
- Index an ordered list of records.
- Relation stored information about the specific relationship between tables.

3.2 OPENING DATABASE

To open an existing DAO Database, you use the DAO `OpenDatabase` method, passing it the name of the Database to open, and these arguments:

```
Set Database = workspace.OpenDatabase (dbname, [options [, read-only _[,  
connect]]])
```

Here are the arguments for `OpenDatabase`:

- `Dbname` The name of an existing Database file, or the data source name (DSN) of an ODBC data source.
- `Options` Setting options to `True` opens the DAO Database in exclusive mode; setting it to `False` (the default) opens the Database in shared mode.
- `Read-only`—`True` if you want to open the Database with read-only access, or `False` (the default) if you want to open the Database with read/write access.
- `Connect`—Optional. A Variant (String subtype) that specifies various connection information, including passwords.

Let's see an example to make this clearer. In our DAO code example, the `daocode` project (see the first topic in this chapter), the user can click the Open Database menu item to open a Database. In the program, we get the name of the Database the user wants to open with a Common Dialog control, and open the Database like this:

```
Private Sub OpenDatabase_Click()
```

```
CommonDialog1.ShowO
```

```
If CommonDialog1.FileName <> "" Then
```

```
Set db = _
```

```
DBEngine.Workspaces(0).OpenDatabase(CommonDialog1.FileName)
```

Next, if you know the name of the table you want to open in the Database, you can open that table by name immediately with the `OpenRecordset` method. However, because we let the user set the name of tables in the Databases we create in the `daocode` project, we don't know the names of the tables in the Database we've opened. Instead, we'll open the first user-defined table in this Database.

When you open a DAO Database, there are a number of system tables already in it, so to open the first user-defined table, we find the index of that table in the `TableDefs` collection by first skipping the system tables (which have the `dbSystemObject` flag set in their `Attributes` properties):

```
Private Sub OpenDatabase_Click()
```

```
    Dim tableIndex As Integer
```

```
    CommonDialog1.ShowOpen
```

```
    If CommonDialog1.FileName <> "" Then
```

```
        Set db = _
```

```
        DBEngine.Workspaces(0).OpenDatabase(CommonDialog1.FileName)
```

```
        tableIndex = 0
```

```
        While (db.TableDefs(tableIndex).Attributes And dbSystemObject)
```

```
            tableIndex = tableIndex + 1
```

```
        Wend
```

We'll open the first table after the system tables. We open a new record set for that table with the OpenRecordset method and fill the text boxes Text1 and Text2 in the program's main window with the fields of the first record in that table (note that in this example program, we are assuming the table we're opening has at least one record):

```
Private Sub OpenDatabase_Click()
```

```
    Dim tableIndex As Integer
```

```
    CommonDialog1.ShowOpen
```

```
    If CommonDialog1.FileName <> "" Then
```

```
        Set db = _
```

```
        DBEngine.Workspaces(0).OpenDatabase(CommonDialog1.FileName)
```

```
        tableIndex = 0
```

```
        While (db.TableDefs(tableIndex).Attributes And dbSystemObject)
```

```
            tableIndex = tableIndex + 1
```

```
        Wend
```

```

Set dbrecordset = db.OpenRecordset_
(db.TableDefs(table1index).Name, dbOpenTable)

Set td = db.TableDefs(table1index)

Text1.Text = dbrecordset.fields(0)

Text2.Text = dbrecordset.fields(1)

End If

End Sub

```

And that's it now we've opened a Database file.

3.3 ADDING A RECORD TO A RECORD SET

To add a new record to a DAO record set, you use the AddNew method (this method takes no parameters). After you've updated the fields of the current record, you save that record to the Database with the Update method.

Here's an example using AddNew. When the user clicks the Add button in our DAO code example, the daocode project (see the first topic in this chapter), we execute the AddNew method on the program's record set and clear the two data field text boxes:

```

Private Sub Command1_Click()

    dbrecordset.AddNew

    Text1.Text = ""

    Text2.Text = ""

End Sub

```

Now users can enter data for the new record's fields and click the program's Update button. When they click the Update Database button, the new data is written to the Database.

3.4 EDITING A RECORD IN A RECORD SET

Besides adding new records to the record set, users might want to edit the existing records. To do that, you use the Edit method like this in our DAO code example, the `daocode` project (see the first topic in this chapter):

```
Private Sub Command2_Click()
```

```
    dbrecordset.Edit
```

```
End Sub
```

After users edit the data in the record's fields (by entering new data in the text fields in the `daocode` project's main window), they must update the Database with the new data, and they do that in the `daocode` project by clicking the Update Database button. That button executes the Update method, as we'll see in the next topic.

3.5 UPDATING A RECORD IN A RECORD SET

When the user changes the data in a record or adds a new record, we must update the Database to record that change, and you use the record set Update method to do that:

```
recordset.Update ([type [, force]])
```

Here are the arguments in this function:

- **Type**—Constant indicating the type of update, as specified in Settings (ODBCDirect workspaces only).
- **Force**—Boolean value indicating whether or not to force the changes into the Database, regardless of whether the data has been changed by another user (ODBCDirect workspaces only).

Let's see an example. When the user clicks the Update button in our DAO code example, the `daocodev` project (see the first topic in this chapter), we will update the Database with the new data for the current record. We get the new data for the current

record from the text boxes Text1 and Text2, where the user has entered that data, and load the data into the record set's fields using the field's collection:

```
Private Sub Command3_Click ()  
  
    dbrecordset.fields(0) = Text1.Text  
  
    dbrecordset.fields(1) = Text2.Text  
  
—  
  
End Sub
```

After loading the data into the current record's fields, we save that record to the Database using the Update method:

```
Private Sub Command3_Click()  
  
    dbrecordset.fields(0) = Text1.Text  
  
    dbrecordset.fields(1) = Text2.Text  
  
    dbrecordset.Update
```

```
End Sub
```

3.6 MOVING TO THE FIRST RECORD IN A RECORD SET

To make the first record in a record set the current record, you use the MoveFirst method. For example, here's how we move to the first record when the user clicks the appropriate button in our DAO code example, the daocode project (see the first topic in this chapter):

```
Private Sub Command4_Click()  
  
    dbrecordset.MoveFirst
```

```
...
```


End Sub

After moving to the first record, we display that record's fields in the two text boxes in the program,

Text1 and Text2:

```
Private Sub Command4_Click()
```

```
    dbrecordset.MoveFirst
```

```
    Text1.Text = dbrecordset.fields(0)
```

```
    Text2.Text = dbrecordset.fields(1)
```

```
End Sub
```

3.7 MOVING TO THE LAST RECORD IN A RECORD SET

To make the last record in a record set the current record, you use the `MoveLast` method. For example, here's how we move to the last record when the user clicks the appropriate button in our DAO code example, the `daocode` project (see the first topic in this chapter):

```
Private Sub Command7_Click()
```

```
    dbrecordset.MoveLast
```

```
    ...
```

```
End Sub
```

After moving to the last record, we display that record's fields in the two text boxes in the program,

Text1 and Text2:

```
Private Sub Command7_Click()
```

```
    dbrecordset.MoveLast
```

```

    Text1.Text = dbrecordset.fields(0)

    Text2.Text = dbrecordset.fields(1)

End Sub

```

3.8 DELETING A RECORD IN A RECORD SET

To delete a record in a DAO record set, you use the Delete method, and then you update the record set.

For example, when the user clicks the Delete button in our DAO code example, the daocode project (see the first topic in this chapter), we clear the two text boxes, Text1 and Text2, that display the data for the current record and delete that record:

```

Private Sub Command8_Click()

    Text1.Text = ""

    Text2.Text = ""

    dbrecordset.Delete

End Sub

```

3.10 SEARCHING A RECORD SET

You can search a record set with an index; we just set its Index property to the index we want to search and then set its Seek property to the string we want to search for. Let's see an example. When the user selects the Search menu item in our DAO code example, the daocode project (see the first topic in this chapter), we install the index based on the first field in the record set and show the dialog box named Search:

```

Private Sub Search_Click()

    Set dbindex = td.Indexes(0)

    dbrecordset.Index = dbindex.Name

    SearchForm.Show

```

End Sub

After the user dismisses the Search... dialog box, we retrieve the text to search for from that dialog box's text box and place that text in the record set's Seek property, along with the command "=", which indicates we want to find exact matches to the search text:

Sub SearchTable()

dbrecordset.Seek "=", SearchForm.Text1.Text

...

Besides =, you can also search using <, <=, >=, and >. When the search is complete, we display the found record in the daocode project's main text boxes, Text1 and Text2:

Sub SearchTable()

dbrecordset.Seek "=", SearchForm.Text1.Text

Text1.Text = dbrecordset.fields(0)

Text2.Text = dbrecordset.fields(1)

End Sub

CHAPTER 4

ACCOUNT FEE STUDENT

4.1 OVER VIEW

This project is an easy way for registering new incoming students, to update all the old students and users.

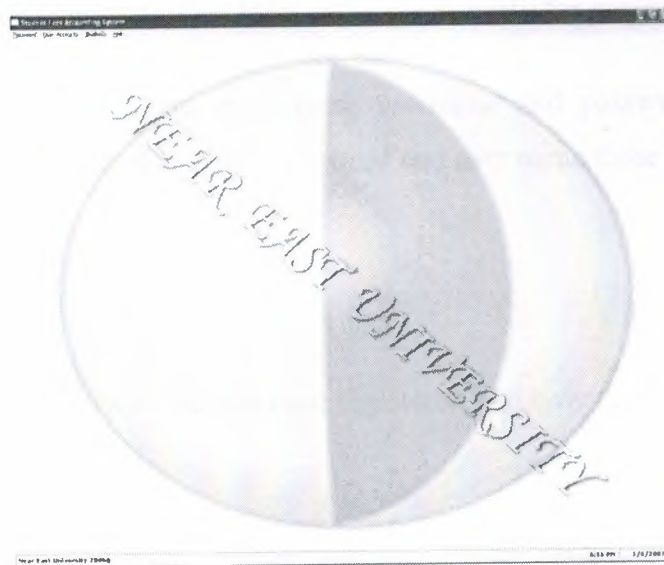


Figure 4.1: This form show the home page

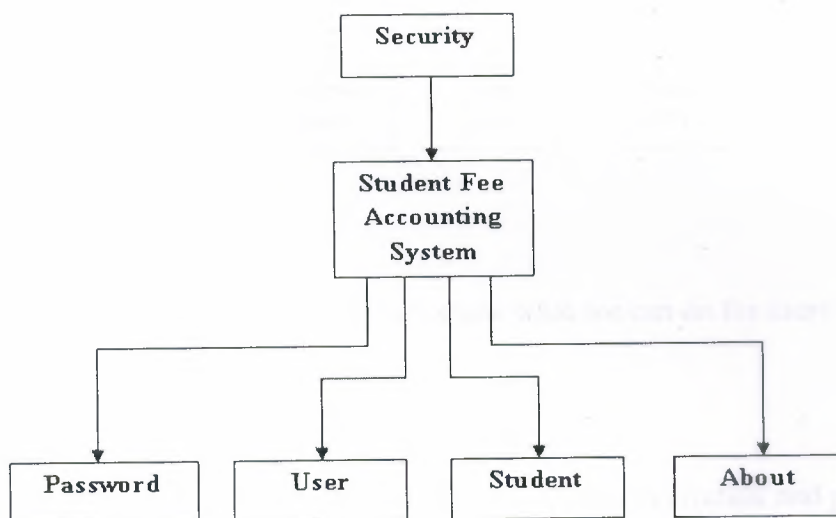


Diagram 4.1: home page diagram

4.2 SECURITY

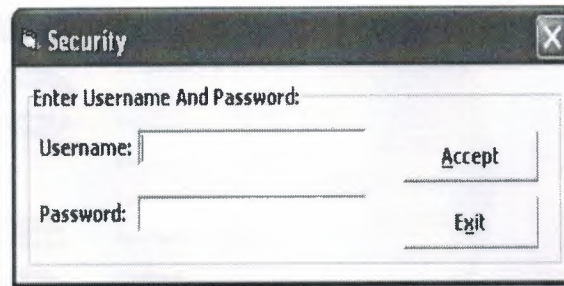


Figure 4.2: This form about the security for project

In this form users should enter their username and password to log in the program, if the user entered a wrong password and user name three times the program will close directly.

4.3 USERS

This project shows how the users are registered and how they gain benefit out of it.

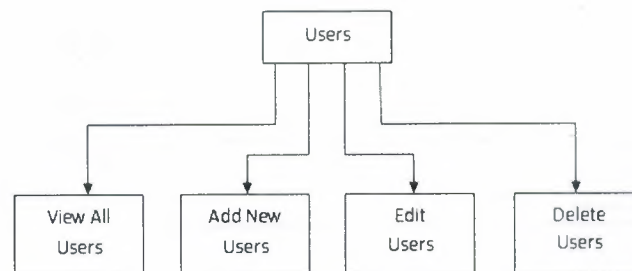


Diagram 4.2: This diagram shows what we can do for users

4.3.1 ADD NEW USERS

In this section we can add a new user to have an authorization and password to run this program as you see in the figure 4.3.

Add New User

NEAR EAST

Please Fill In All The Requested Information And Press [Save] To Update The Database:

ID:

First Name:

Last Name:

User Name:

Password:

Registered In:

Figure 4.3: This form for enter the information for user

When we press user restriction you will go to figure 4.4.

User Restrictions Settings

NEAR EAST

Please Check or Uncheck The Actions That The User Can Access Or Is Prohibited To Be Accessed.

User ID:

User Name:

The User Can:

<input type="checkbox"/> Log In	<input type="checkbox"/> Register New Std	<input type="checkbox"/> Lock System
<input type="checkbox"/> View All Users	<input type="checkbox"/> Register Old Std	<input type="checkbox"/> Minimize Application
<input type="checkbox"/> Add New Users	<input type="checkbox"/> Edit Stdnt Info	<input type="checkbox"/> Exit
<input type="checkbox"/> Edit Users	<input type="checkbox"/> Unreg Stdnt	
<input type="checkbox"/> Delete Users	<input type="checkbox"/> Debts And Payments	

Figure 4.4: Restriction for user.

When we press user ok you will go to figure 4.3.

When press save will save all information for user in data base.

4.3.2 EDIT USERS

In this section we can update user and password information first you must search for the users then update the information form the database as you see in figure 4.5.

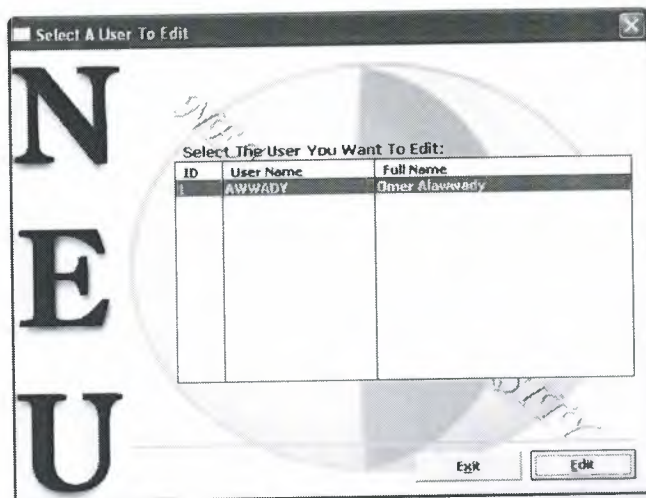


Figure 4.5: List for user

You must select the user from the list then press edit you will go to figure 4.6.

Please Fill In All The Requested Information And Press [Save] To Update The Database:

ID:

First Name:

Last Name:

User Name:

Password:

☐ Show Password

User Restriction

Cancel Save

Figure 4.6: Edit for user

When you press in user restriction you will go to figure 4.4, if you press save it will update in data base and you will go to figure 4.5.

4.3.3 DELETE USERS

In this section we can delete users from the list first you must search for the users then deleted form the database as you see in the figure 4.7.

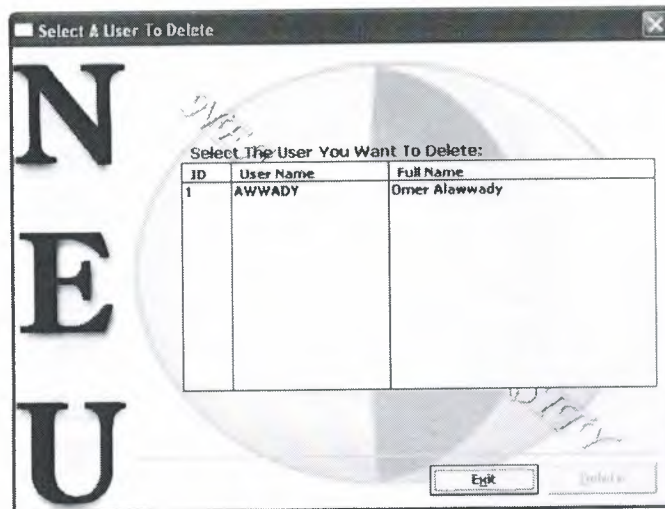


Figure 4.7: Delete for user

You must select the user then press delete to delete the user for data base.

4.3.4 VIEWING ALL USERS

In this section gives information about user (ID, name, full name and the last access), as you see in the figure 4.8.

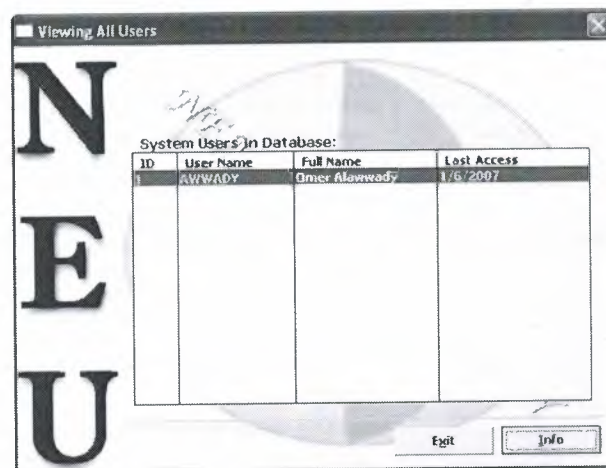


Figure 4.8: list of all the users

If you want more information you must select the users then press (Info) to give you all information as you see in figure 4.9.

User Information

Viewing User Information:

ID:

First Name:

Last Name:

User Name:

Registered In:

Last Accessed In:

[Back](#)

Figure 4.9: This form show all information for the users

4.4 STUDENT

This sections about student as you see in the diagram 4.3.

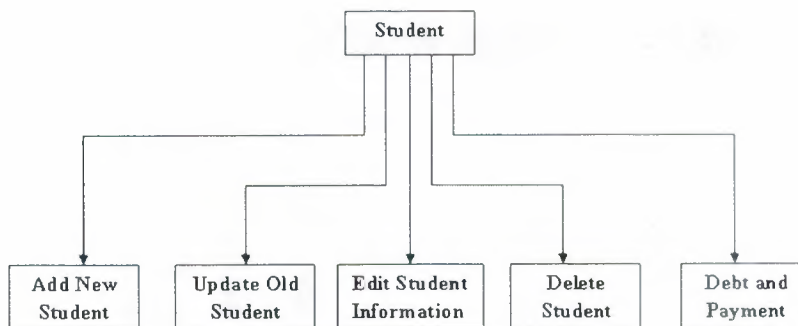


Diagram 4.3: show the forms for student

4.4.1 NEW STUDENT

In this section we registered the new student.

First we enter the information for new coming students, but you must enter all the information, as you see in the figure 4.10.

Add New Student

First Name: Omar

Sir Name: Yasin

Department: Computer Engineering

Date Of Birth: 12/28/1983

Place Of Birth: Amman, Jordan

Passport Number: K12984

Balance: 1250 USD Dollar

Fees: 1250 USD Dollar

Registration Date: 11/18/2027

Load

Back Next

Figure 4.10:

Second when we press next we will get the student number as you see in figure 4.11.

Register New Student

Press [Accept] to save the new student in database, To cancel or to edit, click [Back].

Student Name: Omar Yasin

Student Number: 20070001

Student Department: Computer Engineering

Back Accept

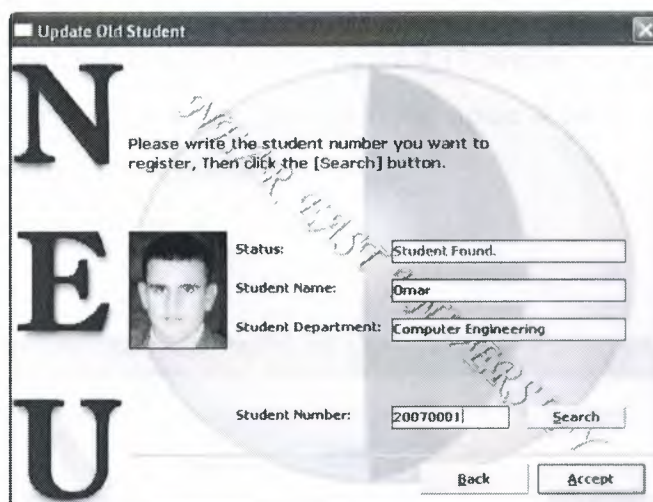
Figure 4.11: for get the number for new student

Third when we press accept the student information will be saved in the database.

4.4.2 UPDATE OLD STUDENT

In this section we registered the old student

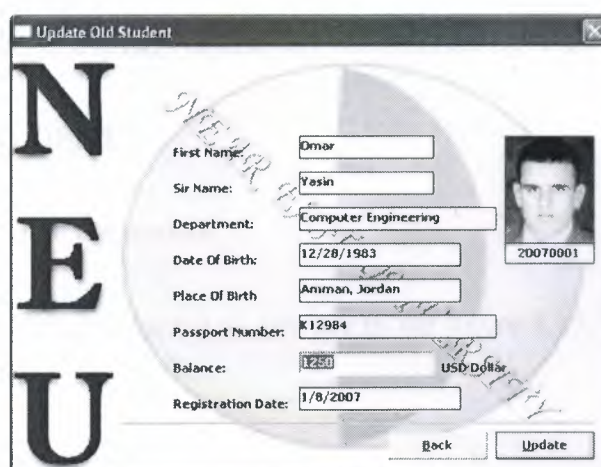
First we must search for student by entering the student number if the student isn't available not found student message will be displayed in status.



The screenshot shows a web application window titled "Update Old Student". On the left, the letters "N", "E", and "U" are stacked vertically. The main area contains a search instruction: "Please write the student number you want to register, Then click the [Search] button." Below this is a small portrait of a man. To the right of the portrait, the following information is displayed in a form-like layout: "Status: Student Found.", "Student Name: Omar", and "Student Department: Computer Engineering". At the bottom, there is a "Student Number" field containing "20070001", a "Search" button, and "Back" and "Accept" buttons.

Figure 4.12: To search for student

If the student number is available the message will be displayed in the status and all information of the student, as you see in figure 4.12. If the student is available after pressing except you will see the next form.



The screenshot shows the same "Update Old Student" window, but now it displays a registration form. The form fields are: "First Name: Omar", "Sir Name: Yasin", "Department: Computer Engineering", "Date Of Birth: 12/28/1983", "Place Of Birth: Amman, Jordan", "Passport Number: K12984", "Balance: 1250 USD Dollar", and "Registration Date: 1/8/2007". A small portrait of the student is shown on the right with the number "20070001" below it. At the bottom, there are "Back" and "Update" buttons.

Figure 4.13: Registration for old student

Second we can't update anything except the balances, when you press update it will be saved in the database.

4.4.3 EDIT STUDENT INFORMATION

This section for edit the information for the student.

After we make search for student as the figure 4.12, we go to next form to change all the information for the student as you see in the figure 4.14, when we press save button it will BE saved in the database.

■ Edit Existing Student: 20070001

First Name:

Sir Name:

Department:

Date Of Birth:

Place Of Birth:

Passport Number:

Balance: USD Dollar

Fees: USD Dollar

Registration Date:

Figure 4.14: For edit any information for student

4.4.4 DELETE STUDENT

This section for delete the student.

After we make search for student as you see in the figure 4.15, then we can delete the student.

Figure 4.15: For delete student

When press delete will delete all the information from the database,

4.4.5 DEBT AND PAYMENT FOR STUDENT

This form for see if the student has debt or credit.

Fist you will search for the student by figure 4.12, then you will see if he have debt or credit as you will see in the figure 4.16.

Figure 4.16: Debt and payment for student

CONCLUSION

Visual Basic is the favorite programming environment of many programmers. When Visual Basic first appeared, it created a revolution in Windows programming, and that revolution continues to this day. Never before had Windows programming been so easy just build the program you want, right before your eyes, and then run it. Visual Basic introduced unheard-of ease to Windows programming and changed programming from a chore to something very fun. When Visual Basic first started working with databases, it used the Microsoft Jet database engine,

- This program is talking about the registering of students in university by using the Visual Basic programming language and data base.
- This project helps us to save all of needed information about any student who registered in university and any new student who come to register in university.
- The most important thing in my working that any one can use it and no need to professional person, so it's very easy and very clear.
- Each user has a particular authority in using this program by defining his actions and that related to his name and password and it's considered to be security for the program
- The project designed to protect the user from himself

REFERENCE

- [1] Visual Basic 6 Black Book
- [2] Sams Teach Yourself Microsoft Access 2000 in 21 Days
- [3] Sam's Teach Yourself MySQL in 21 Days
- [4] <http://www.Wikipedia.com>
- [5] <http://www.sqlcourse.com/table.html>
- [6] <http://www.howstuffworks.com>
- [7] <http://www.Wibopedia.com>
- [8] <http://www.Computerhope.com>

APPENDIX

Student Fee Accounting System:

Option Explicit

Private Const Min_Height = 8700

Private Const Min_Width = 10000

Dim MinStat As Boolean

Public Sub SetRest_Mine()

 If GetRestriction(REST_LOCK) Then mnuSysLock.Enabled = False

 If GetRestriction(REST_EXIT) Then mnuExit.Enabled = False

 If GetRestriction(REST_MINM) Then MinStat = False Else MinStat = True

 If GetRestriction(REST_VEWA) Then mnuViewAll.Enabled = False

 If GetRestriction(REST_ADDDU) Then mnuAddNew.Enabled = False

 If GetRestriction(REST_DELU) Then mnuDelUser.Enabled = False

 If GetRestriction(REST_EDTU) Then mnuEditlUser.Enabled = False

 If GetRestriction(REST_REGN) Then mnuRegNewStd.Enabled = False

 If GetRestriction(REST_REGO) Then mnuRegOldStd.Enabled = False

 If GetRestriction(REST_DBTS) Then mnuStdDbtsAndPayments.Enabled =
False

 If GetRestriction(REST_EDTS) Then mnuEditStd.Enabled = False

 If GetRestriction(REST_DELS) Then mnuUnregStd.Enabled = False

End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)

 Cancel = True

 Call mnuExit_Click

End Sub


```
Private Sub Form_Resize()
```

```
    If Me.WindowState = vbMinimized And Not MinStat Then
```

```
        Me.WindowState = vbMaximized
```

```
        MsgBox "Minimizing The Application Is Prohibited!" & vbCrLf & "Contact  
your system administrator.", vbInformation, "Error: Access Denied"
```

```
        Exit Sub
```

```
    End If
```

```
    If Me.WindowState = vbMinimized Then Exit Sub
```

```
    With img_background
```

```
        .Left = 0
```

```
        .Top = -10
```

```
        .Width = Me.Width
```

```
        .Height = Me.Height - 1100
```

```
    End With
```

```
    With Me
```

```
        If .Height < Min_Height Then .Height = Min_Height: SendKeys "{escape}"
```

```
        If .Width < Min_Width Then .Width = Min_Width: SendKeys "{escape}"
```

```
    End With
```

```
End Sub
```

```
Private Sub mnuRegStd_Click()
```

```
    frmNewStdnt.Show vbModal
```

```
End Sub
```

```
Private Sub mnuAbout_Click()
```

```
    frmAbout.Show vbModal
```

```
End Sub
```

```
Private Sub mnuAddNew_Click()
```

```
    frmAddNewUser.Show vbModal  
End Sub
```

```
Private Sub mnuDelUser_Click()  
    frmUserSlctDel.Show vbModal  
End Sub
```

```
Private Sub mnuEditlUser_Click()  
    frmUserSlct.Show vbModal  
End Sub
```

```
Private Sub mnuEditStd_Click()  
    frmEditStdntNum.Show vbModal  
End Sub
```

```
Private Sub mnuExit_Click()  
    If mnuExit.Enabled = False Then MsgBox "You Cannot Exit This  
Application!!!" & vbCrLf & "Contact your system administrator.", vbInformation:  
Exit Sub  
    If MsgBox("Are You Sure You Want To Exit?", vbYesNo + vbQuestion +  
vbApplicationModal, "Exit") = vbYes Then End  
End Sub
```

```
Private Sub mnuRegNewStd_Click()  
    frmNewStdnt.Show vbModal  
End Sub
```

```
Private Sub mnuRegOldStd_Click()  
    frmOldStdntNum.Show vbModal  
End Sub
```

```
Private Sub mnuStdDbtsAndPayments_Click()  
    frmDebtsStdNum.Show vbModal
```

End Sub

Private Sub mnuSysLock_Click()

Load frmLogIn

frmLogIn.IsSecuring = True

frmLogIn.cmdExit.Enabled = False

frmLogIn.Show vbModal

End Sub

Private Sub mnuUnregStd_Click()

frmUnregStdnt.Show vbModal

End Sub

Private Sub mnuUserPerRes_Click()

End Sub

Private Sub mnuViewAll_Click()

frmVAUsers.Show vbModal

End Sub

Security:

Option Explicit

Dim iCount As Integer, iLoc As Integer

Public IsSecuring As Boolean

Private Sub cmdAccept_Click()

Dim strTheUserName As String, strThePassword As String

strTheUserName = txtUsername.Text

strThePassword = txtPass.Text

Call System_DBConnect

Call System_FindUserName(strTheUserName)

If System_RSEOF Then

MsgBox "Error: Invalid User Name Entered!!!" _
 & vbCrLf & IIf(IsSecuring, "", CStr(iCount) & " Tries Left!"), _
 vbCritical, "ERROR: User Name Error"

txtUsername.SetFocus

Call SendKeys("{home}+{end}")

If IsSecuring = False Then iCount = iCount - 1

Else

If Not (System_GetPassword = strThePassword) Then

MsgBox "Error: Invalid Password Entered!!!" _
 & vbCrLf & IIf(IsSecuring, "", CStr(iCount) & " Tries Left!"), _
 vbCritical, "ERROR: Password Error"

txtPass.SetFocus

Call SendKeys("{home}+{end}")

If IsSecuring = False Then iCount = iCount - 1

Else

Call SetRestriction(strTheUserName)

Call SetLastAccess(strTheUserName)

If Not IsSecuring Then

If GetRestriction(REST_LGIN) = False Then

Load frmMainDialog

frmMainDialog.SetRest_Mine

frmMainDialog.Show


```

        Unload Me
    Else
        MsgBox "This Username Has Been Blocked!" & vbCrLf & "To be
able to login, contact your system administrator.", vbInformation, "Access
Blocked"
        txtUsername.SetFocus
    Exit Sub
End If
Else
    Unload Me
End If
End If
End If

```

```

If (iCount = -1 And Not IsSecuring = True) Then
    MsgBox "User Name And\Or Password was entered 3 times incorrectly," &
vbCrLf & _
    "Please Consult Your System Administrator." & vbCrLf & vbCrLf & _
    "Application Will Now Exit.", vbInformation + vbApplicationModal,
"ERROR"
    End
End If
End Sub

```

```

Private Sub cmdExit_Click()
    If MsgBox("Are you sure you want to exit?", vbYesNo + vbQuestion, "Exit") =
vbYes Then Unload Me
End Sub

```

```

Private Sub Form_Load()
    iCount = 2
End Sub

```

```

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    If UnloadMode = vbFormControlMenu Then
        If cmdExit.Enabled = False Then Cancel = True: Exit Sub
        If MsgBox("Are you sure you want to exit?", vbYesNo + vbQuestion, "Exit")
= vbNo Then Cancel = True
    End If
End Sub

```

```

Private Sub txtPass_KeyDown(KeyCode As Integer, Shift As Integer)
    If KeyCode = vbKeyReturn Then Call cmdAccept_Click
End Sub

```

```

Private Sub txtUsername_Change()
    If txtUsername.SelStart = 0 Then Exit Sub
    iLoc = txtUsername.SelStart
    txtUsername.Text = UCase$(txtUsername.Text)
    txtUsername.SelStart = iLoc
End Sub

```

```

Private Sub txtPass_Change()
    If txtPass.SelStart = 0 Then Exit Sub
    iLoc = txtPass.SelStart
    txtPass.Text = UCase$(txtPass.Text)
    txtPass.SelStart = iLoc
End Sub

```

```

Private Sub txtUsername_KeyDown(KeyCode As Integer, Shift As Integer)
    If KeyCode = vbKeyReturn Then txtPass.SetFocus
End Sub

```

```

Private Sub txtUsername_KeyPress(KeyAscii As Integer)
    KeyAscii = LimitInputToStringAllCase(KeyAscii)

```

End Sub

```
Private Sub txtPass_KeyPress(KeyAscii As Integer)
    KeyAscii = LimitInputToStringAllCase(KeyAscii)
End Sub
```

Users

View All Users

Option Explicit

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

```
Private Sub cmdInfo_Click()
    Call System_DBConnect
    Call System_DBExecute("select * from users where id=" &
        CInt(lstIDs.List(lstIDs.ListIndex)) & "")
```

With frmUserInfo

```
.txtUserDetails(0) = SysRS![id]
.txtUserDetails(1) = SysRS![fname]
.txtUserDetails(2) = SysRS![lname]
.txtUserDetails(3) = SysRS![UserName]
.txtUserDetails(4) = SysRS![RegDate]
.txtUserDetails(5) = SysRS![lastaccess]
.Show vbModal
```

```

End With

Call System_DBClose
End Sub

Private Sub Form_Load()
    Call System_DBConnect
    Call System_DBExecute("select id, username, fname, lname, lastaccess from
users")

    Do While Not SysRS.EOF
        lstIDs.AddItem SysRS![id]
        lstUsers.AddItem SysRS![UserName]
        lstLastAcc.AddItem Str(If(IsNull(SysRS![lastaccess]), "",
SysRS![lastaccess]))
        lstFullName.AddItem SysRS![fname] & " " & SysRS![lname]
        DoEvents
        SysRS.MoveNext
        DoEvents
    Loop

    Call System_DBClose

    If lstIDs.ListCount > 0 Then cmdInfo.Enabled = True: lstIDs.ListIndex = 0
End Sub

Private Sub lstFullName_MouseMove(Button As Integer, Shift As Integer, X As
Single, Y As Single)
    lstUsers.ListIndex = lstIDs.ListIndex
    lstFullName.ListIndex = lstIDs.ListIndex
    lstLastAcc.ListIndex = lstIDs.ListIndex
End Sub

```



```
Private Sub lstIDs_Click()
```

```
    lstUsers.ListIndex = lstIDs.ListIndex
```

```
    lstFullName.ListIndex = lstIDs.ListIndex
```

```
    lstLastAcc.ListIndex = lstIDs.ListIndex
```

```
End Sub
```

```
Private Sub lstFullName_Click()
```

```
    lstUsers.ListIndex = lstIDs.ListIndex
```

```
    lstFullName.ListIndex = lstIDs.ListIndex
```

```
    lstLastAcc.ListIndex = lstIDs.ListIndex
```

```
End Sub
```

```
Private Sub lstUsers_Click()
```

```
    lstUsers.ListIndex = lstIDs.ListIndex
```

```
    lstFullName.ListIndex = lstIDs.ListIndex
```

```
    lstLastAcc.ListIndex = lstIDs.ListIndex
```

```
End Sub
```

```
Private Sub lstLastAcc_Click()
```

```
    lstUsers.ListIndex = lstIDs.ListIndex
```

```
    lstFullName.ListIndex = lstIDs.ListIndex
```

```
    lstLastAcc.ListIndex = lstIDs.ListIndex
```

```
End Sub
```

```
Private Sub lstIDs_MouseDown(Button As Integer, Shift As Integer, X As Single,  
Y As Single)
```

```
    lstUsers.ListIndex = lstIDs.ListIndex
```

```
    lstFullName.ListIndex = lstIDs.ListIndex
```

```
    lstLastAcc.ListIndex = lstIDs.ListIndex
```

```
End Sub
```

```
Private Sub lstFullName_MouseDown(Button As Integer, Shift As Integer, X As  
Single, Y As Single)
```

```
lstUsers.ListIndex = lstIDs.ListIndex
lstFullName.ListIndex = lstIDs.ListIndex
lstLastAcc.ListIndex = lstIDs.ListIndex
End Sub
```

```
Private Sub lstIDs_MouseMove(Button As Integer, Shift As Integer, X As Single,
Y As Single)
    lstUsers.ListIndex = lstIDs.ListIndex
    lstFullName.ListIndex = lstIDs.ListIndex
    lstLastAcc.ListIndex = lstIDs.ListIndex
End Sub
```

```
Private Sub lstUsers_MouseDown(Button As Integer, Shift As Integer, X As
Single, Y As Single)
    lstUsers.ListIndex = lstIDs.ListIndex
    lstFullName.ListIndex = lstIDs.ListIndex
    lstLastAcc.ListIndex = lstIDs.ListIndex
End Sub
```

```
Private Sub lstUsers_MouseMove(Button As Integer, Shift As Integer, X As
Single, Y As Single)
    lstUsers.ListIndex = lstIDs.ListIndex
    lstFullName.ListIndex = lstIDs.ListIndex
    lstLastAcc.ListIndex = lstIDs.ListIndex
End Sub
```

```
Private Sub lstLastAcc_MouseDown(Button As Integer, Shift As Integer, X As
Single, Y As Single)
    lstUsers.ListIndex = lstIDs.ListIndex
    lstFullName.ListIndex = lstIDs.ListIndex
    lstLastAcc.ListIndex = lstIDs.ListIndex
End Sub
```

```

Private Sub lstLastAcc_MouseMove(Button As Integer, Shift As Integer, X As
Single, Y As Single)
    lstUsers.ListIndex = lstIDs.ListIndex
    lstFullName.ListIndex = lstIDs.ListIndex
    lstLastAcc.ListIndex = lstIDs.ListIndex
End Sub

```

User Information

Option Explicit

```

Private Sub cmdExit_Click()
    Unload Me
End Sub

```

Add New Users

Option Explicit

Public UstrRstr As Long

```

Private Sub cmdExit_Click()
    Unload Me
End Sub

```

```

Private Sub cmdSave_Click()
    Dim i As Integer
    For i = 1 To 4
        If txtUserDetails(i).Text = "" Then
            MsgBox "Error: Some Data Missing," & vbCrLf & "Please Fill All
Fields.", vbCritical, "Error: Data Missing"
            txtUserDetails(i).SetFocus
            SendKeys "{home}+{end}"
        End If
    Next i
End Sub

```

Next i

If UstrRstr = 0 Then _

 If MsgBox("Warning: User Restrictions Are Still Unsettled!" & vbCrLf &
 "User Access Will Be Disabled!", _
 vbExclamation + vbQuestion, "User Restriction") = vbYes Then _
 frmSetRestr.Show vbModal: Exit Sub

Call System_DBConnect

Call System_DBExecute("insert into users(id, fname, lname, username,
password, regdate, lastaccess, restr) " & _
 "values(" & txtUserDetails(0).Text & _
 ", '" & txtUserDetails(1).Text & "'" & _
 ", '" & txtUserDetails(2).Text & "'" & _
 ", '" & txtUserDetails(3).Text & "'" & _
 ", '" & txtUserDetails(4).Text & "'" & _
 ", '" & txtUserDetails(5).Text & "'" & _
 ", '" & txtUserDetails(5).Text & "'" & _
 ", " & CStr(UstrRstr) & ")")

Call System_DBClose

MsgBox "User Added Successfully!", vbInformation, "Successful"

Unload Me

End Sub

Private Sub cmdUserRestr_Click()

With frmSetRestr

 .txtUserDetails(0).Text = txtUserDetails(0).Text
 .txtUserDetails(1).Text = txtUserDetails(3).Text
 .Show vbModal



```
End With  
End Sub
```

```
Private Sub Form_Load()  
    txtUserDetails(5).Text = FixDate(Format$(Now, "MM/DD/YYYY"))  
    UstrRstr = 0
```

```
    Call System_DBConnect  
    Call System_DBExecute("select max(id) as tmp from users")
```

```
    If SysRS.EOF Then  
        txtUserDetails(0).Text = "1"  
    Else  
        txtUserDetails(0).Text = SysRS![tmp] + 1
```

```
    End If  
End Sub
```

```
Private Sub txtUserDetails_KeyPress(Index As Integer, KeyAscii As Integer)  
    Dim iStrtPos As Integer
```

```
    If KeyAscii = vbKeyBack Or KeyAscii = vbKeyDelete Then Exit Sub
```

```
    If KeyAscii = vbKeyReturn Then If Index < 5 Then txtUserDetails(Index +  
1).SetFocus: SendKeys "{home}+{end}" Else cmdUserRestr.SetFocus
```

```
    iStrtPos = txtUserDetails(Index).SelStart
```

```
    Select Case Index
```

```
        Case Is = 1
```

```
            KeyAscii = LimitInputToStringAllCaseWithSpace(KeyAscii)
```

```
            txtUserDetails(Index).Text = FirstCaps(txtUserDetails(Index).Text)
```

```

Case Is = 2
KeyAscii = LimitInputToStringAllCaseWithSpace(KeyAscii)
txtUserDetails(Index).Text = FirstCaps(txtUserDetails(Index).Text)

Case Is = 3: KeyAscii = Asc(UCASE$(Chr(KeyAscii))): KeyAscii =
LimitInputToStringLarge(KeyAscii)

Case Is = 4: KeyAscii = Asc(UCASE$(Chr(KeyAscii))): KeyAscii =
LimitInputToStringLarge(KeyAscii)
End Select

txtUserDetails(Index).SelStart = iStrtPos
End Sub

```

Users Restriction

Option Explicit

```
Private Sub cmdBack_Click()
```

```
    Unload Me
```

```
End Sub
```

```
Private Sub cmdOK_Click()
```

```
    Dim i As Integer, sum As Long
```

```
    sum = 0
```

```
    For i = 0 To 12
```

```
        sum = sum + IIf(chkUsrRestr(i).Value = vbChecked, chkUsrRestr(i).Tag, 0)
```

```
    Next i
```

```
    frmEditUser.edtUserRest = sum
```

```
MsgBox "Done.      ", vbInformation, "Successful"  
Unload Me  
End Sub
```

```
Private Sub Form_Load()  
    Call System_DBCConnect  
End Sub
```

```
Private Sub lblRestrTitle_Click(Index As Integer)  
    chkUsrRestr(Index).Value = IIf(chkUsrRestr(Index).Value = vbChecked,  
vbUnchecked, vbChecked)  
End Sub
```

Edit Users

```
Option Explicit  
Public edtUserRest As Long
```

```
Private Sub chkShowHidePass_Click()  
    txtUserDetails(4).PasswordChar = IIf(chkShowHidePass.Value = vbChecked,  
"", "*")  
End Sub
```

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

```
Private Sub cmdSave_Click()  
    Dim i As Integer, strSQL As String  
    For i = 1 To 4  
        If txtUserDetails(i).Text = "" Then
```

```

        MsgBox "Error: Data Missing," & vbCrLf & "Please Fill All Fields.",
vbCritical, "Error: Data Missing"
        txtUserDetails(i).SetFocus
        Exit Sub
    End If
Next i

If edtUserRest = 0 Then _
    If MsgBox("Warning: User Access Will Be Disabled!" & vbCrLf &
"Selected User will not be able to access the system anymore," & _
    "Are You Sure?", _
    vbExclamation + vbYesNo, "User Restriction") = vbYes Then _
        frmSetRestr.Show vbModal: Exit Sub

Call System_DBConnect

strSQL = "update users set fname='" & txtUserDetails(1).Text & "'" & _
    ", lname='" & txtUserDetails(2).Text & "'" & _
    ", username='" & txtUserDetails(3).Text & "'" & _
    ", password='" & txtUserDetails(4).Text & "'" & _
    ", restr=" & edtUserRest & "" & _
    " where id=" & txtUserDetails(0).Text & ""

Call System_DBExecute(strSQL)

Call System_DBClose

MsgBox "User Information Was Saved Successfully!", vbInformation,
"Successfull"

Unload Me
End Sub

```



```
Private Sub cmdUserRestr_Click()
```

```
Dim j As Integer
```

```
Call ParsetmpRestriction(edtUserRest)
```

```
With frmEditRest
```

```
.txtUserDetails(0).Text = txtUserDetails(0).Text
```

```
.txtUserDetails(1).Text = txtUserDetails(3).Text
```

```
For j = 0 To 12
```

```
.chkUsrRestr(j).Value = IIf(tmpRestrictions(j) = True, vbUnchecked,  
vbChecked)
```

```
DoEvents
```

```
Next j
```

```
.Show vbModal
```

```
End With
```

```
End Sub
```

```
Private Sub Form_Load()
```

```
Call System_DBConnect
```

```
End Sub
```

```
Private Sub lblShowHidePass_Click()
```

```
chkShowHidePass.Value = IIf(chkShowHidePass.Value = vbChecked,  
vbUnchecked, vbChecked)
```

```
End Sub
```

```
Private Sub txtUserDetails_KeyPress(Index As Integer, KeyAscii As Integer)
```

```
Dim iStrtPos As Integer
```

```
If KeyAscii = vbKeyBack Or KeyAscii = vbKeyDelete Then Exit Sub
```

```
If KeyAscii = vbKeyReturn Then If Index < 5 Then txtUserDetails(Index +  
1).SetFocus: SendKeys "{home}+{end}" Else cmdUserRestr.SetFocus
```

```
iStrtPos = txtUserDetails(Index).SelStart
```

```
Select Case Index
```

```
Case Is = 1
```

```
KeyAscii = LimitInputToStringAllCaseWithSpace(KeyAscii)
```

```
txtUserDetails(Index).Text = FirstCaps(txtUserDetails(Index).Text)
```

```
Case Is = 2
```

```
KeyAscii = LimitInputToStringAllCaseWithSpace(KeyAscii)
```

```
txtUserDetails(Index).Text = FirstCaps(txtUserDetails(Index).Text)
```

```
Case Is = 3
```

```
KeyAscii = Asc(UCase$(Chr(KeyAscii)))
```

```
KeyAscii = LimitInputToStringLarge(KeyAscii)
```

```
Case Is = 4
```

```
KeyAscii = Asc(UCase$(Chr(KeyAscii)))
```

```
KeyAscii = LimitInputToStringLarge(KeyAscii)
```

```
End Select
```

```
txtUserDetails(Index).SelStart = iStrtPos
```

```
End Sub
```

Delete Users

Option Explicit

```
Private Sub cmdExit_Click()
```

```

Unload Me
End Sub

Private Sub cmdUsrSlctDel_Click()
    Dim strSQL As String

    If MsgBox("Are You Sure You Wish To Delete The Selected User?" & vbCrLf
    & "Changes cannot be restored!" _
    , vbExclamation + vbYesNo, "Delete User") = vbYes Then

        Call System_DBConnect
        strSQL = "delete * from users where id=" &
        CInt(lstIDs.List(lstIDs.ListIndex)) & ""

        Call System_DBExecute(strSQL)

        MsgBox "User Was Deleted Successfully!", vbInformation, "Successful"

        Call System_DBClose

        Call Form_Load
    End If
End Sub

Private Sub Form_Load()
    Call System_DBConnect
    Call System_DBExecute("select id, username, fname, lname, lastaccess from
users")

    lstIDs.Clear
    lstUsers.Clear
    lstFullName.Clear

```

```

Do While Not SysRS.EOF
    lstIDs.AddItem SysRS![id]
    lstUsers.AddItem SysRS![UserName]
    lstFullName.AddItem SysRS![fname] & " " & SysRS![lname]
    DoEvents
    SysRS.MoveNext
    DoEvents
Loop

Call System_DBClose

If lstIDs.ListCount = 1 Then
    cmdUsrSlctDel.Enabled = False
    MsgBox "Only One User Found In Database!!!" & vbCrLf & "You cannot
delete the last user.", vbExclamation, "One User Found!"
    Exit Sub
End If

If lstIDs.ListCount > 0 Then
    cmdUsrSlctDel.Enabled = True
    lstIDs.ListIndex = 0
Else
    MsgBox "No Users Found In Database!" & vbCrLf & "Please add two users
at least before trying to delete one!", _
    vbInformation, "No Users"
    Unload Me
End If
End Sub

Private Sub lstFullName_MouseMove(Button As Integer, Shift As Integer, X As
Single, Y As Single)
    lstUsers.ListIndex = lstIDs.ListIndex
    lstFullName.ListIndex = lstIDs.ListIndex

```


End Sub

Private Sub lstIDs_Click()

 lstUsers.ListIndex = lstIDs.ListIndex

 lstFullName.ListIndex = lstIDs.ListIndex

End Sub

Private Sub lstFullName_Click()

 lstIDs.ListIndex = lstIDs.ListIndex

 lstFullName.ListIndex = lstIDs.ListIndex

End Sub

Private Sub lstUsers_Click()

 lstUsers.ListIndex = lstIDs.ListIndex

End Sub

Private Sub lstIDs_MouseDown(Button As Integer, Shift As Integer, X As Single,
Y As Single)

 lstFullName.ListIndex = lstIDs.ListIndex

 lstUsers.ListIndex = lstIDs.ListIndex

End Sub

Private Sub lstFullName_MouseDown(Button As Integer, Shift As Integer, X As
Single, Y As Single)

 lstFullName.ListIndex = lstIDs.ListIndex

 lstUsers.ListIndex = lstIDs.ListIndex

End Sub

Private Sub lstIDs_MouseMove(Button As Integer, Shift As Integer, X As Single,
Y As Single)

 lstFullName.ListIndex = lstIDs.ListIndex

 lstUsers.ListIndex = lstIDs.ListIndex

End Sub

```
Private Sub lstUsers_MouseDown(Button As Integer, Shift As Integer, X As  
Single, Y As Single)
```

```
    lstFullName.ListIndex = lstIDs.ListIndex
```

```
    lstUsers.ListIndex = lstIDs.ListIndex
```

```
End Sub
```

```
Private Sub lstUsers_MouseMove(Button As Integer, Shift As Integer, X As  
Single, Y As Single)
```

```
    lstFullName.ListIndex = lstIDs.ListIndex
```

```
    lstUsers.ListIndex = lstIDs.ListIndex
```

```
End Sub
```

Student:

Add New Student

Option Explicit

Public IsOK As Boolean

Private Const TXT_MIN = 0

Private Const TXT_MAX = 7

Private Const TXT_FIRSTN = 0

Private Const TXT_LASTN = 1

Private Const TXT_DATEOFBIRTH = 2

Private Const TXT_PLACEOFBIRTH = 3

Private Const TXT_PASSPORTNUM = 4

Private Const TXT_BALANCE = 5

Private Const TXT_REGDATE = 6

Private Const TXT_FEES = 7

Private Const MAX_CHRSZ = 64

```

Private Const COMB_COMP = "Computer Engineering"
Private Const COMB_ELEC = "Electrical Engineering"
Private Const COMB_MECH = "Mechanical Engineering"
Private Const COMB_CIVL = "Civil Engineering"

```

```

Dim IsProcessing As Boolean

```

```

Private Sub cmdBack_Click()
    Unload Me
End Sub

```

```

Private Sub cmdLoadPic_Click()
    On Error GoTo ErrHndlr
    With CDG
        .CancelError = True
        .DialogTitle = "Select Student Picture"
        .Flags = cdlOFNFileMustExist
        .Filter = "JPEG Files(*.jpg)|*.jpg;*.jpeg|Bitmap Images(*.bmp)|*.bmp|GIF
Files(*.gif)|*.gif|All Files(*.*)|*.*"
        .ShowOpen

        If Not .FileName = vbNullString Then
            If FileExist(.FileName) Then _
                stPic.Picture = LoadPicture(.FileName)
            stPic.Tag = .FileName
            stPic.ToolTipText = "Picture Loaded: [" & .FileName & "]"
        Else
            MsgBox "Error: No Picture Loaded!!!", vbCritical + vbApplicationModal,
"Error: File Access Error"
        End If
    End With

```

Exit Sub

ErrHndlr:

End Sub

Private Sub cmdNext_Click()

Dim i As Integer

For i = TXT_MIN To TXT_MAX

If txtDataInput(i).Text = "" Then

MsgBox "Error: " & txtDataInput(i).Tag & " Must Not Be Empty." &
vbCrLf _

& "Cannot Continue.", vbCritical, "Error: Empty Text"

Exit Sub

End If

Next

If comDep.Text = "" Then _

MsgBox "Error: Student's Department Must Not Be Empty." & vbCrLf &
"Cannot Continue.", _

vbCritical, "Error: Data Missing": Exit Sub

If Len(comDep.Text) > MAX_CHRSZ Then _

MsgBox "Error: Student's Department Is Too Long!!!" & vbCrLf & "Cannot
Continue.", _

vbCritical, "Error: Data Exceeded The Maximum Size": Exit Sub

If stPic.Picture = 0 Then _

MsgBox "Error: Student Picture Must Not Be Empty." & vbCrLf & "Can not
Continue.", _

vbCritical, "Error: Data Missing": Exit Sub


```

Load frmNewStdntNum

Call Student_DBConnect

With frmNewStdntNum
    .txtStName = Me.txtDataInput(TXT_FIRSTN).Text & " " &
txtDataInput(TXT_LASTN).Text
    .txtStDep = Me.comDep.Text
    .txtStNumber = modStudents.GenerateStNum
    .stPic.Picture = Me.stPic.Picture
End With

frmNewStdntNum.Show vbModal

While frmNewStdntNum.Visible = True
    DoEvents
Wend

If IsOK Then
    Call Student_DBConnect
    Call AddStudent(CLng(frmNewStdntNum.txtStNumber), txtDataInput(0),
txtDataInput(1), _
        comDep.Text, CDate(txtDataInput(2).Text), txtDataInput(3).Text,
    _
        (txtDataInput(4).Text), CDbl(txtDataInput(5).Text),
CDate(txtDataInput(6).Text), _
        CDbl(txtDataInput(TXT_FEES).Text))
    Call SaveSTPicture(stPic.Picture, frmNewStdntNum.txtStNumber.Text)

Unload frmNewStdntNum
MsgBox "User Added Successfully!", vbInformation, "Successful"
Unload Me
frmMainDialog.Show

```

```
End If  
End Sub
```

```
Private Sub comDep_Change()
```

```
    If Len(comDep.Text) = 0 Then Exit Sub
```

```
    If IsProcessing = False Then
```

```
        IsProcessing = True
```

```
    Else
```

```
        IsProcessing = False
```

```
        Exit Sub
```

```
    End If
```

```
Dim i%, j%, C$
```

```
C = LCase$(comDep.Text)
```

```
For i = 1 To comDep.ListCount
```

```
    If Left$(C, Len(C)) = LCase$(Left$(comDep.List(i - 1), Len(C))) Then
```

```
        j = Len(C)
```

```
        comDep.Text = comDep.Text & Right$(comDep.List(i - 1),
```

```
Len(comDep.List(i - 1)) - Len(comDep.Text))
```

```
        comDep.SelStart = j
```

```
        comDep.SelLength = Len(comDep) - j
```

```
        IsProcessing = False
```

```
        Exit Sub
```

```
    End If
```

```
Next i
```

```
IsProcessing = False
```

```
End Sub
```

```
Private Sub comDep_KeyDown(KeyCode As Integer, Shift As Integer)
```

```
    If KeyCode = vbKeyReturn Then
```

```
        txtDataInput(TXT_DATEOFBIRTH).SetFocus
```

```

        Call SendKeys("{home}+{end}")
    ElseIf KeyCode = vbKeyDelete Or KeyCode = vbKeyBack Then
        IsProcessing = True
    End If
End Sub

Private Sub comDep_LostFocus()
    comDep.Text = FirstCaps(comDep.Text)
End Sub

Private Sub Form_Load()
    Call System_DBConnect
    txtDataInput(TXT_REGDATE).Text = Format$(Now, "mm/dd/yyyy")
    txtDataInput(TXT_DATEOFBIRTH).Format = ("mm/dd/yyyy")
    txtDataInput(TXT_REGDATE).Enabled = False
End Sub

Private Sub txtDataInput_GotFocus(Index As Integer)
    txtDataInput(Index).SelStart = 0
    txtDataInput(Index).SelLength = Len(txtDataInput(Index).Text)
End Sub

Private Sub txtDataInput_KeyDown(Index As Integer, KeyCode As Integer, Shift
As Integer)
    If KeyCode = vbKeyReturn Then
        Dim iTempVal As Integer

        Select Case Index
            Case Is = TXT_FIRSTN: iTempVal = TXT_LASTN
            Case Is = TXT_LASTN: comDep.SetFocus: Call
SendKeys("{home}+{end}"): Exit Sub
            Case Is = TXT_DATEOFBIRTH: iTempVal = TXT_PLACEOFBIRTH
            Case Is = TXT_PLACEOFBIRTH: iTempVal = TXT_PASSPORTNUM

```

```

    Case Is = TXT_PASSPORTNUM: iTempVal = TXT_BALANCE
    Case Is = TXT_BALANCE: iTempVal = TXT_FEES
    Case Is = TXT_FEES: cmdLoadPic.SetFocus: Exit Sub
End Select

    txtDataInput(iTempVal).SetFocus
    Call SendKeys("{home}+{end}")
End If
End Sub

Private Sub txtDataInput_KeyPress(Index As Integer, KeyAscii As Integer)
    If Index = TXT_BALANCE Or Index = TXT_FEES Then
        KeyAscii = LimitInputToCurrency(KeyAscii)

    ElseIf Index = TXT_PLACEOFBIRTH Then
        KeyAscii = LimitInputToStringWithComma(KeyAscii)

    ElseIf Index = TXT_PASSPORTNUM Then
        KeyAscii = Asc(UCase$(Chr(KeyAscii)))
        KeyAscii = LimitInputToStringLarge(KeyAscii)

    Else
        KeyAscii = LimitInputToStringAllCaseWithSpace(KeyAscii)
    End If
End Sub

Private Sub txtDataInput_KeyUp(Index As Integer, KeyCode As Integer, Shift As Integer)
    If (Index = TXT_FIRSTN Or Index = TXT_LASTN Or
    TXT_PLACEOFBIRTH) Then
        Dim iTempVal As Integer

        iTempVal = txtDataInput(Index).SelStart
    End If
End Sub

```



```

        txtDataInput(Index).Text = FirstCaps(txtDataInput(Index).Text)
        txtDataInput(Index).SelStart = iTempVal
    End If
End Sub

```

Number For New Student:

```

Option Explicit

Private Sub cmdBack_Click()
    frmNewStdnt.IsOK = False
    Unload Me
End Sub

```

```

Private Sub cmdNext_Click()
    frmNewStdnt.IsOK = True
    Me.Hide
End Sub

```

Update For Student:

Search For Student:

```

Option Explicit

Private Const FoundTrue_Color = &H8000&
Private Const FoundFalse_Color = &HFF
Dim sqlStr As String
Dim LastNum As Long

Private Sub cmdBack_Click()
    Call Student_DBClose
    Unload Me
End Sub

```

```
Private Sub cmdNext_Click()
```

```
    Load frmOldStdnt
```

```
    Call frmOldStdnt.FillForm(Val(txtStNumber.Text))
```

```
    Set frmOldStdnt.stPic.Picture = Me.stPic.Picture
```

```
    frmOldStdnt.Show vbModal
```

```
End Sub
```

```
Private Sub cmdSearch_Click()
```

```
    If txtStNumber.Text = "" Then MsgBox "Student Number Cannot Be Empty!",
```

```
    vbCritical, "Error": Exit Sub
```

```
    Call Student_DBConnect
```

```
    sqlStr = "select stfname, stsname, stdep from students where stnum = " _  
    & txtStNumber.Text
```

```
    Call Student_DBExecute(sqlStr)
```

```
    If Student_RSEOF Then
```

```
        txtStat.ForeColor = FoundFalse_Color
```

```
        txtStat.Text = "Student Not Found!!!"
```

```
        txtStName.Text = ""
```

```
        txtStDep.Text = ""
```

```
        cmdNext.Enabled = False
```

```
        Call SetStPic(False)
```

```
        LastNum = 0
```

```
        txtStNumber.SetFocus
```

```
        cmdSearch.Default = True
```

```
        SendKeys "{home}+{end}"
```

```
    Else
```

```
        txtStat.ForeColor = FoundTrue_Color
```

```
        txtStat.Text = "Student Found."
```

```
        txtStName = StdRS![stFName]
```

```
        txtStDep = StdRS![stDep]
```

```
        cmdNext.Enabled = True
```

```

    Call SetStPic(True)
    LastNum = Val(txtStNumber.Text)
    cmdNext.Default = True
    cmdNext.SetFocus
End If
End Sub

Private Sub Form_Load()
    Call Student_DBConnect
End Sub

Private Sub txtStNumber_Change()
    If LastNum = 0 Then
        cmdNext.Enabled = False
        cmdSearch.Default = True
    End If

    If Not Val(txtStNumber.Text) = LastNum Then
        LastNum = 0
        cmdNext.Enabled = False
        cmdSearch.Default = True
    End If
End Sub

Private Sub txtStNumber_KeyPress(KeyAscii As Integer)
    KeyAscii = LimitInputToInteger(KeyAscii)
End Sub

Private Sub SetStPic(BStat As Boolean)
    If BStat Then
        Dim PicFile As String
        PicFile = App.Path & "\STDPictures\" & txtStNumber.Text & ".jpg"
    End If
End Sub

```

```

If FileExist(PicFile) Then
    Set stPic.Picture = LoadPicture(PicFile)
Else
    MsgBox "Could Not Load Student Picture!!!" & vbCrLf & "File Access
Error", vbCritical, "Error: Access"
End If

Else
    stPic.Picture = Nothing
End If

End Sub

```

Update For Student:

```

Option Explicit

Private Const TXT_FIRSTN = 0
Private Const TXT_LASTN = 1
Private Const TXT_DEPARTMENT = 2
Private Const TXT_DATEOFBIRTH = 3
Private Const TXT_PLACEOFBIRTH = 4
Private Const TXT_PASSPORTNUM = 5
Private Const TXT_BALANCE = 6
Private Const TXT_REGDATE = 7
Private Const TXT_STNUMBER = 8

Public Function FillForm(StNumber As Long)
    Dim sqlStr As String

    Call Student_DBConnect
    sqlStr = "select * from students where stnum = " & StNumber
    Call Student_DBExecute(sqlStr)

    If Not Student_RSEOF Then

```



```

Dim sqlStr As String

Call Student_DBConnect
sqlStr = "update students set stbal = " &
Val(txtDataOutput(TXT_BALANCE).Text) _
    & " where stnum = " & txtDataOutput(TXT_STNUMBER).Text
Call Student_DBExecute(sqlStr)

MsgBox "Student Information Updated Successfully!", vbInformation,
"Student Information"
Unload Me
Unload frmOldStdntNum
End If
End Sub

```

```

Private Sub Form_Load()
    Call System_DBConnect
End Sub

```

```

Private Sub txtDataOutput_GotFocus(Index As Integer)
    txtDataOutput(Index).SelStart = 0
    txtDataOutput(Index).SelLength = Len(txtDataOutput(Index).Text)
End Sub

```

```

Private Sub txtDataOutput_KeyPress(Index As Integer, KeyAscii As Integer)
    If Index = TXT_BALANCE Then KeyAscii =
LimitInputToCurrency(KeyAscii)
End Sub

```

Edit For Student:

```

Option Explicit

Public StNumVal As Long

```

Private Const TXT_MIN = 0

Private Const TXT_MAX = 7

Private Const TXT_FIRSTN = 0

Private Const TXT_LASTN = 1

Private Const TXT_DATEOFBIRTH = 2

Private Const TXT_PLACEOFBIRTH = 3

Private Const TXT_PASSPORTNUM = 4

Private Const TXT_BALANCE = 5

Private Const TXT_REGDATE = 6

Private Const TXT_FEES = 7

Private Const MAX_CHRSZ = 64

Private Const COMB_COMP = "Computer Engineering"

Private Const COMB_ELEC = "Electrical Engineering"

Private Const COMB_MECH = "Mechanical Engineering"

Private Const COMB_CIVL = "Civil Engineering"

Dim IsProcessing As Boolean

Private Sub cmdBack_Click()

 Unload Me

End Sub

Private Sub cmdLoadPic_Click()

On Error GoTo ErrHndlr

 With CDG

 .CancelError = True

 .DialogTitle = "Select Student Picture"

```

.Flags = cdlOFNFileMustExist
.Filter = "JPEG Files(*.jpg)|*.jpg;|All Files(*.*)|*.*"
.ShowOpen

If Not .FileName = vbNullString Then
    If FileExist(.FileName) Then _
        stPic.Picture = LoadPicture(.FileName)
        stPic.Tag = .FileName
        stPic.ToolTipText = "Picture Loaded: [" & .FileName & "]"
    Else
        MsgBox "Error: No Picture Loaded!!!", vbCritical + vbApplicationModal,
"Error: File Access Error"
    End If
End With
Exit Sub

ErrHndlr:
End Sub

Private Sub cmdNext_Click()
    Dim i As Integer

    For i = TXT_MIN To TXT_MAX
        If txtDataInput(i).Text = "" Then
            MsgBox "Error: " & txtDataInput(i).Tag & " Must Not Be Empty." &
vbCrLf _
            & "Cannot Continue.", vbCritical, "Error: Empty Text"
            Exit Sub
        End If
    Next

    If comDep.Text = "" Then _

```

```

    MsgBox "Error: Student's Department Must Not Be Empty." & vbCrLf &
"Cannot Continue.", _
    vbCritical, "Error: Data Missing": Exit Sub

If Len(comDep.Text) > MAX_CHRSZ Then _
    MsgBox "Error: Student's Department Is Too Long!!!" & vbCrLf & "Cannot
Continue.", _
    vbCritical, "Error: Data Exceeded The Maximum Limit.": Exit Sub

If stPic.Picture = 0 Then _
    MsgBox "Error: Student Picture Can Not Be Empty." & vbCrLf & "Can not
Continue.", _
    vbCritical, "Error: Data Missing": Exit Sub

Call Student_DBConnect
Call UpdateStudentInfo((StNumVal) _
, txtDataInput(TXT_FIRSTN).Text _
, txtDataInput(TXT_LASTN).Text _
, comDep.Text _
, CDate(txtDataInput(TXT_DATEOFBIRTH).Text) _
, txtDataInput(TXT_PLACEOFBIRTH).Text _
, txtDataInput(TXT_PASSPORTNUM).Text _
, CDbl(txtDataInput(TXT_BALANCE).Text) _
, CDate(txtDataInput(TXT_REGDATE).Text) _
, CDbl(txtDataInput(TXT_FEES).Text))

If ReplaceSTPicture(stPic.Picture, Str(StNumVal)) Then
    MsgBox "Student Updated Successfully!", vbInformation, "Successful"
Else
    MsgBox "Student Picture could not be updated," & vbCrLf & "File Access
Error", vbCritical, "Error"
End If

```



```

Unload Me
Unload frmEditStdntNum
End Sub

Private Sub comDep_Change()

    If Len(comDep.Text) = 0 Then Exit Sub

    If IsProcessing = False Then
        IsProcessing = True
    Else
        IsProcessing = False
        Exit Sub
    End If

    Dim i%, j%, C$
    C = LCase$(comDep.Text)
    For i = 1 To comDep.ListCount
        If Left$(C, Len(C)) = LCase$(Left$(comDep.List(i - 1), Len(C))) Then
            j = Len(C)
            comDep.Text = comDep.Text & Right$(comDep.List(i - 1),
Len(comDep.List(i - 1)) - Len(comDep.Text))
            comDep.SelStart = j
            comDep.SelLength = Len(comDep) - j
            IsProcessing = False
            Exit Sub
        End If
    Next i
    IsProcessing = False
End Sub

Private Sub comDep_KeyDown(KeyCode As Integer, Shift As Integer)

```

```

If KeyCode = vbKeyReturn Then
    txtDataInput(TXT_DATEOFBIRTH).SetFocus
    Call SendKeys("{home}+{end}")
ElseIf KeyCode = vbKeyDelete Or KeyCode = vbKeyBack Then
    IsProcessing = True
End If
End Sub

Private Sub comDep_LostFocus()
    comDep.Text = FirstCaps(comDep.Text)
End Sub

Private Sub Form_Load()
    Call System_DBConnect
End Sub

Private Sub txtDataInput_GotFocus(Index As Integer)
    txtDataInput(Index).SelStart = 0
    txtDataInput(Index).SelLength = Len(txtDataInput(Index).Text)
End Sub

Private Sub txtDataInput_KeyDown(Index As Integer, KeyCode As Integer, Shift
As Integer)
    If KeyCode = vbKeyReturn Then
        Dim iTempVal As Integer

        Select Case Index
            Case Is = TXT_FIRSTN: iTempVal = TXT_LASTN
            Case Is = TXT_LASTN: comDep.SetFocus: Call
SendKeys("{home}+{end}"): Exit Sub
            Case Is = TXT_DATEOFBIRTH: iTempVal = TXT_PLACEOFBIRTH
            Case Is = TXT_PLACEOFBIRTH: iTempVal = TXT_PASSPORTNUM
            Case Is = TXT_PASSPORTNUM: iTempVal = TXT_BALANCE

```

```

        Case Is = TXT_BALANCE: iTempVal = TXT_FEES
        Case Is = TXT_FEES: cmdLoadPic.SetFocus: Exit Sub
    End Select

    txtDataInput(iTempVal).SetFocus
    Call SendKeys("{home}+{end}")
End If
End Sub

Private Sub txtDataInput_KeyPress(Index As Integer, KeyAscii As Integer)
    If Index = TXT_BALANCE Or Index = TXT_FEES Then
        KeyAscii = LimitInputToCurrency(KeyAscii)

    ElseIf Index = TXT_PLACEOFBIRTH Then
        KeyAscii = LimitInputToStringWithComma(KeyAscii)

    ElseIf Index = TXT_PASSPORTNUM Then
        KeyAscii = Asc(UCase$(Chr(KeyAscii)))
        KeyAscii = LimitInputToStringLarge(KeyAscii)

    Else
        KeyAscii = LimitInputToStringAllCaseWithSpace(KeyAscii)
    End If
End Sub

Private Sub txtDataInput_KeyUp(Index As Integer, KeyCode As Integer, Shift As Integer)
    If (Index = TXT_FIRSTN) Or (Index = TXT_LASTN) Or (Index = TXT_PLACEOFBIRTH) Then
        Dim iTempVal As Integer

        iTempVal = txtDataInput(Index).SelStart
        txtDataInput(Index).Text = FirstCaps(txtDataInput(Index).Text)
    End If
End Sub

```

```

        txtDataInput(Index).SelStart = iTempVal
    End If
End Sub

Public Function FillForm(StNumber As Long)
    Dim sqlStr As String

    Call Student_DBConnect
    sqlStr = "select * from students where stnum = " & StNumber
    Call Student_DBExecute(sqlStr)

    If Not Student_REOF Then
        With Me
            .txtDataInput(TXT_FIRSTN).Text = StdRS![stFName]
            .txtDataInput(TXT_LASTN).Text = StdRS![stSName]
            .comDep.Text = StdRS![stDep]
            .txtDataInput(TXT_DATEOFBIRTH).Text = FixDate(StdRS![stBD])
            .txtDataInput(TXT_PLACEOFBIRTH).Text = StdRS![stBP]
            .txtDataInput(TXT_PASSPORTNUM).Text = StdRS![stPNum]
            .txtDataInput(TXT_BALANCE).Text = StdRS![stBal]
            .txtDataInput(TXT_FEES).Text = StdRS![stFees]
            .txtDataInput(TXT_REGDATE).Text = FixDate(StdRS![stRegDate])
            .Caption = .Caption & ": " & CStr(StdRS![stNum])
        End With
    Else
        MsgBox "Error: Database Access Error," & vbCrLf & _
            "Cannot Retrieve Data From Database.", vbCritical, _
            "Error: Database Error"

        Call Student_DBClose
        Unload Me
        Unload frmOldStdntNum
    End If
End Function

```



```
End If
End Function
```

Delete For Student:

```
Option Explicit
```

```
Private Const FoundTrue_Color = &H8000&
```

```
Private Const FoundFalse_Color = &HFF
```

```
Dim sqlStr As String
```

```
Dim LastNum As Long
```

```
Private Sub cmdBack_Click()
```

```
    Call Student_DBClose
```

```
    Unload Me
```

```
End Sub
```

```
Private Sub cmdNext_Click()
```

```
    Call EnableCMDs(False)
```

```
    If MsgBox("Are You Sure You Want To Delete The Selected Student?" & _
```

```
        vbCrLf & "Selected Student will be deleted from the database.", _
```

```
        vbQuestion + vbYesNo, "Delete [" & txtStNumber.Text & "]") = vbYes Then
```

```
        Dim sqlStr As String
```

```
        sqlStr = "delete from students where stnum=" & Val(txtStNumber.Text) & ""
```

```
        Call Student_DBConnect
```

```
        Call Student_DBExecute(sqlStr)
```

```
        MsgBox "Student Deleted Successfully!", vbInformation, "Delete Student"
```

```
        Unload Me
```

```
    Else
```

```
        Call EnableCMDs(True)
```

```
        txtStNumber.SetFocus
```

```
    End If
```

```
End Sub
```

```

Private Sub cmdSearch_Click()
    If txtStNumber.Text = "" Then MsgBox "Student Number Cannot Be Empty!",
vbCritical, "Error": Exit Sub
    Call Student_DBConnect
    sqlStr = "select stfname, stsname, stdep from students where stnum = " & _
txtStNumber.Text
    Call Student_DBExecute(sqlStr)

    If Student_RSEOF Then
        txtStat.ForeColor = FoundFalse_Color
        txtStat.Text = "Student Not Found!!!"
        txtStName.Text = ""
        txtStDep.Text = ""
        cmdNext.Enabled = False
        Call SetStPic(False)
        LastNum = 0
        txtStNumber.SetFocus
        cmdSearch.Default = True
        SendKeys "{home}+{end}"

    Else
        txtStat.ForeColor = FoundTrue_Color
        txtStat.Text = "Student Found."
        txtStName = StdRS![stFName]
        txtStDep = StdRS![stDep]
        cmdNext.Enabled = True
        Call SetStPic(True)
        LastNum = Val(txtStNumber.Text)
        cmdNext.Default = True
        cmdNext.SetFocus
    End If
End Sub

```

```
Private Sub Form_Load()  
    Call Student_DBConnect  
End Sub
```

```
Private Sub txtStNumber_Change()  
    If LastNum = 0 Then  
        cmdNext.Enabled = False  
        cmdSearch.Default = True  
    End If
```

```
    If Not Val(txtStNumber.Text) = LastNum Then  
        LastNum = 0  
        cmdNext.Enabled = False  
        cmdSearch.Default = True  
    End If  
End Sub
```

```
Private Sub txtStNumber_KeyPress(KeyAscii As Integer)  
    KeyAscii = LimitInputToInteger(KeyAscii)  
End Sub
```

```
Private Sub SetStPic(BStat As Boolean)  
    If BStat Then  
        Dim PicFile As String  
        PicFile = App.Path & "\STDPictures\" & txtStNumber.Text & ".jpg"  
  
        If FileExist(PicFile) Then  
            Set stPic.Picture = LoadPicture(PicFile)  
        Else  
            MsgBox "Could Not Load Student Picture!!!" & vbCrLf & "File Access  
Error", vbCritical, "Error: Access"  
        End If
```

```

Else
    stPic.Picture = Nothing
End If
End Sub

Private Sub EnableCMDs(BStat As Boolean)
    cmdSearch.Enabled = BStat
    txtStat.Enabled = BStat
    txtStName.Enabled = BStat
    txtStDep.Enabled = BStat
    stPic.Enabled = BStat
End Sub

```

Debt And Payment:

Option Explicit

```

Private Const TXT_NUMB = 0
Private Const TXT_NAME = 1
Private Const TXT_DEPT = 2
Private Const TXT_BLNC = 3
Private Const TXT_FEES = 4
Private Const TXT_CRDT = 5
Private Sub cmdExit_Click()
    Unload Me
End Sub

```

```

Public Sub FillForm(lstdNumber As Long)
Dim strSQL As String
    Call Student_DBConnect

```

```

    strSQL = "select stnum, stfname, stsname, stdep, stBal, stfees from students
where stnum=" & CStr(lstdNumber) & ""

```



```

Call Student_DBExecute(strSQL)
DoEvents

txtDataInput(TXT_NUMB).Text = lstdNumber
txtDataInput(TXT_NAME).Text = StdRS![stFName] & " " &
StdRS![stSName]
txtDataInput(TXT_DEPT).Text = StdRS![stDep]
txtDataInput(TXT_BLNC).Text = StdRS![stBal]
txtDataInput(TXT_FEES).Text = StdRS![stFees]
txtDataInput(TXT_CRDT).Text = MakeCredit(StdRS![stBal], StdRS![stFees])

End Sub

Private Function MakeCredit(dblBal As Double, dblFees As Double)
Dim A, B, C, F As Double

B = dblBal
F = dblFees

C = B - F

If C >= 0 Then
    MakeCredit = C
Else

    A = C * 3 / 100
    B = C + A

    MakeCredit = B
End If

```

End Function

About

Option Explicit

Private Sub Form_Click()

 Unload Me

End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)

 Unload Me

End Sub

Private Sub Form_KeyPress(KeyAscii As Integer)

 Unload Me

End Sub

Private Sub Imgscr1_Click()

 Unload Me

End Sub

Private Sub tmrMain_Timer()

 Imgscr1.Top = Imgscr1.Top - 10

 If Imgscr1.Top <= -15000 Then tmrMain.Enabled = False

End Sub

Mod Function:

Option Explicit

Public Function LimitInputToInteger(AsciiCode As Integer) As Integer

```

    If InStr("0123456789", Chr(AsciiCode)) = 0 And Not AsciiCode = vbKeyBack
And Not AsciiCode = vbKeyDelete Then LimitInputToInteger = 0 Else
LimitInputToInteger = AsciiCode
End Function

```

```

Public Function LimitInputToCurrency(AsciiCode As Integer) As Integer
    If InStr("-.0123456789", Chr(AsciiCode)) = 0 And Not AsciiCode =
vbKeyBack And Not AsciiCode = vbKeyDelete Then LimitInputToCurrency = 0
Else LimitInputToCurrency = AsciiCode
End Function

```

```

Public Function LimitInputToStringSmall(AsciiCode As Integer) As Integer
    If InStr(" 0123456789abcdefghijklmnopqrstuvwxyz", Chr(AsciiCode)) = 0 And
Not AsciiCode = vbKeyBack And Not AsciiCode = vbKeyDelete Then
LimitInputToStringSmall = 0 Else LimitInputToStringSmall = AsciiCode
End Function

```

```

Public Function LimitInputToStringLarge(AsciiCode As Integer) As Integer
    If InStr("0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ",
Chr(AsciiCode)) = 0 And Not AsciiCode = vbKeyBack And Not AsciiCode =
vbKeyDelete Then LimitInputToStringLarge = 0 Else LimitInputToStringLarge =
AsciiCode
End Function

```

```

Public Function LimitInputToStringAllCase(AsciiCode As Integer) As Integer
    If
InStr("0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
UVWXYZ", Chr(AsciiCode)) = 0 And Not AsciiCode = vbKeyBack And Not
AsciiCode = vbKeyDelete Then LimitInputToStringAllCase = 0 Else
LimitInputToStringAllCase = AsciiCode
End Function

```

```
Public Function LimitInputToStringAllCaseWithSpace(AsciiCode As Integer) As Integer
```

```
    If InStr("0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ", Chr(AsciiCode)) = 0 And Not AsciiCode = vbKeyBack And Not AsciiCode = vbKeyDelete Then LimitInputToStringAllCaseWithSpace = 0 Else  
    LimitInputToStringAllCaseWithSpace = AsciiCode  
End Function
```

```
Public Function LimitInputToStringWithComma(AsciiCode As Integer) As Integer
```

```
    If InStr(",0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ", Chr(AsciiCode)) = 0 And Not AsciiCode = vbKeyBack And Not AsciiCode = vbKeyDelete Then LimitInputToStringWithComma = 0 Else  
    LimitInputToStringWithComma = AsciiCode  
End Function
```

```
Public Function FileExist(FileName As String) As Boolean
```

```
    On Error GoTo NotFound  
    Dim X&  
    X = FileLen(FileName)  
    If X >= 0 Then FileExist = True: Exit Function
```

```
NotFound:
```

```
    If Err.Number = 53 Then FileExist = False: Exit Function  
End Function
```

```
Public Function FirstCaps(TheText As String) As String
```

```
    If (Len(TheText) < 2) Then  
        FirstCaps = UCase$(TheText)  
    Else  
        If InStr(TheText, " ") = 0 Then
```



```

        FirstCaps = (UCase(Left$(TheText, 1)) & Right(TheText, Len(TheText) -
1))
    Else
        Dim C$, D$, X$
        C = TheText & " "
        While InStr(C, " ") > 0
            D = Left$(C, InStr(C, " "))
            X = X & UCase$(Left$(C, 1)) & Right$(D, Len(D) - 1)
            C = Right$(C, Len(C) - Len(D))
            DoEvents
        Wend
        X = Left$(X, Len(X) - 1)
        FirstCaps = X
    End If
End If
End Function

```

```

Public Function InSpaces(TheSt As String) As Integer
    Dim j As Integer, C As String
    C = TheSt
    While InStr(C, " ") > 0
        C = Mid(C, InStr(C, " ") + 1, Len(C) - InStr(C, " "))
        j = j + 1
    Wend
    InSpaces = j
End Function

```

```

Public Sub SaveSTPicture(PIC As IPictureDisp, STDNum As String)
On Error GoTo SystemErr
    Dim StPicPath As String

    StPicPath = App.Path & "\STDPictures\"
    FixPath (StPicPath)

```

```
StPicPath = App.Path & "\STDPictures\" & STDNum & ".jpg"
```

```
Call SavePicture(PIC, StPicPath)
```

```
Exit Sub
```

```
SystemErr:
```

```
MsgBox "Error: " & Hex(Err.Number) & vbCrLf & Err.Description, vbCritical,  
"Error: " & Err.Number
```

```
Err.Clear
```

```
End Sub
```

```
Public Function ReplaceSTPicture(PIC As IPictureDisp, STDNum As String) As  
Boolean
```

```
On Error GoTo SystemErr
```

```
Dim StPicPath As String, iErr As Integer
```

```
iErr = 0
```

```
StPicPath = App.Path & "\STDPictures\"
```

```
FixPath (StPicPath)
```

```
StPicPath = App.Path & "\STDPictures\" & Trim$(STDNum) & ".jpg"
```

```
Call SetAttr(StPicPath, vbNormal)
```

```
Call Kill(StPicPath)
```

```
DoEvents
```

```
Call SavePicture(PIC, StPicPath)
```

```
ReplaceSTPicture = True
```

```
Exit Function
```

```
SystemErr:
```

```
iErr = iErr + 1
```

```
Err.Clear
```

```
If iErr >= 3 Then
```

```

        ReplaceSTPicture = False: Exit Function
    Else
        Resume Next
    End If
End Function

Public Function LoadSTPicture(ByRef PIC As PictureBox, STDNum As String)
    As Boolean
    On Error GoTo SystemErr
        Dim StPicPath As String

        StPicPath = App.Path & "\\STDPictures\\"
        FixPath (StPicPath)
        StPicPath = App.Path & "\\STDPictures\\" & STDNum & ".jpg"

        If FileExist(StPicPath) Then
            Set PIC.Picture = LoadPicture(StPicPath)
            LoadSTPicture = True
            Exit Function
        Else
            LoadSTPicture = False
            Exit Function
        End If

    SystemErr:
        MsgBox "Error: " & Hex(Err.Number) & vbCrLf & Err.Description, vbCritical,
        "Error: " & Err.Number
        LoadSTPicture = False
        Err.Clear
        Exit Function
    End Function

```

```

        ReplaceSTPicture = False: Exit Function
    Else
        Resume Next
    End If
End Function

Public Function LoadSTPicture(ByRef PIC As PictureBox, STDNum As String)
    As Boolean
    On Error GoTo SystemErr
        Dim StPicPath As String

        StPicPath = App.Path & "\\STDPictures\\"
        FixPath (StPicPath)
        StPicPath = App.Path & "\\STDPictures\\" & STDNum & ".jpg"

        If FileExist(StPicPath) Then
            Set PIC.Picture = LoadPicture(StPicPath)
            LoadSTPicture = True
            Exit Function
        Else
            LoadSTPicture = False
            Exit Function
        End If

    SystemErr:
        MsgBox "Error: " & Hex(Err.Number) & vbCrLf & Err.Description, vbCritical,
        "Error: " & Err.Number
        LoadSTPicture = False
        Err.Clear
        Exit Function
    End Function

```



```
Public Sub FixPath(PathStr$)
```

```
On Error GoTo PathNotExist
```

```
    ChDir (PathStr)
```

```
    ChDir ("C:\")
```

```
    Exit Sub
```

```
PathNotExist:
```

```
    MkDir (PathStr)
```

```
    Err.Clear
```

```
    Exit Sub
```

```
End Sub
```

'This Function Converts The Date Into A Valid String

'For example the date 1/1/2006 is invalid,

'we cannot use such a date in a MaskedEdit textbox

'The valid string of it is 01/01/2006

'Now we may have four cases, ###/###/####

' ###/####

' ##/##/####

' #/#/####

'And this function makes the four cases a valid string.

```
Public Function FixDate(TheDate As Date) As String
```

```
    Dim D$, M$, Y$, T$, F$, i%
```

```
    T = CStr(TheDate)
```

'Format: 'mm/dd/yyyy'

'Incase Of Mismatch Errors:

```
    T = Replace(T, "\", "/")
```

'Remove The Year: ("####"+"/"=5Chars)

```
    Y = Right$(T, 4)
```

T = Left\$(T, Len(T) - 5)

'Month Cases:

M = Split(T, "/")(0)

M = If(Len(M) = 1, "0" + M, M)

'Day Cases:

D = Split(T, "/")(1)

D = If(Len(D) = 1, "0" + D, D)

'Fix The Date:

F = M & "/" & D & "/" & Y

'Return The Date

FixDate = F

End Function

Mod Main:

Option Explicit

Sub Main()

On Error GoTo X

Load frmLogIn

frmLogIn.Show

Exit Sub

X:

MsgBox "Click [Ok] To Continue.", vbInformation

Err.Clear

Resume Next

End Sub

Public Sub ShowDialog(prvFrmName As Form, NewfrmName As Form)

```
    prvFrmName.Hide  
    NewfrmName.Show  
End Sub
```

```
Public Sub GoBack(frmName As Form, prvFrm As Form)  
    frmName.Hide  
    prvFrm.Show  
End Sub
```

Mod Restriction:

```
Option Explicit
```

```
Global Restrictions(0 To 13) As Boolean  
Global tmpRestrictions(0 To 13) As Boolean
```

```
Private lRestValues(0 To 13) As Long
```

```
Private Const REST_MAX = 4096  
Private Const REST_MIN = 0  
Private Const REST_SUM = 8191
```

```
Private Const REST_START = 0  
Private Const REST_END = 13
```

```
Public Enum RESTR_TYP
```

```
    REST_LGIN = 0
```

```
    REST_VEWA = 1
```

```
    REST_ADDU = 2
```

```
    REST_EDTU = 3
```

```
    REST_DELU = 4
```

```

REST_REGN = 5
REST_REGO = 6
REST_EDTS = 7
REST_DELS = 8
REST_DBTS = 9

REST_LOCK = 10
REST_MINM = 11
REST_EXIT = 12
End Enum

'
'
'
'

Public Function GetRestriction(ByVal CodeNum As RESTR_TYP) As Boolean
    GetRestriction = Restrictions(CodeNum)
End Function

Public Sub SetRestriction(ByVal strUserName As String)
    Dim lRestr As Long, tmpSql As String

    If Not strUserName = "" Then
        System_DBConnect
        tmpSql = "select restr from users where username='" & strUserName & "'"

        System_DBExecute (tmpSql)

        If IsNull(SysRS![restr]) Then
            lRestr = 0
        Else
            lRestr = SysRS![restr]
        End If
    End If
End Sub

```

```

End If

System_DBClose
End If

Call SetRestValues
Call ParseRestriction(lRestr)
End Sub

Private Sub ParseRestriction(ByVal lRstr As Long)
Dim i As Integer, lRestVal As Long, lRestSum As Long

lRestVal = lRstr
lRestSum = REST_SUM

If lRestVal = REST_SUM Then _
    Call SetAdmin: Exit Sub

For i = REST_END To REST_START Step -1

    If lRestVal >= lRestSum Then          'TRUE: USER HAS ACCESS
        lRestVal = lRestVal - lRestValues(i + 1)
        Restrictions(i) = False          'SET PROHIBITED
RESTRICTION=FALSE
    Else
        Restrictions(i) = True           'FALSE: USER ACCESS WILL BE
DENIED!
    End If                               'SET PROHIBITED RESTRICTION=TRUE

    lRestSum = lRestSum / 2
Next i
End Sub

```



```

Public Sub ParsetmpRestriction(ByVal lRstr As Long)
Dim i As Integer, lRestVal As Long, lRestSum As Long

lRestVal = lRstr
lRestSum = REST_SUM

SetRestValues

If lRestVal = REST_SUM Then _
    Call SettmpAdmin: Exit Sub

For i = REST_END To REST_START Step -1

    If lRestVal >= lRestSum Then          'TRUE: USER HAS ACCESS
        lRestVal = lRestVal - lRestValues(i + 1)
        tmpRestrictions(i) = False      'SET PROHIBITED
RESTRICTION=FALSE
    Else
        tmpRestrictions(i) = True       'FALSE: USER ACCESS WILL BE
DENIED!
    End If                               'SET PROHIBITED RESTRICTION=TRUE

    lRestSum = lRestSum / 2
Next i
End Sub

Public Sub SetLastAccess(strUserName As String)
Dim strLastAcc As String, strSQL As String

strLastAcc = FixDate(Format$(Now, "MM/DD/YYYY"))

System_DBConnect

```

```
strSQL = "update users set lastaccess='" & strLastAcc & "' where username='"  
& strUserName & "'"
```

```
Call System_DBExecute(strSQL)
```

```
End Sub
```

```
Private Sub SetRestValues()
```

```
Dim j As Integer, s As Long
```

```
s = 1
```

```
lRestValues(0) = 0
```

```
For j = 1 To 13
```

```
lRestValues(j) = s
```

```
s = s * 2
```

```
Next j
```

```
End Sub
```

```
Private Sub SetAdmin()
```

```
Dim k As Integer
```

```
For k = 0 To 13
```

```
Restrictions(k) = False
```

```
Next k
```

```
End Sub
```

```
Private Sub SettmpAdmin()
```

```
Dim k As Integer
```

```
For k = 0 To 13
```

```
tmpRestrictions(k) = False
```

```
Next k  
End Sub
```

Mod Student:

```
Option Explicit
```

```
Dim StdDBConn As ADODB.Connection  
Public StdRS As ADODB.Recordset
```

```
Dim ConnStr As String  
Dim sqlStr As String
```

```
Public Sub Student_DBConnect()  
Set StdDBConn = New ADODB.Connection  
StdDBConn.CursorLocation = adUseServer  
ConnStr = "provider=Microsoft.jet.oledb.3.51; Data Source=" & App.Path &  
"Data\Students.mdb"  
StdDBConn.Open ConnStr  
End Sub
```

```
Public Sub AddStudent(stNum&, stFName$, stSName$, stDep$, stBD As Date, _  
stBP$, stPNum$, stBal As Double, stRegDate As Date, stFees As  
Double)  
sqlStr = "insert into students(stnum, stfname, stsname, stdep, stbd, stbp,  
stpnum, stbal, stregdate, stfees) values("  
sqlStr = sqlStr & "'" & stNum & ","  
sqlStr = sqlStr & "'" & stFName & ","  
sqlStr = sqlStr & "'" & stSName & ","  
sqlStr = sqlStr & "'" & stDep & ","  
sqlStr = sqlStr & "'" & stBD & ","  
sqlStr = sqlStr & "'" & stBP & ","  
sqlStr = sqlStr & "'" & stPNum & ","
```

```

sqlStr = sqlStr & "" & stBal & ","
sqlStr = sqlStr & "" & stRegDate & ","
sqlStr = sqlStr & "" & stFees & ")"

```

```

Call StdDBConn.Execute(sqlStr)

```

```

End Sub

```

```

Public Sub UpdateStudentInfo(stNum&, stFName$, stSName$, stDep$, stBD As
Date, _

```

```

    stBP$, stPNum$, stBal As Double, stRegDate As Date, stFees As

```

```

Double)

```

```

    sqlStr = "update students set stfname='" & stFName & "'" & _
        ", stlname='" & stSName & "'" & _
        ", stdep='" & stDep & "'" & _
        ", stbd='" & stBD & "'" & _
        ", stbp='" & stBP & "'" & _
        ", stpnum='" & stPNum & "'" & _
        ", stbal=" & stBal & "" & _
        ", stregdate='" & stRegDate & "'" & _
        ", stfees=" & stFees & _
        " where stnum=" & stNum & ""

```

```

    Call StdDBConn.Execute(sqlStr)

```

```

End Sub

```

```

Public Function GenerateStNum() As Long

```

```

    sqlStr = "select max(stnum) as lstnum from students"

```

```

    Set StdRS = New ADODB.Recordset

```

```

    Set StdRS = StdDBConn.Execute(sqlStr)

```

```

    If (Not StdRS.EOF = True) And (Not StdRS![lstnum] = "") Then

```

```

        If (Right$(Str(StdRS![lstnum]), 4) = "9999") Then

```

```

        MsgBox "Student Numbers Are Full!!!" & vbCrLf & "Next Year Code
Will Be Used!", _
        vbExclamation, "Error"
        GenerateStNum = CLng(Format$(Now, "yyyy") & "0001")
        Exit Function
    Else
        GenerateStNum = CLng((StdRS![lstnum]) + 1)
    End If
Else
    GenerateStNum = CLng(Format$(Now, "yyyy") & "0001")
End If
End Function

```

```

Public Sub Student_DBClose()
    On Error Resume Next
    StdDBConn.Close
End Sub

```

```

Public Sub Student_RSClose()
    StdRS.Close
End Sub

```

```

Public Sub Student_DBExecute(TheStr As String)
    Set StdRS = New ADODB.Recordset
    Set StdRS = StdDBConn.Execute(TheStr)
End Sub

```

```

Public Function Student_RSEOF() As Boolean
    Student_RSEOF = StdRS.EOF
End Function

```

Mod System:

Option Explicit

Dim SysDBConn As ADODB.Connection

Public SysRS As ADODB.Recordset

Dim ConnStr As String

Dim sqlStr As String

Public Sub System_DBConnect()

Set SysDBConn = New ADODB.Connection

SysDBConn.CursorLocation = adUseServer

ConnStr = "provider=Microsoft.jet.oledb.3.51; Data Source=" & App.Path &

"\Data\System.mdb"

SysDBConn.Open ConnStr

End Sub

Public Sub System_DBClose()

SysDBConn.Close

End Sub

Public Sub System_FindUserName(UsernameStr As String)

Set SysRS = New ADODB.Recordset

sqlStr = "select password from users where username=" & UsernameStr & ""

Set SysRS = SysDBConn.Execute(sqlStr)

End Sub

Public Function System_REOF() As Boolean

System_REOF = SysRS.EOF

End Function

Public Function System_GetPassword()

System_GetPassword = SysRS![Password]

End Function

```
Public Sub System_RSClose()
```

```
    SysRS.Close
```

```
End Sub
```

```
Public Sub System_DBExecute(TheSqlStr As String)
```

```
    Set SysRS = New ADODB.Recordset
```

```
    Set SysRS = SysDBConn.Execute(TheSqlStr)
```

```
End Sub
```