NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

Using Computer Programming for Office Reception Desk in Prisons

Graduation Project
COM 400

Student : Hatice Özsaltık
(20030076)

Supervisor : Mr Ümit SOYER

Nicosia – 2008

ACKNOWLEDGEMENTS

First of all, I would like to thank my supervisor Mr. Ümit SOYER for his invaluable advice and belief in my work and myself over the course of this graduation project.

Second, I want to say thanks to my family for their endless support. I will never forget the things that my father Mr Kemal Özsaltık and my mother Mrs Pembe Özsaltık did for me during my educational life .I thank my fiance Mr Idris Oral for his constant encouragement, support and patience during the preparation of this project.

Finally, I would also thank Mr Samir Mirghani for his help in Visual Basic programming and helpful ideas.

ABSTRACT

There is not any computer program in North Cyprus prisons for making the visitors and prisoners controlling. It must be, because it is very important for security. This project consists of make the program for controlling visitor's and prisoner's in prisons. We implemented this project using Visual Basic 6.0. The aim of this project is the dependable entering, exiting and controlling of visitors and prisoners in the prison. There are lots of advantages of this program for prisons. Only administrators and users will be using this program. Each administrator and user must have a username and a password for accessing the program functions. The implemented program allows users to add new visitor information, search visitor records, and find information about visitors and prisoners. It provides information such as the visitor's name, contact information about which the visitor visited, the reason of the visit, the date and time of the visit. The system can generate reports about both the prisoner's records and the visitor's records. When the user enters the visit information will be required to take and attach a photo of the visitor to the record by using web cam. This program will be helpful to users and administrators in finding information about visitors and prisoners. This information may then help prisoner officials and control illegal activities.

TABLE OF CONTENTS

INTRODUCTION

In this project concerned a program to visitors and prisoners automation system for prison using Visual Basic Language. The aim of this project is the dependable entering, exiting and controlling visitors and prisoners in the prison. There are lost of advantages of this program for prisons. Only administrators and users will be using this program. Administrators and users must have username and password for using access to the program functions. The program's main functions are; users can add new visitor information, to search the visitor's records, find information about visitors and prisoners. It shows information such as the visitor's name, surname, ID number, contact number, to who they came, the reason of the visit, date and finally their in and out going time.

Users can take to generate printable reports for both the prisoner's records and the visitor's records. When the user enters the visit information will be required to take and attach a photo of the visitor to the record by using web cam. This program will be helpful to users and administrators in finding information about visitors and prisoners. This information may then help prisoner officials and control illegal activities.

We implemented this project using Visual Basic 6.0, allow developers to target Windows, Web, and mobile devices. As with all languages targeting the Microsoft .NET Framework, programs written in Visual Basic benefit from security and language interoperability.

Microsoft Office Access, is a relational database management system from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software development tools.

This project consists of the introduction, five chapters and conclusion.

First four chapters give a basic introduction about the visual basic, Microsoft access, SQL and ActiveX Data Objects (ADO). Last chapter give description about project.

In Chapter 1, the visual basic programming language will be explained. After brief introduction about visual studio, framework .net, types, architectures and security and VB handling errors will be described.

In Chapter 2, the application Microsoft access and database will be explained. These include both file extensions and versions. The applications of database with general models and algorithms are also discussed.

In Chapter 3, the general information about SQL, SQL table basics, creating and inserting table and updating, deleting records will be discussed.

In Chapter 4, presents the property and fonksiyons the ADO with existing technologies and new features. In this chapter will describe the meaning of the ADO connection objects, ADO Recordsets and ADO Attributes Property.

In Chapter 5, the description about project will be presented with login, main windows and their functions and forms.

In Conclusion, the general results of this work are explained.

CHAPTER 1
VISAUL BASIC

1.1 Overview

In this chapter, the general information about visual basic programming language, visual studio and framework .NET, types, architectures and security and Visual Basic handling errors will be explained briefly.

1.2 What is Programming Language?

People express themselves using a language with many words. Computers use a simple language consisting of only 1s and 0s, with a 1 meaning "on" and a 0 meaning "off." Trying to talk to a computer in its own language would be like trying to talk to your friends. A programming language acts as a translator between you and the computer. Rather than learning the computer's native language (known as machine language), you can use a programming language to instruct the computer in a way that is easier to learn and understand.

A specialized program known as a compiler takes the instructions written in the programming language and converts them to machine language. This means that as a Visual Basic programmer, you don't need to understand what the computer is doing or how it does it, you just need to understand how the Visual Basic programming language works.

1.3 Introduction to the Visual Basic Programming Language

Visual Basic allows developers to target Windows, Web, and mobile devices. As with all languages targeting the Microsoft .NET Framework, programs written in Visual Basic benefit from security and language interoperability.

This version of Visual Basic brings back support for Edit and Continue and has new features for rapid application development. One of these features, called My, provides quick access to common tasks provided by the .NET Framework, as well as information and default object instances that are related to the application and its run-time environment.

New language features include loop continuation, guaranteed resource disposal, operator overloading, generic types, and custom events.

"Visual" refers to the method used to create what the user sees the graphical user interface, or GUI. "Basic" refers to the BASIC (Beginners All-Purpose Symbolic Instruction Code) programming language, a language used by more programmers than any other language in the history of computing

Inside the Visual Basic Language; in many ways, Visual Basic is a lot like the language that you use every day. When you speak or write, you use different types of words, such as nouns or verbs, which define how they are used. Visual Basic also has different types of words known as programming elements that define how they are used to write programs.

Programming elements in Visual Basic include statements, declarations, methods, operators, and keywords. Written and spoken language also has rules, or syntax, that defines the order of words in a sentence. The language you write and speak also has structure: for example, a book has chapters with paragraphs that contain sentences. Programs written in Visual Basic also have a structure: modules are like chapters. Visual Basic also has syntax at first it may look strange, but it is actually very simple. For example, to state "The maximum speed of my car is 120", you would write:

```
Car.Speed.Maximum = 120
```

1.4 How to Install Visual Basic

Visual basic 6.0 comes as a part of the Microsoft Visual Studio 6.0, and by obtaining the Microsoft Visual Studio 6.0 CDs which is 5 CDs visual basic can be installed as a part of the full installation or just setting it up alone in the system. By running the setup you can follow the steps guiding you to your need.

1.5 How Visual Basic Programming Works

On its own, a computer isn't very smart. A computer is essentially just a big bunch of tiny electronic switches that are either on or off. By setting different combinations of these switches, you can make the computer do something, for example, display something on the screen or make a sound. That's what programming is at its most basic telling a computer what to do. Of course, understanding which combination of switches make the computer will do what you want would be a daunting task that's where programming languages come in. Inside the Visual Basic Language; in many ways, Visual Basic is a lot like the language that you use every day. When you speak or write, you use different types of words, such as nouns or verbs, which define how they are used. Visual Basic also has different types of words known as programming elements that define how they are used to write programs.

1.5.1 Representing Words, Numbers, and Values with Variables in VB

Variables are an important concept in computer programming. A variable is a letter or name that can store a value. When you create computer programs, you can use variables to store numbers, such as the height of a building, or words, such as a person's name. Simply put, you can use variables to represent any kind of information your program needs.

There are steps to using a variable:

- **Declare the variable:** Tell the program the name and kinds of variable you want to use. You declare a variable using the **Dim** and **As** keywords.
  Dim aNumber **As** Integer, This line of code tells the program that you want to use a variable named aNumber, and that you want it to be a variable that stores whole numbers (the Integer data type). Because aNumber is an Integer, it can store only whole numbers. If you had wanted to store 42.5, for example, you would have used the Double data type (Dim aDouble As Double). And if you wanted to store a word, you'd use a data type called a String (Dim aName As String).

- **Assign the variable:** Give the variable a value to hold. You assign a value to your variable with the = sign, which is sometimes called the **assignment operator**, as shown in the following example: aNumber = 42 This line of code takes the value 42 and stores it in the previously declared variable named aNumber.

- **Use the variable:** Retrieve the value held in the variable and use it in your program. You can declare a variable on one line of code, and then later assign the value on another line. This can result in an error if you try to use the variable before assigning it a value. For that reason, it is a better idea to declare and assign variables on a single line. Even if you don't yet know what value the variable will hold, you can assign a default value. The code for declaring and assigning the same variables shown earlier would look like the following: Dim aName As String = "default string" Dim YesOrNo As Boolean = True

1.5.2 Words and Text: Using String Variables to Organize Words in VB

A string is any series of text characters, such as letters, numbers, special characters, and spaces. Strings can be human-readable phrases or sentences. String variables are created just as other variables: by first declaring the variable and assigning it a value, as shown below:

```
Dim aString As String = "This is a string"
```

When assigning actual text (also called a string literal) to a String variable, the text must be enclosed in quotation marks (""). You can also use the = character to assign one String variable to another String variable. You can use the ampersand (&) character to sequentially combine two or more strings into a new string, as shown below:

```
Dim aString As String = "Across the Wide"
Dim bString As String = "Missouri"
Dim cString As String = ""
cString = aString & bString
```

The previous example declares three String variables and respectively assigns "Across the Wide" and "Missouri" to the first two, and then assigns the combined values of the first two to the third variable. The value is "Across the WideMissouri" because there is no space at the end of aString or at the beginning of bString.

The two strings are simply joined together. If you want to add spaces or anything else between two strings, you must do so with a string literal, such as " ", as shown below:

```
Dim aString As String = "Across the Wide"
Dim bString As String = "Missouri"
Dim cString As String = ""
cString = aString & " " & bString
```

The text contained in cString now reads as "Across the Wide Missouri".

1.5.3 Arrays: Variables That Represent More Than One Value

Variables are used to store different types of data for use by your program. There is another type of variable called an array that provides a convenient way to store several values of the same type.

For example, suppose you were writing a program for a baseball team and you wanted to store the names of all of the players on the field. You could create nine separate string variables, one for each player, or you could declare an array variable that looks something like the code below:

```
Dim players() As String
```

You declare an array variable by putting parentheses after the variable name. If you know how many values you need to store, you can also specify the size of the array in the declaration as follows.

```
Dim players(9) As String
```

As with other types of values, you need to assign values to arrays. To do so, you refer to the element number as part of the assignment, as shown below:

```
players(0) = "hatice"
players(3) = "idris"
```

In the above code, the value hatice is assigned to the first element of the array (element 0) and the value idris is assigned to the fourth element (element 3).

As with other types of values, you can declare and assign values to an array on a single line as follows:

```
Dim players() As Integer = {1, 2, 3, 4, 5, 6, 7, 8, 9}
```

Retrieving Values from Arrays, Just as you use numbers to specify an item's position in an array; you use the element number to specify which value you want to retrieve.

```
Dim AtBat As String
AtBat = players(3)
```

The above code retrieves the fourth element of the array and assigns it to the string variable AtBat.

1.5.4 What To Do When Something Goes Wrong: Handling Errors

In this topic, you will learn how to create basic error-handling code for your programs. Even the best designed programs sometimes encounter errors. Some errors are defects in your code that can be found and corrected. Other errors are a natural consequence of the program; for example, your program might attempt to open a file that is already in use. In cases like this, errors can be predicted but not prevented. As a programmer, it is your job to predict these errors and help your program deal with them.

- **Run-Time Errors:** An error that occurs while a program is running is called a run-time error. A run-time error occurs when a program tries to do something it wasn't designed to do. For example, if your program attempts to perform an illegal operation, such as converting a non-numeric string to a numeric value, a run-time error occurs. When a run-time error occurs, the program issues an exception, which deals with errors by looking for code within the program to handle the error. If no such code is found, the program stops and has to be restarted. Because this can lead to the loss of data, it is wise to create error-handling code wherever you anticipate errors occurring.
- **The Try...Catch...Finally block:** You can use the Try...Catch...Finally block to handle run-time errors in your code. You can Try a segment of code if an exception is issued by that code, it jumps to the Catch block, and then the code in the Catch

block is executed. After that code has finished, any code in the finally block is executed. The entire Try...Catch...Finally block is closed by an End Try statement. The following example illustrates how each block is used:

```
Try
' Code here attempts to do something.
Catch
 ' If an error occurs, code here will be run.
Finally
 ' Code in this block will always be run.
End Try
```

First, the code in the Try block is executed. If it runs without error, the program skips the Catch block and runs the code in the Finally block. If an error does occur in the Try block, execution immediately jumps to the Catch block, and the code there is run; then the code in the Finally block is run.

1.6 Introducing Visual Studio

Visual Studio is a complete set of development tools for building ASP.NET Web applications, XML Web Services, desktop applications, and mobile applications. Visual Basic, Visual C++, Visual C#, and Visual J# all use the same integrated development environment (IDE), which allows them to share tools and facilitates in the creation of mixed-language solutions. In addition, these languages leverage the functionality of the .NET Framework, which provides access to key technologies that simplify the development of ASP Web applications and XML Web Services.

1.6.1 Visual Studio Highlights

This section contains information about some of the latest tools and technologies available in this release of Visual Studio.

- **Visual Studio Tools for Office:** Microsoft Visual Studio 2005 Tools for the Microsoft Office System can help you create solutions by extending Word 2003 documents and Excel 2003 workbooks using Visual Basic and Visual C#.

- **Visual Web Developer:** Visual Studio features a new Web page designer named Visual Web Developer that includes many enhancements for creating and editing ASP.NET Web pages and HTML pages. It provides a simpler, faster way to create and maintain Web sites as local folders, in Internet Information Services (IIS), or on an FTP or SharePoint server.
- **Web Forms:** Web Forms render themselves as browser-compatible HTML and script, which allows any browser on any platform to view the pages. Using Web Forms, you create Web pages by dragging and dropping controls onto the designer and then adding code, similar to the way that you create Visual Basic forms.
- **Windows Forms:** Windows Forms is for creating Microsoft Windows applications on the .NET Framework. This framework provides a clear, object-oriented, extensible set of classes that enables you to develop rich Windows applications.
- **XML Web Services:** In Visual Studio, you can quickly create and include XML Web Services using Visual Basic, Visual C#, JScript, or ATL Server.

1.6.2 About Visual Studio Team System

Visual Studio Team System is a productive, integrated, and extensible software development life-cycle tools platform that helps software teams by improving communication and collaboration throughout the software development process. It consists of the following:

- **Team Foundation:** is an extensible team collaboration server that provides work item tracking, source control, reporting, and process guidance.
- **Team Edition for Architects:** is a set of integrated application design tools for service-oriented development.
- **Team Edition for Developers:** provides code quality and performance tools that enable teams to build reliable, mission-critical services and applications.
- **Team Edition for Testers:** provides advanced load testing tools that enable teams to verify the performance of applications before deployment.

1.6.3 Description The .NET Framework

The .NET Framework is a multi-language environment for building, deploying, and running XML Web Services and applications. It consists of three main parts:

- **Common Language Runtime**: The runtime actually has a role in both a component's runtime and development time experiences. While the component is running, the runtime is responsible for managing memory allocation, starting up and stopping threads and processes, and enforcing security policy, as well as satisfying any dependencies that the component might have on other components. At development time, the runtime's role changes slightly; because it automates so much (for example, memory management).
- **Unified programming classes**: The framework provides developers with a unified, object-oriented, hierarchical, and extensible set of class libraries (APIs). Currently, C++ developers use the Microsoft Foundation Classes and Java developers use the Windows Foundation Classes. The framework unifies these disparate models and gives Visual Basic and JScript programmer's access to class libraries as well.
- **ASP.NET builds on the programming classes of the .NET Framework**: providing a Web application model with a set of controls and infrastructure that make it simple to build Web applications. ASP.NET includes a set of controls that encapsulate common HTML user interface elements, such as text boxes, buttons, and list boxes. These controls run on the Web server, however, and render their user interface as HTML to the browser.

1.6.4 What is .NET

.NET is both a business strategy from Microsoft and its collection of programming support for what are known as Web services, the ability to use the Web rather than your own computer for various services. Microsoft's goal is to provide individual and business users with a seamlessly interoperable and Web enabled interface for applications and computing devices and to make computing activities increasingly Web browser oriented. The .NET platform includes servers; building-block services, such as Web-based data storage; and device software.

The .NET platform was designed to provide:

- The ability to make the entire range of computing devices work together and to have user information automatically updated and synchronized on all of them
- Increased interactive capability for Web sites, enabled by greater use of XML (Extensible Markup Language) rather than HTML
- A premium online subscription service, that will feature customized access and delivery of products and services to the user from a central starting point for the management of various applications, such as e-mail, for example, or software, such as Office .NET
- Centralized data storage, which will increase efficiency and ease of access to information, as well as synchronization of information among users and devices
- The ability to integrate various communications media, such as e-mail, faxes, and telephones
- For developers, the ability to create reusable modules, which should increase productivity and reduce the number of programming errors.

The full release of .NET is expected to take several years to complete, with intermittent releases of products such as a personal security service and new versions of Windows and Office that implement the .NET strategy coming on the market separately. Visual Studio .NET is a development environment that is now available. Windows XP supports certain .NET capabilities.

1.6.5 What is Visual Studio .NET

Visual Studio .NET is Microsoft's visual programming environment for creating Web services based on use of the Extensible Markup Language (XML). The product suite provides a visual interface for identifying a program as a Web service, forms for building a user interface (including support for mobile device interfaces), features for integrating existing application data, and for debugging. Visual Studio .NET comes with the; .NET Framework, including the Common Language Runtime, and includes several programming languages including Visual Basic, Visual C++, and Visual C#.

Visual Studio .NET comes in any of three levels of capability and price: Professional, Enterprise Developer (which includes Microsoft's SQL Server), and Enterprise Architect (which includes the Visio product for modeling an application program). In Microsoft's view, Visual Studio .NET aims at setting a benchmark of ease in application development for the Web in the present decade just as its Visual Basic set a benchmark for visual programming in the 1990s. Existing users of Microsoft's Visual line and related languages may upgrade to Visual Studio .NET for a discount from the full price.

1.7 Summary

Chapter 1, a general information about visual basic is explained. The types visual studio, framework .net of together with their securitys and meaning of the basic topics are presented.

CHAPTER 2
MICROSOFT ACCESS AND DATABASES

2.1 Overview

In this chapter, the application Microsoft access and database are covered. We will introduce meaning of Microsoft access with their file extensions and versions and then we will cover the application of database with general models and an algorithm.

2.2 Introduction to Microsoft Access

Microsoft Office Access, previously known as Microsoft Access, is a relational database management system from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software development tools. It is a member of the 2007 Microsoft Office system.

Access can use data stored in Access/Jet, Microsoft SQL Server, Oracle, or any ODBC-compliant data container (including MySQL and PostgreSQL). Skilled software developers and data architects use it to develop application software. Relatively unskilled programmers and non-programmer "power users" can use it to build simple applications. It supports some object-oriented techniques but falls short of being a fully object-oriented development tool. Access was also the name of a communications program from Microsoft, meant to compete with ProComm and other programs. This proved a failure and was dropped. One Years later Microsoft reused the name for its database software.

2.2.1 History of Microsoft Access

Access version 1.0 was released in November 1992. Since that time, the following versions have been released: 2.0, 95, 97, 2000, 2002 (also called XP), 2003, and the latest, 2007.Microsoft specified the minimum operating system for Version 2.0 as Microsoft Windows v3.0 with 4 MB of RAM. 6 MB RAM was recommended along with a minimum of 8 MB of available hard disk space (14 MB hard disk space recommended). The product was shipped on seven 1.44 MB diskettes. The manual shows a 1993 copyright date.

Access's initial codename was Cirrus; the forms engine was called Ruby. This was before Visual Basic - Bill Gates saw the prototypes and decided that the BASIC language component should be co-developed as a separate expandable application, a project called Thunder. The two projects were developed separately as the underlying forms engines were incompatible with each other; however, these were merged together again after VBA.

2.2.2 Uses of Access

Access is used by small businesses, within departments of large corporations, and by hobby programmers to create ad hoc customized desktop systems for handling the creation and manipulation of data. Access can be used as a database for basic web based applications hosted on Microsoft's Internet Information Services and utilizing Microsoft Active Server Pages ASP. Most typical web applications should use tools like ASP/Microsoft SQL Server or the LAMP stack.

Some professional application developers use Access for rapid application development, especially for the creation of prototypes and standalone applications that serve as tools for on-the-road salesmen. Access does not scale well if data access is via a network, so applications that are used by more than a handful of people tend to rely on Client-Server based solutions. However, an Access "front end" (the forms, reports, queries and VB code) can be used against a host of database backend, including JET (file-based database engine, used in Access by default), Microsoft SQL Server, Oracle, and any other ODBC-compliant product.

2.2.3 Features of Access

One of the benefits of Access from a programmer's perspective is its relative compatibility with SQL (structured query language) queries may be viewed and edited as SQL statements, and SQL statements can be used directly in Macros and VBA Modules to manipulate Access tables. Users may mix and use both VBA and "Macros" for programming forms and logic and offers object-oriented possibilities.

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

2.2.4 Development of Access

Access allows relatively quick development because all database tables, queries, forms, and reports are stored in the database. For query development, Access utilizes the Query Design Grid, a graphical user interface that allows users to create queries without knowledge of the SQL programming language. In the Query Design Grid, users can "show" the source tables of the query and select the fields they want returned by clicking and dragging them into the grid. Joins can be created by clicking and dragging fields in tables to fields in other tables. Access allows users to view and manipulate the SQL code if desired.

The programming language available in Access is, as in other products of the Microsoft Office suite, Microsoft Visual Basic for Applications. Two database access libraries of COM components are provided: the legacy Data Access Objects (DAO), which was superseded for a time (but still accessible) by ActiveX Data Objects (ADO); however (DAO) has been reintroduced in the latest version, Microsoft Access 2007.

Since all database queries, forms, and reports are stored in the database, and in keeping with the ideals of the relational model, there is no possibility of making a physically structured hierarchy with them. One recommended technique is to migrate to SQL Server and utilize Access Data Projects. This allows stored procedures, views, and constraints which are greatly superior to anything found in Jet.

Access allows no relative paths when linking, so the development environment should have the same path as the production environment (though it is possible to write a "dynamic-linker" routine in VBA that can search out a certain back-end file by searching through the directory tree, if it can't find it in the current path). This technique also allows the developer to divide the application among different files, so some structure is possible.

2.2.5 File extensions of Access

Microsoft Access saves information under the following file extensions:

.mdb - Access Database (2003 and earlier)

.mde - Protected Access Database, with compiled VBA (2003 and earlier)

.accdb - Access Database (2007)

.accde - Protected Access Database, with compiled VBA (2007)

.mam - Windows Shortcut: Access Macro

.maq - Windows Shortcut: Access Query

.mar - Windows Shortcut: Access Report

.mat - Windows Shortcut: Access Table

.maf - Windows Shortcut: Access Form

.adp - Access Project

.adn - Access Blank Project Template

.mda - Access Database, used for adding (2, 95,97), previously used for workgroups

.mdw - Access Workgroup, database for user-level security.

.mdf - Access (SQL Server) detached database (2000)

2.2.6 Versions of Access

Table 2.1 Versions of access

Date	Version	Version number	Supported OS	Office suite version
1992	Access 1.1	1	Windows 3.1x	
1993	Access 2.0	2.0	Windows 3.1x	Office 4.3 Pro
1995	Access for Windows 95	7.0	Windows 95	Office 95 Professional

1997	Access 97	8.0	Windows 9x, NT 3.5/4.0	Office 97 Professional and Developer
1999	Access 2000	9.0	Windows 9x, NT 4.0, 2000	Office 2000 Professional, Premium and Developer
2001	Access 2002	10	Windows 98, Me, 2000, XP	Office XP Professional and Developer
2003	Access 2003	11	Windows 2000, XP	Office 2003 Professional and Professional Enterprise
2007	Microsoft Office Access 2007	12	Windows XP SP2, Vista	Office 2007 Professional, Plus, Ultimate and Enterprise

All of the Office 95 products have OLE 2 capabilities, and Access 7 shows that it was compatible with Word 7.

2.3 What is a Database?

A database is a structured collection of records or data. A computer database relies upon software to organize the storage of data. The software models the database structure in what are known as database models. The model in most common use today is the relational model. Other models such as the hierarchical model and the network model use a more explicit representation of relationships.

Database management systems are the software used to organize and maintain the database. These are categorized according to the database model that they support. The model tends to determine the query languages that are available to access the database. A great deal of the internal engineering of a DBMS, however, is independent of the data model, and is concerned with managing factors such as performance, concurrency, integrity, and recovery from hardware failures. In these areas there are large differences between products.

2.3.1 History of Computer Databases

The first database management systems were developed in the 1960s. A pioneer in the field was Bachman's early papers show that his aim was to make more effective use of the new direct access storage devices becoming available: until then, data processing had been based on punched cards and magnetic tape, so serial processing was the dominant activity.

The relational model was proposed by E. F. Cod in 1970. He criticized existing models for confusing the abstract description of information structure with descriptions of physical access mechanisms. For a long while, however, the relational model remained of academic interest only. While CODASYL products (IDMS) and network model products (IMS) were conceived as practical engineering solutions taking account of the technology as it existed.

During the 1980s, research activity focused on distributed database systems and database machines. Another important theoretical idea was the Functional Data Model, but apart from some specialized applications in genetics, molecular biology, and fraud investigation, the world took little notice.

In the 1990s, attention shifted to object-oriented databases. These had some success in fields where it was necessary to handle more complex data than relational systems could easily cope with, such as spatial databases, engineering data, including software, and multimedia data. Some of these ideas were adopted by the relational vendors, who integrated new features into their products as a result. The 1990s also saw the spread of Open Source databases, such as PostgreSQL and MySQL.

In the 2000s, the fashionable area for innovation is the XML database. As with object databases, this has spawned a new collection of start-up companies, but at the same time the key ideas are being integrated into the established relational products. XML databases aim to remove the traditional divide between documents and data, allowing all of an organization's information resources to be held in one place, whether they are highly structured or not.

2.3.2 Database Models

Various techniques are used to model data structure. Most database systems are built around one particular data model, although it is increasingly common for products to offer support for more than one model. For any one logical model various physical implementations may be possible, and most products will offer the user some level of control in tuning the physical implementation, since the choices that are made have a significant effect on performance.

- **Hierarchical model:** In a hierarchical model, data is organized into an inverted tree-like structure, implying a multiple downward link in each node to describe the nesting, and a sort field to keep the records in a particular order in each same-level list. This structure arranges the various data elements in a hierarchy and helps to establish logical relationships among data elements of multiple files. Each unit in the model is a record which is also known as a node. In such a model, each record on one level can be related to multiple records on the next lower level. A record that has subsidiary records is called a parent and the subsidiary records are called children. Data elements in this model are well suited for one-to-many relationships with other data elements in the database. This model is advantageous when the data elements are inherently hierarchical. The disadvantage is that in order to prepare the database it becomes necessary to identify the requisite groups of files that are to be logically integrated.

- **Network Model:** The network model tends to store records with links to other records. Each record in the database can have multiple parents, i.e., the relationships among data elements can have a many to many relationship. Associations are tracked via "pointers". These pointers can be node numbers or disk addresses. Most network databases tend to also include some form of hierarchical model. Databases can be translated from hierarchical model to network and vice versa. The main difference between the network model and hierarchical model is that in a network model, a child can have a number of parents whereas in a hierarchical model, a child can have only one parent.

- **Relational Model**: The basic data structure of the relational model is a table where information about a particular entity (say, an employee) is represented in columns and rows. The columns enumerate the various attributes of an entity (e.g. employee_name, address, phone_number). Rows (also called records) represent instances of an entity (e.g. specific employees).The "relation" in "relational database" comes from the mathematical notion of relations from the field of set theory. A relation is a set of tupelos, so rows are sometimes called tuple. All tables in a relational database have these rules: The ordering of columns is immaterial, Identical rows are not allowed in a table, each row has a single (separate) value for each of its columns (each tuple has an atomic value).

Tables can have a designated column or set of columns that act as a "key" to select rows from that table with the same or similar key values. A "primary key" is a key that has a unique value for each row in the table. Keys are commonly used to join or combine data from two or more tables. For example, an employee table may contain a column named address which contains a value that matches the key of an address table. Keys are also critical in the creation of indexes, which facilitate fast retrieval of data from large tables. It is not necessary to define all the keys in advance; a column can be used as a key even if it was not originally intended to be one.

2.3.4 DBMS Internals

- **Storage and Physical Database Design:** Database tables/indexes are typically stored in memory or on hard disk in one of many forms, ordered/unordered flat files, ISAM, heaps, hash buckets or B+ trees. These have various advantages and disadvantages discussed further in the main article on this topic. The most commonly used are B+ trees and ISAM. Other important design choices relate to the clustering of data by category (such as grouping data by month, or location), creating pre-computed views known as materialized views, partitioning data by range or hash.
- **Security:** Database security denotes the system, processes, and procedures that protect a database from unintended activity. In the United Kingdom legislation protecting the public from unauthorized disclosure of personal information held on databases falls under the Office of the Information Commissioner. United Kingdom based organizations holding personal data in electronic format (databases for example) are required to register with the Data Commissioner.
- **Locking:** Locking is the act of putting a lock (access restriction) on an aspect of a database which at a particular given instance is being modified. Such locks can be applied on a row level, or on other levels such as an entire table. This helps maintain the integrity of the data by ensuring that only one user at a time can modify the data. Databases can also be locked for other reasons, like access restrictions for given levels of user. Databases are also locked for routine database maintenance, which prevents changes being made during the maintenance.
- **Architecture**: Depending on the intended use, there are a number of database architectures in use. Many databases use a combination of strategies. On-line Transaction Processing systems (OLTP) often use row-oriented data store architecture, while data-warehouse and other retrieval-focused applications like Google's Bitable, or bibliographic database (library catalogue) systems may use column-oriented data store architecture.

Indexing: All of these databases can take advantage of indexing to increase their speed, and this technology has advanced tremendously. The most common kind of index is a sorted list of the contents of some particular table column, with pointers to the row associated with the value. An index allows a set of table rows matching some criterion to be located quickly. Typically, indexes are also stored in the various forms of data-structure mentioned above. Usually, a specific technique is chosen by the database designer to increase efficiency in the particular case of the type of index required.

Relational DBMSs have the advantage that indexes can be created or dropped without changing existing applications making use of it. The database chooses between many different strategies based on which one it estimates will run the fastest.

2.3.5 Applications of Databases

Databases are used in many applications, spanning virtually the entire range of computer software. Databases are the preferred method of storage for large multi-user applications, where coordination between many users is needed. Even individual users find them convenient, and many electronic mail programs and personal organizers are based on standard database technology. Software database drivers are available for most database platforms so that application software can use a common Application Programming Interface to retrieve the information stored in a database. Two commonly used database APIs are JDBC and ODBC. For example supplier's database contains the data relating to suppliers such as; supplier name, supplier code, supplier address it is often used by schools to teach students and grade them.

2.4 Summary

Chapter 2, introduced Microsoft Access with meaning, file extensions, versions and it covered the application of database by general models and an algorithm.

CHAPTER 3
SQL

3.1 Overview

In this chapter, the general information about SQL, SQL table basics, creating and inserting table and updating, deleting records will be explained briefly.

3.2 What is SQL?

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

3.2.1 Tables Basic

A relational database system contains one or more objects called tables. The data or information for the database is stored in these tables. Tables are uniquely identified by their names and are comprised of columns and rows. Columns contain the column name, data type, and any other attributes for the column. Rows contain the records or data for the columns.

The Microsoft Jet database engine searches the specified table or tables, extracts the chosen columns, selects rows that meet the criterion and sorts or groups the resulting rows into the order specified. SELECT statements don't change data in the database. SELECT is usually the first word in an SQL statement. Most SQL statements are either SELECT or SELECT...INTO statements.

SELECT Statement: Instructs the Microsoft Jet database engine to return information from the database as a set of records.

```
SELECT [predicate] {* | table.* | [table.]field1 [AS
alias1] [, [table.]field2 [AS alias2] [,..]}
FROM tableexpression [, ...] [IN externaldatabase]
[WHERE... ]
[GROUP BY... ]
[HAVING... ]
[ORDER BY... ]
[WITH OWNERACCESS OPTION]
```

The SELECT statement has the following parts:

- **Predicate**: One of the following predicates: ALL, DISTINCT, DISTINCTROW, or TOP. You use the predicate to restrict the number of records returned.
- *: Specifies that all fields from the specified table or tables are selected.
- **Table**: The name of the table containing the fields from which records are selected.
- **field1, field2:** The names of the fields containing the data you want to retrieve. If you include more than one field, they are retrieved in the order listed.
- **alias1, alias2**: The names to use as column headers instead of the original column names in table.
- **table expression:** The name of the table or tables containing the data you want to retrieve.
- **external database:** The name of the database containing the tables in table expression if they are not in the current database.

The minimum syntax for a SELECT statement is:

```
SELECT fields FROM table
```

You can use an asterisk (*) to select all fields in a table. The following example selects all of the fields in the Employees table:

```
SELECT * FROM Employees;
```

SELECT Statement: Instructs the Microsoft Jet database engine to return information from the database as a set of records.

```
SELECT [predicate] {* | table.* | [table.]field1 [AS
alias1] [, [table.]field2 [AS alias2] [,...]}
FROM tableexpression [, ...] [IN externaldatabase]
[WHERE... ]
[GROUP BY... ]
[HAVING... ]
[ORDER BY... ]
[WITH OWNERACCESS OPTION]
```

The SELECT statement has the following parts:

- **Predicate**: One of the following predicates: ALL, DISTINCT, DISTINCTROW, or TOP. You use the predicate to restrict the number of records returned.
- *: Specifies that all fields from the specified table or tables are selected.
- **Table**: The name of the table containing the fields from which records are selected.
- **field1, field2:** The names of the fields containing the data you want to retrieve. If you include more than one field, they are retrieved in the order listed.
- **alias1, alias2**: The names to use as column headers instead of the original column names in table.
- **table expression:** The name of the table or tables containing the data you want to retrieve.
- **external database:** The name of the database containing the tables in table expression if they are not in the current database.

The minimum syntax for a SELECT statement is:

```
SELECT fields FROM table
```

You can use an asterisk (*) to select all fields in a table. The following example selects all of the fields in the Employees table:

```
SELECT * FROM Employees;
```

If a field name is included in more than one table in the FROM clause, precede it with the table name and the . (dot) operator. In the following example, the Department field is in both the Employees table and the Supervisors table. The SQL statement selects departments from the Employees table and supervisor names from the Supervisors table:

```
SELECT Employees.Department, Supervisors.SupvName
FROM Employees INNER JOIN Supervisors
WHERE Employees.Department = Supervisors.Department;
```

When a Recordset object is created, the Microsoft Jet database engine uses the table's field name as the Field object name in the Recordset object. If you want a different field name or a name isn't implied by the expression used to generate the field, use the AS reserved word. The following example uses the title Birth to name the returned Field object in the resulting Recordset object:

```
SELECT BirthDate
AS Birth FROM Employees;
```

Whenever you use aggregate functions or queries that return ambiguous or duplicate Field object names, you must use the AS clause to provide an alternate name for the Field object. The following example uses the title HeadCount to name the returned Field object in the resulting Recordset object:

```
SELECT COUNT (EmployeeID)
AS HeadCount FROM Employees;
```

3.2.2 Selecting Data

The select statement is used to query the database and retrieve selected data that match the criteria that you specify. Here is the format of a simple select statement:

```
select "column1"
[,"column2",etc]
from "tablename"
[where "condition"];
[] = optional
```

The column names that follow the select keyword determine which columns will be returned in the results. You can select as many column names that you'd like, or you can use a "*" to select all columns.

The where clause (optional) specifies which data values or rows will be returned or displayed, based on the criteria described after the keyword where.

Conditional selections used in the where clause:

=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal to

LIKE *See note below

The LIKE pattern matching operator can also be used in the conditional selection of the where clause. Like is a very powerful operator that allows you to select only rows that are "like" what you specify. The percent sign "%" can be used as a wild card to match any possible character that might appear before or after the characters specified. For example:

```
select first, last, city
 from empinfo
 where first LIKE 'Er%';
```

This SQL statement will match any first names that start with 'Er'. Strings must be in single quotes. Or you can specify,

```
select first, last
 from empinfo
 where last LIKE '%s';
```

This statement will match any last names that end in a 's'.

```
select * from empinfo
where first = 'Eric';
```

This will only select rows where the first name equals 'Eric' exactly.

3.2.3 Creating Tables

The create table statement is used to create a new table. Here is the format of a simple creates table statement:

```
create table "tablename"
("column1" "data type",
 "column2" "data type",
 "column3" "data type");
```

Format of create table if you were to use optional constraints:

```
create table "tablename"
("column1" "data type"
[constraint],
 "column2" "data type"
  [constraint],
 "column3" "data type"
[constraint]);
 [ ] = optional
```

You may have as many columns as you'd like, and the constraints are optional.

Example: create table employee

```
(first varchar(15),      last varchar(20),
   age number(3),
   address varchar(30),
   city varchar(20),
```

To create a new table, enter the keywords create table followed by the table name, followed by an open parenthesis, followed by the first column name, followed by the data type for that column, followed by any optional constraints, and followed by a closing parenthesis. It is important to make sure you use an open parenthesis before the beginning table and a closing parenthesis after the end of the last column definition. All SQL statements should end with a ";".

The table and column names must start with a letter and can be followed by letters, numbers, or underscores - not to exceed a total of 30 characters in length. Do not use any SQL reserved keywords as names for tables or column names (such as "select", "create", "insert", etc).

Data types specify what the type of data can be for that particular column. If a column called "Last_Name", is to be used to hold names, then that particular column should have a "varchar" (variable-length character) data type.

Here are the most common Data types:

- **char(size):** Fixed-length character string. Size is specified in parenthesis.
- **varchar(size)** :Variable-length character string. Max size is specified in parenthesis.
- **number(size):** Number value with a max number of column digits specified in parenthesis.
- **Date:** Date value
- **number(size,d):** Number value with a maximum number of digits of "size" total, with a maximum number of "d" digits to the right of the decimal.

What are constraints: When tables are created, it is common for one or more columns to have constraints associated with them. A constraint is basically a rule associated with a column that the data entered into that column must follow. For example, a "unique" constraint specifies that no two records can have the same value in a particular column. They must all be unique. The other two most popular constraints are "not null" which specifies that a column can't be left blank, and "primary key". A "primary key" constraint defines a unique identification of each record (or row) in a table. Constraints can be entered in this SQL interpreter, however, they are not supported in this Intro to SQL tutorial & interpreter.

3.2.4 Inserting into a Table

The insert statement is used to insert or add a row of data into the table.To insert records into a table, enter the key words insert into followed by the table name, followed by an open parenthesis, followed by a list of column names separated by commas, followed by a closing parenthesis, followed by the keyword values, followed by the list of values enclosed in parenthesis. The values that you enter will be held in the rows and they will match up with the column names that you specify. Strings should be enclosed in single quotes, and numbers should not.

```
insert into "tablename"
(first_column,...last_column)
values (first_value,...last_value);
```

In the example below, the column name first will match up with the value 'Hatice', and the column name state will match up with the value 'Ozsaltik'.

Example:
```
insert into employee
  (first, last, age, address, city, state)
   values ('Hatice', 'Ozsaltik', 00, '2130 Boars
Nest', 'Hazard Co', 'Georgia');
```

3.2.5 Updating Records

The update statement is used to update or change records that match specified criteria. This is accomplished by carefully constructing a where clause.

```
update "tablename"   set "columnname" =  "newvalue"
 [,"nextcolumn" =      "newvalue2"...]
where "columnname"
 OPERATOR "value"
[and|or "column"
OPERATOR "value"];
```

Examples:
```
update phone_book
set area_code = 623
where prefix = 979;
```

3.2.6 Deleting Records

The delete statement is used to delete records or rows from the table.

```
Delete from "tablename"
where "columnname"
OPERATOR "value"
[and|or "column"
OPERATOR "value"];
[ ] = optional
```

Examples: delete from employee
    where lastname = 'May';

To delete an entire record/row from a table, enter "delete from" followed by the table name, followed by the where clause which contains the conditions to delete. If you leave off the where clause, all records will be deleted.

3.2.7 Removing Tables

The drop table command is used to delete a table and all rows in the table. To delete an entire table including all of its rows, issue the drop table command followed by the table name. Drop table is different from deleting all of the records in the table. Deleting all of the records in the table leaves the table including column and constraint information. Dropping the table removes the table definition as well as all of its rows.

```
drop table "tablename"
```

Example:

```
drop table myemployees_ts0211;
```

3.3 Summary

Chapter 3, a general information about SQL is explained. The creating, inserting table and updating ,deleting records are presented.

CHAPTER 4
ACTIVEX DATA OBJECTS (ADO)

4.1 Overview

This chapter will present the property and fonksiyons the ActiveX Data Objects (ADO) with existing technologies and new features and this chapter focuses on general purpose ADO connection objects, ADO Recordsets and ADO Attributes Property will include with general applications.

4.2 What is ADO?

ActiveX Data Objects (ADO) and Object Linking and Embedding Database (OLE DB), its underlying technology, currently play a big part in data access. Microsoft has unequivocally committed its future to it, and rightly so. The paperless office has yet to appear, but the amount of data stored on computer systems increases every day. This is illustrated by the rate at which the Web is expanding and that's just the public face of data. Much more data is hidden from general view in corporate applications or intranets. ADO is central to Microsoft's data access strategy, so it's important to understand why it came about and what sort of a future it has.

What is Data: Make a mental note of how many separate pieces of information you've got: databases, documents, spreadsheets, e-mail messages, HTML and Active Server Pages (ASP) documents, etc. They are all pockets of data, but are stored in different forms. This might seem obvious, but traditionally data has been thought of as being stored only in a database; if you built a business application, the data had to be in a database. In fact, as computers become more powerful, the term "data" is starting to include multimedia items such as music and video, as well as objects and the more typical document-based data. So, by "data" I mean any piece of information whatever its contents. Whether it's your address book, you're monthly expenses spreadsheet, or a pleading letter to the taxman, its all data.

4.3 About Universal Data Access

Universal Data Access (UDA) is Microsoft's strategy for dealing with all this data. It's aimed at providing high-performance access to a variety of data stores. That data is stored in many different ways, and there is no central way of accessing it all. UDA offers an easy-to-use methodology that allows access to multiple sources of data in a single way. Build in high performance and support for existing data access methods, and you're on your way to something that could make a real difference. It's important to remember that UDA is simply Microsoft's strategy for accessing data, not a technology. UDA is physically implemented as a collection of four technologies: ADO, OLE DB, Remote Data Services (RDS), and Open Database Connectivity (ODBC). Collectively, these four technologies are known as the Microsoft Data Access Components (MDAC). This means that you don't have to bundle all your data into a single data store. Here's how it can work:

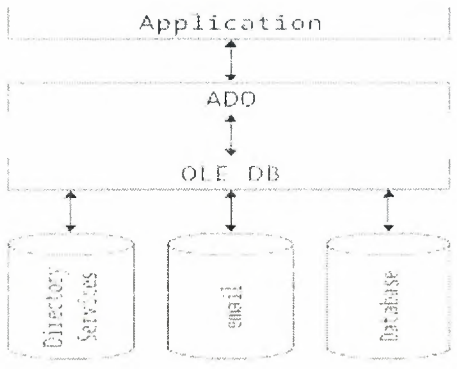


Figure 4.1. Shown Microsoft data access components (MDAC)

When building an application, you can make sure it uses ADO for its data access, and ADO will talk to all the data sources required. This means that programming is easier, because you need learn how only one programming syntax, as shown in the following illustration. Because ADO provides fast, transparent access to different types of data, there's no reason to use any other method. The three main design goals for the Data Access Components:

- Meeting the key customer requirements, such as performance, reliability, and broad industry support
- Giving access to the widest range of data sources through a common interface
- Providing an easy migration path for existing data access technologies

4.4 ADO Existing Technologies

- **DB-Library (DBLib):** This is the underlying technology for connecting to SQL Server. It is primarily designed for C, but is often used in Visual Basic. Because it is specific to SQL Server, it is extremely fast and functional. For this very reason, however, it doesn't allow access to any other source of data. Other databases, such as Oracle and Sybase, have similar native communication libraries.
- **ODBC:** Open Database Connectivity (ODBC) was the first step on the road to a universal data access strategy. ODBC was designed as a cross-platform, database-independent method for accessing data in any relational database through the use of an Application Program Interface (API), known as the ODBC API. Although ODBC was designed for multi database use, it is often used only on single relational databases.
- **DAO:** The Data Access Objects (DAO), introduced with Microsoft Access, provided a strictly hierarchical set of objects for manipulating data in Jet and other Indexed Sequential Access Method (ISAM) and SQL databases. These objects were first available with Visual Basic 3.0 and quickly became the most commonly used data access method for early Visual Basic programs. DAO also had the advantage of being able to sit on top of ODBC, which allowed it to communicate with many different databases.
- **RDO:** RDO also brought the world of remote database servers to the world of many programmers. RDO and ODBC share the same relationship as ADO and OLE DB: a thin layer on top of an underlying data access mechanism.
- **ODBCDirect:** An extension to DAO, ODBCDirect combined portions of DAO and RDO. It allows programmers to use the DAO programming model and also allows access to ODBC data sources without having the Jet database engine loaded.

4.5 Why ADO?

OLE DB is a COM-based set of object-oriented interfaces, so it is too complex for a large portion of the programming community to use, or it is not suitable because they use programming languages that don't have access to custom COM interfaces. For example, accessing OLE DB directly requires C++ because of the OLE DB interface's complexity.

ADO is the higher-level model that most people will use, because it allows access from dual-interface COM components that can be accessed from Visual Basic and scripting languages. It equates fairly well to the DAO level, where you create an object and call its methods and properties. As a COM component, it can be used from any language that supports COM, such as Visual Basic, VBA, scripting languages, and Visual C++.

Various languages all have the ability to use a central data access strategy. Some languages (like Java and Visual C++) can talk directly to OLE DB in addition to talking to the easier ADO.ADO also improves speed and ease of development by providing an object model that allows data to be retrieved from a data source with as little as one line of code.

4.6 Data Providers and Data Consumers

OLE DB introduces two new terms that help to explain how OLE DB and ADO fit together:

1. **A Data Consumer** is something that uses (or consumes) data. Strictly speaking, ADO is actually a consumer, because it uses data provided by OLE DB.
2. **A Data Provider** is something that provides data. This isn't the physical source of the data, but the mechanism that connects you to the physical data store. The provider may get the data directly from the data store, or it may go through another layer (such as ODBC) to get to the data store.

4.7 New Features of ADO

ADO 2.5 and 2.6 have a host of new features that make programming easier and extend the goal of Universal Data Access.

ADO 2.5

- **The Record Object:** The Record object is designed to deal with Document Source Providers, which are OLE DB Providers that don't access databases, but provide data from semi structured data stores. Two examples of this are Microsoft's Exchange Server 2000 and Internet Information Server 5.0; both are sources of large amounts of data, and the OLE DB Provider for Internet Publishing allows you to access the data storage structure and the stored objects themselves.

- **The Stream Object:** The Stream object is a component that wraps the COM IStream interface, allowing easy access to streams of memory. This provides a way to transfer Recordsets directly to other components (such as the ASP 3.0 Request and Response objects) that support streams. The Stream is also used with Document Source Providers to allow access to file contents.

ADO 2.6

- **Command Streams:** Command streams allow a Stream object to be used as the source of a command. A good example of this is a Stream containing an XML command to be executed against SQL Server 2000.
- **Results in Streams:** Along with command streams, ADO 2.6 allows the results of a data query to be returned into a Stream object. This is particularly useful for obtaining XML data directly from SQL Server 2000.
- **Field Status Values:** The Status property of the Field object is now filled with information to help with the dreaded "Errors Occurred" error.
- **SQL Variant Support for Cursor Service: Extended** support for variant types has been added to the OLE DB Cursor Service.
- **ADOX Group and User Properties:** The Properties collection has been added to the ActiveX Data Objects Extensions (ADOX) Group and User properties to allow access to provider-specific properties.
- **ADO MD UniqueName support:** The UniqueName property can now be used to access ActiveX Data Objects Multidimensional ADO MD objects. This means that parent collections no longer need to be populated to retrieve schema objects.

4.8 ADO Connection Object

The ADO Connection Object is used to create an open connection to a data source. Through this connection, you can access and manipulate a database.If you want to access a database multiple times; you should establish a connection using the Connection object. You can also make a connection to a database by passing a connection string via a Command or Recordset object. However, this type of connection is only good for one specific, single query.

ProgID

set objConnection=Server.CreateObject("ADODB.connection")

Properties

Property	Description
Property	Description
Attributes	Sets or returns the attributes of a Connection object
CommandTimeout	Sets or returns the number of seconds to wait while attempting to execute a command
ConnectionString	Sets or returns the details used to create a connection to a data source
ConnectionTimeout	Sets or returns the number of seconds to wait for a connection to open
CursorLocation	Sets or returns the location of the cursor service
DefaultDatabase	Sets or returns the default database name
IsolationLevel	Sets or returns the isolation level
Mode	Sets or returns the provider access permission
Provider	Sets or returns the provider name
State	Returns a value describing if the connection is open or closed
Version	Returns the ADO version number

Methods

Method	Description
BeginTrans	Begins a new transaction
Cancel	Cancels an execution
Close	Closes a connection
CommitTrans	Saves any changes and ends the current transaction
Execute	Executes a query, statement, procedure or provider specific text

Open	Opens a connection
OpenSchema	Returns schema information from the provider about the data source
RollbackTrans	Cancels any changes in the current transaction and ends the transaction

Events

Event	Description
BeginTransComplete	Triggered after the BeginTrans operation
CommitTransComplete	Triggered after the CommitTrans operation
ConnectComplete	Triggered after a connection starts
Disconnect	Triggered after a connection ends
ExecuteComplete	Triggered after a command has finished executing
InfoMessage	Triggered if a warning occurs during a ConnectionEvent operation
RollbackTransComplete	Triggered after the RollbackTrans operation
WillConnect	Triggered before a connection starts
WillExecute	Triggered before a command is executed

4.9 ADO Attributes Property

The Attributes property sets or returns a long value that indicates one or more characteristics of an object. When setting multiple attributes, it is possible to sum the values. Syntax:

object.Attributes

Object	Description of the Attributes Property
Connection	The Attributes property has read/write permissions on a Connection object. Its value can be the sum of one or more XactAttributeEnum values. Default value is 0

Parameter	The Attributes property has read/write permissions on a Parameter object. Its value can be the sum of one or more ParameterAttributesEnum values. Default value is adParamSigned
Field	The Attributes property has read/write permissions when used to create a Recordset, but it has read-only permissions when you open an existing Recordset. Its value can be the sum of one or more FieldAttributeEnum values
Property	The Attributes property is read-only for a Property object. Its value can be the sum of one or more PropertyAttributesEnum values

4.10 ADO Database Connection

Before a database can be accessed from a web page, a database connection has to be established.

4.10.1 Create a DSN-less Database Connection

The easiest way to connect to a database is to use a DSN-less connection. A DSN-less connection can be used against any Microsoft Access database on your web site.

If you have a database called "northwind.mdb" located in a web directory like "c:/webdata/", you can connect to the database with the following ASP code:

```
<%    set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"        %>
```

4.10.2 Create an ODBC Database Connection

If you have an ODBC database called "northwind" you can connect to the database with the following ASP code:

```
<%   set conn=Server.CreateObject("ADODB.Connection")
          conn.Open "northwind"    %>
```

With an ODBC connection, you can connect to any database, on any computer in your network, as long as an ODBC connection is available.

4.11 ADO Recordset

Record: A set of related data about a person, place, event, or some other item. Table data is stored in records (rows) in the database. Each record is composed of a set of related fields (columns) each field defining one attribute of information for the record. Taken together, a record defines one specific unit of retrievable information in a database.

Create an ADO Table Recordset

Suppose we have a database named "Northwind", we can get access to the "Customers" table inside the database with the following lines:

```
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"
set rs=Server.CreateObject("ADODB.recordset")
rs.Open "Customers", conn
 %>
```

Create an ADO SQL Recordset

We can also get access to the data in the "Customers" table using SQL:

```
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"
set rs=Server.CreateObject("ADODB.recordset")
rs.Open "Select * from Customers", conn
%>
```

Extract Data from the Recordset

After a recordset is opened, we can extract data from recordset.

Suppose we have a database named "Northwind", we can get access to the "Customers" table inside the database with the following lines:

```
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"
set rs=Server.CreateObject("ADODB.recordset")
rs.Open "Select * from Customers", conn
for each x in rs.fields
response.write(x.name)
 response.write(" = ")
response.write(x.value)
next
%>
```

When creating a Recordset object using a non-linked TableDef object in a Microsoft Jet workspace, table-type Recordset objects are created. Only dynaset-type or snapshot-type Recordset objects can be created with linked tables or tables in Microsoft Jet-connected ODBC databases

4.12 ADO Recordset Object

You use Recordset objects to manipulate data in a database at the record level. When you use DAO objects, you manipulate data almost entirely using Recordset objects. All Recordset objects are constructed using records (rows) and fields (columns). There are five types of Recordset objects:

- **Table-type Recordset**: representation in code of a base table that you can use to add, change, or delete records from a single database table (Microsoft Jet workspaces only).
- **Dynaset-type Recordset**: the result of a query that can have updatable records. A dynaset-type Recordset object is a dynamic set of records that you can use to add, change, or delete records from an underlying database table or tables. A dynaset-type Recordset object can contain fields from one or more tables in a database. This type corresponds to an ODBC keyset cursor.

- **Snapshot-type Recordset**: a static copy of a set of records that you can use to find data or generate reports. A snapshot-type Recordset object can contain fields from one or more tables in a database but can't be updated. This type corresponds to an ODBC static cursor.
- **Forward-only-type Recordset** — identical to a snapshot except that no cursor is provided. You can only scroll forward through records. This improves performance in situations where you only need to make a single pass through a result set. This type corresponds to an ODBC forward-only cursor.
- **Dynamic-type Recordset** — a query result set from one or more base tables in which you can add, change, or delete records from a row-returning query. Further, records other users add, delete, or edit in the base tables also appear in your Recordset. This type corresponds to an ODBC dynamic cursor.

In a Microsoft Jet workspace, if you don't specify a type, DAO attempts to create the type of Recordset with the most functionality available, starting with table. If this type isn't available, DAO attempts a dynaset, then a snapshot, and finally a forward-only type Recordset object.

A new Recordset object is automatically added to the Recordsets collection when you open the object, and is automatically removed when you close it. You can create as many Recordset object variables as needed. Different Recordset objects can access the same tables, queries, and fields without conflicting

If you use variables to represent a Recordset object and the Database object that contains the Recordset, make sure the variables have the same scope, or lifetime. For example, if you declare a public variable that represents a Recordset object, make sure the variable that represents the Database containing the Recordset is also public, or is declared in a Sub or Function procedure using the Static keyword.

The default collection of a Recordset object is the Fields collection, and the default property of a Field object is the Value property. Use these defaults to simplify your code.

When you create a Recordset object, the current record is positioned to the first record if there are any records. If there are no records, the RecordCount property setting is 0, and the BOF and EOF property settings are True.

You can use the MoveNext, MovePrevious, MoveFirst, and MoveLast methods to reposition the current record. Forward-only–type Recordset objects support only the MoveNext method. When using the Move methods to visit each record (or "walk" through the Recordset), you can use the BOF and EOF properties to check for the beginning or end of the Recordset object.

With dynaset- and snapshot-type Recordset objects in a Microsoft Jet workspace, you can also use the Find methods, such as FindFirst, to locate a specific record based on criteria. If the record isn't found, the NoMatch property is set to True. For table-type Recordset objects, you can scan records using the Seek method.

The Type property indicates the type of Recordset object created, and the Updatable property indicates whether you can change the object's records. Information about the structure of a base table, such as the names and data types of each Field object and any Index objects, is stored in a TableDef object.

To refer to a Recordset object in a collection by its ordinal number or by its Name property setting, use any of the following syntax forms:

```
Recordsets(0)
Recordsets("name")
Recordsets![name]
```

4.13 Summary

Chapter 4, explained the meaning and property of the ActiveX Data Objects (ADO) with existing technologies and new features. ADO connection objects, ADO Recordsets and ADO Attributes Property are presented with general applications.

CHAPTER 5
AUTOMATION SYSTEM FOR MANAGING PRISON VISITATION

5.1 Overview

In this chapter, description about project will be presented with login, main windows and their functions and forms.

5.2 Project Explanation

There are two important parts in my project. First, Login window and second one the main window. Now these parts will be presented.

5.2.1 Explanation of Login Window

To be able to use the program the user must enter the right username and password to have access to the program functions. The username is not case sensitive but the password is case sensitive. This form will first control is the username exist in the database or not and then moves forward to check the password. Once the login is successful the program will move to the main form and the user can proceed to use the program.

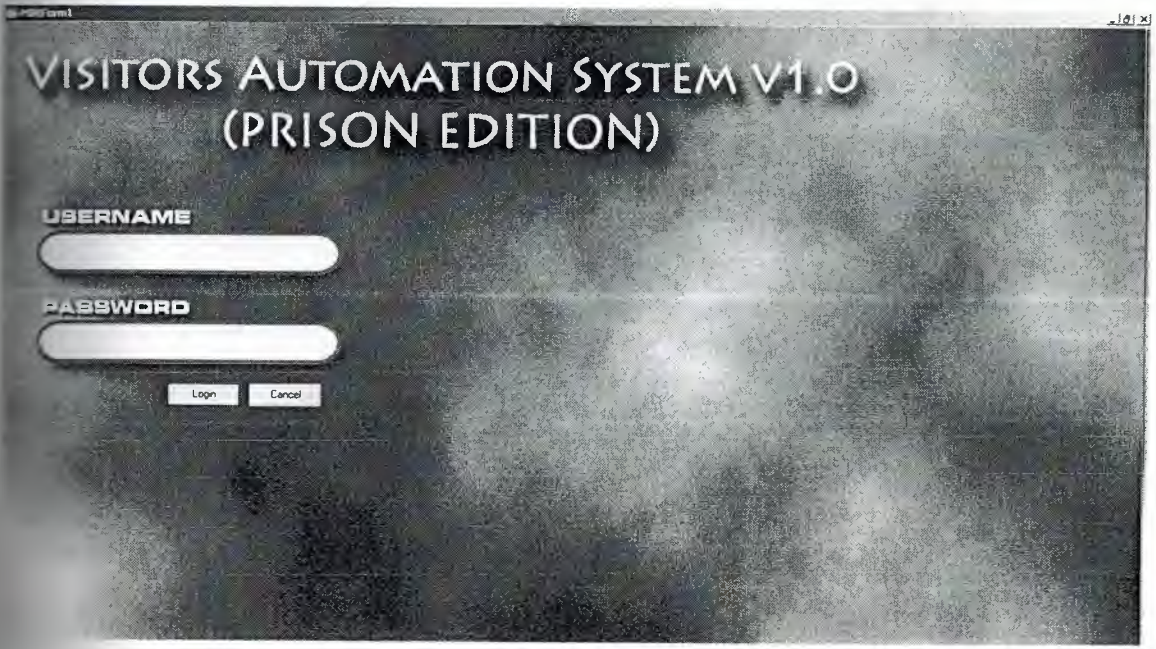


Figure 5.1 Shows the Login window

5.2.2 Explanation of Main Window

Once the user has a successful login this form will appear to the user. In the right side is the program's main functions panel, which consist of:

Today: to show the today's records of visitors.

New Record: to add new visitor information.

Find Record: to search the visitors records for a specific name.

Prisoners: managing prisoner's information.

Reports: to generate printable reports for both the prisoner's records and the visitors records.

User Management: managing the users of the program and defining how can login.

Logout: to logout of the system or change active user. (Will not exit the program)

Exit: to close the program.

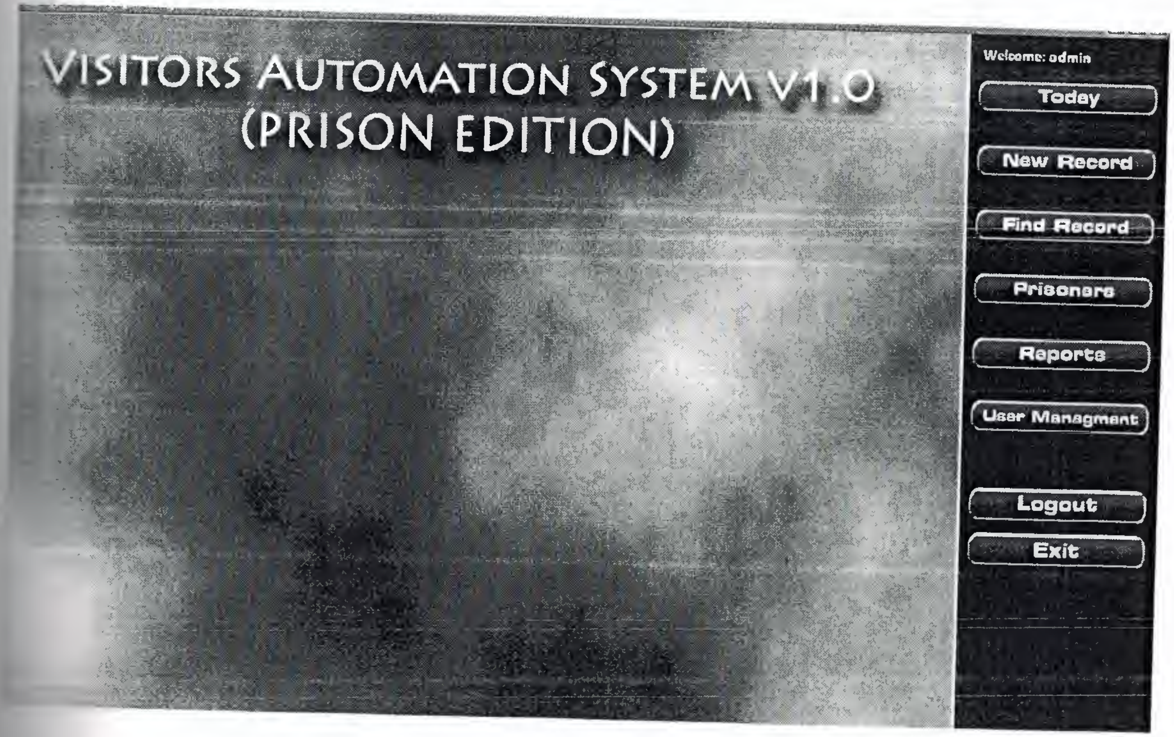
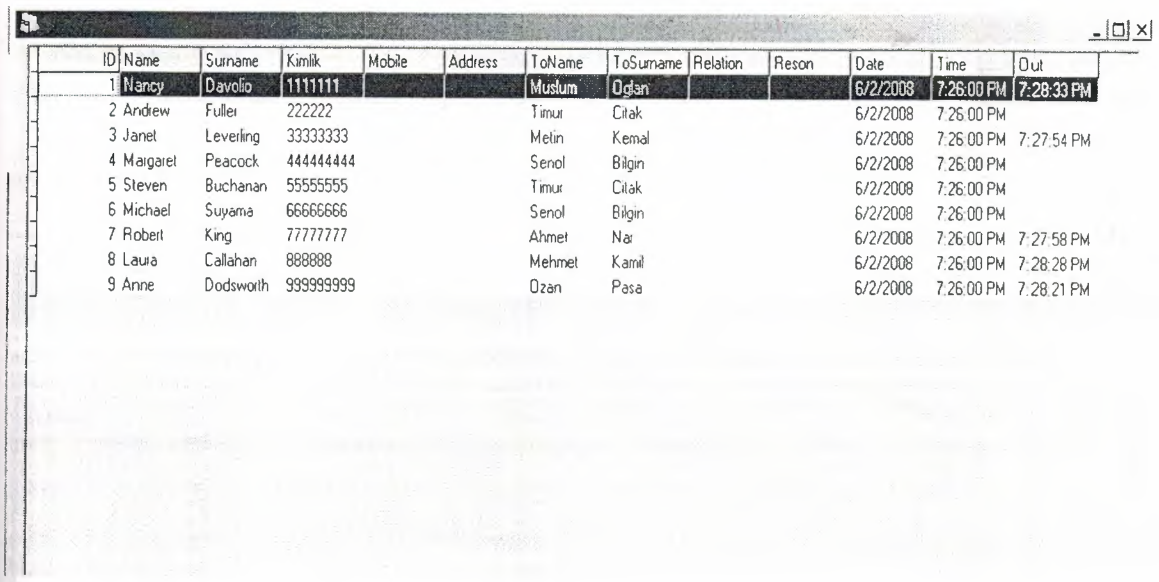


Figure 5.2 Shows the Main window

The Today Window

This form shows all the current records of the visitors came in and out in the current day. It shows information such as the visitor's name and surname, his/hers ID number, Contact number, to who they came, the reason of the visit, date and finally their in and out time.

Information of the each record can be modified by double-clicking on the record, and then a form will open to show the information of the record and will allow the user to modify. By selecting a record and right-clicking on it will allow the user to end a visit, by doing so the out time will be added to the record for future use.



ID	Name	Surname	Kimlik	Mobile	Address	ToName	ToSurname	Relation	Reson	Date	Time	Out
1	Nancy	Davolio	1111111			Muslum	Oglan			6/2/2008	7:26:00 PM	7:28:33 PM
2	Andrew	Fuller	222222			Timur	Citak			6/2/2008	7:26:00 PM	
3	Janet	Leverling	33333333			Metin	Kemal			6/2/2008	7:26:00 PM	7:27:54 PM
4	Margaret	Peacock	444444444			Senol	Bilgin			6/2/2008	7:26:00 PM	
5	Steven	Buchanan	55555555			Timur	Citak			6/2/2008	7:26:00 PM	
6	Michael	Suyama	66666666			Senol	Bilgin			6/2/2008	7:26:00 PM	
7	Robert	King	77777777			Ahmet	Nar			6/2/2008	7:26:00 PM	7:27:58 PM
8	Laura	Callahan	888888			Mehmet	Kamil			6/2/2008	7:26:00 PM	7:28:28 PM
9	Anne	Dodsworth	999999999			Ozan	Pasa			6/2/2008	7:26:00 PM	7:28:21 PM

Figure 5.3 Shows the Today window

The Add New Record Window

When a visitor come to visit a prisoner the user will have to add the visit information to the database. Such information will be both related to the visitor and the prisoner, such as their names and the reason of the visit and the relationship, some additional information will be what the visitor got for the prisoner. An additional function of this program is to be able to capture the photos of the visitors and save them for complete identification. When any visitor comes in, any video capturing device attached to the operating system such as a webcam will allow the program to capture an image of the visitor and saves it for facial identification

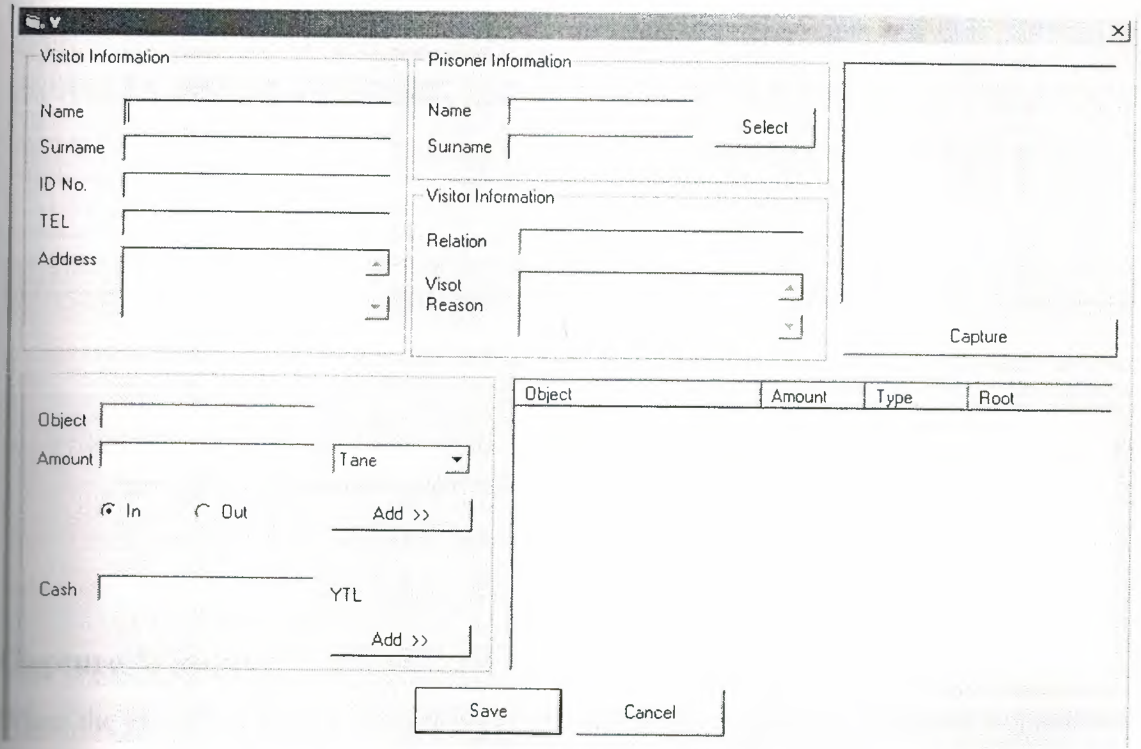
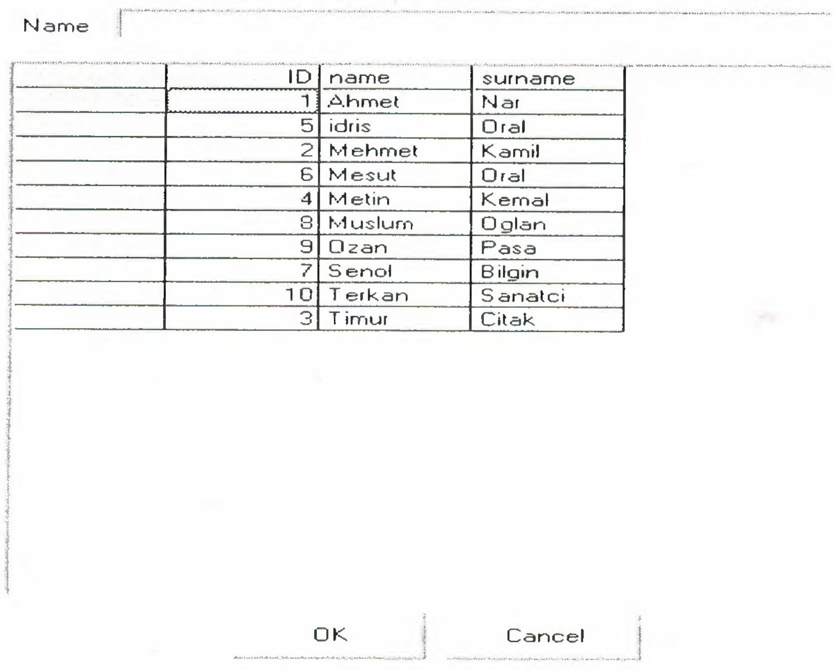


Figure 5.4 Shows the Add new record window

Prisoner Select Window

When adding visit information the user must enter the prisoner name that the visitor is going to visit. By clicking on the select button in the add window this window will appear to easily select a prisoner, by writing their name or just choosing from the list.



	ID	name	surname
	1	Ahmet	Nar
	5	idris	Oral
	2	Mehmet	Kamil
	6	Mesut	Oral
	4	Metin	Kemal
	8	Muslum	Oglan
	9	Ozan	Pasa
	7	Senol	Bilgin
	10	Terkan	Sanatci
	3	Timur	Citak

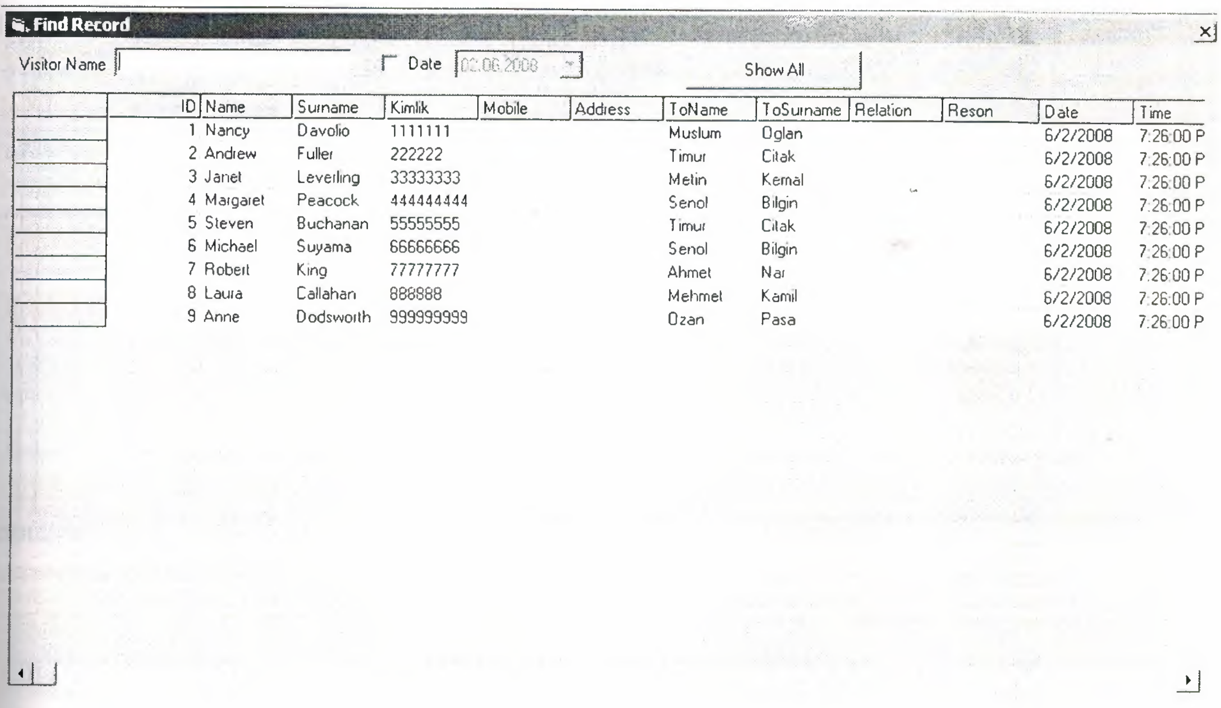
Figure 5.5 Shows the Prisoner select window

Capture Window

When the user enter the visit information he will be required to snap and attach a photo of the visitor to the record, by clicking on the capture button a window will open and starts the attached video capture device to the system in real-time mode and then by clicking on the capture button again the window will capture a single frame and send it to the add window so it will be saved with the record.

The Find Record Window

For looking at the saved record that has been saved in the database the user can search the database by entering a name or a date. User can find information both related to the visitor and the prisoner, such as their names, surnames, IDS, mobiles, address, in come dates, and in-out going times.

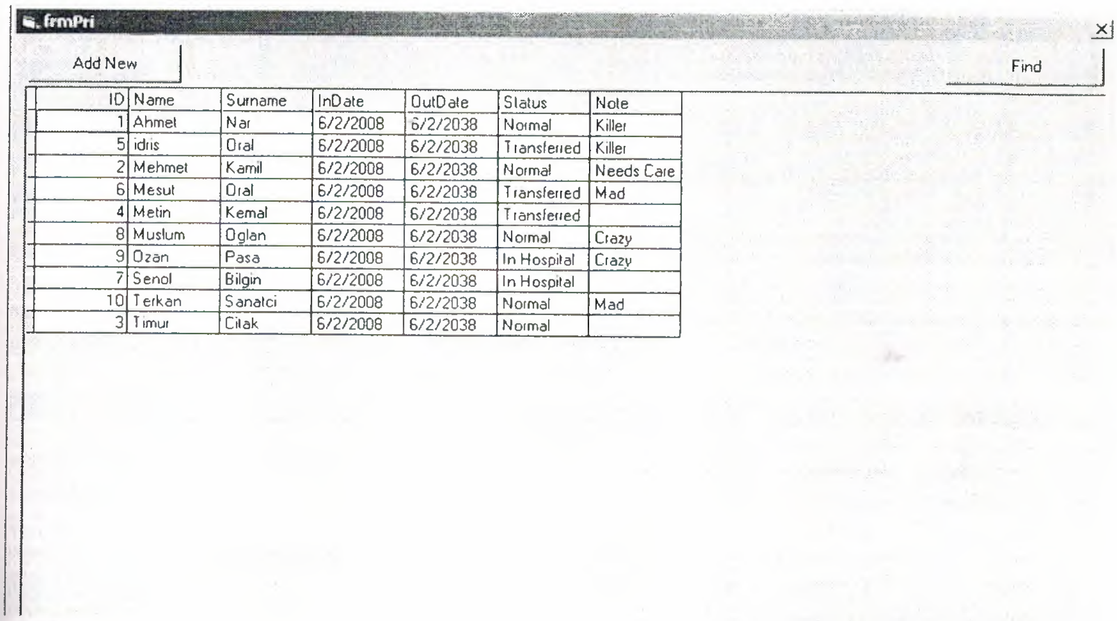


ID	Name	Surname	Kimlik	Mobile	Address	ToName	ToSurname	Relation	Reson	Date	Time
1	Nancy	Davolio	1111111			Muslum	Oglan			6/2/2008	7:26:00 P
2	Andrew	Fuller	222222			Timur	Citak			6/2/2008	7:26:00 P
3	Janet	Leverling	33333333			Metin	Kemal			6/2/2008	7:26:00 P
4	Margaret	Peacock	444444444			Senol	Bilgin			6/2/2008	7:26:00 P
5	Steven	Buchanan	55555555			Timur	Citak			6/2/2008	7:26:00 P
6	Michael	Suyama	66666666			Senol	Bilgin			6/2/2008	7:26:00 P
7	Robert	King	77777777			Ahmet	Nar			6/2/2008	7:26:00 P
8	Laura	Callahan	888888			Mehmet	Kamil			6/2/2008	7:26:00 P
9	Anne	Dodsworth	999999999			Ozan	Pasa			6/2/2008	7:26:00 P

Figure 5.6 Shows the Find record window

The Prisoners Window

In this window all the records of the prisoners will be managed from here. The user can add a new prisoner record or modify an existing one by double clicking on it. And user can make search with prisoner's name and status. Status shows the user if prisoner inside or



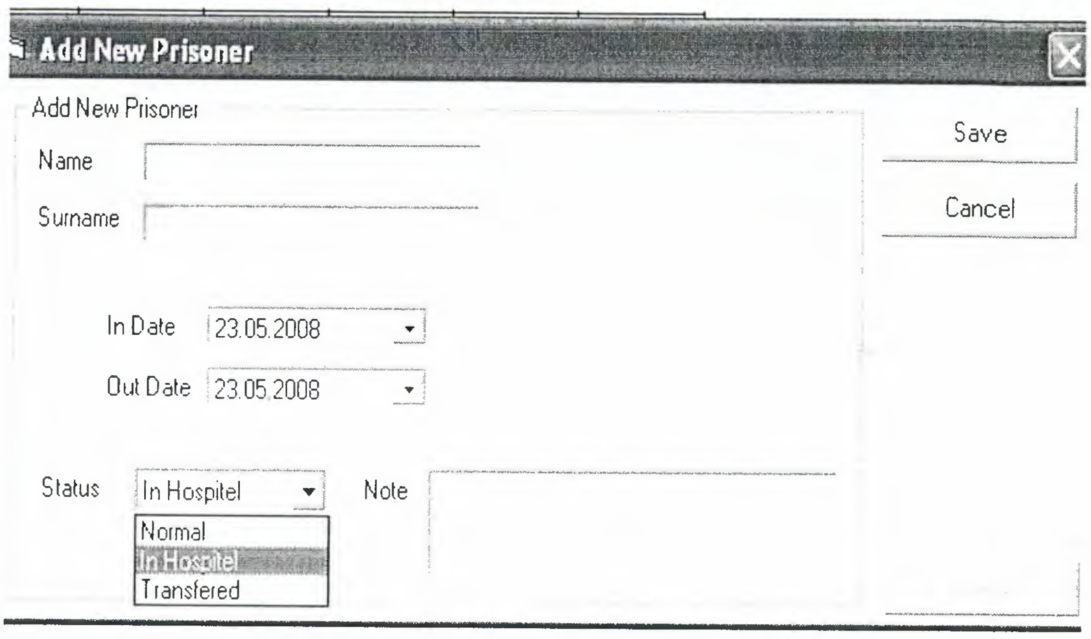
ID	Name	Surname	InDate	OutDate	Status	Note
1	Ahmet	Nar	6/2/2008	6/2/2038	Normal	Killer
5	idris	Oral	6/2/2008	6/2/2038	Transferred	Killer
2	Mehmet	Kamil	6/2/2008	6/2/2038	Normal	Needs Care
6	Mesut	Oral	6/2/2008	6/2/2038	Transferred	Mad
4	Metin	Kemal	6/2/2008	6/2/2038	Transferred	
8	Muslum	Oglan	6/2/2008	6/2/2038	Normal	Crazy
9	Ozan	Pasa	6/2/2008	6/2/2038	In Hospital	Crazy
7	Senol	Bilgin	6/2/2008	6/2/2038	In Hospital	
10	Terkan	Sanatci	6/2/2008	6/2/2038	Normal	Mad
3	Timur	Citak	6/2/2008	6/2/2038	Normal	

outside.

Figure 5.7 Shows the Prisoners window

Add New Prisoner

By clicking on add new in the prisoner's window, this window will appear to enter a new prisoner record. The user must enter the prisoner information to be saved in the database.



In Date 23.05.2008

Out Date 23.05.2008

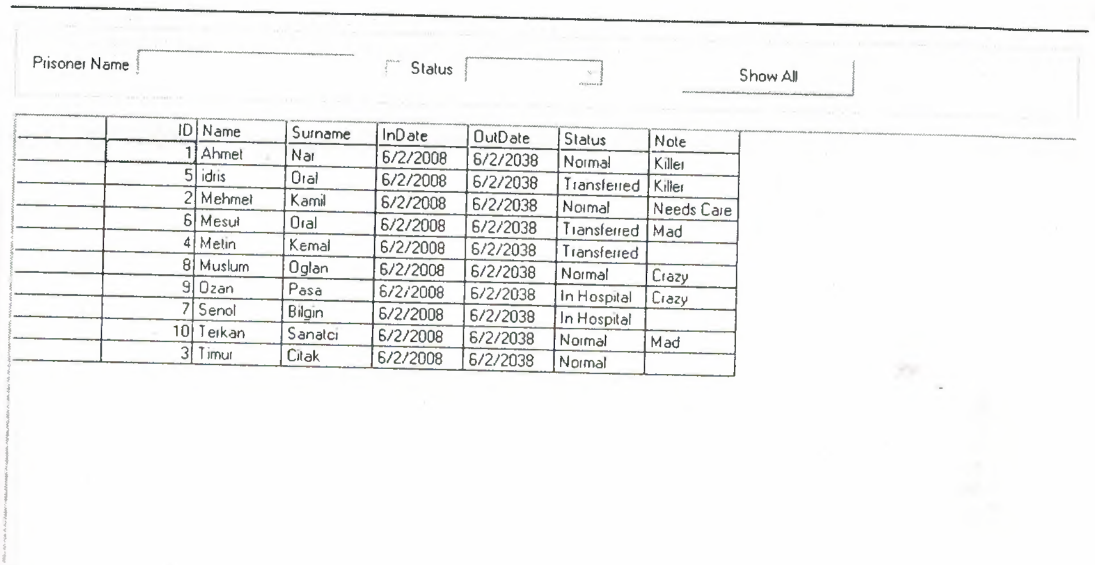
Status In Hospitel

Normal
In Hospitel
Transfered

Figure 5.8 Shows the Prisoners window

Find Prisoner

To find a prisoner to be modified the user can search the database by using this window. User can make search with prisoner name and their status (if he/she inside or outside).

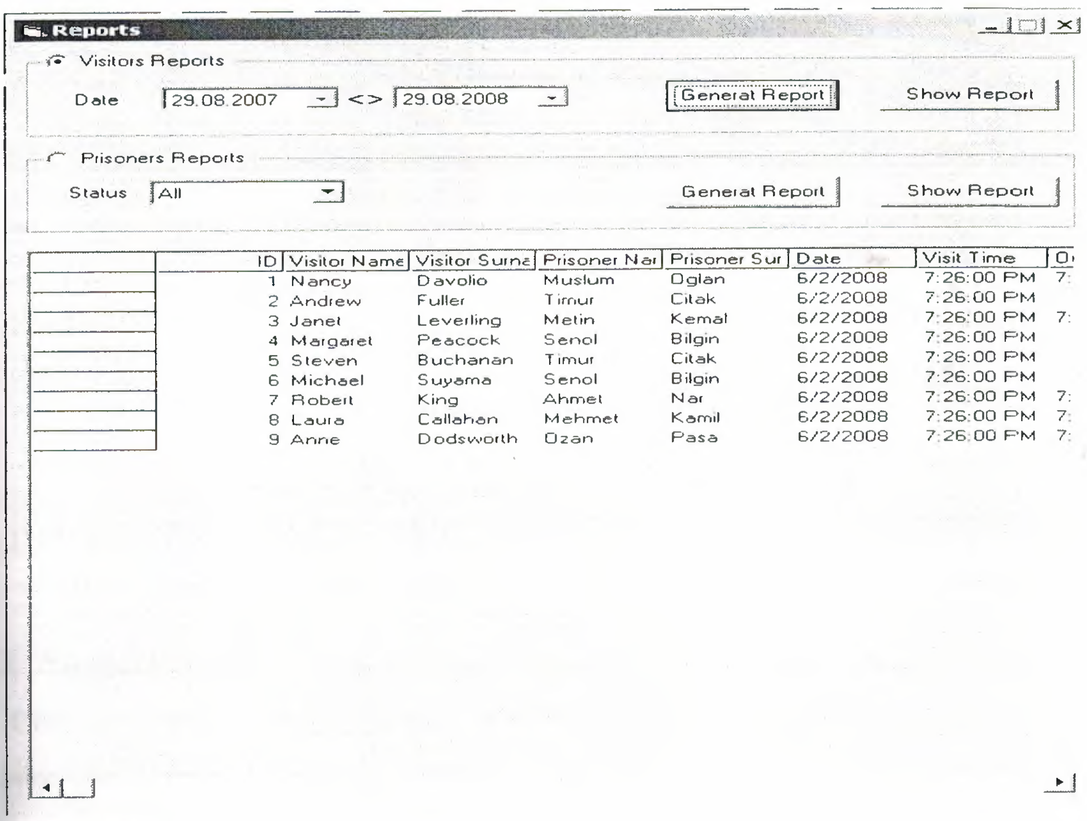


ID	Name	Surname	InDate	OutDate	Status	Note
1	Ahmet	Nar	6/2/2008	6/2/2038	Normal	Killer
5	idris	Oral	6/2/2008	6/2/2038	Transferred	Killer
2	Mehmet	Kamil	6/2/2008	6/2/2038	Normal	Needs Care
6	Mesut	Oral	6/2/2008	6/2/2038	Transferred	Mad
4	Metin	Kemal	6/2/2008	6/2/2038	Transferred	
8	Muslum	Oglan	6/2/2008	6/2/2038	Normal	Crazy
9	Ozan	Pasa	6/2/2008	6/2/2038	In Hospital	Crazy
7	Senol	Bilgin	6/2/2008	6/2/2038	In Hospital	
10	Terkan	Sanatci	6/2/2008	6/2/2038	Normal	Mad
3	Timur	Citak	6/2/2008	6/2/2038	Normal	

Figure 5.9 Shows the Find prisoner window

The Reports Window

This window can prepare printable reports for the user, for both the visitors and the prisoner's records. The visitor's reports can be prepared by choosing records selecting 2 dates, the start date and the end date. The prisoner reports can be prepared by selecting the



Reports

Visitors Reports

Date 29.08.2007 <> 29.08.2008 General Report Show Report

Prisoners Reports

Status All General Report Show Report

ID	Visitor Name	Visitor Surna	Prisoner Nar	Prisoner Sur	Date	Visit Time	O
1	Nancy	Davolio	Muslum	Oglan	6/2/2008	7:26:00 PM	7:
2	Andrew	Fuller	Timur	Citak	6/2/2008	7:26:00 PM	
3	Janet	Leverling	Metin	Kemal	6/2/2008	7:26:00 PM	7:
4	Margaret	Peacock	Senol	Bilgin	6/2/2008	7:26:00 PM	
5	Steven	Buchanan	Timur	Citak	6/2/2008	7:26:00 PM	
6	Michael	Suyama	Senol	Bilgin	6/2/2008	7:26:00 PM	
7	Robert	King	Ahmet	Nar	6/2/2008	7:26:00 PM	7:
8	Laura	Callahan	Mehmet	Kamil	6/2/2008	7:26:00 PM	7:
9	Anne	Dodsworth	Ozan	Pasa	6/2/2008	7:26:00 PM	7:

prisoner status.

Figure 5.10 Shows the reports

The Users Management Window

The administrators managing the users of the program and defining how can login. This information's automatically will be written in user.db data table.

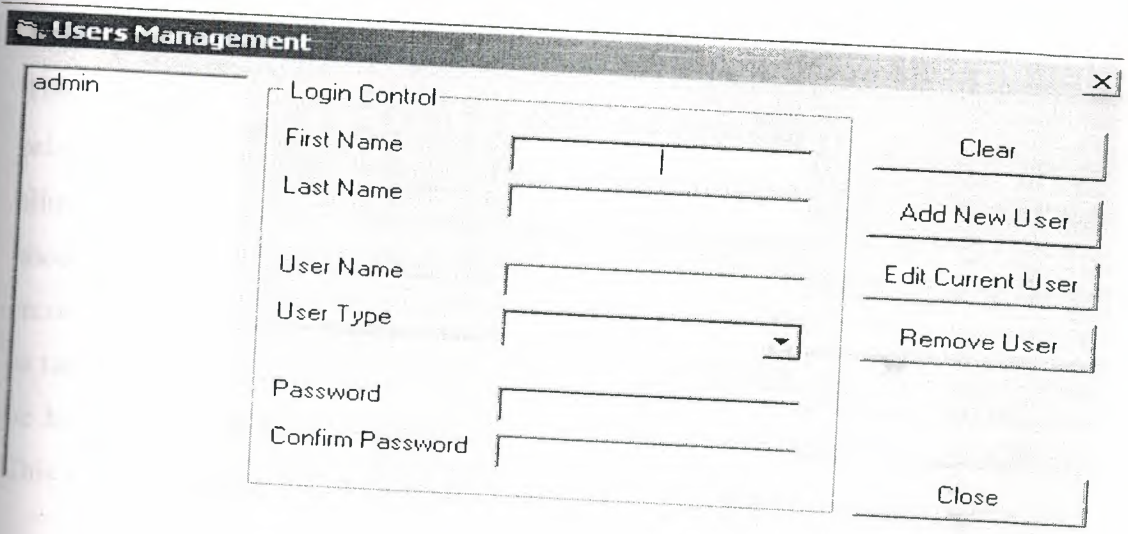


Figure 5.11 Shows the users management window

5.3 Summary

In Chapter 5, a general information about project is explained. The login and main windows are presented with their fuctions.

CONCLUSION

This project introduced the program used in visitor's automation system for prisons. The software tools Microsoft Visual Basic, SQL and Microsoft Office Access was implementing this program.

In this project concerned a program to visitors and prisoners automation system for prison. This project is the dependable entering, exiting and controlling visitors and prisoners in the prison. There are lost of advantages of this program for prisons. The implemented program allows users to add new visitor information, search visitor records, and find information about visitors and prisoners. The system can generate reports about both the prisoner's records and the visitor's records. When the user enters the visit information will be required to take and attach a photo of the visitor to the record by using web cam. This program will be helpful to users and administrators in finding information about visitors and prisoners. This information may then help prisoner officials and control illegal activities.

This project is the first version and has future works. I want to make second version in my master education. It will develop this program with using .NET for online interface web pages. This program will can make database archives and it is possible to take database feedback. I am thinking to make Work log functions for learning which user make records and when user do records, and only administrator will use this function.

REFERENCES

1. Peter Wrights, Beginning Visual Basic 6 Objects, Publishing Wrox Press, August 1998.

2. John Smiley, Learn to Program with Visual Basic 6, 1st ed., Publisher Peer Information, 1998.

3. Valery V Shmeleff, "Visual Basic 6.0 User Guide E-book", http://www.oflameron.ru/

4. Beginning Visual Basic 6 Database Programming online paper. From the World Wide Web: http://www.universaldataacces.org

5. Jeffrey P. McManus , Database Access with Visual Basic 6, Publisher Sams; Pap/Cdr edition, January 1999.

6. Steven Holzner, ADO Programming in Visual Basic 6, Prentice Hall PTR; Pap/Cdr edition (November 24, 1999)

7. Adobe Photoshop CS Help Files.

8. John Connell 's Web Page, http://www.amazon.com.

APPENDIX A
PROGRAM CODE

```
Option Explicit
Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" ( _
   lpvDest As Any, lpvSource As Any, ByVal cbCopy As Long)
Private Type SAFEARRAYBOUND
   cElements As Long
   lLbound As Long
End Type
Private Type SAFEARRAY2D
   cDims As Integer
   fFeatures As Integer
   cbElements As Long
   cLocks As Long
   pvData As Long
   Bounds(0 To 1) As SAFEARRAYBOUND
End Type
Private Declare Function VarPtrArray Lib "msvbvm50.dll" Alias "VarPtr" (Ptr() As Any)
As Long
Private Type RGBQUAD
   rgbBlue As Byte
   rgbGreen As Byte
   rgbRed As Byte
   rgbReserved As Byte
End Type
Private Type BITMAPINFOHEADER '40 bytes
   biSize As Long
   biWidth As Long
   biHeight As Long
   biPlanes As Integer
   biBitCount As Integer
```

```
    biCompression As Long
    biSizeImage As Long
    biXPelsPerMeter As Long
    biYPelsPerMeter As Long
    biClrUsed As Long
    biClrImportant As Long
End Type
Private Type BITMAPINFO
    bmiHeader As BITMAPINFOHEADER
    bmiColors As RGBQUAD
End Type
Private Declare Function CreateCompatibleDC Lib "gdi32" (ByVal hdc As Long) As Long
Private Declare Function GetDC Lib "USER32" (ByVal hwnd As Long) As Long
Private Declare Function GetDesktopWindow Lib "USER32" () As Long
' Note - this is not the declare in the API viewer - modify lplpVoid to be
' Byref so we get the pointer back:
Private Declare Function CreateDIBSection Lib "gdi32" _
    (ByVal hdc As Long, _
    pBitmapInfo As BITMAPINFO, _
    ByVal un As Long, _
    lplpVoid As Long, _
    ByVal handle As Long, _
    ByVal dw As Long) As Long
Private Declare Function BitBlt Lib "gdi32" (ByVal hDestDC As Long, ByVal x As Long,
ByVal y As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal hSrcDC As
Long, ByVal xSrc As Long, ByVal ySrc As Long, ByVal dwRop As Long) As Long
Private Declare Function SelectObject Lib "gdi32" (ByVal hdc As Long, ByVal hObject
As Long) As Long
Private Declare Function DeleteObject Lib "gdi32" (ByVal hObject As Long) As Long
Private Declare Function DeleteDC Lib "gdi32" (ByVal hdc As Long) As Long
Private Declare Function ReleaseDC Lib "USER32" (ByVal hwnd As Long, ByVal hdc As Long) As Long
Private Declare Function LoadImage Lib "USER32" Alias "LoadImageA" (ByVal hInst As Long, ByVal lpsz As String, ByVal un1 As Long, ByVal n1 As Long, ByVal n2 As Long, ByVal un2 As Long) As Long
Private Const BI_RGB = 0&
Private Const BI_RLE4 = 2&
Private Const BI_RLE8 = 1&
Private Const DIB_RGB_COLORS = 0 '  color table in RGBs
Private Type BITMAP
    bmType As Long
    bmWidth As Long
    bmHeight As Long
    bmWidthBytes As Long
    bmPlanes As Integer
    bmBitsPixel As Integer
    bmBits As Long
End Type
Private Declare Function GetObjectAPI Lib "gdi32" Alias "GetObjectA" (ByVal hObject As Long, ByVal nCount As Long, lpObject As Any) As Long
Private Declare Function timeGetTime Lib "winmm.dll" () As Long
Private Declare Function CreateCompatibleBitmap Lib "gdi32" (ByVal hdc As Long, ByVal nWidth As Long, ByVal nHeight As Long) As Long
' Clipboard functions:
Private Declare Function OpenClipboard Lib "USER32" (ByVal hwnd As Long) As Long
Private Declare Function CloseClipboard Lib "USER32" () As Long
Private Declare Function SetClipboardData Lib "USER32" (ByVal wFormat As Long, ByVal hMem As Long) As Long
Private Declare Function EmptyClipboard Lib "USER32" () As Long
Private Const CF_BITMAP = 2
Private Const CF_DIB = 8
```

```
' Handle to the current DIBSection:
Private m_hDIb As Long
' Handle to the old bitmap in the DC, for clear up:
Private m_hBmpOld As Long
' Handle to the Device context holding the DIBSection:
Private m_hDC As Long
' Address of memory pointing to the DIBSection's bits:
Private m_lPtr As Long
' Type containing the Bitmap information:
Private m_tBI As BITMAPINFO
Public Function CopyToClipboard( _
      Optional ByVal bAsDIB As Boolean = True _
   ) As Boolean
Dim lhDCDesktop As Long
Dim lhDC As Long
Dim lhBmpOld As Long
Dim hObj As Long
Dim lFmt As Long
Dim b() As Byte
Dim tBI As BITMAPINFO
Dim lPtr As Long
Dim hDibCopy As Long
   lhDCDesktop = GetDC(GetDesktopWindow())
   If (lhDCDesktop <> 0) Then
      lhDC = CreateCompatibleDC(lhDCDesktop)
      If (lhDC <> 0) Then
        If (bAsDIB) Then
       MsgBox "I don't know how to put a DIB on the clipboard! Copy as bitmap instead!!!"
           ' Create a duplicate DIBSection and copy
           ' to the clipboard:
           'LSet tBI = m_tBI
```

```
  'hDibCopy = CreateDIBSection( _
  '     lhDC, _
  '     m_tBI, _
  '     DIB_RGB_COLORS, _
  '     lPtr, _
  '     0, 0)
  'If (hDibCopy <> 0) Then
  '   lhBmpOld = SelectObject(lhDC, hObj)
  '   BitBlt lhDC, 0, 0, Width, Height, m_hDC, 0, 0, vbSrcCopy
  '   SelectObject lhDC, lhBmpOld
  '   lFmt = CF_DIB
  '
  '     '....
  'Else
  '   hObj = 0
  'End If
Else
  ' Create a compatible bitmap and copy to
  ' the clipboard:
  hObj = CreateCompatibleBitmap(lhDCDesktop, Width, Height)
  If (hObj <> 0) Then
    lhBmpOld = SelectObject(lhDC, hObj)
    PaintPicture lhDC
    SelectObject lhDC, lhBmpOld
    lFmt = CF_BITMAP
    ' Now set the clipboard to the bitmap:
    If (OpenClipboard(0) <> 0) Then
      EmptyClipboard
      If (SetClipboardData(lFmt, hObj) <> 0) Then
        CopyToClipboard = True
      End If
```

```
                CloseClipboard
             End If
          End If
       End If
       DeleteDC lhDC
    End If
    DeleteDC lhDCDesktop
 End If
End Function
Public Function CreateDIB( _
      ByVal lhDC As Long, _
      ByVal lWidth As Long, _
      ByVal lHeight As Long, _
      ByRef hDib As Long _
   ) As Boolean
   With m_tBI.bmiHeader
      .biSize = Len(m_tBI.bmiHeader)
      .biWidth = lWidth
      .biHeight = lHeight
      .biPlanes = 1
      .biBitCount = 24
      .biCompression = BI_RGB
      .biSizeImage = BytesPerScanLine * .biHeight
   End With
   hDib = CreateDIBSection( _
         lhDC, _
         m_tBI, _
         DIB_RGB_COLORS, _
         m_lPtr, _
         0, 0)
   CreateDIB = (hDib <> 0)
```

```
End Function
Public Function CreateFromPicture( _
      ByRef picThis As StdPicture _
   )
Dim lhDC As Long
Dim lhDCDesktop As Long
Dim lhBmpOld As Long
Dim tBMP As BITMAP
Dim lhWnd As Long
   GetObjectAPI picThis.handle, Len(tBMP), tBMP
   If (Create(tBMP.bmWidth, tBMP.bmHeight)) Then
      lhWnd = GetDesktopWindow()
      lhDCDesktop = GetDC(lhWnd)
      If (lhDCDesktop <> 0) Then
         lhDC = CreateCompatibleDC(lhDCDesktop)
         ReleaseDC lhWnd, lhDCDesktop
         If (lhDC <> 0) Then
            lhBmpOld = SelectObject(lhDC, picThis.handle)
            LoadPictureBlt lhDC
            SelectObject lhDC, lhBmpOld
            DeleteDC lhDC
         End If
      End If
   End If
End Function
Public Function Create( _
      ByVal lWidth As Long, _
      ByVal lHeight As Long _
   ) As Boolean
   ClearUp
   m_hDC = CreateCompatibleDC(0)
```

```
      If (m_hDC <> 0) Then
         If (CreateDIB(m_hDC, lWidth, lHeight, m_hDIb)) Then
            m_hBmpOld = SelectObject(m_hDC, m_hDIb)
            Create = True
         Else
            DeleteDC m_hDC
            m_hDC = 0
         End If
   End If
End Function
Public Property Get BytesPerScanLine() As Long
   ' Scans must align on dword boundaries:
   BytesPerScanLine = (m_tBI.bmiHeader.biWidth * 3 + 3) And &HFFFFFFFC
End Property
Public Property Get Width() As Long
   Width = m_tBI.bmiHeader.biWidth
End Property
Public Property Get Height() As Long
   Height = m_tBI.bmiHeader.biHeight
End Property
Public Sub LoadPictureBlt( _
      ByVal lhDC As Long, _
      Optional ByVal lSrcLeft As Long = 0, _
      Optional ByVal lSrcTop As Long = 0, _
      Optional ByVal lSrcWidth As Long = -1, _
      Optional ByVal lSrcHeight As Long = -1, _
      Optional ByVal eRop As RasterOpConstants = vbSrcCopy _
   )
   If lSrcWidth < 0 Then lSrcWidth = m_tBI.bmiHeader.biWidth
   If lSrcHeight < 0 Then lSrcHeight = m_tBI.bmiHeader.biHeight
   BitBlt m_hDC, 0, 0, lSrcWidth, lSrcHeight, lhDC, lSrcLeft, lSrcTop, eRop
```

```
End Sub
Public Sub PaintPicture( _
        ByVal lhDC As Long, _
        Optional ByVal lDestLeft As Long = 0, _
        Optional ByVal lDestTop As Long = 0, _
        Optional ByVal lDestWidth As Long = -1, _
        Optional ByVal lDestHeight As Long = -1, _
        Optional ByVal lSrcLeft As Long = 0, _
        Optional ByVal lSrcTop As Long = 0, _
        Optional ByVal eRop As RasterOpConstants = vbSrcCopy _
    )
    If (lDestWidth < 0) Then lDestWidth = m_tBI.bmiHeader.biWidth
    If (lDestHeight < 0) Then lDestHeight = m_tBI.bmiHeader.biHeight
    BitBlt lhDC, lDestLeft, lDestTop, lDestWidth, lDestHeight, m_hDC, lSrcLeft, lSrcTop,
eRop
End Sub
Public Property Get hdc() As Long
    hdc = m_hDC
End Property
Public Property Get hDib() As Long
    hDib = m_hDIb
End Property
Public Property Get DIBSectionBitsPtr() As Long
    DIBSectionBitsPtr = m_lPtr
End Property
Public Sub RandomiseBits( _
        Optional ByVal bGray As Boolean = False _
    )
Dim bDib() As Byte
Dim x As Long, y As Long
Dim lC As Long
Dim tSA As SAFEARRAY2D
Dim xEnd As Long
    ' Get the bits in the from DIB section:
   With tSA
      .cbElements = 1
      .cDims = 2
      .Bounds(0).lLbound = 0
      .Bounds(0).cElements = m_tBI.bmiHeader.biHeight
      .Bounds(1).lLbound = 0
      .Bounds(1).cElements = BytesPerScanLine()
      .pvData = m_lPtr
   End With
   CopyMemory ByVal VarPtrArray(bDib()), VarPtr(tSA), 4
' random:
   Randomize Timer
   xEnd = (Width - 1) * 3
   If (bGray) Then
      For y = 0 To m_tBI.bmiHeader.biHeight - 1
         For x = 0 To xEnd Step 3
            lC = Rnd * 255
            bDib(x, y) = lC
            bDib(x + 1, y) = lC
            bDib(x + 2, y) = lC
         Next x
      Next y
   Else
      For x = 0 To xEnd Step 3
         For y = 0 To m_tBI.bmiHeader.biHeight - 1
            bDib(x, y) = 0
            bDib(x + 1, y) = Rnd * 255
            bDib(x + 2, y) = Rnd * 255
            Next y
          Next x
      End If
      ' Clear the temporary array descriptor
      ' (This does not appear to be necessary, but
      ' for safety do it anyway)
      CopyMemory ByVal VarPtrArray(bDib), 0&, 4
      End Sub
   Public Sub ClearUp()
      If (m_hDC <> 0) Then
         If (m_hDIb <> 0) Then
            SelectObject m_hDC, m_hBmpOld
            DeleteObject m_hDIb
         End If
         DeleteDC m_hDC
      End If
      m_hDC = 0: m_hDIb = 0: m_hBmpOld = 0: m_lPtr = 0
   End Sub
   Public Function Resample( _
         ByVal lNewHeight As Long, _
         ByVal lNewWidth As Long _
      ) As cDIBSection
   Dim cDib As cDIBSection
      Set cDib = New cDIBSection
      If cDib.Create(lNewWidth, lNewHeight) Then
         If (lNewWidth <> m_tBI.bmiHeader.biWidth) Or (lNewHeight <>
   m_tBI.bmiHeader.biHeight) Then
            ' Change in size, do resample:
            ResampleDib cDib
         Else
            ' No size change so just return a copy:
```

```
        cDib.LoadPictureBlt m_hDC
      End If
      Set Resample = cDib
   End If
End Function
Private Function ResampleDib(ByRef cDibTo As cDIBSection) As Boolean
Dim bDibFrom() As Byte
Dim bDibTo() As Byte
Dim tSAFrom As SAFEARRAY2D
Dim tSATo As SAFEARRAY2D
' Get the bits in the from DIB section:
   With tSAFrom
      .cbElements = 1
      .cDims = 2
      .Bounds(0).lLbound = 0
      .Bounds(0).cElements = m_tBI.bmiHeader.biHeight
      .Bounds(1).lLbound = 0
      .Bounds(1).cElements = BytesPerScanLine()
      .pvData = m_lPtr
   End With
   CopyMemory ByVal VarPtrArray(bDibFrom()), VarPtr(tSAFrom), 4
 ' Get the bits in the to DIB section:
   With tSATo
      .cbElements = 1
      .cDims = 2
      .Bounds(0).lLbound = 0
      .Bounds(0).cElements = cDibTo.Height
      .Bounds(1).lLbound = 0
      .Bounds(1).cElements = cDibTo.BytesPerScanLine()
      .pvData = cDibTo.DIBSectionBitsPtr
   End With
```

```
    CopyMemory ByVal VarPtrArray(bDibTo()), VarPtr(tSATo), 4
Dim xScale As Single
Dim yScale As Single
Dim x As Long, y As Long, xEnd As Long, xOut As Long
Dim fX As Single, fY As Single
Dim ifY As Long, ifX As Long
Dim dX As Single, dy As Single
Dim r As Long, r1 As Single, r2 As Single, r3 As Single, r4 As Single
Dim g As Long, g1 As Single, g2 As Single, g3 As Single, g4 As Single
Dim b As Long, b1 As Single, b2 As Single, b3 As Single, b4 As Single
Dim ir1 As Long, ig1 As Long, ib1 As Long
Dim ir2 As Long, ig2 As Long, ib2 As Long
xScale = (Width - 1) / cDibTo.Width
yScale = (Height - 1) / cDibTo.Height
    xEnd = cDibTo.Width - 1
     For y = 0 To cDibTo.Height - 1
        fY = y * yScale
        ifY = Int(fY)
        dy = fY - ifY
         For x = 0 To xEnd
            fX = x * xScale
            ifX = Int(fX)
            dX = fX - ifX
            ifX = ifX * 3
            ' Interpolate using the four nearest pixels in the source
b1 = bDibFrom(ifX, ifY): g1 = bDibFrom(ifX + 1, ifY): r1 = bDibFrom(ifX + 2, ifY)
b2 = bDibFrom(ifX + 3, ifY): g2 = bDibFrom(ifX + 4, ifY): r2 = bDibFrom(ifX + 5, ifY)
b3 = bDibFrom(ifX, ifY + 1): g3 = bDibFrom(ifX + 1, ifY + 1): r3 = bDibFrom(ifX + 2,
ifY + 1)
b4 = bDibFrom(ifX + 3, ifY + 1): g4 = bDibFrom(ifX + 4, ifY + 1): r4 = bDibFrom(ifX +
5, ifY + 1)
```

```
        ' Interplate in x direction:
  ir1 = r1 * (1 - dy) + r3 * dy: ig1 = g1 * (1 - dy) + g3 * dy: ib1 = b1 * (1 - dy) + b3 * dy
  ir2 = r2 * (1 - dy) + r4 * dy: ig2 = g2 * (1 - dy) + g4 * dy: ib2 = b2 * (1 - dy) + b4 * dy
    ' Interpolate in y:
  r = ir1 * (1 - dX) + ir2 * dX: g = ig1 * (1 - dX) + ig2 * dX: b = ib1 * (1 - dX) + ib2 * dX
        ' Set output:
        If (r < 0) Then r = 0
        If (r > 255) Then r = 255
        If (g < 0) Then g = 0
        If (g > 255) Then g = 255
        If (b < 0) Then b = 0
        If (b > 255) Then
            b = 255
        End If
        xOut = x * 3
        bDibTo(xOut, y) = b
        bDibTo(xOut + 1, y) = g
        bDibTo(xOut + 2, y) = r
        Next x
        Next y
    ' Clear the temporary array descriptor
    ' (This does not appear to be necessary, but  ' for safety do it anyway)
    CopyMemory ByVal VarPtrArray(bDibFrom), 0&, 4
    CopyMemory ByVal VarPtrArray(bDibTo), 0&, 4
End Function
Private Sub Class_Terminate()
ClearUp
End Sub
```

Add New Form:

```
Dim Conn As ADODB.Connection
Dim Re As ADODB.Recordset
Dim CMD As ADODB.Command
Dim LsItem1(), LsItem2(), LsItem3(), LsItem4()
Dim LsCount As Integer
Private Sub Command2_Click()
Load frmPriSelect
frmPriSelect.Show vbModal
If Pname <> Empty Then
Text1(5).Text = Pname
Text1(6).Text = Psurname
End If
End Sub
Private Sub Command1_Click()
Load frmCap
frmCap.Show vbModal
End Sub
Private Sub Command4_Click()
Dim M As ListItem
If Text2(0).Text = Empty Or Text2(1).Text = Empty Then
  MsgBox "Enter all fields!"
  Exit Sub
Else
  If IsNumeric(Text2(1).Text) = False Then
    MsgBox "Amount must be a number!"
    Text2(1).Text = Empty
    Text2(1).SetFocus
    Exit Sub
  End If
  If Command4.Caption = "Add  >>" Then
```

```
      Set M = Ls.ListItems.Add(1, , Text2(0).Text)
      M.SubItems(1) = Text2(1).Text
      M.SubItems(2) = Combo1.Text
      If Option1.Value = True Then M.SubItems(3) = "In" Else M.SubItems(3) = "Out"
      Text2(0).Text = Empty
      Text2(1).Text = Empty
    Else
      Ls.ListItems(Ls.SelectedItem.Index).Text = Text2(0).Text
      Ls.ListItems(Ls.SelectedItem.Index).SubItems(1) = Text2(1).Text
      Ls.ListItems(Ls.SelectedItem.Index).SubItems(2) = Combo1.Text
      If Option1.Value = True Then
        Ls.ListItems(Ls.SelectedItem.Index).SubItems(3) = "In"
      Else
        Ls.ListItems(Ls.SelectedItem.Index).SubItems(3) = "Out"
      End If
    End If
    Text2(0).SetFocus
End If
End Sub
Private Sub Command5_Click()
On Error GoTo e1:
Dim IDn, mName, mSurname, Root
Dim jDIB As cDIBSection
mName = Text1(0).Text
mSurname = Text1(1).Text
Set Re = New ADODB.Recordset
If EditMode = False Then
  Re.Open "table1", Conn, adOpenDynamic, adLockOptimistic
  Re.AddNew
Else
```

```
Re.Open "select * from table1 where ID=" & Zid, Conn, adOpenDynamic,
adLockOptimistic, adCmdUnknown
End If
For i = 0 To 8
  If Text1(i).Text <> Empty Then Re.Fields(i + 1) = Text1(i).Text Else Re.Fields(i + 1) =
" "
Next
Re.Fields(10) = Format(Date, "dd.mm.yyyy")
Re.Fields(11) = Format(Time, "hh:mm")
Re.Update
Re.Close
Re.Open "select ID from table1 order by ID", Conn, adOpenStatic, adLockReadOnly,
adCmdUnknown
Re.MoveLast
Set jDIB = New cDIBSection
jDIB.CreateFromPicture Image1.Picture
If Image1.Picture = Empty Then
  Image1.Picture = picOutput.Image
End If
If EditMode = True Then
  If SaveJPG(jDIB, App.Path & "\Data\" & Zid & ".jpg", 80) Then
  Else
    MsgBox "Failed to save picture."
  End If
  'SavePicture picOutput.Image, App.Path & "\Data\" & Zid & ".bmp"
Else
  If SaveJPG(jDIB, App.Path & "\Data\" & Re.Fields("ID") & ".jpg", 80) Then
  Else
    MsgBox "Failed to save picture."
  End If
  'SavePicture picOutput.Image, App.Path & "\Data\" & Re.Fields("ID") & ".bmp"
```

```
End If
jDIB.ClearUp
Set jDIB = Nothing
Re.Close
'Command6_Click
Re.Open "select ID from table1 order by ID", Conn, adOpenStatic, adLockReadOnly,
adCmdUnknown
Re.MoveLast
IDn = Re.Fields(0)
Re.Close
ReDim LsItem1(Ls.ListItems.Count)
ReDim LsItem2(Ls.ListItems.Count)
ReDim LsItem3(Ls.ListItems.Count)
ReDim LsItem4(Ls.ListItems.Count)
For i = 1 To Ls.ListItems.Count
  LsItem1(i) = Ls.ListItems(i).Text
  LsItem2(i) = Ls.ListItems(i).SubItems(1)
  LsItem3(i) = Ls.ListItems(i).SubItems(2)
  LsItem4(i) = Ls.ListItems(i).SubItems(3)
Next
LsCount = Ls.ListItems.Count
If LsCount <> 0 Then
  If EditMode = True Then
    Set CMD = New ADODB.Command
    CMD.ActiveConnection = Conn
    CMD.CommandText = "delete * from table2 where ID=" & Zid
    CMD.Execute
    CMD.Cancel
    Set CMD = Nothing
  End If
```

```
        Set Re = New ADODB.Recordset
        Re.Open "table2", Conn, adOpenDynamic, adLockOptimistic
        For i = 1 To LsCount
            Re.AddNew
            Re.Fields(0) = IDn
            Re.Fields(1) = mName
            Re.Fields(2) = mSurname
            Re.Fields(3) = LsItem1(i)
            Re.Fields(4) = LsItem2(i)
            Re.Fields(5) = LsItem3(i)
            Re.Fields(6) = LsItem4(i)
            Re.Update
        Next
        Re.Close
        Set Re = Nothing
    End If
    Set Re = Nothing
    Unload Me
    Exit Sub
el:
    MsgBox Err.Description
End Sub
Private Sub Command6_Click()
For i = 0 To 8
    Text1(i).Text = Empty
    Text1(i).Enabled = False
Next
'Command2.Enabled = True
'Command5.Enabled = False
'Command6.Enabled = False
'MS1.Enabled = True
```

```
Image1.Picture = Nothing
Ls.ListItems.Clear
'Load frmMain
'frmMain.Show
Unload Me
End Sub
Private Sub Command7_Click()
Dim M As ListItem
If Text2(2).Text = Empty Or IsNumeric(Text2(2).Text) = False Then
  MsgBox "Please check data!"
  Text2(2).Text = Empty
  Text2(2).SetFocus
  Exit Sub
Else
  If Command7.Caption = "Add  >>" Then
    Set M = Ls.ListItems.Add(1, , "Cash")
    M.SubItems(1) = Text2(2).Text
    M.SubItems(2) = "YTL"
    M.SubItems(3) = "In"
    Text2(2).Text = Empty
  Else
    Ls.ListItems(Ls.SelectedItem.Index).SubItems(1) = Text2(2).Text
  End If
End If
End Sub
Private Sub Form_Load()
Dim r As ListItem
Set Conn = New ADODB.Connection
Conn.Open    "Provider=Microsoft.Jet.OLEDB.4.0;Data    Source="    &    App.Path    &
"\Data.mdb;Persist Security Info=False"
Combo1.ListIndex = 0
```

```
If EditMode = True Then
  Set Re = New ADODB.Recordset
  Re.Open "select * from table1 where ID=" & Zid, Conn, adOpenStatic,
adLockReadOnly, adCmdUnknown
  Image1.Picture = LoadPicture(App.Path & "\Data\" & Re.Fields(0) & ".JPG")
  picOutput.Picture = LoadPicture(App.Path & "\Data\" & Re.Fields(0) & ".JPG")
  For i = 0 To 8
    If IsNull(Re.Fields(i + 1)) = False Then Text1(i).Text = Re.Fields(i + 1) Else
Text1(i).Text = Empty
  Next
  Re.Close
  Re.Open "select * from table2 where ID=" & Zid, Conn, adOpenStatic,
adLockReadOnly, adCmdUnknown
  If Re.RecordCount <> 0 Then
    While Not Re.EOF
      Set r = Ls.ListItems.Add(1, , Re.Fields(3))
      r.SubItems(1) = Re.Fields(4)
      r.SubItems(2) = Re.Fields(5)
      r.SubItems(3) = Re.Fields(6)
      Re.MoveNext
    Wend
  End If
End If
End Sub
Private Sub Ls_DblClick()
If Ls.ListItems.Count <> 0 Then
  LSid = Ls.SelectedItem.Index
  If Ls.ListItems(Ls.SelectedItem.Index).SubItems(2) = "YTL" Then
    Load frmMonEdit
    frmMonEdit.Text2(2).Text = Ls.ListItems(Ls.SelectedItem.Index).SubItems(1)
```

```
        frmMonEdit.Show vbModal
    Else
        Load frmObjEdit
        frmObjEdit.Text2(0).Text = Ls.ListItems(Ls.SelectedItem.Index).Text
        frmObjEdit.Text2(1).Text = Ls.ListItems(Ls.SelectedItem.Index).SubItems(1)
        frmObjEdit.Combo1.Text = Ls.ListItems(Ls.SelectedItem.Index).SubItems(2)
        If Ls.ListItems(Ls.SelectedItem.Index).SubItems(3) = "In" Then
            frmObjEdit.Option1.Value = True
        ElseIf Ls.ListItems(Ls.SelectedItem.Index).SubItems(3) = "Out" Then
            frmObjEdit.Option2.Value = True
        End If
        frmObjEdit.Show vbModal
    End If
    ReDim LsItem1(Ls.ListItems.Count)
    ReDim LsItem2(Ls.ListItems.Count)
    ReDim LsItem3(Ls.ListItems.Count)
    ReDim LsItem4(Ls.ListItems.Count)
    For i = 1 To Ls.ListItems.Count
        LsItem1(i) = Ls.ListItems(i).Text
        LsItem2(i) = Ls.ListItems(i).SubItems(1)
        LsItem3(i) = Ls.ListItems(i).SubItems(2)
        LsItem4(i) = Ls.ListItems(i).SubItems(3)
    Next
    LsCount = Ls.ListItems.Count
End If
End Sub
Private Sub Text1_GotFocus(Index As Integer)
Command5.Default = True
End Sub
Private Sub Text2_GotFocus(Index As Integer)
If Index = 0 Or Index = 1 Then
```

```
  Command4.Default = True
Else
  Command7.Default = True
End If
End Sub
```

Form Capture:

```
Private Sub Command1_Click()
frmAddNew.Image1.Picture = picOutput.Image
frmAddNew.picOutput.Picture = picOutput.Image
End Sub
Private Sub Command2_Click()
Unload Me
End Sub
Private Sub Form_Load()
'cmdStart.Enabled = False
'cmdStop.Enabled = True
mCapHwnd = capCreateCaptureWindow("WebcamCapture", 0, 0, 0, 320, 240, Me.hwnd,
0)
DoEvents: SendMessage mCapHwnd, CONNECT, 0, 0
tmrMain.Enabled = True
End Sub
Private Sub tmrMain_Timer()
On Error Resume Next
    SendMessage mCapHwnd, GET_FRAME, 0, 0
    SendMessage mCapHwnd, COPY, 0, 0
    picOutput.Picture = Clipboard.GetData
    Clipboard.Clear
End Sub
```

Form Find:

```
Private Sub Check1_Click()
If Check1.Value = vbChecked Then
   DT1.Enabled = True
Else
   DT1.Enabled = False
End If
End Sub
Private Sub Command1_Click()
Adodc1.RecordSource = "select * from table1 where date=datevalue('" & DT1.Value & "')"
Adodc1.Refresh
MS1.Refresh
End Sub
Private Sub Form_Load()
DT1.Value = Date
DT1.Enabled = False
Adodc1.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
App.Path & "\Data.mdb;Persist Security Info=False"
Adodc1.RecordSource = "select * from table1" ' where date=datevalue('" & Date & "')"
Adodc1.Refresh
End Sub
Private Sub MS1_DblClick()
If MS1.Text <> Empty Then
   EditMode = True
   Zid = MS1.Text
   Zname = MS1.TextMatrix(MS1.Row, 1)
   Zsurname = MS1.TextMatrix(MS1.Row, 2)
   Load frmAddNew
   'Me.Hide
   frmAddNew.Show vbModal, MDIForm1
   'Unload Me
```

```
End If
End Sub
Private Sub Text1_KeyUp(KeyCode As Integer, Shift As Integer)
Dim D
If Check1.Value = vbChecked Then
   D = " and date=datevalue('" & DT1.Value & "')"
Else
   D = Empty
End If
Adodc1.RecordSource = "select * from table1 where name like '" & Text1.Text & "%'"& D
Adodc1.Refresh
End Sub
```

Form Login:

```
Dim Conn As ADODB.Connection
Dim Re As ADODB.Recordset
Private Sub cmdCancel_Click()
End
End Sub
Private Sub cmdOK_Click()
Dim a
Set Re = New ADODB.Recordset
Re.Open "select username,password from users where username='" & Text1.Text & "'",
Conn, adOpenStatic, adLockReadOnly, adCmdUnknown
If Re.RecordCount = 0 Then
   a = MsgBox("User Name Not Found! Try again?", vbExclamation + vbYesNo)
   If a = vbYes Then
      Text1.Text = Empty
      Text2.Text = Empty
      Text1.SetFocus
     Exit Sub
```

```
    Else
      End
    End If
  Else
    If UCase(Re!Password) <> UCase(Text2.Text) Then
      a = MsgBox("Wrong password! Try again?", vbExclamation + vbYesNo)
      If a = vbYes Then
        Text2.Text = Empty
        Text2.SetFocus
        Exit Sub
      Else
        End
      End If
    Else
      MDIForm1.Picture1.Visible = True
      MDIForm1.Enabled = True
      Unload Me
      Load frmMain
      frmMain.Show
    End If
End If
End Sub
Private Sub Form_Load()
Set Conn = New ADODB.Connection
Conn.Open    "Provider=Microsoft.Jet.OLEDB.4.0;Data    Source="    &    App.Path    &
"\Data.mdb;Persist Security Info=False"
End Sub
```

Form Main:

```
Dim Conn As ADODB.Connection
Dim Re As ADODB.Recordset
```

```
Private Sub cmdStart_Click()
cmdStart.Enabled = False
cmdStop.Enabled = True
mCapHwnd = capCreateCaptureWindow("WebcamCapture", 0, 0, 0, 320, 240, Me.hwnd,
0)
DoEvents: SendMessage mCapHwnd, CONNECT, 0, 0
tmrMain.Enabled = True
End Sub
Private Sub cmdStop_Click()
cmdStart.Enabled = True
cmdStop.Enabled = False
tmrMain.Enabled = False
DoEvents: SendMessage mCapHwnd, DISCONNECT, 0, 0
End Sub
Private Sub Command1_Click()
Image1.Picture = picOutput.Image
'cmdStop_Click
End Sub
Private Sub Command2_Click()
'cmdStart_Click
Image1.Picture = Nothing
For i = 0 To 8
   Text1(i).Enabled = True
   Text1(i).Text = Empty
Next
Command2.Enabled = False
Command5.Enabled = True
Command6.Enabled = True
EditMode = False
Text1(0).SetFocus
MS1.Enabled = False
```

```
End Sub
Private Sub Command5_Click()
Set Re = New ADODB.Recordset
If EditMode = False Then
   Re.Open "table1", Conn, adOpenDynamic, adLockOptimistic
   Re.AddNew
Else
   Re.Open "select * from table1 where ID='" & MS1.Text & "'", Conn, adOpenDynamic,
adLockOptimistic, adCmdUnknown
End If
For i = 0 To 8
   If Text1(i).Text <> Empty Then Re.Fields(i + 1) = Text1(i).Text Else Re.Fields(i + 1) =
" "
Next
Re.Fields(9) = Format(Date, "dd.mm.yyyy")
Re.Fields(10) = Format(Time, "hh:mm")
Re.Update
Re.Close
Adodc1.Refresh
Re.Open "select ID from table1 order by ID", Conn, adOpenStatic, adLockReadOnly,
adCmdUnknown
Re.MoveLast
If Image1.Picture = Empty Then
   Image1.Picture = picOutput.Image
End If
SavePicture picOutput.Image, App.Path & "\Data\" & Re.Fields("ID") & ".bmp"
Re.Close
Set Re = Nothing
Command6_Click
Timer1.Enabled = True
End Sub
```

```
Private Sub Command6_Click()
For i = 0 To 8
   Text1(i).Text = Empty
   Text1(i).Enabled = False
Next
Command2.Enabled = True
Command5.Enabled = False
Command6.Enabled = False
MS1.Enabled = True
Image1.Picture = Nothing
End Sub
Private Sub en1_Click()
Zid = MS1.Text
Load frmOut
frmOut.Show vbModal
Adodc1.Refresh
End Sub
Private Sub Form_Load()
Set Conn = New ADODB.Connection
Conn.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & App.Path &
"\Data.mdb;Persist Security Info=False"
Adodc1.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
App.Path & "\Data.mdb;Persist Security Info=False"
Adodc1.RecordSource = "select * from table1 where date=datevalue('" & Date & "')"
Adodc1.Refresh
MS1.ColWidth(0) = 100
End Sub
Private Sub MS1_Click()
If MS1.Text <> Empty Then
   Set Re = New ADODB.Recordset
```

```
 Re.Open "select * from table1 where ID=" & MS1.Text, Conn, adOpenStatic,
adLockReadOnly, adCmdUnknown
  Image1.Picture = LoadPicture(App.Path & "\Data\" & Re.Fields(0) & ".JPG")
  For i = 0 To 8
    If IsNull(Re.Fields(i + 1)) = False Then Text1(i).Text = Re.Fields(i + 1) Else
Text1(i).Text = Empty
  Next
  Zid = MS1.Text
  Zname = MS1.TextMatrix(MS1.Row, 1)
  Zsurname = MS1.TextMatrix(MS1.Row, 2)
End If
End Sub
Private Sub MS1_DblClick()
If MS1.Text <> Empty Then
 EditMode = True
 Zid = MS1.Text
 Zname = MS1.TextMatrix(MS1.Row, 1)
 Zsurname = MS1.TextMatrix(MS1.Row, 2)
 Load frmAddNew
 frmAddNew.Show vbModal, MDIForm1
 End If
End Sub
Private Sub MS1_MouseUp(Button As Integer, Shift As Integer, x As Single, y As Single)
If MS1.TextMatrix(MS1.Row, 13) = Empty Then
  If Button = vbRightButton Then
    PopupMenu en
  End If
End If
End Sub
Private Sub Timer1_Timer()
Adodc1.Refresh
```

```
End Sub
```

Form Money Edit:

```
Private Sub Command1_Click()
If Text2(2).Text <> Empty Then
  If IsNumeric(Text2(2).Text) = True Then
    frmAddNew.Ls.ListItems(LSid).SubItems(1) = Text2(2).Text
    Unload Me
  Else
    MsgBox "Check Data!", vbCritical
    Text2(2).Text = Empty
    Text2(2).SetFocus
    Exit Sub
  End If
Else
  MsgBox "Check Data!", vbCritical
  Text2(2).Text = Empty
  Text2(2).SetFocus
  Exit Sub
End If
End Sub
Private Sub Command2_Click()
If MsgBox("are you sure!", vbYesNo + vbQuestion) = vbYes Then
  frmAddNew.Ls.ListItems.Remove (LSid)
  Unload Me
End If
End Sub

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

Form Object Edit:

```
Private Sub Command1_Click()
If Text2(0).Text <> Empty Or Text2(1).Text <> Empty Then
  If IsNumeric(Text2(1).Text) = True Then
    frmAddNew.Ls.ListItems(LSid).Text = Text2(0).Text
    frmAddNew.Ls.ListItems(LSid).SubItems(1) = Text2(1).Text
    frmAddNew.Ls.ListItems(LSid).SubItems(2) = Combo1.Text
    If Option1.Value = True Then
      frmAddNew.Ls.ListItems(LSid).SubItems(3) = "In"
    Else
      frmAddNew.Ls.ListItems(LSid).SubItems(3) = "Out"
    End If
    Unload Me
  Else
    MsgBox "Check Data!", vbCritical
    Text2(0).Text = Empty
    Text2(0).SetFocus
    Exit Sub
  End If
Else
  MsgBox "Check Data!", vbCritical
  Text2(0).Text = Empty
  Text2(0).SetFocus
  Exit Sub
End If
End Sub
Private Sub Command2_Click()
If MsgBox("are you sure!", vbYesNo + vbQuestion) = vbYes Then
  frmAddNew.Ls.ListItems.Remove (LSid)
  Unload Me
```

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

Form Out:

```
Dim Conn As ADODB.Connection
Dim Re As ADODB.Recordset
Private Sub Command1_Click()
Re.Fields(12) = Time
Re.Update
frmMain.Adodc1.Refresh
Unload Me
End Sub
Private Sub Command2_Click()
Unload Me
End Sub
Private Sub Form_Load()
Set Conn = New ADODB.Connection
Conn.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & App.Path & "\Data.mdb;Persist Security Info=False"
Set Re = New ADODB.Recordset
 Re.Open "select * from table1 where ID=" & Zid, Conn, adOpenDynamic, adLockOptimistic, adCmdUnknown
 Image1.Picture = LoadPicture(App.Path & "\Data\" & Re.Fields(0) & ".JPG")
If IsNull(Re.Fields(1)) = False Then Text1(0).Text = Re.Fields(1) Else Text1(0).Text = Empty
 If IsNull(Re.Fields(2)) = False Then Text1(1).Text = Re.Fields(2) Else Text1(0).Text = Empty
If IsNull(Re.Fields(6)) = False Then Text1(5).Text = Re.Fields(6) Else Text1(5).Text = Empty
If IsNull(Re.Fields(7)) = False Then Text1(6).Text = Re.Fields(7) Else Text1(6).Text = Empty
End Sub
```

Form Prison:

```
Dim Conn As ADODB.Connection
Dim Re As ADODB.Recordset
Private Sub Command1_Click()
Load frmPriAdd
frmPriAdd.Show vbModal, MDIForm1
End Sub
Private Sub Command2_Click()
Load frmPriFind
frmPriFind.Show vbModal
EditMode2 = False
End Sub
Private Sub Form_Load()
Set Conn = New ADODB.Connection
Conn.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & App.Path & "\Data.mdb;Persist Security Info=False"
Adodc1.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & App.Path & "\Data.mdb;Persist Security Info=False"
Adodc1.RecordSource = "select * from pri order by name,surname"
Adodc1.Refresh
MS1.ColWidth(0) = 100
End Sub
Private Sub MS1_DblClick()
EditMode2 = True
Pid = MS1.Text
```

```
Load frmPriAdd
frmPriAdd.Show vbModal
Timer1.Enabled = True
EditMode2 = False
End Sub
Private Sub Timer1_Timer()
Adodc1.RecordSource = "select * from pri order by name,surname"
Adodc1.Refresh
End Sub
```

Form Prison Add:

```
Dim Conn As ADODB.Connection
Dim Re As ADODB.Recordset
Private Sub Command1_Click(Index As Integer)
If Index = 0 Then
  Set Re = New ADODB.Recordset
  If EditMode2 = False Then
    Re.Open "pri", Conn, adOpenDynamic, adLockOptimistic
    Re.AddNew
  Else
    Re.Open "select * from pri where ID=" & Pid, Conn, adOpenDynamic,
adLockOptimistic, adCmdUnknown
  End If
  Re.Fields(1) = Text1(0).Text
  Re.Fields(2) = Text1(1).Text
  Re.Fields(3) = DTPicker1(0).Value
  Re.Fields(4) = DTPicker1(1).Value
  Re.Fields(5) = Combo1.Text
  Re.Fields(6) = Text1(2).Text
  Re.Update
  Re.Close
```

```
End If
   frmPri.Adodc1.Refresh
Unload Me
End Sub
Private Sub Command2_Click()
If MsgBox("Are you sure? YES/NO", vbQuestion + vbYesNo) = vbYes Then
   Set Re = New ADODB.Recordset
   Re.Open "delete * from pri where ID=" & Pid, Conn, adOpenDynamic,
adLockOptimistic, adCmdUnknown
   frmPri.Adodc1.Refresh
   Unload Me
   End If
End Sub
Private Sub Form_Load()
Combo1.ListIndex = 0
Set Conn = New ADODB.Connection
Conn.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & App.Path &
"\Data.mdb;Persist Security Info=False"
If EditMode2 = True Then
   Set Re = New ADODB.Recordset
   Re.Open "select * from pri where ID=" & Pid, Conn, adOpenStatic, adLockReadOnly,
adCmdUnknown
   For i = 0 To 1
      If IsNull(Re.Fields(i + 1)) = False Then Text1(i).Text = Re.Fields(i + 1) Else
Text1(i).Text = Empty
   Next
   If IsNull(Re.Fields(6)) = False Then Text1(2).Text = Re.Fields(6) Else Text1(2).Text =
Empty
   Combo1.Text = Re.Fields(5)
   DTPicker1(0).Value = Re.Fields(3)
   DTPicker1(1).Value = Re.Fields(4)
```

```
    Re.Close
    Command2.Enabled = True
End If
```

Form Find Prison:

```
Private Sub Check1_Click()
If Check1.Value = vbChecked Then
    Combo1.Enabled = True
Else
    Combo1.Enabled = False
End If
End Sub
Private Sub Command1_Click()
Adodc1.RecordSource = "select * from pri order by name,surname"
Adodc1.Refresh
MS1.Refresh
End Sub
Private Sub Form_Load()
Adodc1.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
App.Path & "\Data.mdb;Persist Security Info=False"
Adodc1.RecordSource = "select * from pri order by name, surname" ' where
date=datevalue('" & Date & "')"
Adodc1.Refresh
End Sub
Private Sub MS1_DblClick()
If MS1.Text <> Empty Then
    EditMode2 = True
    Pid = MS1.Text
    Load frmPriAdd
    frmPriAdd.Show vbModal, MDIForm1
End If
End Sub
Private Sub Text1_KeyUp(KeyCode As Integer, Shift As Integer)
Dim D
If Check1.Value = vbChecked Then
   D = " and status='" & Combo1.Text & "')"
Else
   D = Empty
End If
Adodc1.RecordSource = "select * from pri where name like '" & Text1.Text & "%'" & D
Adodc1.Refresh
End Sub
Private Sub Timer1_Timer()
Adodc1.Refresh
End Sub
End Sub
```

Form Prison Select:

```
Private Sub Command1_Click()
Pname = Empty
Psurname = Empty
Unload Me
End Sub
Private Sub Command2_Click()
If MS1.Text <> Empty Then
   Pname = MS1.TextMatrix(MS1.Row, 2)
   Psurname = MS1.TextMatrix(MS1.Row, 3)
   Unload Me
End If
End Sub
Private Sub Form_Load()
```

```
Adodc1.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
App.Path & "\Data.mdb;Persist Security Info=False"
Adodc1.RecordSource = "select ID,name,surname from pri order by name, surname" '
where date=datevalue('" & Date & "')"
Adodc1.Refresh
End Sub
Private Sub MS1_DblClick()
If MS1.Text <> Empty Then
   Pname = MS1.TextMatrix(MS1.Row, 2)
   Psurname = MS1.TextMatrix(MS1.Row, 3)
   Unload Me
End If
End Sub
Private Sub Text1_KeyUp(KeyCode As Integer, Shift As Integer)
Dim D
Adodc1.RecordSource = "select * from pri where name like '" & Text1.Text & "%' and
status='Normal'"
Adodc1.Refresht
End Sub
```

Form Reports:

```
Dim D1, D2, D3, D4 As Date
Private Sub Check1_Click()
End Sub
Private Sub Check2_Click()
If Check2.Value = vbChecked Then
   DTPicker1(0).Enabled = True
   DTPicker1(1).Enabled = True
Else
   DTPicker1(0).Enabled = False
   DTPicker1(1).Enabled = False
```

```
End If
End Sub
Private Sub Command1_Click()
Dim x
If Combo1.Text = "All" Then
   x = Empty
Else
   x = "where status ='" & Combo1.Text & "'"
End If
Adodc1.RecordSource = "select ID, Name ,Surname ,Indate as [In Date],outdate as [Out
Date],Status from Pri " & x
Adodc1.Refresh
End Sub
Private Sub Command2_Click()
Set DataReport2.DataSource = Adodc1
DataReport2.DataMember = Adodc1.Recordset.DataMember
DataReport2.Show
End Sub
Private Sub Command3_Click()
Set DataReport1.DataSource = Adodc1
DataReport1.DataMember = Adodc1.Recordset.DataMember
DataReport1.Show
End Sub
Private Sub Command4_Click()
   D3 = DTPicker1(2).Value
   D4 = DTPicker1(3).Value
'If Check1(1).Value = vbChecked Then
'   D1 = DTPicker1(0).Value
'   D2 = DTPicker1(1).Value
'Else
'   D1 = DTPicker1(0).Value
'   D2 = DTPicker1(1).Value
'End If
Adodc1.RecordSource = "select ID, name as [Visitor Name],surname as [Visitor Surname],toname as [Prisoner Name],tosurname as [Prisoner Surname],date as [Date], time as [Visit Time], out as [Out Time] from table1 where date between datevalue('" & D3 & "') and datevalue('" & D4 & "')"
Adodc1.Refresh
End Sub
Private Sub DTPicker1_Click(Index As Integer)
If Index = 0 Then
  DTPicker1(1).Value = DTPicker1(0).Value
End If
End Sub
Private Sub DTPicker1_CloseUp(Index As Integer)
If Index = 0 Then
  DTPicker1(1).Value = DTPicker1(0).Value
End If
D1 = DTPicker1(0).Value
D2 = DTPicker1(1).Value
End Sub
Private Sub DTPicker1_KeyPress(Index As Integer, KeyAscii As Integer)
If Index = 0 Then
  DTPicker1(1).Value = DTPicker1(0).Value
End If
D1 = DTPicker1(0).Value
D2 = DTPicker1(1).Value
End Sub
Private Sub Form_Load()
Combo1.ListIndex = 0
Adodc1.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & App.Path & "\Data.mdb;Persist Security Info=False"
```

```
End Sub
Private Sub Option1_Click()
If Option1.Value = True Then
   Frame1.Enabled = True
   Frame2.Enabled = False
Else
   Frame1.Enabled = False
   Frame2.Enabled = True
End If
End Sub
Private Sub Option2_Click()
If Option2.Value = True Then
   Frame2.Enabled = True
   Frame1.Enabled = False
Else
   Frame2.Enabled = False
   Frame1.Enabled = False
End If
End Sub
```

Form Users:

```
Dim Conn As ADODB.Connection
Dim Re As ADODB.Recordset
Dim User
Private Sub Command1_Click()
Unload Me
End Sub
Private Sub Command2_Click()
For i = 0 To 4
   If Text1(i).Text = Empty Then
      MsgBox "Please enter all fields!", vbExclamation
```

```
        Text1(1).SetFocus
        Exit Sub
      End If
    Next
    If IsNumeric(Text1(0).Text) = True Then
      MsgBox "You can not use numeric user names!", vbCritical
      Text1(0).Text = Empty
      Text1(0).SetFocus
      Exit Sub
    End If
    Set Re = New ADODB.Recordset
    Re.Open "select * from users where username='" & Text1(0).Text & "'", Conn,
adOpenStatic, adLockReadOnly, adCmdUnknown
    If Re.RecordCount = 0 Then
      If Text1(3).Text <> Text1(4).Text Then
        MsgBox "Check the password!", vbExclamation
        Text1(3).Text = Empty
        Text1(4).Text = Empty
        Text1(3).SetFocus
        Exit Sub
      End If
      Set Re = New ADODB.Recordset
      Re.Open ("users"), Conn, adOpenDynamic, adLockOptimistic, adCmdTable
      Re.AddNew
      For i = 0 To 3
        Re.Fields(i) = Text1(i).Text
        Text1(i).Text = Empty
      Next
      Text1(4).Text = Empty
      Re.Fields(4) = Combo1.Text
      Re.Update
```

```
Else
   MsgBox "This User Name already exist!", vbCritical
   Text1(0).Text = Empty
   Text1(0).SetFocus
End If
RefreshList
End Sub
Private Sub Command3_Click()
If LCase(List1.Text) = "admin" And LCase(Text1(0).Text) <> "admin" Then
   MsgBox "You can not change [admin] user name!", vbCritical
   Exit Sub
ElseIf List1.Text = "admin" And Combo1.Text <> "Administrator" Then
   MsgBox "You can not change this user type!", vbCritical
   Exit Sub
End If
For i = 0 To 4
   If Text1(i).Text = Empty Then
      MsgBox "Please enter all fields!", vbExclamation
      Text1(i).SetFocus
      Exit Sub
   End If
Next
If IsNumeric(Text1(0).Text) = True Then
   MsgBox "You can not use numeric user names!", vbCritical
   Text1(0).Text = Empty
   Text1(0).SetFocus
   Exit Sub
End If
Set Re = New ADODB.Recordset
Re.Open "select * from users where username='" & Text1(0).Text & "'", Conn,
adOpenDynamic, adLockOptimistic, adCmdUnknown
```

```
If LCase(Text1(0).Text) = LCase(List1.Text) And Re.RecordCount <> 0 Then
MsgBox "This User Name already exist!", vbCritical
   Text1(0).Text = Empty
   Text1(0).SetFocus
   Exit Sub
Else
If Text1(3).Text <> Text1(4).Text Then
      MsgBox "Check the password!", vbExclamation
      Text1(3).Text = Empty
      Text1(4).Text = Empty
      Text1(3).SetFocus
      Exit Sub
   End If
   Set Re = New ADODB.Recordset
   Re.Open "select * from users where username='" & List1.Text & "'", Conn,
adOpenKeyset, adLockOptimistic, adCmdUnknown
   For i = 0 To 3
      Re.Fields(i) = Text1(i).Text
   Next
   Re.Fields(4) = Combo1.Text
   Re.Update
   RefreshList
End If
Command3.Enabled = False
End Sub
Private Sub Command4_Click()
If LCase(List1.Text) <> "admin" Then
   Set Re = New ADODB.Recordset
   Re.Open "select * from users where username='" & List1.Text & "'", Conn,
adOpenDynamic, adLockOptimistic, adCmdUnknown
   Re.Delete
```

```
  RefreshList
   For i = 0 To 4
      Text1(i).Text = Empty
   Next
Else
   MsgBox "You can not delete [admin] user.", vbCritical
End If
End Sub
Private Sub Command5_Click()
For i = 0 To 4
   Text1(i).Text = Empty
Next
Text1(1).SetFocus
End Sub
Private Sub Form_Load()
Set Conn = New ADODB.Connection
Conn.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & App.Path &
"\Data.mdb;Persist Security Info=False"
RefreshList
End Sub
Private Sub List1_Click()
On Error Resume Next
Set Re = New ADODB.Recordset
Re.Open "select * from users where username='" & List1.Text & "'", Conn
For i = 0 To 3
   Text1(i).Text = Re.Fields(i)
Next
Text1(4).Text = Re.Fields(3)
Combo1.Text = Re.Fields(4)
User = Text1(0).Text
Command3.Enabled = True
```

```
End Sub
Private Sub RefreshList()
List1.Clear
Set Re = New ADODB.Recordset
Re.Open "select * from users order by username", Conn
If Re.RecordCount <> 0 Then
   Re.MoveFirst
   While Not Re.EOF
      List1.AddItem Re!UserName
      Re.MoveNext
   Wend
End If
End Sub
```

mCAPTURE:

```
Option Explicit
'*************************************************************
' mCapture
'
' Written By   : Shawn K. Hall [Reliable Answers.com]
'              :
' Description  : Visual Basic Screen Capture Routines
'              :
' Requires     : Reference to "Standard OLE Types"
'*************************************************************
'[Types]
Private Type PALETTEENTRY
peRed As Byte
peGreen As Byte
peBlue As Byte
peFlags As Byte
```

```
End Type
Private Type LOGPALETTE
palVersion As Integer
palNumEntries As Integer
palPalEntry(255) As PALETTEENTRY
End Type
Private Type GUID
   Data1 As Long
   Data2 As Integer
   Data3 As Integer
   Data4(7) As Byte
End Type
Private Type RECT
  Left As Long
  Top As Long
  Right As Long
  Bottom As Long
End Type
Private Type PicBmp
  Size As Long
  Type As Long
  hBmp As Long
  hPal As Long
  Reserved As Long
End Type
'[Declares]
Private Declare Function CreateCompatibleDC _
  Lib "gdi32" _
  (ByVal hdc As Long) _
    As Long
Private Declare Function CreateCompatibleBitmap _
```

```
  Lib "gdi32" _
   (ByVal hdc As Long, _
    ByVal nWidth As Long, _
    ByVal nHeight As Long) _
      As Long
Private Declare Function GetDeviceCaps _
  Lib "gdi32" _
   (ByVal hdc As Long, _
    ByVal iCapabilitiy As Long) _
      As Long
Private Declare Function GetSystemPaletteEntries _
  Lib "gdi32" _
   (ByVal hdc As Long, _
    ByVal wStartIndex As Long, _
    ByVal wNumEntries As Long, _
    lpPaletteEntries As PALETTEENTRY) _
      As Long
Private Declare Function CreatePalette _
  Lib "gdi32" _
   (lpLogPalette As LOGPALETTE) _
      As Long
Private Declare Function SelectObject _
  Lib "gdi32" _
   (ByVal hdc As Long, _
    ByVal hObject As Long) _
      As Long
Private Declare Function BitBlt _
  Lib "gdi32" _
   (ByVal hDCDest As Long, _
    ByVal XDest As Long, _
    ByVal YDest As Long, _
    ByVal nWidth As Long, _
    ByVal nHeight As Long, _
    ByVal hDCSrc As Long, _
    ByVal xSrc As Long, _
    ByVal ySrc As Long, _
    ByVal dwRop As Long) _
     As Long
Private Declare Function DeleteDC _
  Lib "gdi32" _
  (ByVal hdc As Long) _
     As Long
Private Declare Function GetForegroundWindow _
  Lib "USER32" () _
     As Long
Private Declare Function SelectPalette _
  Lib "gdi32" _
  (ByVal hdc As Long, _
    ByVal hPalette As Long, _
    ByVal bForceBackground As Long) _
     As Long
Private Declare Function RealizePalette _
  Lib "gdi32" _
  (ByVal hdc As Long) _
     As Long
Private Declare Function GetWindowDC _
  Lib "USER32" _
  (ByVal hwnd As Long) _
     As Long
Private Declare Function GetDC _
  Lib "USER32" _
  (ByVal hwnd As Long) _
```

```
       As Long
   Private Declare Function GetWindowRect _
     Lib "USER32" _
     (ByVal hwnd As Long, _
      lpRect As RECT) _
       As Long
   Private Declare Function ReleaseDC _
     Lib "USER32" _
     (ByVal hwnd As Long, _
      ByVal hdc As Long) _
       As Long
   Private Declare Function GetDesktopWindow _
     Lib "USER32" () _
       As Long
   Private Declare Function OleCreatePictureIndirect _
     Lib "olepro32.dll" _
     (PicDesc As PicBmp, _
      RefIID As GUID, _
      ByVal fPictureOwnsHandle As Long, _
      IPic As IPicture) _
       As Long
'[Constants]
 Private Const RASTERCAPS As Long = 38
 Private Const RC_PALETTE As Long = &H100
 Private Const SIZEPALETTE As Long = 104
'[Code]
'*************************************************************
' CreateBitmapPicture
' Inputs      : ByVal hBmp& = Handle to a bitmap
'             : ByVal hPal& = Handle to a Palette -
'             :            null if no palette
' Returns      : Picture = containing the bitmap
' Description  : Creates a bitmap type Picture object from
'              : a bitmap and palette
'*************************************************************
Public Function _
 CreateBitmapPicture( _
  ByVal hBmp As Long, _
  ByVal hPal As Long) _
   As Picture
 On Error GoTo proc_err
' Variables
 Dim r&, Pic As PicBmp
 Dim IPic As IPicture, IID_IDispatch As GUID
' Fill in with IDispatch Interface ID.
 With IID_IDispatch
  .Data1 = &H20400
  .Data4(0) = &HC0
  .Data4(7) = &H46
 End With
' Fill Pic with necessary parts.
 With Pic
  .Size = Len(Pic)        ' Length of structure.
  .Type = vbPicTypeBitmap ' Type of Picture (bitmap).
  .hBmp = hBmp            ' Handle to bitmap.
  .hPal = hPal            ' Handle to palette (may be null).
 End With
' Create Picture object.
 r = OleCreatePictureIndirect(Pic, IID_IDispatch, 1, IPic)
' Return the new Picture object.
 Set CreateBitmapPicture = IPic
proc_exit:
  Exit Function
 proc_err:
  MsgBox Err.Number & " - " & Err.Description, _
      vbExclamation, _
      "Error: CaptureBitmapPicture()"
  Resume proc_exit
  Resume
 End Function
'*************************************************************
' CaptureWindow
' Written By   : Shawn K. Hall [Reliable Answers.com]
' Inputs       : ByVal hWndSrc&   = Handle to the window
'              :                  to be captured
'              : ByVal Client:B   = Capture the client
'              :                  area of the window
'              : ByVal LeftSrc&   = Area of window to
'              :                  capture, in pixels
'              : ByVal TopSrc&    = ...
'              : ByVal WidthSrc&  = ...
'              : ByVal HeightSrc& = ...
' Returns      : Picture = bitmap of the specified portion
'              :          of the window that was captured
' Description  : Captures any portion of a window
'*************************************************************
Public Function _
 CaptureWindow( _
  ByVal hWndSrc As Long, _
  ByVal Client As Boolean, _
  ByVal LeftSrc As Long, _
  ByVal TopSrc As Long, _
  ByVal WidthSrc As Long, _
  ByVal HeightSrc As Long) _
    As Picture
 On Error GoTo proc_err
' Variables
 Dim hDCMemory&, hBmp, hBmpPrev&, r&, hDCSrc&
 Dim hPal&, hPalPrev&, RasterCapsScrn&, HasPaletteScrn&
 Dim PaletteSizeScrn&, LogPal As LOGPALETTE
' Depending on the value of Client get the proper device context.
 If Client Then
 ' Get device context for client area.
  hDCSrc = GetDC(hWndSrc)
 Else
 ' Get device context for entire window.
  hDCSrc = GetWindowDC(hWndSrc)
 End If
' Create a memory device context for the copy process.
 hDCMemory = CreateCompatibleDC(hDCSrc)
' Create a bitmap and place it in the memory DC.
 hBmp = CreateCompatibleBitmap(hDCSrc, WidthSrc, HeightSrc)
 hBmpPrev = SelectObject(hDCMemory, hBmp)
' Get screen properties.
 RasterCapsScrn = GetDeviceCaps(hDCSrc, RASTERCAPS)    ' Raster capabilities.
 HasPaletteScrn = RasterCapsScrn And RC_PALETTE        ' Palette support.
 PaletteSizeScrn = GetDeviceCaps(hDCSrc, SIZEPALETTE)  ' Size of palette.
 ' If the screen has a palette make a copy and realize it.
  If HasPaletteScrn And (PaletteSizeScrn = 256) Then
   ' Create a copy of the system palette.
   LogPal.palVersion = &H300
   LogPal.palNumEntries = 256
   r = GetSystemPaletteEntries(hDCSrc, 0, 256, LogPal.palPalEntry(0))
   hPal = CreatePalette(LogPal)
```

```
      ' Select the new palette into the memory DC and realize it.
      hPalPrev = SelectPalette(hDCMemory, hPal, 0)
      r = RealizePalette(hDCMemory)
    End If
   ' Copy the on-screen image into the memory DC.
    r  =  BitBlt(hDCMemory,  0,  0,  WidthSrc,  HeightSrc,  hDCSrc,  LeftSrc,  TopSrc,
  vbSrcCopy)
  ' Remove the new copy of the  on-screen image.
   hBmp = SelectObject(hDCMemory, hBmpPrev)
  ' If the screen has a palette get back the palette that was
  ' selected in previously.
   If HasPaletteScrn And (PaletteSizeScrn = 256) Then
    hPal = SelectPalette(hDCMemory, hPalPrev, 0)
   End If
  ' Release the device context resources back to the system.
   r = DeleteDC(hDCMemory)
   r = ReleaseDC(hWndSrc, hDCSrc)
  ' Call CreateBitmapPicture to create a picture object from the
  ' bitmap and palette handles. Then return the resulting Picture
  ' object.
  Set CaptureWindow = CreateBitmapPicture(hBmp, hPal)
proc_exit:
  Exit Function
proc_err:
  MsgBox Err.Number & " - " & Err.Description, _
      vbExclamation, _
      "Error: CaptureWindow()"
 Resume proc_exit
 Resume
End Function
```

```
'**********************************************************
' CaptureScreen
' Written By   : Shawn K. Hall [Reliable Answers.com]
' Inputs       : N/A
' Returns      : Picture = bitmap of the screen
' Description  : Captures the entire screen
'**********************************************************
Public Function _
 CaptureScreen() _
    As Picture
 On Error GoTo proc_err
' Variables
 Dim hWndScreen&
' Get a handle to the desktop window.
 hWndScreen = GetDesktopWindow()
' Call CaptureWindow to capture the entire desktop give the Handle
' and return the resulting Picture object.
 Set CaptureScreen = _
     CaptureWindow( _
      hWndScreen, False, 0, 0, _
      Screen.Width \ Screen.TwipsPerPixelX, _
      Screen.Height \ Screen.TwipsPerPixelY)
proc_exit:
 Exit Function
proc_err:
 MsgBox Err.Number & " - " & Err.Description, _
     vbExclamation, _
     "Error: CaptureScreen()"
 Resume proc_exit
 Resume
End Function
```

```
'*******************************************************
' CaptureForm
' Written By   : Shawn K. Hall [Reliable Answers.com]
' Inputs       : frmSrc : Form = object to capture
' Returns      : Picture = bitmap of the entire form
' Description  : Captures an entire form including title
'              : bar and border
'*******************************************************
Public Function _
 CaptureForm( _
  frmSrc As Form) _
   As Picture
 On Error GoTo proc_err
' Call CaptureWindow to capture the entire form given its window
' handle and then return the resulting Picture object.
 Set CaptureForm = _
    CaptureWindow( _
     frmSrc.hwnd, False, 0, 0, _
     frmSrc.ScaleX( _
      frmSrc.Width, _
      vbTwips, _
      vbPixels), _
     frmSrc.ScaleY( _
      frmSrc.Height, _
      vbTwips, _
      vbPixels) _
     )
proc_exit:
 Exit Function
proc_err:
 MsgBox Err.Number & " - " & Err.Description, _
```

```
        vbExclamation, _
        "Error: CaptureForm()"
    Resume proc_exit
    Resume
  End Function
'*******************************************************
' CaptureClient
' Written By   : Shawn K. Hall [Reliable Answers.com]
' Inputs       : frmSrc : Form = object to capture
' Returns      : Picture = bitmap of frmSrc's client area
' Description  : Captures the client area of a form
'*******************************************************
Public Function _
  CaptureClient( _
    frmSrc As Form) _
      As Picture
  On Error GoTo proc_err
' Call CaptureWindow to capture the client area of the form given
' its window handle and return the resulting Picture object.
  Set CaptureClient = _
        CaptureWindow( _
          frmSrc.hwnd, True, 0, 0, _
          frmSrc.ScaleX( _
            frmSrc.ScaleWidth, _
            frmSrc.ScaleMode, _
            vbPixels), _
          frmSrc.ScaleY( _
            frmSrc.ScaleHeight, _
            frmSrc.ScaleMode, _
            vbPixels) _
        )
```

```
proc_exit:
 Exit Function
proc_err:
 MsgBox Err.Number & " - " & Err.Description, _
     vbExclamation, _
     "Error: CaptureClient()"
 Resume proc_exit
 Resume
End Function
'*******************************************************
' CaptureActiveWindow
' Written By   : Shawn K. Hall [Reliable Answers.com]
' Returns      : Picture = bitmap of the active window
' Description  : Captures the currently active window
'*******************************************************
Public Function _
 CaptureActiveWindow() _
    As Picture
 On Error GoTo proc_err
' Variables
 Dim hWndActive&, r&, RectActive As RECT
' Get a handle to the active/foreground window.
 hWndActive = GetForegroundWindow()
' Get the dimensions of the window.
 r = GetWindowRect(hWndActive, RectActive)
' Call CaptureWindow to capture the active window given its
' handle and return the Resulting Picture object.
 Set CaptureActiveWindow = _
     CaptureWindow( _
      hWndActive, False, 0, 0, _
      RectActive.Right - RectActive.Left, _
```

```
        RectActive.Bottom - RectActive.Top)
proc_exit:
  Exit Function
proc_err:
  MsgBox Err.Number & " - " & Err.Description, _
        vbExclamation, _
        "Error: CaptureActiveWindow()"
  Resume proc_exit
  Resume
End Function
'*********************************************************
' PrintPictureToFitPage
' Written By   : Shawn K. Hall [Reliable Answers.com]
' Inputs       : Prn : Printer = Destination Printer object
'              : Pic : Picture = Source Picture object
' Returns      : N/A
' Description  : Prints a Picture object as large as
'              : possible
'*********************************************************
Public Sub _
  PrintPictureToFitPage( _
    Prn As Printer, _
    Pic As Picture)
  On Error GoTo proc_err
' Variables
  Dim PicRatio#, PrnWidth#, PrnHeight#
  Dim PrnRatio#, PrnPicWidth#, PrnPicHeight#
' Determine if picture should be printed in landscape or
' portrait and set the orientation.
  If Pic.Height >= Pic.Width Then
    Prn.Orientation = vbPRORPortrait  ' Taller than wide.
  Else
   Prn.Orientation = vbPRORLandscape ' Wider than tall.
  End If
 ' Calculate device independent Width-to-Height ratio for picture.
  PicRatio = Pic.Width / Pic.Height
 ' Calculate the dimentions of the printable area in HiMetric.
  PrnWidth = Prn.ScaleX(Prn.ScaleWidth, Prn.ScaleMode, vbHimetric)
  PrnHeight = Prn.ScaleY(Prn.ScaleHeight, Prn.ScaleMode, vbHimetric)
 ' Calculate device independent Width to Height ratio for printer.
  PrnRatio = PrnWidth / PrnHeight
 ' Scale the output to the printable area.
  If PicRatio >= PrnRatio Then
  ' Scale picture to fit full width of printable area.
   PrnPicWidth = Prn.ScaleX(PrnWidth, vbHimetric, Prn.ScaleMode)
   PrnPicHeight = Prn.ScaleY(PrnWidth / PicRatio, vbHimetric, Prn.ScaleMode)
  Else
  ' Scale picture to fit full height of printable area.
   PrnPicHeight = Prn.ScaleY(PrnHeight, vbHimetric, Prn.ScaleMode)
   PrnPicWidth = Prn.ScaleX(PrnHeight * PicRatio, vbHimetric, Prn.ScaleMode)
  End If
 ' Print the picture using the PaintPicture method.
  Prn.PaintPicture Pic, 0, 0, PrnPicWidth, PrnPicHeight
proc_exit:
  Exit Sub
proc_err:
  MsgBox Err.Number & " - " & Err.Description, _
      vbExclamation, _
      "Error: PrintPictureToFitPage()"
  Resume proc_exit
  Resume
End Sub
```

mDIFORM1:

```
Dim Conn As ADODB.Connection
Dim Re As ADODB.Recordset
Private Sub Command10_Click()
End
End Sub
Private Sub Command9_Click()
Dim a
Set Re = New ADODB.Recordset
Re.Open "select username,password from users where username='" & Text1.Text & "'",
Conn, adOpenStatic, adLockReadOnly, adCmdUnknown
If Re.RecordCount = 0 Then
  a = MsgBox("User Name Not Found! Try again?", vbExclamation + vbYesNo)
  If a = vbYes Then
    Text1.Text = Empty
    Text2.Text = Empty
    Text1.SetFocus
    Exit Sub
  Else
    End
  End If
Else
  If UCase(Re!Password) <> UCase(Text2.Text) Then
    a = MsgBox("Wrong password! Try again?", vbExclamation + vbYesNo)
    If a = vbYes Then
      Text2.Text = Empty
      Text2.SetFocus
      Exit Sub
    Else
      End
```

```
        End If
      Else
        'MDIForm1.Enabled = True
        'Unload Me
        'Load frmMain
        'frmMain.Show
        Picture2.Picture = LoadPicture(App.Path & "\Skin\back2.jpg")
       Picture2. Visible = False
      Picture1.Visible = True
      Text1.Text = Empty
        Text2.Text = Empty
        End If
    End If
    End Sub
    Private Sub Image1_Click(Index As Integer)
    Unload frmUsers
    Unload frmPri
    Unload frmAddNew
    Unload frmReports
    Unload frmFind
    Load frmMain
    frmMain.Show
    End Sub
    Private Sub Image1_MouseMove(Index As Integer, Button As Integer, Shift As Integer, x
    As Single, y As Single)
    Image1(0).Visible = False
    Image1(1).Visible = True
    If Image2(0).Visible = False Then
      Image2(0).Visible = True
      Image2(1).Visible = False
    End If
```

```
If Image3(0).Visible = False Then
    Image3(0).Visible = True
    Image3(1).Visible = False
End If
If Image4(0).Visible = False Then
    Image4(0).Visible = True
    Image4(1).Visible = False
End If
If Image5(0).Visible = False Then
    Image5(0).Visible = True
    Image5(1).Visible = False
End If
If Image6(0).Visible = False Then
    Image6(0).Visible = True
    Image6(1).Visible = False
End If
If Image7(0).Visible = False Then
    Image7(0).Visible = True
    Image7(1).Visible = False
End If
If Image8(0).Visible = False Then
    Image8(0).Visible = True
    Image8(1).Visible = False
End If
End Sub
Private Sub Image2_Click(Index As Integer)
Unload frmMain
Unload frmPri
Unload frmReports
Unload frmUsers
Unload frmFind
```

```
EditMode = False
Load frmAddNew
frmAddNew.Show , Me
End Sub
Private Sub Image2_MouseMove(Index As Integer, Button As Integer, Shift As Integer, x
As Single, y As Single)
Image2(0).Visible = False
Image2(1).Visible = True
If Image1(0).Visible = False Then
   Image1(0).Visible = True
   Image1(1).Visible = False
End If
If Image3(0).Visible = False Then
   Image3(0).Visible = True
   Image3(1).Visible = False
End If
If Image4(0).Visible = False Then
   Image4(0).Visible = True
   Image4(1).Visible = False
End If
If Image5(0).Visible = False Then
   Image5(0).Visible = True
   Image5(1).Visible = False
End If
If Image6(0).Visible = False Then
   Image6(0).Visible = True
   Image6(1).Visible = False
End If
If Image7(0).Visible = False Then
   Image7(0).Visible = True
   Image7(1).Visible = False
```

```
End If
If Image8(0).Visible = False Then
   Image8(0).Visible = True
   Image8(1).Visible = False
End If
End Sub
Private Sub Image3_Click(Index As Integer)
Unload frmUsers
Unload frmMain
Unload frmAddNew
Unload frmReports
Unload frmPri
Load frmFind
frmFind.Show vbModal, MDIForm1
End Sub
Private Sub Image3_MouseMove(Index As Integer, Button As Integer, Shift As Integer, x
As Single, y As Single)
Image3(0).Visible = False
Image3(1).Visible = True
If Image1(0).Visible = False Then
   Image1(0).Visible = True
   Image1(1).Visible = False
End If
If Image2(0).Visible = False Then
   Image2(0).Visible = True
   Image2(1).Visible = False
End If
If Image4(0).Visible = False Then
   Image4(0).Visible = True
   Image4(1).Visible = False
End If
```

```
  If Image5(0).Visible = False Then
     Image5(0).Visible = True
     Image5(1).Visible = False
  End If
  If Image6(0).Visible = False Then
     Image6(0).Visible = True
     Image6(1).Visible = False
  End If
  If Image7(0).Visible = False Then
     Image7(0).Visible = True
     Image7(1).Visible = False
  End If
  If Image8(0).Visible = False Then
     Image8(0).Visible = True
     Image8(1).Visible = False
  End If
End Sub
Private Sub Image4_Click(Index As Integer)
Unload frmUsers
Unload frmMain
Unload frmAddNew
Unload frmReports
Unload frmFind
Load frmPri
frmPri.Show vbModal, MDIForm1
End Sub
Private Sub Image4_MouseMove(Index As Integer, Button As Integer, Shift As Integer, x
As Single, y As Single)
Image4(0).Visible = False
Image4(1).Visible = True
If Image1(0).Visible = False Then
```

```
        Image1(0).Visible = True
        Image1(1).Visible = False
    End If
    If Image2(0).Visible = False Then
        Image2(0).Visible = True
        Image2(1).Visible = False
    End If
    If Image3(0).Visible = False Then
        Image3(0).Visible = True
        Image3(1).Visible = False
    End If
    If Image5(0).Visible = False Then
        Image5(0).Visible = True
        Image5(1).Visible = False
    End If
    If Image6(0).Visible = False Then
        Image6(0).Visible = True
        Image6(1).Visible = False
    End If
    If Image7(0).Visible = False Then
        Image7(0).Visible = True
        Image7(1).Visible = False
    End If
    If Image8(0).Visible = False Then
        Image8(0).Visible = True
        Image8(1).Visible = False
    End If
End Sub
Private Sub Image5_Click(Index As Integer)
Unload frmUsers
Unload frmMain
Unload frmAddNew
Unload frmPri
Unload frmFind
Load frmReports
frmReports.Show
End Sub
Private Sub Image5_MouseMove(Index As Integer, Button As Integer, Shift As Integer, x
As Single, y As Single)
Image5(0).Visible = False
Image5(1).Visible = True
If Image1(0).Visible = False Then
   Image1(0).Visible = True
   Image1(1).Visible = False
End If
If Image2(0).Visible = False Then
   Image2(0).Visible = True
   Image2(1).Visible = False
End If
If Image3(0).Visible = False Then
   Image3(0).Visible = True
   Image3(1).Visible = False
End If
If Image4(0).Visible = False Then
   Image4(0).Visible = True
   Image4(1).Visible = False
End If
If Image6(0).Visible = False Then
   Image6(0).Visible = True
   Image6(1).Visible = False
End If
If Image7(0).Visible = False Then
```

```
    Image7(0).Visible = True
    Image7(1).Visible = False
 End If
 If Image8(0).Visible = False Then
    Image8(0).Visible = True
    Image8(1).Visible = False
 End If
 End Sub
 Private Sub Image6_Click(Index As Integer)
 Unload frmReports
 Unload frmMain
 Unload frmAddNew
 Unload frmPri
 Unload frmFind
 Load frmUsers
 frmUsers.Show
 End Sub
 Private Sub Image6_MouseMove(Index As Integer, Button As Integer, Shift As Integer, x
 As Single, y As Single)
 Image6(0).Visible = False
 Image6(1).Visible = True
 If Image1(0).Visible = False Then
    Image1(0).Visible = True
    Image1(1).Visible = False
 End If
 If Image2(0).Visible = False Then
    Image2(0).Visible = True
    Image2(1).Visible = False
 End If
 If Image3(0).Visible = False Then
    Image3(0).Visible = True
    Image3(1).Visible = False
 End If
 If Image4(0).Visible = False Then
    Image4(0).Visible = True
    Image4(1).Visible = False
 End If
 If Image5(0).Visible = False Then
    Image5(0).Visible = True
    Image5(1).Visible = False
 End If
 If Image7(0).Visible = False Then
    Image7(0).Visible = True
    Image7(1).Visible = False
 End If
 If Image8(0).Visible = False Then
    Image8(0).Visible = True
    Image8(1).Visible = False
 End If
End Sub
Private Sub Image7_Click(Index As Integer)
Unload frmUsers
Unload frmMain
Unload frmAddNew
Picture2.Picture = LoadPicture(App.Path & "\Skin\back1.jpg")
Picture1.Visible = False
Picture2.Visible = True
'Load frmLogin
'frmLogin.Show vbModal, MDIForm1
End Sub
Private Sub Image7_MouseMove(Index As Integer, Button As Integer, Shift As Integer, x
As Single, y As Single)
```

```
Image7(0).Visible = False
Image7(1).Visible = True
If Image1(0).Visible = False Then
   Image1(0).Visible = True
   Image1(1).Visible = False
End If
If Image2(0).Visible = False Then
   Image2(0).Visible = True
   Image2(1).Visible = False
End If
If Image3(0).Visible = False Then
   Image3(0).Visible = True
   Image3(1).Visible = False
End If
If Image4(0).Visible = False Then
   Image4(0).Visible = True
   Image4(1).Visible = False
End If
If Image5(0).Visible = False Then
   Image5(0).Visible = True
   Image5(1).Visible = False
End If
If Image6(0).Visible = False Then
   Image6(0).Visible = True
   Image6(1).Visible = False
End If
If Image8(0).Visible = False Then
   Image8(0).Visible = True
   Image8(1).Visible = False
End If
End Sub
```

```
Private Sub Image8_Click(Index As Integer)
End
End Sub
Private Sub Image8_MouseMove(Index As Integer, Button As Integer, Shift As Integer, x
As Single, y As Single)
Image8(0).Visible = False
Image8(1).Visible = True
If Image1(0).Visible = False Then
   Image1(0).Visible = True
   Image1(1).Visible = False
End If
If Image2(0).Visible = False Then
   Image2(0).Visible = True
   Image2(1).Visible = False
End If
If Image3(0).Visible = False Then
   Image3(0).Visible = True
   Image3(1).Visible = False
End If
If Image4(0).Visible = False Then
   Image4(0).Visible = True
   Image4(1).Visible = False
End If
If Image5(0).Visible = False Then
   Image5(0).Visible = True
   Image5(1).Visible = False
End If
If Image6(0).Visible = False Then
   Image6(0).Visible = True
   Image6(1).Visible = False
End If
```

```
If Image7(0).Visible = False Then
    Image7(0).Visible = True
    Image7(1).Visible = False
End If
End Sub
Private Sub MDIForm_Load()
Set Conn = New ADODB.Connection
Conn.Open    "Provider=Microsoft.Jet.OLEDB.4.0;Data    Source="    &    App.Path    &
"\Data.mdb;Persist Security Info=False"
'Load frmLogin
'frmLogin.Show , MDIForm1
End Sub
Private Sub Picture1_MouseMove(Button As Integer, Shift As Integer, x As Single, y As
Single)
If Image1(0).Visible = False Then
    Image1(0).Visible = True
    Image1(1).Visible = False
End If
If Image2(0).Visible = False Then
    Image2(0).Visible = True
    Image2(1).Visible = False
End If
If Image3(0).Visible = False Then
    Image3(0).Visible = True
    Image3(1).Visible = False
End If
If Image4(0).Visible = False Then
    Image4(0).Visible = True
    Image4(1).Visible = False
End If
If Image5(0).Visible = False Then
```

```
    Image5(0).Visible = True
    Image5(1).Visible = False
End If
If Image6(0).Visible = False Then
    Image6(0).Visible = True
    Image6(1).Visible = False
End If
If Image7(0).Visible = False Then
    Image7(0).Visible = True
    Image7(1).Visible = False
End If
If Image8(0).Visible = False Then
    Image8(0).Visible = True
    Image8(1).Visible = False
End If
End Sub
```

mINTELJPEGLIBRARY:

```
Option Explicit
'
=====================================================================
=================
' Filename:    mIntelJPEGLibrary.bas
' Author:      Steve McMahon
' Date:        15 March 1999
' Requires:    cDIBSection.cls (vbAccelerator)
'              IJL11.DLL (Intel)
'
' An interface to Intel's IJL (Intel JPG Library) for use in VB.
'
' Copyright © 1999 Steve McMahon for vbAccelerator
' http://vbaccelerator.com/
'
' Modifications
' 16 Jan 2000  SPM
' * Modified to declares to access v1.1. of Intel's IJL DLL
' * SaveJPG - if you were overwriting an existing JPG, the file could never be
'   reduced in size, only increased.  Old bytes were simply left trailing as
'   an unnecessary payload at the JPG.  The JPG could be loaded, but this was
'   not ideal.  The new version ensures the file size is always set.
' * LoadJPGFromPtr - new function, allows you to read a JPG from a memory
'   address (e.g. resource etc)
'
'
' Copyright.
' IJL.DLL is a copyright © Intel, which is a registered trade mark of the Intel
' Corporation.
'
'
' Note.
' Intel are not responsible for any errors in this code and should not be
' mentioned in any Help, About or support in any product using the Intel library.
'
'
'
'
=================================================================
=================
' IJL Declares:
Private Enum IJLERR
  '// The following "error" values indicate an "OK" condition.
  IJL_OK = 0
```

```
IJL_INTERRUPT_OK = 1
IJL_ROI_OK = 2
'// The following "error" values indicate an error has occurred.
IJL_EXCEPTION_DETECTED = -1
IJL_INVALID_ENCODER = -2
IJL_UNSUPPORTED_SUBSAMPLING = -3
IJL_UNSUPPORTED_BYTES_PER_PIXEL = -4
IJL_MEMORY_ERROR = -5
IJL_BAD_HUFFMAN_TABLE = -6
IJL_BAD_QUANT_TABLE = -7
IJL_INVALID_JPEG_PROPERTIES = -8
IJL_ERR_FILECLOSE = -9
IJL_INVALID_FILENAME = -10
IJL_ERROR_EOF = -11
IJL_PROG_NOT_SUPPORTED = -12
IJL_ERR_NOT_JPEG = -13
IJL_ERR_COMP = -14
IJL_ERR_SOF = -15
IJL_ERR_DNL = -16
IJL_ERR_NO_HUF = -17
IJL_ERR_NO_QUAN = -18
IJL_ERR_NO_FRAME = -19
IJL_ERR_MULT_FRAME = -20
IJL_ERR_DATA = -21
IJL_ERR_NO_IMAGE = -22
IJL_FILE_ERROR = -23
IJL_INTERNAL_ERROR = -24
IJL_BAD_RST_MARKER = -25
IJL_THUMBNAIL_DIB_TOO_SMALL = -26
IJL_THUMBNAIL_DIB_WRONG_COLOR = -27
IJL_RESERVED = -99
```

```
End Enum
Private Enum IJLIOTYPE
 IJL_SETUP = -1&
 "// Read JPEG parameters (i.e., height, width, channels,
 "// sampling, etc.) from a JPEG bit stream.
 IJL_JFILE_READPARAMS = 0&
 IJL_JBUFF_READPARAMS = 1&
 "// Read a JPEG Interchange Format image.
 IJL_JFILE_READWHOLEIMAGE = 2&
 IJL_JBUFF_READWHOLEIMAGE = 3&
 "// Read JPEG tables from a JPEG Abbreviated Format bit stream.
 IJL_JFILE_READHEADER = 4&
 IJL_JBUFF_READHEADER = 5&
 "// Read image info from a JPEG Abbreviated Format bit stream.
 IJL_JFILE_READENTROPY = 6&
 IJL_JBUFF_READENTROPY = 7&
 "// Write an entire JFIF bit stream.
 IJL_JFILE_WRITEWHOLEIMAGE = 8&
 IJL_JBUFF_WRITEWHOLEIMAGE = 9&
 "// Write a JPEG Abbreviated Format bit stream.
 IJL_JFILE_WRITEHEADER = 10&
 IJL_JBUFF_WRITEHEADER = 11&
 "// Write image info to a JPEG Abbreviated Format bit stream.
 IJL_JFILE_WRITEENTROPY = 12&
 IJL_JBUFF_WRITEENTROPY = 13&
 "// Scaled Decoding Options:
 "// Reads a JPEG image scaled to 1/2 size.
 IJL_JFILE_READONEHALF = 14&
 IJL_JBUFF_READONEHALF = 15&
 "// Reads a JPEG image scaled to 1/4 size.
 IJL_JFILE_READONEQUARTER = 16&
 IJL_JBUFF_READONEQUARTER = 17&
 ''// Reads a JPEG image scaled to 1/8 size.
 IJL_JFILE_READONEEIGHTH = 18&
 IJL_JBUFF_READONEEIGHTH = 19&
 ''// Reads an embedded thumbnail from a JFIF bit stream.
 IJL_JFILE_READTHUMBNAIL = 20&
 IJL_JBUFF_READTHUMBNAIL = 21&
End Enum
Private Type JPEG_CORE_PROPERTIES_VB ' Sadly, due to a limitation in VB (UDT
variable count)
          ' we can't encode the full JPEG_CORE_PROPERTIES structure
 UseJPEGPROPERTIES As Long                    '// default = 0
'// DIB specific I/O data specifiers.
 DIBBytes As Long ';              '// default = NULL 4
 DIBWidth As Long ';              '// default = 0 8
 DIBHeight As Long ';             '// default = 0 12
 DIBPadBytes As Long ';             '// default = 0 16
 DIBChannels As Long ';             '// default = 3 20
 DIBColor As Long ';              '// default = IJL_BGR 24
 DIBSubsampling As Long  ';           '// default = IJL_NONE 28
'// JPEG specific I/O data specifiers.
 JPGFile As Long 'LPTSTR          JPGFile;          32  '// default = NULL
 JPGBytes As Long ';              '// default = NULL 36
 JPGSizeBytes As Long ';            '// default = 0 40
 JPGWidth As Long ';              '// default = 0 44
 JPGHeight As Long ';             '// default = 0 48
 JPGChannels As Long ';             '// default = 3
 JPGColor As Long          ';            '// default = IJL_YCBCR
 JPGSubsampling As Long  ';           '// default = IJL_411
 JPGThumbWidth As Long ' ;           '// default = 0
 JPGThumbHeight As Long ';          '// default = 0
 '// JPEG conversion properties.
 cconversion_reqd As Long ';        '// default = TRUE
 upsampling_reqd As Long ';         '// default = TRUE
 jquality As Long ';                '// default = 75.  100 is my preferred quality setting.
'// Low-level properties - 20,000 bytes.  If the whole structure
 ' is written out then VB fails with an obscure error message
 ' "Too Many Local Variables" !
 '
 ' These all default if they are not otherwise specified so there
 ' is no trouble to just assign a sufficient buffer in memory:
 jprops(0 To 19999) As Byte
End Type
Private Declare Function ijlInit Lib "ijl11.dll" (jcprops As Any) As Long
Private Declare Function ijlFree Lib "ijl11.dll" (jcprops As Any) As Long
Private Declare Function ijlRead Lib "ijl11.dll" (jcprops As Any, ByVal ioType As Long)
As Long
Private Declare Function ijlWrite Lib "ijl11.dll" (jcprops As Any, ByVal ioType As Long)
As Long
Private Declare Function ijlGetLibVersion Lib "ijl11.dll" () As Long
Private Declare Function ijlGetErrorString Lib "ijl11.dll" (ByVal code As Long) As Long
' Win32 Declares
Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" ( _
   lpvDest As Any, lpvSource As Any, ByVal cbCopy As Long)
Private Declare Function GlobalAlloc Lib "kernel32" (ByVal wFlags As Long, ByVal
dwBytes As Long) As Long
Private Declare Function GlobalFree Lib "kernel32" (ByVal hMem As Long) As Long
Private Declare Function GlobalLock Lib "kernel32" (ByVal hMem As Long) As Long
Private Declare Function GlobalUnlock Lib "kernel32" (ByVal hMem As Long) As Long
Private Const GMEM_DDESHARE = &H2000
Private Const GMEM_DISCARDABLE = &H100
Private Const GMEM_DISCARDED = &H4000
Private Const GMEM_FIXED = &H0
Private Const GMEM_INVALID_HANDLE = &H8000
Private Const GMEM_LOCKCOUNT = &HFF
Private Const GMEM_MODIFY = &H80
Private Const GMEM_MOVEABLE = &H2
Private Const GMEM_NOCOMPACT = &H10
Private Const GMEM_NODISCARD = &H20
Private Const GMEM_NOT_BANKED = &H1000
Private Const GMEM_NOTIFY = &H4000
Private Const GMEM_SHARE = &H2000
Private Const GMEM_VALID_FLAGS = &H7F72
Private Const GMEM_ZEROINIT = &H40
Private Const GPTR = (GMEM_FIXED Or GMEM_ZEROINIT)
' Stuff for replacing a file when you have to Kill the original:
Private Const MAX_PATH = 260
Private Type FILETIME
  dwLowDateTime As Long
  dwHighDateTime As Long
End Type
Private Type WIN32_FIND_DATA
  dwFileAttributes As Long
  ftCreationTime As FILETIME
  ftLastAccessTime As FILETIME
  ftLastWriteTime As FILETIME
  nFileSizeHigh As Long
  nFileSizeLow As Long
  dwReserved0 As Long
  dwReserved1 As Long
  cFileName As String * MAX_PATH
  cAlternate As String * 14
```

```
End Type
Private Declare Function FindFirstFile Lib "kernel32" Alias "FindFirstFileA" (ByVal
lpFileName As String, lpFindFileData As WIN32_FIND_DATA) As Long
Private Declare Function lopen Lib "kernel32" Alias "_lopen" (ByVal lpPathName As
String, ByVal iReadWrite As Long) As Long
Private Declare Function lclose Lib "kernel32" Alias "_lclose" (ByVal hFile As Long) As
Long
Private Declare Function SetFileTime Lib "kernel32" (ByVal hFile As Long,
lpCreationTime As FILETIME, lpLastAccessTime As FILETIME, lpLastWriteTime As
FILETIME) As Long
Private Declare Function SetFileAttributes Lib "kernel32" Alias "SetFileAttributesA"
(ByVal lpFileName As String, ByVal dwFileAttributes As Long) As Long
Private Const OF_WRITE = &H1
Private Const OF_SHARE_DENY_WRITE = &H20
Private Const GENERIC_WRITE = &H40000000
Private Const GENERIC_READ = &H80000000
Private Const FILE_SHARE_WRITE = &H2
Private Const CREATE_ALWAYS = 2
Private Const FILE_BEGIN = 0
Private Const SECTION_MAP_WRITE = &H2
Public Function LoadJPG( _
    ByRef cDib As cDIBSection, _
    ByVal sFile As String _
  ) As Boolean
Dim tJ As JPEG_CORE_PROPERTIES_VB
Dim bFile() As Byte
Dim lR As Long
Dim lPtr As Long
Dim lJPGWidth As Long, lJPGHeight As Long
lR = ijlInit(tJ)
  If lR = IJL_OK Then
```

```
' Write the filename to the jcprops.JPGFile member:
bFile = StrConv(sFile, vbFromUnicode)
ReDim Preserve bFile(0 To UBound(bFile) + 1) As Byte
bFile(UBound(bFile)) = 0
lPtr = VarPtr(bFile(0))
CopyMemory tJ.JPGFile, lPtr, 4
' Read the JPEG file parameters:
lR = ijlRead(tJ, IJL_JFILE_READPARAMS)
If lR <> IJL_OK Then
  ' Throw error
  MsgBox "Failed to read JPG", vbExclamation
Else
 ' set JPG color
  If tJ.JPGChannels = 1 Then
    tJ.JPGColor = 4& ' IJL_G
  Else
    tJ.JPGColor = 3& ' IJL_YCBCR
  End If
     ' Get the JPGWidth ...
  lJPGWidth = tJ.JPGWidth
  ' .. & JPGHeight member values:
  lJPGHeight = tJ.JPGHeight
' Create a buffer of sufficient size to hold the image:
  If cDib.Create(lJPGWidth, lJPGHeight) Then
    ' Store DIBWidth:
    tJ.DIBWidth = lJPGWidth
    ' Very important: tell IJL how many bytes extra there
    ' are on each DIB scan line to pad to 32 bit boundaries:
    tJ.DIBPadBytes = cDib.BytesPerScanLine - lJPGWidth * 3
    ' Store DIBHeight:
    tJ.DIBHeight = -lJPGHeight
```

```
          ' Store Channels:
          tJ.DIBChannels = 3&
          ' Store DIBBytes (pointer to uncompressed JPG data):
          tJ.DIBBytes = cDib.DIBSectionBitsPtr
          ' Now decompress the JPG into the DIBSection:
          lR = ijlRead(tJ, IJL_JFILE_READWHOLEIMAGE)
          If lR = IJL_OK Then
            ' That's it!  cDib now contains the uncompressed JPG.
            LoadJPG = True
          Else
            ' Throw error:
            MsgBox "Cannot read Image Data from file.", vbExclamation
          End If
        Else
          ' failed to create the DIB...
        End If
      End If
      ' Ensure we have freed memory:
      ijlFree tJ
    Else
      ' Throw error:
      MsgBox "Failed to initialise the IJL library: " & lR, vbExclamation
    End If
    End Function
  Public Function LoadJPGFromPtr( _
      ByRef cDib As cDIBSection, _
      ByVal lPtr As Long, _
      ByVal lSize As Long _
    ) As Boolean
  Dim tJ As JPEG_CORE_PROPERTIES_VB
  Dim bFile() As Byte
```

```
Dim lR As Long
Dim lJPGWidth As Long, lJPGHeight As Long
lR = ijlInit(tJ)
  If lR = IJL_OK Then
        ' set JPEG buffer
    tJ.JPGBytes = lPtr
    tJ.JPGSizeBytes = lSize
        ' Read the JPEG parameters:
    lR = ijlRead(tJ, IJL_JBUFF_READPARAMS)
    If lR <> IJL_OK Then
      ' Throw error
      MsgBox "Failed to read JPG", vbExclamation
    Else
     ' set JPG color
     If tJ.JPGChannels = 1 Then
       tJ.JPGColor = 4& ' IJL_G
     Else
       tJ.JPGColor = 3& ' IJL_YCBCR
     End If
   ' Get the JPGWidth ...
     lJPGWidth = tJ.JPGWidth
     ' .. & JPGHeight member values:
     lJPGHeight = tJ.JPGHeight
  ' Create a buffer of sufficient size to hold the image:
    If cDib.Create(lJPGWidth, lJPGHeight) Then
      ' Store DIBWidth:
      tJ.DIBWidth = lJPGWidth
      ' Very important: tell IJL how many bytes extra there
      ' are on each DIB scan line to pad to 32 bit boundaries:
      tJ.DIBPadBytes = cDib.BytesPerScanLine - lJPGWidth * 3
      ' Store DIBHeight:
        tJ.DIBHeight = -lJPGHeight
         ' Store Channels:
         tJ.DIBChannels = 3&
         ' Store DIBBytes (pointer to uncompressed JPG data):
         tJ.DIBBytes = cDib.DIBSectionBitsPtr
         ' Now decompress the JPG into the DIBSection:
         lR = ijlRead(tJ, IJL_JBUFF_READWHOLEIMAGE)
         If lR = IJL_OK Then
           ' That's it!  cDib now contains the uncompressed JPG.
           LoadJPGFromPtr = True
         Else
           ' Throw error:
           MsgBox "Cannot read Image Data from file.", vbExclamation
         End If
       Else
         ' failed to create the DIB...
       End If
     End If
    ' Ensure we have freed memory:
     ijlFree tJ
   Else
     ' Throw error:
     MsgBox "Failed to initialise the IJL library: " & lR, vbExclamation
   End If
  End Function
Public Function SaveJPG( _
     ByRef cDib As cDIBSection, _
     ByVal sFile As String, _
     Optional ByVal lQuality As Long = 90 _
   ) As Boolean
Dim tJ As JPEG_CORE_PROPERTIES_VB
Dim bFile() As Byte
Dim lPtr As Long
Dim lR As Long
Dim tFnd As WIN32_FIND_DATA
Dim hFile As Long
Dim bFileExisted As Boolean
Dim lFileSize As Long
hFile = -1
  lR = ijlInit(tJ)
  If lR = IJL_OK Then
  ' Check if we're attempting to overwrite an existing file.
    ' If so hFile <> INVALID_FILE_HANDLE:
    bFileExisted = (FindFirstFile(sFile, tFnd) <> -1)
    If bFileExisted Then
       Kill sFile
    End If
     ' Set up the DIB information:
    ' Store DIBWidth:
    tJ.DIBWidth = cDib.Width
    ' Store DIBHeight:
    tJ.DIBHeight = -cDib.Height
    ' Store DIBBytes (pointer to uncompressed JPG data):
    tJ.DIBBytes = cDib.DIBSectionBitsPtr
    ' Very important: tell IJL how many bytes extra there
    ' are on each DIB scan line to pad to 32 bit boundaries:
    tJ.DIBPadBytes = cDib.BytesPerScanLine - cDib.Width * 3
    ' Set up the JPEG information:
    ' Store JPGFile:
    bFile = StrConv(sFile, vbFromUnicode)
    ReDim Preserve bFile(0 To UBound(bFile) + 1) As Byte
    bFile(UBound(bFile)) = 0
```

```
lPtr = VarPtr(bFile(0))
CopyMemory tJ.JPGFile, lPtr, 4
' Store JPGWidth:
tJ.JPGWidth = cDib.Width
' .. & JPGHeight member values:
tJ.JPGHeight = cDib.Height
' Set the quality/compression to save:
tJ.jquality = lQuality
' Write the image:
lR = ijlWrite(tJ, IJL_JFILE_WRITEWHOLEIMAGE)
' Check for success:
If lR = IJL_OK Then
' Now if we are replacing an existing file, then we want to
  ' put the file creation and archive information back again:
  If bFileExisted Then
     hFile = lopen(sFile, OF_WRITE Or OF_SHARE_DENY_WRITE)
    If hFile = 0 Then
      ' problem
    Else
      SetFileTime        hFile,        tFnd.ftCreationTime,        tFnd.ftLastAccessTime,
tFnd.ftLastWriteTime
      lclose hFile
      SetFileAttributes sFile, tFnd.dwFileAttributes
    End If
    End If
lFileSize = tJ.JPGSizeBytes - tJ.JPGBytes
 ' Success:
SaveJPG = True
Else
' Throw error
```

```
      Err.Raise 26001, App.EXEName & ".mIntelJPEGLibrary", "Failed to save to JPG "
& lR, vbExclamation
    End If
  ' Ensure we have freed memory:
    ijlFree tJ
  Else
    ' Throw error:
    Err.Raise 26001, App.EXEName & ".mIntelJPEGLibrary", "Failed to initialise the IJL
library: " & lR
  End If
  End Function
Public Function SaveJPGToPtr( _
    ByRef cDib As cDIBSection, _
    ByVal lPtr As Long, _
    ByRef lBufSize As Long, _
    Optional ByVal lQuality As Long = 90 _
  ) As Boolean
Dim tJ As JPEG_CORE_PROPERTIES_VB
Dim bFile() As Byte
Dim lR As Long
Dim tFnd As WIN32_FIND_DATA
Dim hFile As Long
Dim bFileExisted As Boolean
Dim b As Boolean
  hFile = -1
  lR = ijlInit(tJ)
  If lR = IJL_OK Then
     ' Set up the DIB information:
    ' Store DIBWidth:
    tJ.DIBWidth = cDib.Width
    ' Store DIBHeight:
```

```
    tJ.DIBHeight = -cDib.Height
    ' Store DIBBytes (pointer to uncompressed JPG data):
    tJ.DIBBytes = cDib.DIBSectionBitsPtr
    ' Very important: tell IJL how many bytes extra there
    ' are on each DIB scan line to pad to 32 bit boundaries:
    tJ.DIBPadBytes = cDib.BytesPerScanLine - cDib.Width * 3
    ' Set up the JPEG information:
    ' Store JPGWidth:
    tJ.JPGWidth = cDib.Width
    ' .. & JPGHeight member values:
    tJ.JPGHeight = cDib.Height
    ' Set the quality/compression to save:
    tJ.jquality = lQuality
    ' set JPEG buffer
    tJ.JPGBytes = lPtr
    tJ.JPGSizeBytes = lBufSize
    ' Write the image:
    lR = ijlWrite(tJ, IJL_JBUFF_WRITEWHOLEIMAGE)
    ' Check for success:
    If lR = IJL_OK Then
    lBufSize = tJ.JPGSizeBytes
      ' Success:
      SaveJPGToPtr = True
      Else
      ' Throw error
      Err.Raise 26001, App.EXEName & ".mIntelJPEGLibrary", "Failed to save to JPG "
& lR, vbExclamation
   End If
   ' Ensure we have freed memory:
   ijlFree tJ
  Else
```

```
    ' Throw error:
    Err.Raise 26001, App.EXEName & ".mIntelJPEGLibrary", "Failed to initialise the IJL library: " & IR
  End If
  End Function
```

mODWEBCAM:

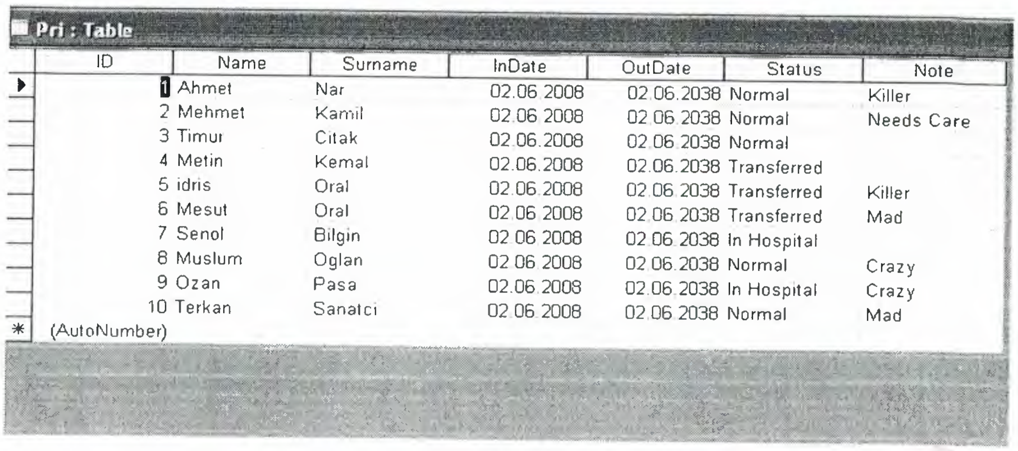
```
Public Declare Function SendMessage Lib "USER32" Alias "SendMessageA" (ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam As Long, lParam As Any) As Long
Public Declare Function capCreateCaptureWindow Lib "avicap32.dll" Alias "capCreateCaptureWindowA" (ByVal lpszWindowName As String, ByVal dwStyle As Long, ByVal x As Long, ByVal y As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal hwndParent As Long, ByVal nID As Long) As Long

Public mCapHwnd As Long
Public Const CONNECT As Long = 1034
Public Const DISCONNECT As Long = 1035
Public Const GET_FRAME As Long = 1084
Public Const COPY As Long = 1054
Global Zname, Zsurname, mName, Msurnam, Zid
Global LSid As Integer
Global EditMode As Boolean
Global EditMode2 As Boolean, Pid, Pname, Psurname
```

APPENDIX B

TABLE OF THE DATABASE

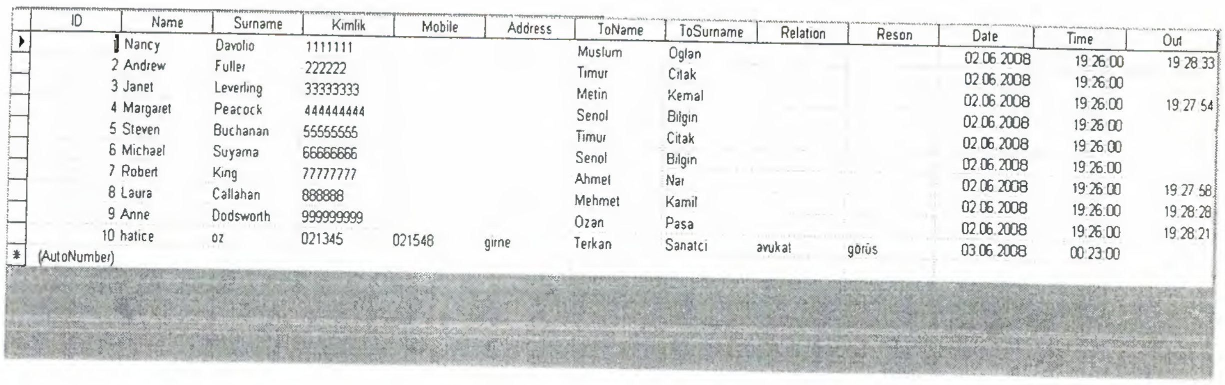
This table, Pri shows for prisoners. It has included the prisoner's name, surname, in date, out date, status and information about his/her.


Pri : Table

ID	Name	Surname	InDate	OutDate	Status	Note
1	Ahmet	Nar	02.06.2008	02.06.2038	Normal	Killer
2	Mehmet	Kamil	02.06.2008	02.06.2038	Normal	Needs Care
3	Timur	Citak	02.06.2008	02.06.2038	Normal	
4	Metin	Kemal	02.06.2008	02.06.2038	Transferred	
5	idris	Oral	02.06.2008	02.06.2038	Transferred	Killer
6	Mesut	Oral	02.06.2008	02.06.2038	Transferred	Mad
7	Senol	Bilgin	02.06.2008	02.06.2038	In Hospital	
8	Muslum	Oglan	02.06.2008	02.06.2038	Normal	Crazy
9	Ozan	Pasa	02.06.2008	02.06.2038	In Hospital	Crazy
10	Terkan	Sanatci	02.06.2008	02.06.2038	Normal	Mad
(AutoNumber)						

Figure 1. Prisoner data table

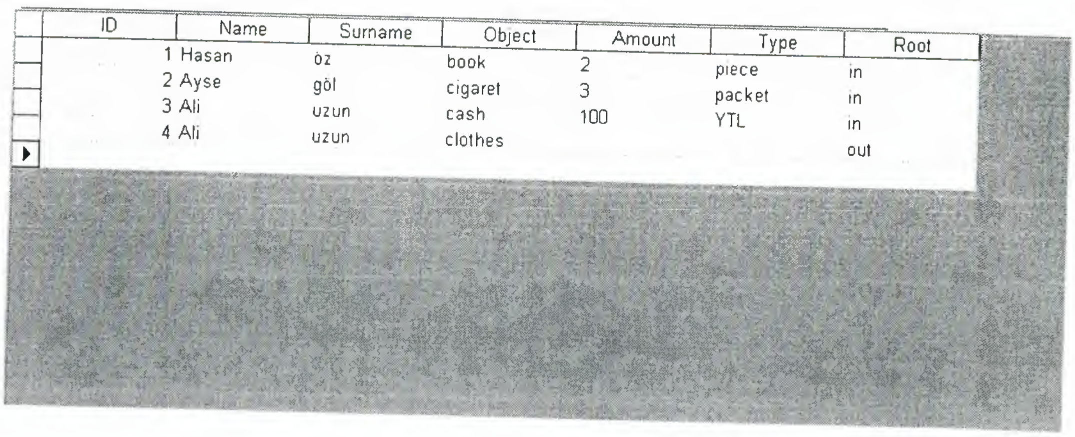
This table, table1 shows all the current records of the visitors came in and out in the current day. It shows information such as the visitor's name and surname, his/hers ID number, mobile, address, to who they came, the reason of the visit, date and finally their in and out time.



ID	Name	Surname	Kimlik	Mobile	Address	ToName	ToSurname	Relation	Reson	Date	Time	Out
1	Nancy	Davolio	1111111			Muslum	Oglan			02.06.2008	19:26:00	19:28:33
2	Andrew	Fuller	222222			Timur	Citak			02.06.2008	19:26:00	
3	Janet	Leverling	33333333			Metin	Kemal			02.06.2008	19:26:00	19:27:54
4	Margaret	Peacock	444444444			Senol	Bilgin			02.06.2008	19:26:00	
5	Steven	Buchanan	55555555			Timur	Citak			02.06.2008	19:26:00	
6	Michael	Suyama	66666666			Senol	Bilgin			02.06.2008	19:26:00	
7	Robert	King	77777777			Ahmet	Nar			02.06.2008	19:26:00	19:27:58
8	Laura	Callahan	888888			Mehmet	Kamil			02.06.2008	19:26:00	19:28:28
9	Anne	Dodsworth	999999999			Ozan	Pasa			02.06.2008	19:26:00	19:28:21
10	hatice	oz	021345	021548	girne	Terkan	Sanatci	avukat	görüs	03.06.2008	00:23:00	
(AutoNumber)												

Figure 2. Visitors and prisoners information data table

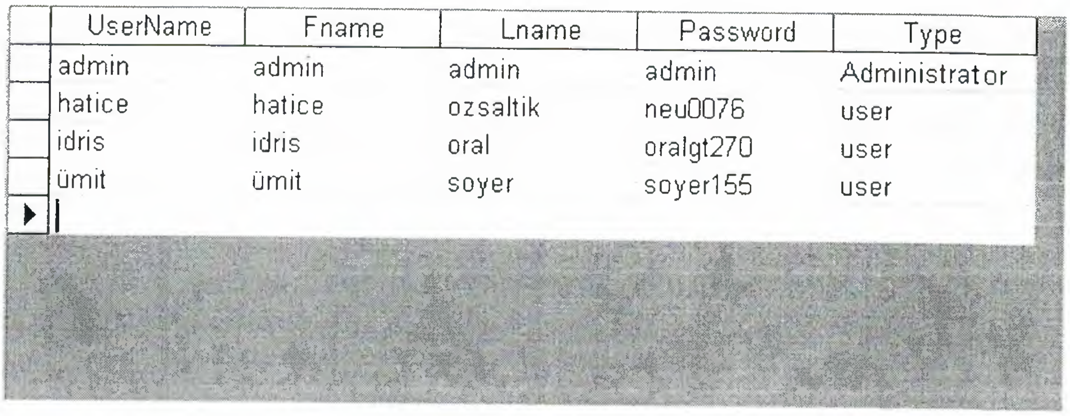
This table table2 contains the visitors objects to bring inside and take outside with visitors name, surname, object types (piece, packet, box, money) , amounts and root directions.



ID	Name	Surname	Object	Amount	Type	Root
1	Hasan	öz	book	2	piece	in
2	Ayse	göl	cigaret	3	packet	in
3	Ali	uzun	cash	100	YTL	in
4	Ali	uzun	clothes			out

Figure 3. Visitor's objects to bring inside and taking outside

The table of users including the administrators and users first name, last name, passwords and types. For entering the program, needed this database table.



UserName	Fname	Lname	Password	Type
admin	admin	admin	admin	Administrator
hatice	hatice	ozsaltik	neu0076	user
idris	idris	oral	oralgt270	user
ümit	ümit	soyer	soyer155	user

Figure 4. User name and password data table