NEAR EAST UNIVERSITY

Faculty of Engineering

Department Of Computer Engineering

STUDENT INFORMATION SYSTEM USING VISUAL BASIC

Graduation Project
COM – 400

Student : Bekir Şimşek

Number : 20001739

Supervisor: Mr. Ümit İlhan

Nicosia - 2005

ACKNOWLEDGEMENTS

"First I would like to thank my supervisor Mr Umit Ilhan for his great advice and recommendations to finish this work properly.

Although I faced many problem collections data but has guiding me the appropriate references.(Dear Abiyev,Okan Donangil) thanks a lot for your invaluable and continual support.

Second, I would like to thank my family for their constant encouragement and support during the preparation of this work specially my father (Derviş Şimşek),my mother (Emine Şimşek) my brothers(Adem ,Ömer ,Yusuf Şimsek),My sister (Havva Şimşek), .

Third, I thank all the staff of the faculty of engineering for giving me the facilities to practice and solving any problem I was facing during working in this project.

Forth I do not want to forget my best friends (MehmetKamiloğlu),(S.Hüseyin Özgün), (Özgür Berbergil),(Necati Ağbulut),(Hakan Topal),(Müslüm Şenel) and all friends for helping me to finish this work in short time by their invaluable encouragement.

Finally thanks for all of my friends for their advices and support.

ABSTRACT

Visual Basic Applications provides a complete integrated development environment (IDE) that features the same elements familiar to developers using Microsoft Visual Basic, including a Project Window, a Properties Window, and debugging tools. VBA also includes support for Microsoft Forms, for creating custom dialog boxes, and ActiveX Controls, for rapidly building user interfaces. Integrated directly into a host application, VBA offers the advantages of fast, in-process performance (up to 200 times faster than other stand-alone development tools), tight integration with the host application (code behind documents, cells, and so forth), and the ability to build solutions without the use of additional tools. To gain access to a remote ODBC data source, you usually have to provide a valid user ID and password combination. These values can be provided in the Connect property of the Data control or in the connect string that is supplied as an argument to the OpenDatabase method. If these values are not supplied, the ODBC Driver Manager exposes a dialog to collect the user name, password, and other missing information needed to establish the connection. There is no way to disable this dialog with ADO and Jet. However, by using the ODBCDirect or RDO prompt arguments, this dialog can be disabled and your code can intercept a trappable error.

TABLE 0F CONTENTS

ACKNOWLEDGEMENTS i

ABSTRACT ii

TABLE OF CONTENTS iii

INTRODUCTION 1

CHAPTER ONE: VISUAL BASIC 2

What you need to know before starting this training 2

What Is Visual Basic for Applications? 3

Visual Basic Editions 3

Competitive Advantage 5

Benefits to Developers 6

CHAPTER TWO : DATABESES *11*

Accessing SQL Views Using ADO 12

Accessing Stored Procedures Using ADO 12

Closing ADO ODBC Connections with Jet 12

Providing User ID and Password 14

Opening Connections Directly 14

Opening Connections Indirectly 14

Handling Remote ADO Messages and Errors 15

Managing ADO Data Source Name Entries 15

Setting the Default Database 16

Managing ADO Network Traffic 16

Managing ADO ODBC Users 17

Managing ADO Query Result Set Size 18

Remote Data Access Using ADO and ODBCDirect 18

Remote Data Access Using ADO and ODBCDirect 18

The T-SQL Debugger 19
Setup and Compatibility 19
Server-Side Setup 20
Unassigned Parameters Dialog box 21
Views and Options 22
Exiting from the T-SQL Debugger 22
Using ADO to Select a Remote Query Processor 23
Using ADO to Select the Jet Query Processor 24
Using ADO to Share Remote Data 24
Using SQL PassThrough Queries with ADO 25
Microsoft Acces Database 25

CHAPTER THREE : SCHOOL REGISTRATON PROGRAM 30

3.1 Pasword Dialog 30
3.2 Main Page 31
3.3 Select School Year 32
3.4 Student Record 33
3.5 Record New Sudent 34
3.6 Existing Studen Option 35
3.7 Student Advisor Selections 36
3.8 Advanced Search For Student 37
3.9 View Option 38
3.10 Select Level 39
3.11 List Of Advisor For Semester Year 40
3.12 List Of Levels 41
3.13 Add New Level 42
3.14 List Of School Year 43
3.15 Print Option 44
3.16 Regitration Slip Report 45
3.17 Print Option For Student List 46

APPENDIX 47

Program Source Codes 47

CONCLUSION 123

REFERENCES 124

INTRODUCTION

In first chapter, we will see the visual basic and aplications. Visual Basic for Applications delivers a competitive advantage for ISVs seeking to provide full customization and integration capabilities to customers. With VBA-enabled products, ISVs can build broad capabilities into their core product while providing a technology for customers to tailor the application and add features and functionality specific to their requirements. The Enterprise edition allows professionals to create robust distributed applications in a team setting. It includes all the features of the Professional edition, plus Back Office tools such as SQL Server, Microsoft Transaction Server, Internet Information Server, Visual SourceSafe, SNA Server, and more. Printed documentation provided with the Enterprise edition includes the Visual Studio Enterprise Features book plus Microsoft Developer Network CDs containing full online documentation.

The next chapter begins with the acces and sql database. There are two other advantages to using Acces as a production tool. First, it provides exactly the same options for the problems you write as it does for the problems you select from a database. Second, the process of writing or selecting problems is almost completely independent of page layout decisions. This means you can do things in almost any order: select some problems, see how they look on a certain type of document, make some changes, try a different type of document, and so on. Acces handles all the finer details of production and, as you will quickly discover, it is extremely good at what it does. SQL Server's server-side cursors support multiple operations on a single connection as implemented with ODBCDirect and RDO. However, there is no support for server-side cursors with Jet.When Jet needs to open a connection, it first checks its internal connection cache. If there is a connection in the cache that uses the same DSN and database parameters, and there are no uncompleted queries pending on the connection, it is reused. Back-end database systems that support pending results on a single connection may not need additional connections to perform simultaneous read/write operations

The last chapter introduce student registration systems program .The chapter shows all program form and information.And explain how we can use this program.

CHAPTER ONE :VISUAL BASIC

What you need to know before starting this training

If you are new to computer programming, you should consider Visual Basic as your language of choice for learning how to develop computer programs. Another popular language used in teaching programming is Pascal. But Visual Basic is generally more popular than Pascal. Microsoft claims that there are over 3 million Visual Basic programmers. Whatever the number, Visual Basic is popular because it is easy to learn, fun to use, and has evolved into a very powerful development tool. But programming in Visual Basic or any other programming language is a challenging task. It is a little more complicated than learning how to use Microsoft Word or Excel. But if you want to become a computer programmer, you will find Visual Basic to be the easiest way to teach yourself programming. You can get the training series (above) that includes the complete set of 17 Multimedia CD-ROMs or Videos.

You will hear and see your trainer on your computer screen as he teaches you beginner to advanced level Visual Basic concepts. But most of the time, you will learn Visual Basic by seeing your trainer perform real world examples with Visual Basic 6.0 and by writing the example programs or your own programs and running them with the Visual Basic 6.0 software included in this package. The best way to learn Visual Basic is by examples, and that is what this course presents a lot of. If you are new to Visual Basic, you should start with the first 8 levels (i.e. Level 1 to Level 8). Then later as you build your skills, you can purchase the rest of the training if you want to become a certified or advanced level Visual Basic Developer. If you spend about 1 to 2 hours each day training with Visual Basic, you should become proficient in about 3 months. The rule to becoming a good or competent programmer is practice, practice, practice! If you want tips on how to become more productive with Visual Basic very quickly, especially if you are new to programming, please do not hesitate to contact us toll free at 1-888-797-4040 (or 301-589-3349) and a technical support person with development experience will assist you. Remember, the best programmers are those who enjoy programming; so just relax and program for fun and treat programming as a hobby. You will be amazed at the progress you will make and success you will have as a developer.

What Is Visual Basic for Applications?

Microsoft Visual Basic for Applications (VBA) is a powerful development technology for rapidly customizing rich-client desktop packaged applications and integrating them with existing data and systems. VBA offers a sophisticated set of programming tools based on the Microsoft Visual Basic development system, the world's most popular rapid application development system, which developers can use to harness the power of packaged applications. VBA enables customers to buy off-the-shelf software and customize it to meet their specific business processes, rather than build solutions from scratch. This helps them save time and money, reduce risks, leverage their programming skills, and deliver precisely what users need.

Visual Basic for Applications provides a complete integrated development environment (IDE) that features the same elements familiar to developers using Microsoft Visual Basic, including a Project Window, a Properties Window, and debugging tools. VBA also includes support for Microsoft Forms, for creating custom dialog boxes, and ActiveX Controls, for rapidly building user interfaces. Integrated directly into a host application, VBA offers the advantages of fast, in-process performance (up to 200 times faster than other stand-alone development tools), tight integration with the host application (code behind documents, cells, and so forth), and the ability to build solutions without the use of additional tools.

Software programs that include VBA are called customizable applications—applications that can be tailored to fit specific business needs. This class of applications enables developers to quickly build solutions that require less end-user training. For MIS and business managers, customization means that solutions can be developed quickly and deployed easily, with minimal maintenance. In an industry familiar with two-year backlogs for new applications and high end-user training costs, these solutions provide a tremendous business benefit in terms of return on investment (ROI) and timeliness.

Visual Basic Editions

Visual Basic is available in three editions, each geared to meet a specific set of development requirements. The features available to you depend on which product you have purchased.

Visual Basic Enterprise Edition Features

The Enterprise edition allows professionals to create robust distributed applications in a team setting. It includes all the features of the Professional edition, plus Back Office tools such as SQL Server, Microsoft Transaction Server, Internet Information Server, Visual SourceSafe, SNA Server, and more. Printed documentation provided with the Enterprise edition includes the Visual Studio Enterprise Features book plus Microsoft Developer Network CDs containing full online documentation.

Visual Basic Professional Edition

The Professional edition provides computer professionals with a full-featured set of tools for developing solutions for others. It includes all the features of the Learning edition, plus additional ActiveX controls, the Internet Information Server Application Designer, Integrated Data Tools and Data Environment, and the Dynamic HTML Page Designer. Documentation provided with the Professional edition includes the Visual Studio Professional Features book plus Microsoft Developer Network CDs containing full online documentation.

Visual Basic Learning Edition

The Visual Basic Learning edition allows programmers to easily create powerful applications for Microsoft Windows and Windows NT. It includes all intrinsic controls, plus grid, tab, and data-bound controls. Documentation provided with this edition includes Learn VB Now, plus Microsoft Developer Network CDs containing full online documentation.

Benefits of Visual Basic for Applications Licensing

The increasing number of VBA-enabled applications provides opportunities for greater application customization and integration by developers, allowing them to leverage their investments in training in and knowledge of Visual Basic. Ultimately, these developer benefits extend to the organizations and users who select VBA-enabled applications over "build from scratch" solutions. Additional benefits are outlined here:

Licensing Visual Basic for Applications (VBA) enables ISVs to concentrate on their core competency, rather than on language development. It enables them to offer customers an award-winning development environment, and means that ISVs don't have to build proprietary technologies with differing tools and languages.

Competitive Advantage

Visual Basic for Applications delivers a competitive advantage for ISVs seeking to provide full customization and integration capabilities to customers. With VBA-enabled products, ISVs can build broad capabilities into their core product while providing a technology for customers to tailor the application and add features and functionality specific to their requirements.

Simplified and extended applications.

VBA provides ISVs with a way to build VBA-based wizards directly into their products to walk users through simple or complex operations. After products ship, VBA enables ISVs to provide Web-based updates to the core application, delivering new features and functionality between product cycles.

Macro Recording.

With VBA and Macro Recording, ISVs can provide a simple way for end users to automate repetitive tasks while providing developers with an easy way of learning the application programming model.

An enormous developer community.

By licensing Visual Basic for Applications, ISVs can take advantage of the 3.2 million developers already familiar with the Visual Basic programming technology who can use an ISV's packaged applications as development platforms.

ISVs investing in VBA can extend their applications and deliver the tools for meeting customers' specific demands. VBA-enabled products impact the bottom line by providing a

built-in customization technology, enabling customers to pursue a "buy and customize" alternative to building applications from scratch.

Benefits to Developers

Each VBA-hosted application exposes its functionality through an object model, expanding the ActiveX-based component set available for developers to use as building blocks for custom solutions.

Developers can become more marketable because they can use their skill set across many applications.

The ability to reuse code is an immediate advantage because the same Visual Basic is used everywhere.

Visual Basic for Applications enables customization of applications to provide solutions tailored to customers' needs.

With the increasing availability of VBA-enabled applications, developers can now integrate these applications to share data and information more easily and seamlessly.

Perhaps most dramatically, Visual Basic for Applications enables developers to build solutions that previously were cost-prohibitive, because functionality is now available through the integration of different applications or from different vendors.

With VBA available across a broad range of applications, developers can customize and integrate line-of-business applications while leveraging their existing skill set.

Benefits to MIS Managers

Developer knowledge can be used across a broad range of applications.

MIS managers can choose to buy instead of build, while enabling application customization to meet specific business requirements.

MIS managers can adapt to changing resource requirements by taking advantage of the huge number of developers skilled in Visual Basic (over 3.2 million worldwide).

The backlog of end-user application demands can be reduced through code reuse, resulting in a faster response.

Developers can be moved across development projects easily.

Visual Basic for Applications can also play a large role in helping MIS managers and their companies lower training costs by reducing the number of development environments or languages in which their developers need to be trained.

Benefits to End Users of Application-Based Solutions

Solutions perform faster, thanks to tight integration between VBA and host applications.
Solutions look and work like the applications users already know, so less training is required.
Solutions can be user-customized, with respect to print options or query creation, for example.
There is greater participation in the solution design process—users can create the output, reports, and documents that they want automatically generated.
Overall, users will benefit the most from improved solution quality and customized functionality, as the applications they use today incorporate richer functionality and integration, and are tailored to meet their needs.

Visual Basic for Applications

With the release of VBA 6.3 in March 2001, Microsoft has built on the power of VBA 6.0, and has included new features that extend the power, flexibility, and security of the development environment. This has opened the door for new ISVs to develop even more powerful solutions using new features, such as multithreaded VBA-based projects, developer productivity add-ins, and support for digital signatures. And with new integration technologies built by Microsoft, ISVs can integrate VBA into their applications more quickly and easily than ever.

Visual Basic for Applications 6.3 is a core component of Microsoft Office XP (it's now in the Microsoft Outlook messaging and collaboration client and the FrontPage Web site creation and management tool, as well as Microsoft Access, Microsoft Excel, Microsoft Word, and the Microsoft PowerPoint presentation graphics program). Through the VBA licensing program, Microsoft is making the same version of Visual Basic for Applications in Microsoft Office broadly available for use in non-Microsoft applications, providing the same ease of use and power of Visual Basic to a broad range of new applications.

How Does Visual Basic for Applications Fit with Other Microsoft Tools?

Microsoft offers a number of development tools aimed at specific developer skills and needs. These include the Microsoft Visual C#, Microsoft Visual C++, Microsoft Visual J++, and Microsoft Visual FoxPro development systems; Microsoft Office Developer; and the Visual Basic family: Visual Basic .NET, Visual Basic for Applications, and Visual Basic Scripting Edition (VBScript). Tools such as Visual C#, Visual C++, Visual J++, Visual FoxPro, and the Visual Basic programming system support developers who build their solutions from scratch to meet highly specific market needs. Microsoft Office Developer and Visual Basic for Applications support those developers who choose to buy and customize packaged applications rather than build from scratch. Buying and customizing off-the-shelf software reduces the cost and time of solution development when compared with building from scratch. The Visual Basic family is designed to offer powerful programming capabilities based on an easy-to-learn and easy-to-use programming language.

Each member of the Visual Basic family also has specific uses. VBScript is designed to offer lightweight scripting capabilities for low-memory environments, such as Web browsers, and is most commonly used in creating HTML Web pages. Visual Basic is the world's most popular rapid-application development tool for creating stand-alone software components, including executable programs, ActiveX Controls, and COM components. Finally, Visual Basic for Applications takes the same power available through the Visual Basic programming system and applies it to highly functional applications, enabling infinite levels of automation, customization, and integration.

Visual Basic

You can access the Visual Basic sample code files in one of two ways:

- Look through the documentation abstracts. When you find an interesting sample, click on the link at the top of the abstract to download the sample files.

The following lists show the Visual Basic sample programs, organized by category.

ActiveX

ActXDoc.vbp ActiveX Document tutorial. AXData.vbg ActiveX components acting as data sources for other controls. Coffee Creating and using ActiveX components.

CtlPlus.vbp Creating an ActiveX control. DatAware.vbp Creating classes that can act as sources or consumers of data. GeoFacts.vbp Demonstrates the use of Excel objects in a Visual Basic application.

Controls

ChrtSamp.vbp Using MSChart control to display data from an Excel worksheet. Controls.vbp Shows use of controls such as the TextBox, CommandButton, and Image. CtlsAdd.vbp Adding controls to an application at run time.Datatree.vbp Using the TreeView, ListView, and ProgressBar.Dialer.vbp Using MSComm control and a modem to dial a phone number. ListCmbo.vbp Data-binding to a list box and combo box.MCITest.vbp Shows basic functionality of the Multimedia MCI Control. OleCont.vbp OLE Container control. RedTop.vbp Creates an animation of a spinning top. VBMail Demonstrates the use of the MAPI controls by sending and receiving electronic mail. VBTerm.vbp Terminal emulation using the MSComm control. WinSeek.vbp Searching for specific files; uses ListBox controls.

Data Access and Data Binding

AXData.vbg ActiveX components acting as data sources for other controls. BookSale.vbp Uses an Automation server to encapsulate the logic of business policies and rules. Data Environment Demonstrates the new Data Environment designer. Data Report Demonstrates the new Data Report designer. DatAware.vbp Creating classes that can act as sources or consumers of data. Datatree.vbp Using the TreeView, ListView, and ProgressBar. FirstApp.vbp Using the Data control and other data-aware controls.ListCmbo.vbp Data-binding to a list box and combo box.Loan.vbp Using ADO and creatable recordsets to dynamically populate a DataGrid control. MSFlexGd.vbp Using the MS FlexGrid control. Visdata.vbp ADO techniques.

Enterprise

Callback Server-initiated callback to the client. Hello World Remote Automation Simple remote automation. Interface Uses the COM apartment model resource allocation algorithm. Message Queue Enterprise messaging. Passthrough Server Simple pass-through server. Pool Manager Clients ask the pool manager for a pointer to an object.

General Programming

ATM.vbp How to use a resource file.CallDlls.vbp Calling procedures in dynamic-link libraries.Controls.vbp Shows use of controls such as the TextBox, CommandButton, and Image. Errors.vbp Error-handling techniques.FirstApp.vbp Using the Data control and other data-aware controls.MdiNote.vbp Making a simple multiple-document interface application. Menu creation.Optimize.vbp Optimization techniques. ProgWOb.vbp Programming with objects. SdiNote.vbp Making a simple single-document interface application. Menu and toolbar creation.TabOrder.vbp Reset the tab order of a given form using Visual Basic Extensibility model.

Graphics

Blanker.vbp General graphics techniques. Palettes.vbp PaletteMode settings; the Picture object.

Web

ActXDoc.vbp ActiveX Document tutorial. DhShowMe.vbp DHTML techniques.PropBag.vbp Storing state values between HTML pages. Support1.vbp Using webclass andADOtechnology to create an application. Wcdemo.vbp WebClass demonstration.

Caching ADO ODBC Connections with Jet

A key part of connection management is that Jet caches either one or two connections, depending on the server. For servers such as Oracle, which allow pending results on a connection, Jet caches one connection. For servers such as Microsoft SQL Server, which do not allow pending results on a connection, Jet caches two connections.

Note SQL Server's server-side cursors support multiple operations on a single connection as implemented with ODBCDirect and RDO. However, there is no support for server-side cursors with Jet.

When Jet needs to open a connection, it first checks its internal connection cache. If there is a connection in the cache that uses the same DSN and database parameters, and there are no uncompleted queries pending on the connection, it is reused. Back-end database systems that support pending results on a single connection may not need additional connections to perform simultaneous read/write operations.

Note Jet caches the user ID and password along with the connection, so that youre not repeatedly prompted. This means that if your application needs to log on to the server with a different user ID and password, you will be unable to do so unless you force the closure of any existing connections.

Jet ages each connection based on elapsed time and its activity. After a configurable connection timeout period (which defaults to 10 minutes), Jet automatically closes and drops any dormant connections. For a connection to be considered dormant, it must have no open Database or Workspace objects associated with it. Jet will not close connections if there are uncommitted transactions, or queries with unfetched results. Since Jet automatically closes connections, this implies that Jet automatically re-opens connections as needed.

Note The ConnectionTimeout setting can be adjusted by accessing the Windows system registry.If your application needs access to a connection that Jet has timed out and closed, the connection is automatically reopened. Assuming that the connection is re-established, this should not cause a problem with your application.

In some cases, if a shared DSN is identical, queries against a second Database object might be blocked while Jet waits for the DSN to become available.

Accessing SQL Views Using ADO

If the remote database only exposes SQL views, you can access this data by attaching those views to a Jet database and creating pseudo indexes to the view using a ADO action query. Although not actually an index, a *pseudo index* allows Jet to create an updatable recordset on the view. You dont need to create a pseudo index if you are not updating server data.SQL views can also be accessed by ADO through ODBCDirect. In some cases these views are updatable using the indexes already available on the remote server.

Accessing Stored Procedures Using ADO

In some environments, access to server data is limited to a set of server-based stored procedures. In this case, some or all data requests and updates are carried out through these stored procedures — especially when you have no direct access to the remote tables. In such an environment, you must use SQL pass-through queries exclusively or use ODBCDirect if you choose to use ADO. If your server forces all queries and updates to be executed through stored procedures, then you can use SQL pass-through queries to execute the UPDATE stored procedures as well as the SELECT stored procedures. You can then base other Jet QueryDef objects on these queries as if they were attached tables.

Closing ADO ODBC Connections with Jet

When you close a Recordset or Database object, or when these objects are no longer in scope, the connections they use are released to the connection cache. For example, if you declare a Recordset object in a procedure, and that procedure ends, the recordset is automatically closed and any connections needed to support it are released to the cache.When your code visits the last record of a Recordset object, as when you execute the MoveLast method, the connection used to populate the recordset is released to the cache. A single connection is maintained to perform updates or other action queries against all open Recordset objects.When your code closes a Database object or the object loses scope, Jet closes the Database and any associated Recordset objects. Any connections associated with those objects are releasedtothe cache Each Data control functions like an OpenRecordset method. That is, each Data control creates one or two connections (depending on the size of the result set and the functionality of the server being accessed) when they are initialized. Visual Basic automatically populates

Recordset objects created by the Data control to release connections as quickly as possible. This happens during idle time, and at a configurable rate determined by the MSysConf table settings. Generally, Jet maintains a single connection to perform updates, but until the result set associated with each Data control is fully populated, a second connection must remain open to return the rows. When your code positions the recordset to the last row, as when you execute a MoveLast method, this extra connection is no longer needed and is returned to the pool.

ADO Remote Data Access Using Jet

This section discusses ADO functionality when it is connected to the Jet engine. The Microsoft Jet database engine is a stand-alone database management system that is capable of both processing queries and routing queries to remote servers as needed. Accessing Jet through ADO adds to Microsoft Visual Basics ease of development by providing an object-oriented development paradigm and accessibility to data-aware bound controls.

By using the Data control, ADO, or Microsoft Access, you can create code that is virtually database-independent, because Jet automatically performs all syntax and data manipulation translations for you. For example, you can write an application that accesses different types of data sources without making reference to specific remote server features. These data sources could be Open Database Connectivity (ODBC) databases, such as Microsoft SQL Server; Index Sequential Access Method (ISAM) databases, such as Microsoft FoxPro, Paradox, or dBASE; or other Jet databases.Unlike most stand-alone database engines, Jet can perform heterogeneous joins across several dissimilar databases. If you are working with departmental data stored in ISAM format, and need to merge it with data on a centralized server, this is an essential feature.

Establishing ADO ODBC Connections with Jet

Jet requires at least one connection when fetching data from a remote data source. If you indicate that the result set is to be updated, Jet attempts to open an additional connection unless an existing connection can be used. That is, one connection is used to populate the result set, and another to update it. However, once the result set is fully populated — as when you use the MoveLast method — the first connection can be closed or returned to the connection cache.

Providing User ID and Password

To gain access to a remote ODBC data source, you usually have to provide a valid user ID and password combination. These values can be provided in the Connect property of the Data control or in the connect string that is supplied as an argument to the OpenDatabase method. If these values are not supplied, the ODBC Driver Manager exposes a dialog to collect the user name, password, and other missing information needed to establish the connection. There is no way to disable this dialog with ADO and Jet. However, by using the ODBCDirect or RDO prompt arguments, this dialog can be disabled and your code can intercept a trappable error.

Opening Connections Directly

Jet follows these guidelines when managing connections to Microsoft SQL Server:

- When using the OpenDatabase method, Jet opens a new connection, or attempts to re-use an existing connection if an identical DSN exists in the cache. The connection remains open after the Database object is closed (in anticipation of later use), unless there is already a cached connection to that server available. Only one connection for each DSN remains open in the cache.

- When you open a connection directly using the OpenDatabase method, the ADO/Jet model is forced to query the database to determine the name of each available table there. This information is cached in the Database object and exists (does not have to be refetched) as long as the Database object remains instantiated.

- With OpenRecordset, Jet tries to share an existing connection, reuse a cached connection, or, failing both of those, opens a new connection to the server and executes a query based on the *source* argument (or the SQL property of a QueryDef object). As soon as a MoveNext method is executed, Jet fetches the first 100 rows. If this does not complete the query, an additional connection is opened to support updates. The first connection must remain open until the recordset is fully populated or closed to support updates.

Opening Connections Indirectly

It usually more efficient to open connections to a remote data source by having Jet perform the operation. This is accomplished by simply opening a Jet database that contains linkages to remote database tables or views. When you access these attached (linked) objects, Jet

establishes the connection using cached connection information that you provided when creating the attachments. However, if Jet is unable to complete the connection for whatever reason, the ODBC driver manager exposes a series of dialogs to attempt to collect logon and DSN information so the connection can be established. Using ADO with Jet, there is no way to disable these dialogs.Another alternative is to provide the needed connection information to ADO and Jet by setting the Connect property on an open ADO/Jet Database object. Using this technique, you can then use SQL PassThrough queries just as if you had opened the connection directly.

Handling Remote ADO Messages and Errors

Remote server systems generate their own litany of errors and messages. The database itself may contain procedures that generate user-defined messages or errors. Once ADO and the Jet database engine receive any error, regardless of the cause, the query that triggered the error is terminated. For those databases that use the SQL Server RaisError function to indicate warning-level messages, this may be problematic.When using QueryDef objects to execute SQL pass-through queries, other messages received from ODBC and the remote server can be trapped. For example, SQL Server SQL PRINT statements generate a message that can be trapped by your code. To enable message trapping, your code must create a property named LogMessages for a specific QueryDef object, and set this property to True. Once set, messages generated by the selected QueryDef are recorded in a Jet table.Each SQL query that Jet or the remote query processor executes can generate one or more ODBC or other remote-engine errors. All of these errors are stored in the Errors collection, which is accessible either during break mode or at run time. Documentation is available for some of these messages, especially those mapped by Jet to its own error numbers. Most ODBC operations will generate a generic ODBC trappable error that is explained more fully in messages found in other members of the Errors collection.

Managing ADO Data Source Name Entries

Generally, an ODBC connection requires a Data Source Name (DSN) entry. Depending on the operating system, these entries are either kept in the ODBC.ini file (16-bit systems) or in the system registry (32-bit systems). You should not attempt to change these entries manually. Instead, use the Windows Control Panel ODBC Administration applet or the RegisterDatabase method.

Note When using the ADO/Jet model, ODBCDirect, Remote Data Objects, or the ODBC API, it is not always necessary to create or reference a registered DSN if enough information about the remote server is provided in the connect string.

Setting the Default Database

Your code should ensure that the correct default database is set during the connection process. The user may specify a user ID that does not have permission to access the database your application expects to use, or uses a different default database. The default database can be established by:

- Including the default database name in the DSN entry.
- Including the DATABASE= argument in the connect string.
- Establishing a default database on the server based on the user name.
- Submitting an action query that changes the default database once the connection is open

Managing ADO Network Traffic

When using ADO with the Jet engine, a primary consideration is the amount of data that your network is required to carry. This is especially true if your design includes a shared Jet database that contains local, unattached data. In this case, the network will carry all disk I/O traffic as multiple users compete for shared data pages. If the shared database is simply a repository for one or more attached tables, in most cases only the query results need to be transmitted over the network.Your application can control network traffic indirectly, through judicious use of Recordset object size and choice of query processor. In many cases, the Jet query processor can create Recordset objects with comparatively little network traffic. However, some designs may not accommodate its use, and may consequently create more network traffic than would occur with other programming models. By tuning the SQL query passed to the Jet query processor, you can often make better use of its power while improving network performance.When accessing attached tables with the Jet engine, only the linkage information and the results of the query need to be transmitted over the network. If, however, the query processor is forced to download part or all of a remote table, network load increases dramatically.

Using a wide area network (WAN) with a Jet database is possible, and with careful error management, WAN applications can be implemented with a degree of security. Your design should, however, take additional precautions and include extremely robust error management that anticipates the loss of network access to the remote server and often dramatically longer response times. Since WAN networks can be significantly slower than conventional local area networks, special care should be given to the amount of network traffic generated and ADO timeout values. Examine the SQL trace logs for a better understanding of the number and complexity of the queries generated to remote ODBC servers. It is always good design practice to use more robust error handling for all network operations regardless of the topology.

Enabling Trace Logs

One of the most helpful debugging and tuning tools you have at your disposal is the ability of the ODBC Driver Manager to log all ODBC operations to an external file. You can enable this file using the ODBCDirect LogMessages property or by selecting the associated option in the Windows control panel ODBC Administration dialog. Be sure to turn off logging before your application goes into production, as the logging process can significantly impact performance.Another option available to Microsoft SQL Server developers is the new SQLTrace utility that can let developers interactively view the queries submitted by all applications against a SQL Server. Once started, the SQLTrace utility exposes a window that displays each query or other operational request made.

For More Information See Choosing a ADO Query Processor for Use with Jet and Managing ADO ODBC Connections with Jet.

Managing ADO ODBC Users

Each instance of your database application uses some number of connections and data page (or row) locks on the server, and creates a measurable load on the network. Since each additional user contends for many of the same resources, the number of users the system can support is directly proportional to the number of resources each instance of your application requires.To reduce the number of locks, users should not be permitted to sit on unpopulated Recordset objects. The application should populate the recordset as quickly as possible using ADO, the Data control, or one of the background population techniques.

Your design should also include management of user logon IDs and passwords. If your design uses a shared Jet (.mdb) database, you must also address Jet security systems.Because all users must disconnect from Jet databases (that contain data) for periodic maintenance, you should include a way to notify users to disconnect from the shared Jet database or provide a way to signal applications to disconnect automatically on their own. If the maintenance operations are executed during nonpeak hours, and applications automatically disconnect from the Jet database after a length of idle time, maintenance programs can execute without disturbing uncompleted result sets or pending updates.

Managing ADO Query Result Set Size

The Jet database engine is capable of retrieving data from databases of any size. However, as the number of records processed increases, be aware of the increased amount of required TEMP storage space. Any design that opens tables directly, without benefit of a SQL query that limits scope, should be reconsidered. Jet will generate a trappable error if local disk space is exhausted while Jet is building a keyset. This is not really a limitation in Jet but of system resources, and it is characteristic of poor application design. Any application design that requires the database engine to create a physical pointer to each row of the result set data, such as when a keyset is created, has a theoretical upper limit set by the capacity of the media where the keyset is stored.In some cursor models, only a subset of the keyset is maintained on the client machine, or the keysets are built on the server. Although Jet supports a table-type recordset that permits browsing tables without impacting client resources, this is not available when accessing remote (ODBC) data. Instead, only the dynaset-type and snapshot-type Recordset objects are supported.Both of these build the keyset on the client system, overflowing to TEMP space on disk if necessary. The snapshot-type recordset also downloads data, which may further limit the size of the recordset that can be built. In any case, your data access strategy should involve restraining the result set scope. That is, you should limit the number of rows returned by the query.

Remote Data Access Using ADO and ODBCDirect

Visual Basic version offers an additional option that can be used with ADO to access remote database engines: *ODBCDirect*. This ADO option permits your application to choose the database engine and interface used by ADO. Basically, you have two choices:

- The Microsoft Jet database engine. By default, ADO uses Jet to perform all data access operations.

- ODBCDirect. When this option is enabled, ADO loads the Remote Data Objects (RDO) 2.0 libraries and delegates all data access operations to the ODBC data source.

Basically, ODBCDirect maps each of the Data Access Objects to an equivalent Remote Data Object. While not all of the RDO functionality is implemented with ODBCDirect, this approach permits you to leverage existing ADO-based applications using a familiar object model when accessing remote database systems.

For More Information Information about ODBCDirects relationship to RDO is also discussed throughout "Using Remote Data Objects and the Remote Data Control."

The T-SQL Debugger

The T-SQL debugger is integrated with the Data Environment designer. It allows you to interactively debug remote stored procedures written in Microsoft SQL Server's Transact SQL dialect, from within the Visual Basic development environment. Using the T-SQL debugger, you can:

- Display the SQL call stack, local variables, and parameters for the SQL stored procedure.

- Control and manage breakpoints.

- View and modify local variables and parameters.

- View global variables.

Setup and Compatibility

In order to use the T-SQL debugger, you must have SQL Server version 6.5 with Service Pack 3 or later installed as your database server. The debugger uses the functionality exposed by SQL Server's Sdi.dll, and exposes that functionality through Remote Automation.The client-side components of the T-SQL debugger are correctly installed and configured when you choose to install all the Enterprise tools in your Visual Basic installation. If it is necessary to repeat the setup process, select "Custom" from the CD Installation dialog box, and choose "Select All" for the Enterprise Tools selection.

Server-Side Setup

With SQL Server version 6.5 and Service Pack 3 or later installed, you can install and register the SQL Debugger interface and Remote Automation component on the server. These components are located at \Program files\Common Files\Microsoft Shared\SQL Debugging. On Windows NT 4.0 or later, simply run the setup program Sdi_nt4.exe.**Note** For setup on NT Server 3.51, you must manually copy and register the necessary files. Complete instructions for this process are included in the Readme.txt file in the \Program Files\Common Files\Microsoft Shared\SQL Debugging folder.

Using the T-SQL Debugger

There are different methods you can use to invoke T-SQL debugging.

1. To debug a stored procedure or batch query at design time, add the T-SQL Debugger Add-In via Visual Basic's Add-In Manager (on the Add-Ins menu). Then you can start the add-in by clicking **T-SQL Debugger** on the Add-Ins menu. You then simply select a DSN, and either **Stored Procedure** or **Batch SQL** and click the **Execute** button. This will invoke the debugger and allow you to debug the SQL you are interested in.

2. To debug stored procedures while debugging Visual Basic code (run-time debugging), select **T-SQL Debugging Options** on Visual Basic's **Tools** menu. The options dialog box allows you to:

   - Turn on automatic step into stored procedures, which will bring up the T-SQL Debugger whenever you step into an ADO or RDO method that executes a stored procedure.
   - Turn Safe Mode on, which will automatically roll back any design-time queries that you debug.
   - Limit the number of rows that appear in the T-SQL Debugger output window when debugging design time queries.
   - Set the login timeout value that the debugger uses to connect to the database, to get internal SQL State. Once you have selected the **Automatically step into Stored Procedures** check box, if you step into (F8) a line of code that executes an ADO or RDO method that invokes a stored procedure, the debugger will automatically be

started. You can then step through the stored procedure and then continue debugging your Visual Basic code.

Note SQL Server will return from a stored procedure before it has finished executing if the stored procedure returns enough data to fill its buffers. If this happens, both the T-SQL Debugger and the Visual Basic debugger will be active at the same time. Your Visual Basic code must fetch the results from ADO or RDO before the stored procedure will complete its execution. If this happens, make sure your basic code reads the result sets by placing Visual Basic in Run Mode (F5) and setting breakpoints where you would like to stop execution. You can toggle back and forth between Visual Basic and the T-SQL Debugger by using the taskbar or using the ALT+TAB key combination.

3. You can also launch the T-SQL Debugger:

- From the Data Environment designer
- While stepping through ADO or RDO code
- By right-clicking a stored procedure in the Data View window and choosing the Debug command
- From the UserConnection designer

Once you have started the debugger, it establishes the ODBC connection and displays the Enter Unassigned Parameters dialog box, as shown.

Unassigned Parameters Dialog box

Enter values for any unassigned parameters in the Value field, then click OK. The T-SQL debugger interface appears and displays the text of the stored procedure:

Debugging Options

With the SQL statement displayed, several debugging options are available on the toolbar buttons and on the **Debug** menu. These options include:

- 2Go
- Set and clear breakpoints

- Step
- Step into subexpression
- Step over subexpression
- Run to cursor
- Stop debugging
- Restart

Views and Options

In addition to the code window containing the SQL statement you are debugging, the T-SQL debugger interface presents separate output windows for local and global variables, and for the output (result set) of the query. The View menu also allows you to open a separate Call Stack window and a Temp Table Dump window, so that you can examine these as the code executes.The Options menu lets you customize the appearance of the T-SQL debugger by changing the fonts and colors used for display.

Exiting from the T-SQL Debugger

When you are finished with your debugging session, click Exit on the File menu to close the debugger. To execute a query again, click Restart on the Debug menu.

Troubleshooting

If you are having problems getting T-SQL debugging to work, you will need to check the event log on the server. SDI.DLL will log events in the application section of the event viewer. COM or distributed COM errors will log events in the system section of the viewer.

- Make sure that the two computers can communicate with each other. The easiest mechanism to do this is by typing **ping** and the computer name of the client at a command prompt on the server if you are running TCP/IP. If this fails, fix the connectivity problem between the machines.
- Make sure the file SDI.DLL resides in the same directory as SQLSERVR.EXE. This will be in the binn sub-directory under the main SQL Server directory. The default is c:\mssql\binn.

- Ensure that the RPC services are started on the server machine. You do this by starting the control panel, opening the services application and checking that the Remote Procedure Call (RPC) Service is running and set to start automatically, as well as the Remote Procedure Call (RPC) Locator.

- Ensure that SQL Server is not set to log on as the SystemAccount. You do this by starting the control panel, opening the services application and double clicking on the MSSQLServer service. If the service is set to run as the SystemAccount, change this so the server will log on to a specific account that is valid to the domain that you are in. If debugging still fails, make sure that the account SQL Server started as has sufficient rights to launch an automation server on the client machine.

- If you see COM error 80080005 in the event log, make sure that you did not start remote automation (autmgr32) from the command prompt. Autmgr32.exe should only be running in the winstation of the account that SQL Server logged in as. Any other winstation will cause problems. If this is the case, close down autmgr32.exe via the task manager and let the sdi.dll and autprx32.dll load autmgr32 via COM.

- Make sure Remote Automation is successfully installed on the server and client machines, if both the client and server do not have Distributed COM (DCOM) installed and loaded.

- If your client system is running Windows NT 4.0 or later, run DCOMCNFG and make sure that everyone has launch and access permission for vbsdicli.exe.

Using ADO to Select a Remote Query Processor

Sometimes you need to force the remote query processor to execute the query. As discussed in "Managing ADO ODBC Connections with Jet," earlier in this chapter, opening a Database object directly can be very costly in time and network traffic because the structure of the remote database and its tables must be determined by sending a number of queries to the remote database.In cases where you must use remote database-specific SQL syntax, or you want to use the remote database engine's query processor, you must bypass the Jet query processor by using the ***dbSQLPassthrough*** option with the ADO Execute or OpenRecordset methods. It is also possible to create ADO QueryDef objects that bypass the Jet query processor.To use QueryDef objects, your application will need access to a Jet .mdb database. There are two kinds of QueryDef objects:

- QueryDef objects used for pass-through queries, which use the servers syntax and cant refer to attached tables.

- QueryDef objects used for non-pass-through queries, which use attached tables and attempt to translate and send as much of the query to the server as possible.

Using ADO to Select the Jet Query Processor

All queries executed by the Jet query processor must be written using Jet SQL syntax. However, Jets SQL syntax is not always the same as the SQL syntax used on your server database. Jets SQL dialect is the same, however, regardless of the database it needs to access. This feature can provide significant portability in your code and the ability to seamlessly access heterogeneous data.By default, the Jet query processor is invoked when any ADO query is executed. In other words, unless you use the ***dbSQLPassThrough*** option with the Execute or OpenRecordset methods or create a SQLPassThrough QueryDef object, the Jet query processor will parse and execute the query's SQL syntax, and attempt to perform whatever operations are needed on the workstation and the remote server to carry out the request.

Using ADO to Share Remote Data

In any client/server application, one of your primary design concerns will be how to best share the data resources. When using either the Jet or remote engine query processor, your design must include code that deals with conflicts caused by instances of your database application and other applications trying to access the same database. Any design must include robust error handling to deal with a variety of contingencies caused by conflicts that arise as multiple applications vie for the same server resources.If your design calls for a centrally shared Jet database that includes attached tables, each client system must also contain DSNs that are compatible with your database. Since the DSN is maintained on the client and only referenced by name in the shared database attachments, your setup routines must ensure the client DSN description is correct — and remains so — to eliminate any chance that the user might change one or more parameters.Although the remote database engine is responsible for managing its own page locks and resources, any application, including those that use the Jet query processor, can lock pages on the server for indefinite periods of time.

Using SQL PassThrough Queries with ADO

In many applications, youll use both Jet queries (that is, queries executed by the Jet database engine) based on attached remote tables *and* SQL pass-through queries. With a Jet query, the query engine determines which parts of the query can be sent to the server and which parts must be processed locally, thereby combining the power of the server with the capabilities of the Jet database engine. With a ADO Jet SQL pass-through query, your code provides a SQL statement that Jet sends directly to the server without stopping to compile the query. Once the SQL pass-through query is complete, if it creates a result set, the Jet recordset processor creates a snapshot-type Recordset object to manage it.

MICROSOFT ACCES DATABASE

Overview

Acces is an electronic publishing system for teachers, sort of a combination between a database, desktop publisher, and word processor. The program stores test items and curricular material very efficiently, supports many different page layouts, and produces beautiful, typeset-quality documents.

While some people like to call Acces a "test generator," it is really much more than that. In fact, it can be a great help to teachers in their regular instruction, and it can benefit students immensely. For example, teachers can use the software to: customize lessons for students with special needs. supplement textbooks with interesting and challenging questions.

create a variety of classroom materials, such as overheads, flash cards, game cards, assignment schedules, and calendars. prepare students for state assessments and standardized tests like the SAT. produce daily assignments, review worksheets, class warm-ups, and other documents, that are closely aligned with a school district's or state's curriculum.

Although Acces addresses many subjects, it is especially well-suited for mathematics, because it has built-in support for formulas, graphics, and special symbols. Therefore, most of our modules are designed for math teachers. However, do not be disappointed if you work in a different field; we also have science and language arts modules. History will be covered soon. If you are a curriculum supervisor, technology specialist, or testing coordinator, then you will definitely want to read on about Acces' capabilities. Making use of database modules

One of Acces' most impressive features is the size of its database. Currently, the program offers more than 300,000 problems in over 40 modules. This is no doubt the largest computer-based collection of math problems available—and it continues to grow!

When we use the term database, we are really talking about two things: a computerized storage and retrieval system, and various "add-on modules" or item banks that are available for Acces. In this section, we go over some general points about the database system and describe features that are common to all modules. We invite you to see our Add-on Modules page to determine which, if any, modules are appropriate for your needs. But please keep in mind that you can also use Acces to write your own problems and store them on the computer. Here is some general information about the database: Acces is a print-based system. This means that you select problems by looking at a printed catalog and telling the computer what you want. We adopted this method because it is much faster than scrolling through problems on the screen. It also makes locating specific kinds of problems very simple, because the catalogs are divided into sections (or topics) with dozens of related problems on each page. The process is actually similar to flipping through a teacher's edition of a textbook. But the software has a tremendous advantage: it puts the equivalent of dozens of textbooks and your entire filing cabinet on the computer. Plus, it handles all of the cutting and pasting, so problem sets and exams are just a few keystrokes away.

All of the material in Acces' database is authored by real people, then stored on the computer. The software does not "generate" items as do some other testing programs. This means you get lots of interesting and subtle variations of problems, rather than endless repeats. Put differently, there is no trade-off between quality and quantity; Acces not only gives you a huge number of problems, it provides an excellent balance between introductory and advanced topics, or between basic and higher-order thinking skills. (Another advantage of a real database system, as opposed to a "test generator," is in the archiving of existing material. The module containing New York Regents Exam questions or any of the math contest databases are good examples of this archival capability.)

Acces never tries to outguess you. You select the items you want from the database in the order you want them. Since the questions are not generated on the fly, there are no surprises. Even the answers are shown in the printed catalog, so you know exactly what you are getting. But don't get us wrong, there are lots of ways to automate the selection process, if you wish. Acces can be told to pick problems at random, scramble their order, or produce an alternate version of a test or a quiz. The important point is that Acces follows your instructions: it is designed to make a teacher's job easier; it does not pretend to do the job better.

Acces' database can be used to store just about any kind of problem: multiple-choice, free-response, fill-in-the-blank, true-false, column-match, etc. The program can even handle some

very unusual question types, such as quantitative-comparisons found on the SAT I or grid-in answers found on many modern assessments. The kind of material you write and store on the computer is entirely your decision. EducAide's database modules tend to include either multiple-choice or free-response questions, but also have excellent open-ended questions. By the way, one of the options in the software is to hide the answers, so that a multiple-choice question can do "double duty" as a free-response question, or even be used with an answer grid. All database modules come complete with clipart (pictures), tables, charts, and any auxiliary files, such as reading passages, that are necessary to make use of the items. You simply install a module into Acces and the items are ready to go. As a side benefit, you can also make use of the included clipart in any new problems that you write, or you can "extract" an existing problem and make any changes to it that you like. This makes the material in the database go even further. Amazingly, everything is also portable between computer platforms. EducAide supplies different data discs for PCs and Macintoshes to make installation easier, but you pay for a module only once, regardless of how many platforms you intend to run it on. (You will, of course, have to license Acces for the different platforms.) If you build your own database module, it will also work on either a PC or Macintosh. Desktop publishing capabilitiesIn addition to managing database modules (or banks of questions), Acces is a very powerful authoring and publishing tool. If you like to create your own problem sets and exams on the computer, then Acces can serve as a complete replacement for your word-processor or desktop publisher. In other words, you can do all of your writing inside the program, or do a combination of writing, selecting problems from the database, and modifying existing problems.

As mentioned earlier, Acces is especially suitable for mathematical and scientific material, because it can create virtually every mathematical notation that you can think of. In some ways, Acces works like an equation editor, but it goes one step further: math is totally integrated with text, so problems are fast to write and easy to modify later. We have commented several times on Acces' documents. So as not to sound immodest, we should explain that Acces is built around a very powerful typesetting system called TeX (pronounced tech). The system is widely known in academic and publishing circles for producing beautiful documents. The system has particular advantages for Acces, because it is programmable. This has allowed us to create numerous document types or "templates" and to support page layouts that are too difficult even to consider doing with a word-processor—say, a two-column document with varying amounts of "workspace" and a rectangular answer box next to each problem.

Currently, Acces supports six document types. These are: Test/worksheet, Standardized test, Overheads, Flash cards, Weekly calendar, and Monthly calendar. There are numerous options for each document type, and you have essentially complete control over font size, page headers, margins, spacing, numbering, and many other fine-tuning features. You will likely find many creative uses for each document type. For example, with the overhead and flash card options, you can produce game cards, bookmarks, bulletin board items, and other classroom materials. The calendar templates are just as interesting; you can use them to create assignment schedules or to provide students with daily warm-up exercises. To switch from one document type to another, you simply indicate your preference to Acces.

There are two other advantages to using Acces as a production tool. First, it provides exactly the same options for the problems you write as it does for the problems you select from a database. Second, the process of writing or selecting problems is almost completely independent of page layout decisions. This means you can do things in almost any order: select some problems, see how they look on a certain type of document, make some changes, try a different type of document, and so on. Acces handles all the finer details of production and, as you will quickly discover, it is extremely good at what it does.

Finally, the reason Acces is so flexible is that it does something called dynamic formatting. Problems are not formatted when they are written or stored on the computer, but rather when they are made part of an actual document. This approach is considered state-of-the-art in electronic publishing, and it has much in common with HTML, the language of the World Wide Web. From the standpoint of database development, there are many advantages, including more efficient problem writing, where the focus is on content, not layout. Even if you are most interested in using existing problems from a database, and not writing your own, you still benefit from some amazing formatting capabilities. There is no question that Acces saves time, reduces the drudgery of certain tasks, and makes teachers' jobs more enjoyable. The time-savings alone is significant; Acces users tell us they are able to spend much more "quality time" with students, helping with their lessons and evaluating their work. But that is not actually what impresses teachers the most...

Acces is exceptionally easy to use. You can produce professional-looking documents in just a few minutes, regardless of your level of experience. In fact, Acces is a great way to get "non-computer" persons to start using technology.

Acces is highly adaptable. You do not have to make any compromises in the way you do things. Acces' database modules are excellent sources of material. There are lots of interesting problems from which to choose and most topics in the curriculum are well covered.

Acces makes testing much simpler and more secure. You can easily produce multiple versions of a test or quiz, and offer make-up tests without the usual hassles.

Acces is really liked by students. No kidding! Many students are relieved not to be given handwritten problem sets (or word processing documents with missing symbols). But they are most happy to see materials targeted to their own needs. Plus, they feel better prepared for tests when they can get plenty of extra practice or review problems. Acces and its various database modules are designed for use by elementary and secondary schools or college departments. Therefore, all of the list prices on our Web site include a Site License.

Briefly, the Site License allows you to install Acces on any computer at your school (or department office) or at a teacher's home. Database modules are licensed in the same way, but there is an important rule governing the use of their items: you may reprint the items freely and as often as you like, but you may distribute them only to students and other teachers at your site. We have prepared a sample site license agreement for your review. Please note that this is only a copy of the Acces Site License Agreement that has been shipped with orders prior to March 2003. Since the Acces Site License Agreement is subject to change, the copy posted on this web site is for review only. All use of the software is governed by the license that is included in each shipment of Acces.

CHAPTER THREE :SCHOOL REGISTRATON SYSTEMS PROGRAM

3.1 Pasword Dialog

Figure 3.1 Pasword Dialog Form

Password dioalog form supply entry student information systems program.User can try three times for the entry.İf password entry wrong program will be close.

3.2 Main Page

There is a main form that contents connections for all sub forms(Look Figure 3.2)

Figure3.2 Main Form

In the main form there are 13 buttons for connections that mentioned before . With these buttons user can open related forms. These buttons can be forbidden for some users by Administrator. First button for Modify Student Record button, Second button for Choose Advisors, Third button select for t Student level, Fourth button select for School Year, Fifth button for Make Registration Slip.Sixth button for Ividual Report, Seventh button for Student Per Sections , Eight button for Student List, Ninety button for Population Report, Tenth button for Calculator,Eleventh button for Notepad,Twelfth button for Calendar and last button for About.

3.3 Select School Year

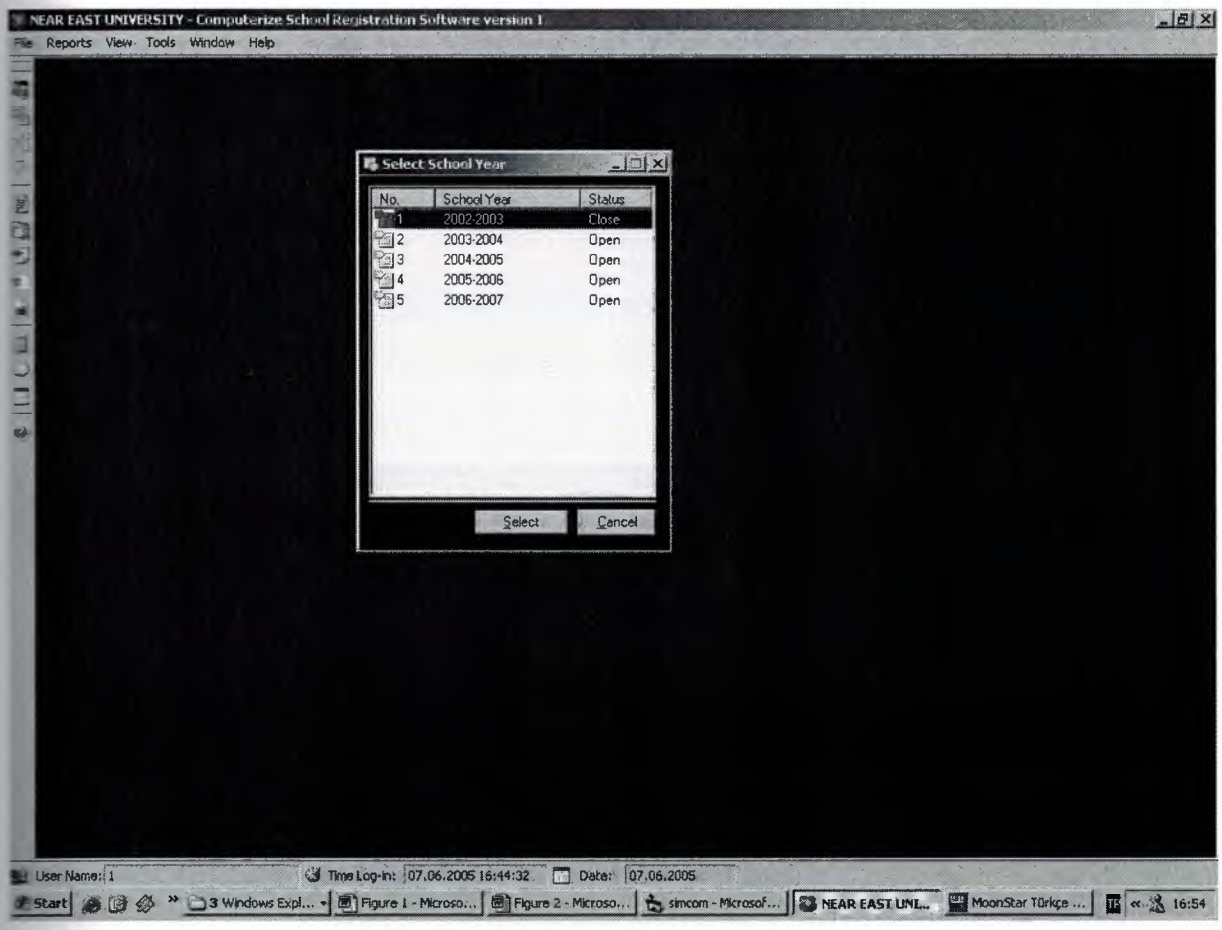

Figure 3.3 Select School Year Form

This form shows years and provides selected semester.

3.4 Student Record

Figure 3.4 Student Record For School Year Form

Student Record For School Year Form shows old recorded student .This form put forward new record button,Edit Button,Show Assign Section Button,Delete button,Reload record button,close button,View option button,Advanced search button.

3.5 Record New Sudent

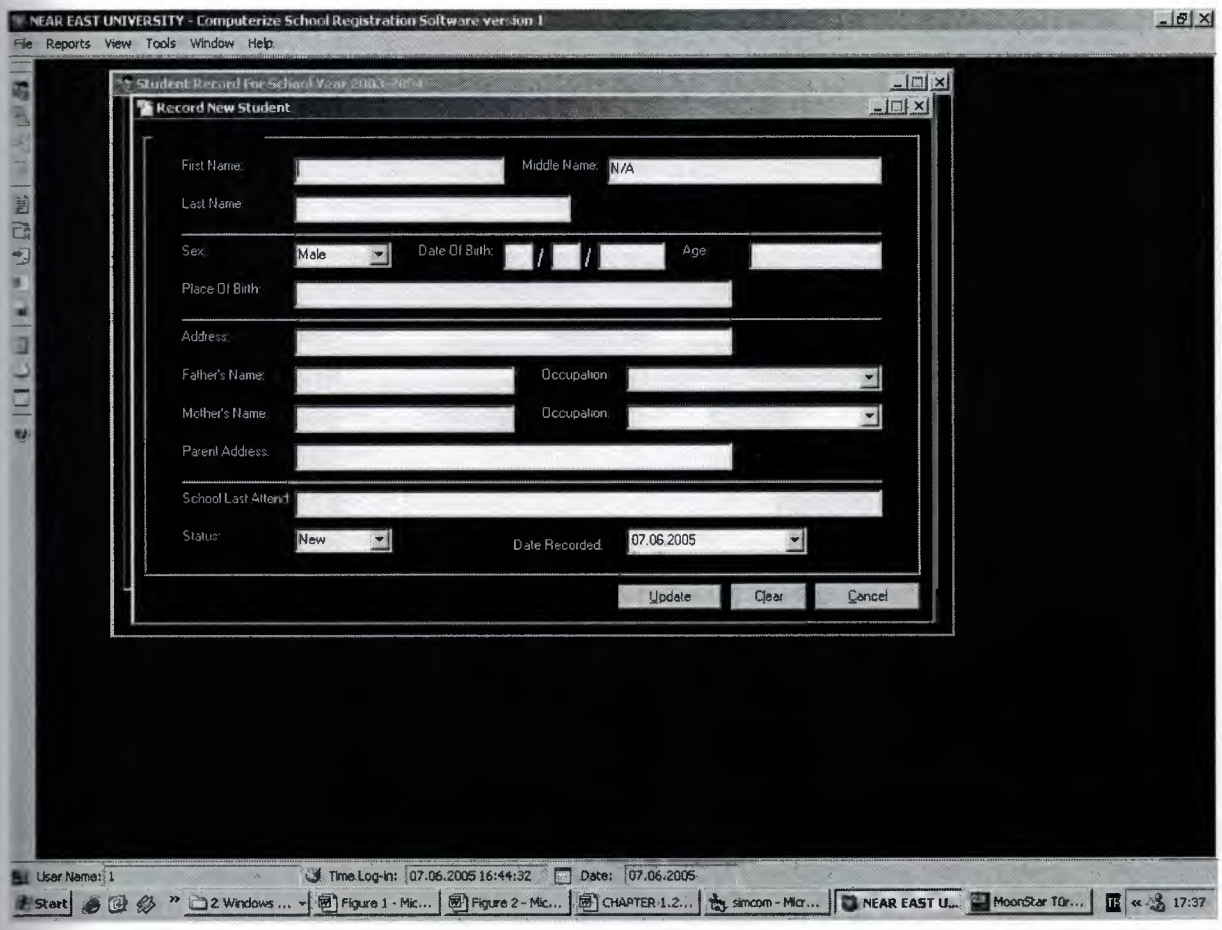

Figure 3.5 Record New Student Form

This form supply new student record.Users fill in the all text box .if user click update button all information will be record.

3.6 Existing Studen Option

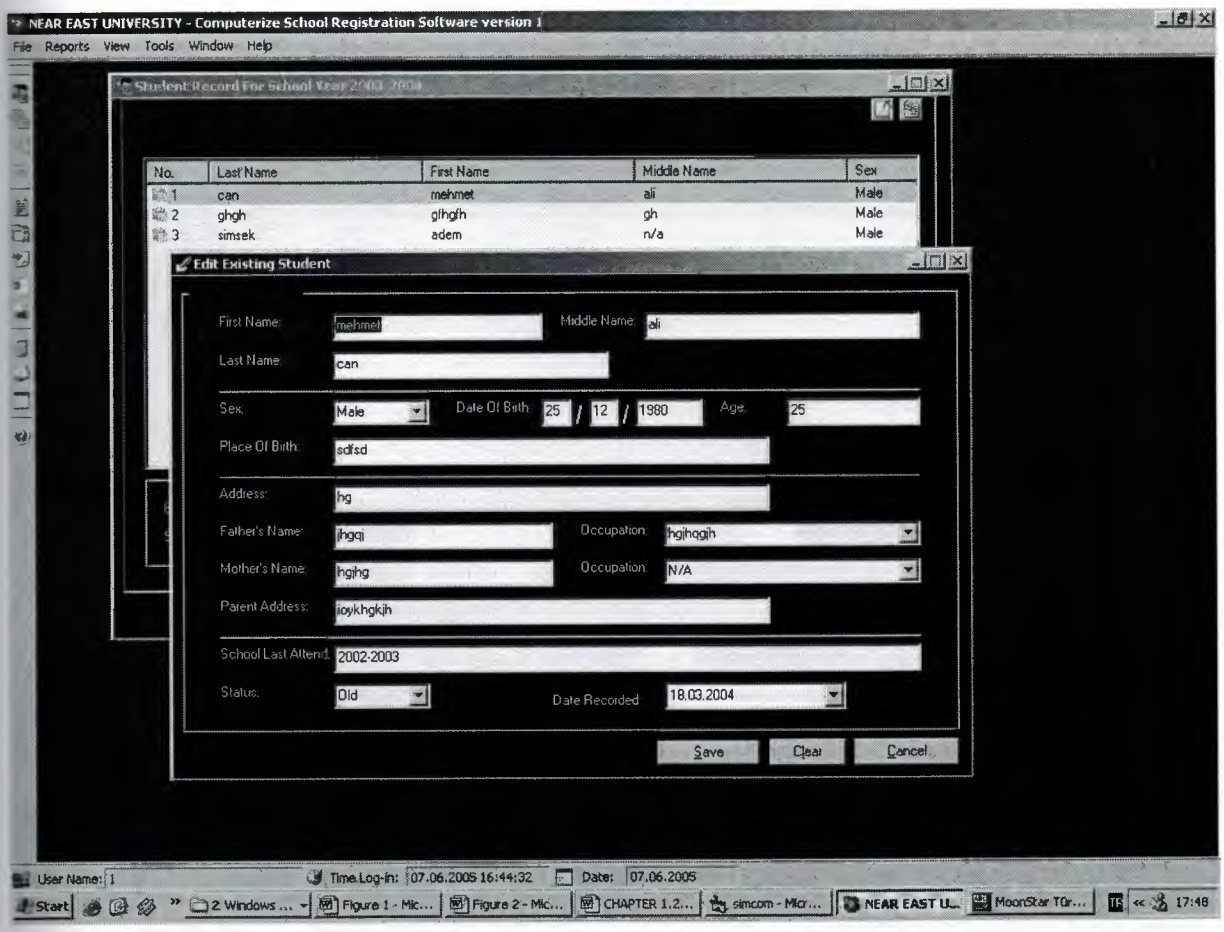

Figure 3.6 Edit Existing Student Form

Using this form users can changes information about exist students.

3.7 Student Advisor Selections

Figure 3.7 Edit Student Advisor Form

This form shows advisor .User can changes and saves advisor name ,year and semester.

3.8 Advanced Search For Student

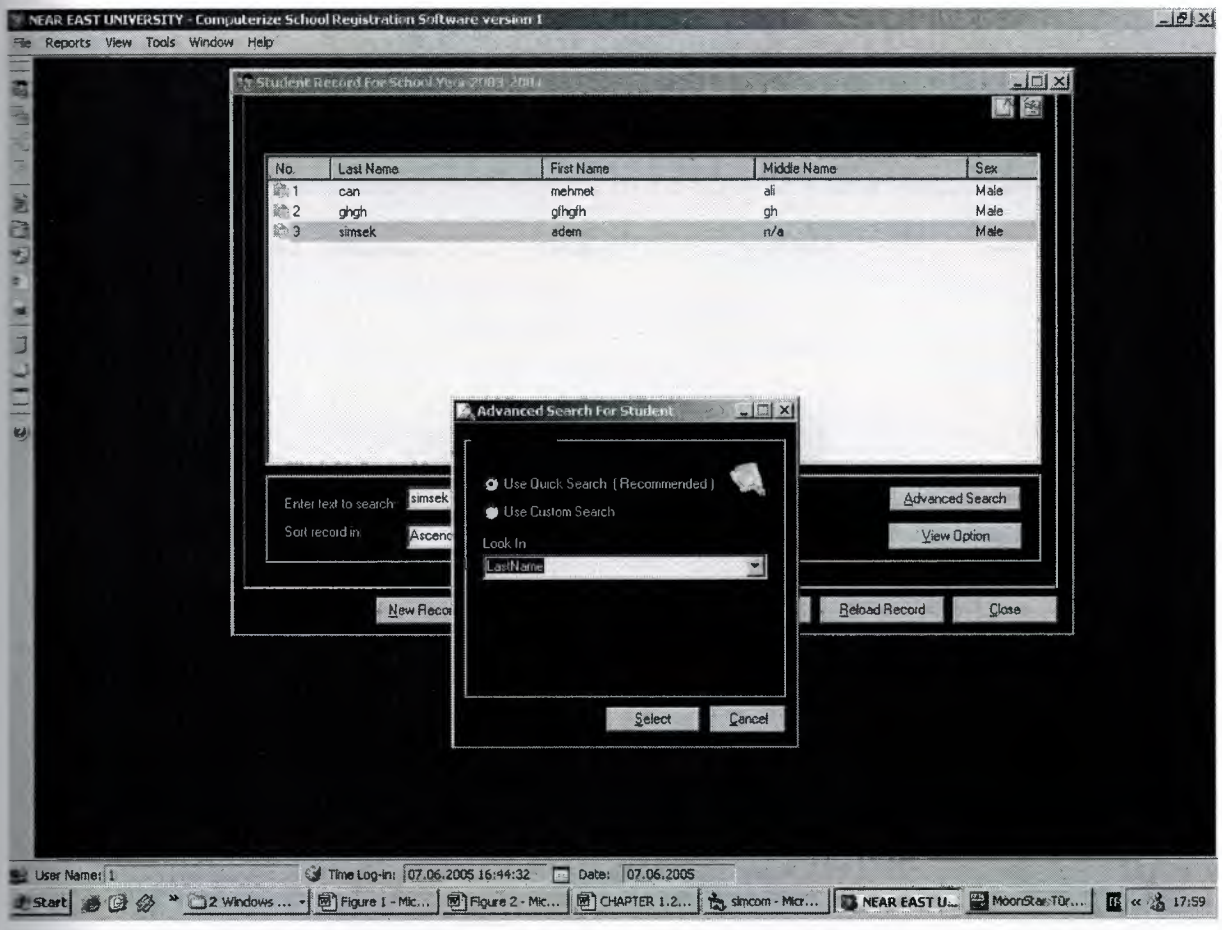

Figure 3.8 Advanced Search For Student Form

This form provides advanced searc.Users writes any contents in the text box and chose which types search form will be shows searc results.

3.9 View Option

Figure 3.9 View Option Form

View Option Form has got four radio box and two menu box.Radio box visible or hide about student information.By menu box users can choose advisor and school year.

3.10 Select Level

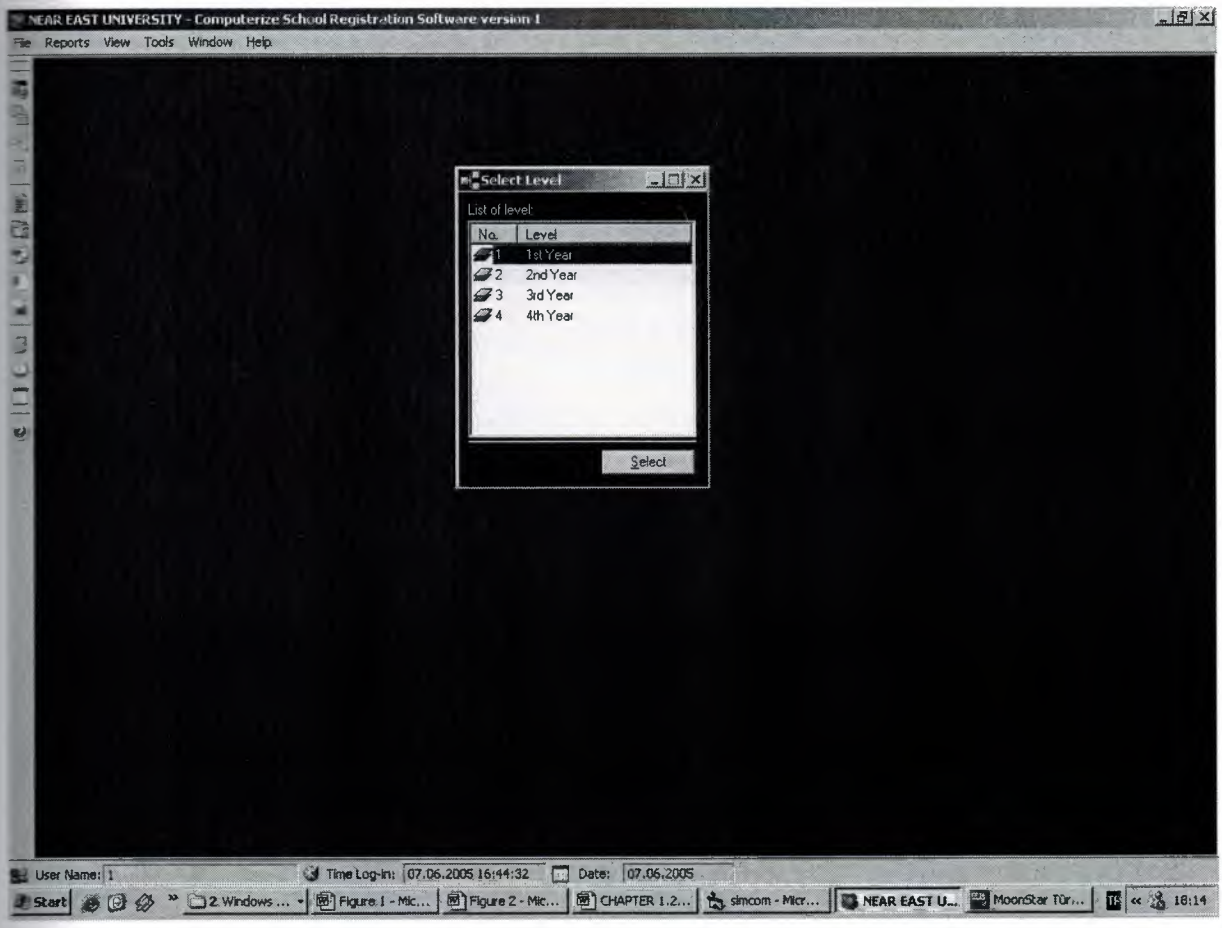

Figure 3.10 Select Level Form

By using this form users chooses semester year.

3.11 List Of Advisor For Semester Year

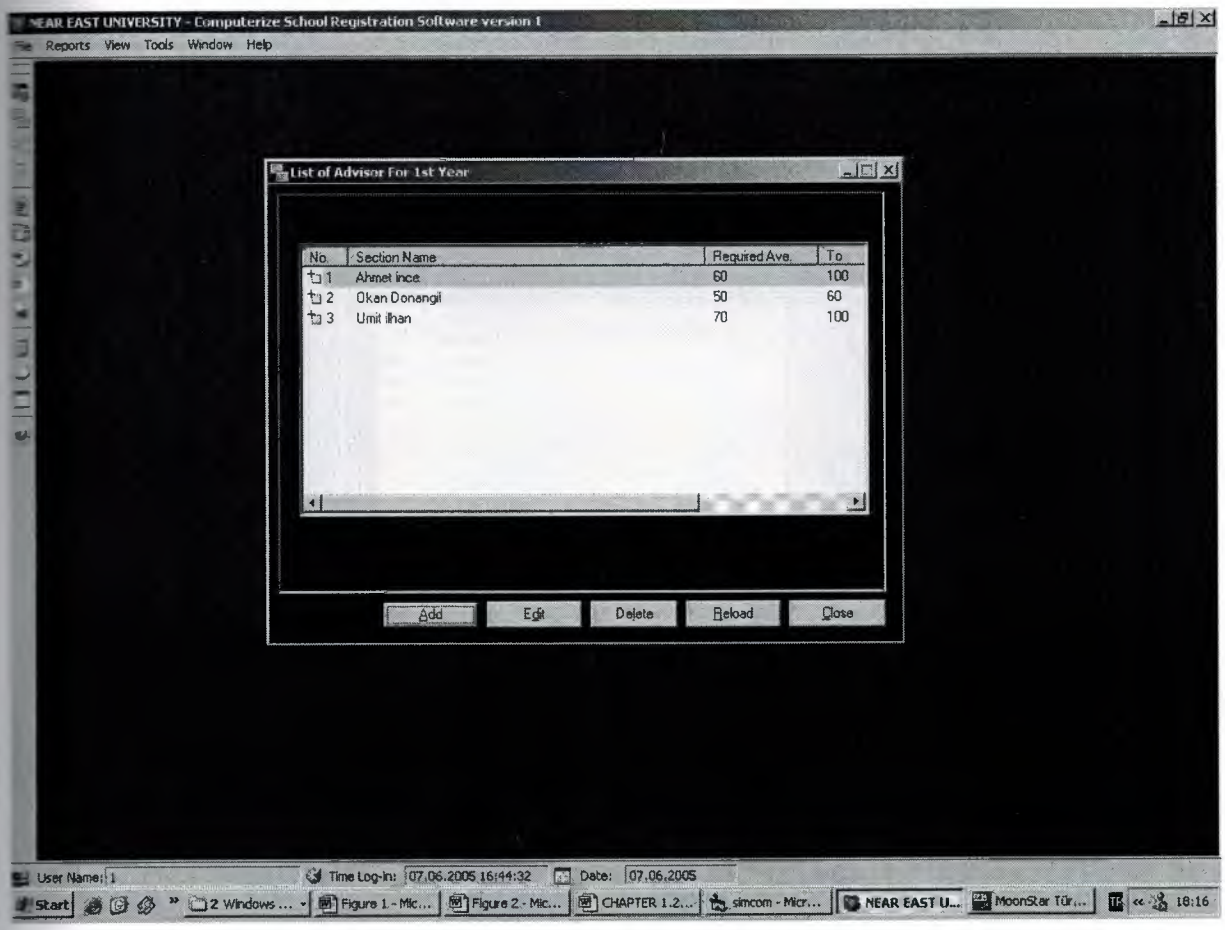

Figure 3.11 List Of Advisor For Semester Year Form

Using List Of Advisor Form users can add new advisor and change exist advisor options.

3.12 List Of Levels

Figure 3.12 List Of Levels Form

Using Thid form users can add new semester ,change exist semester and delete old semester.

3.13 Add New Level

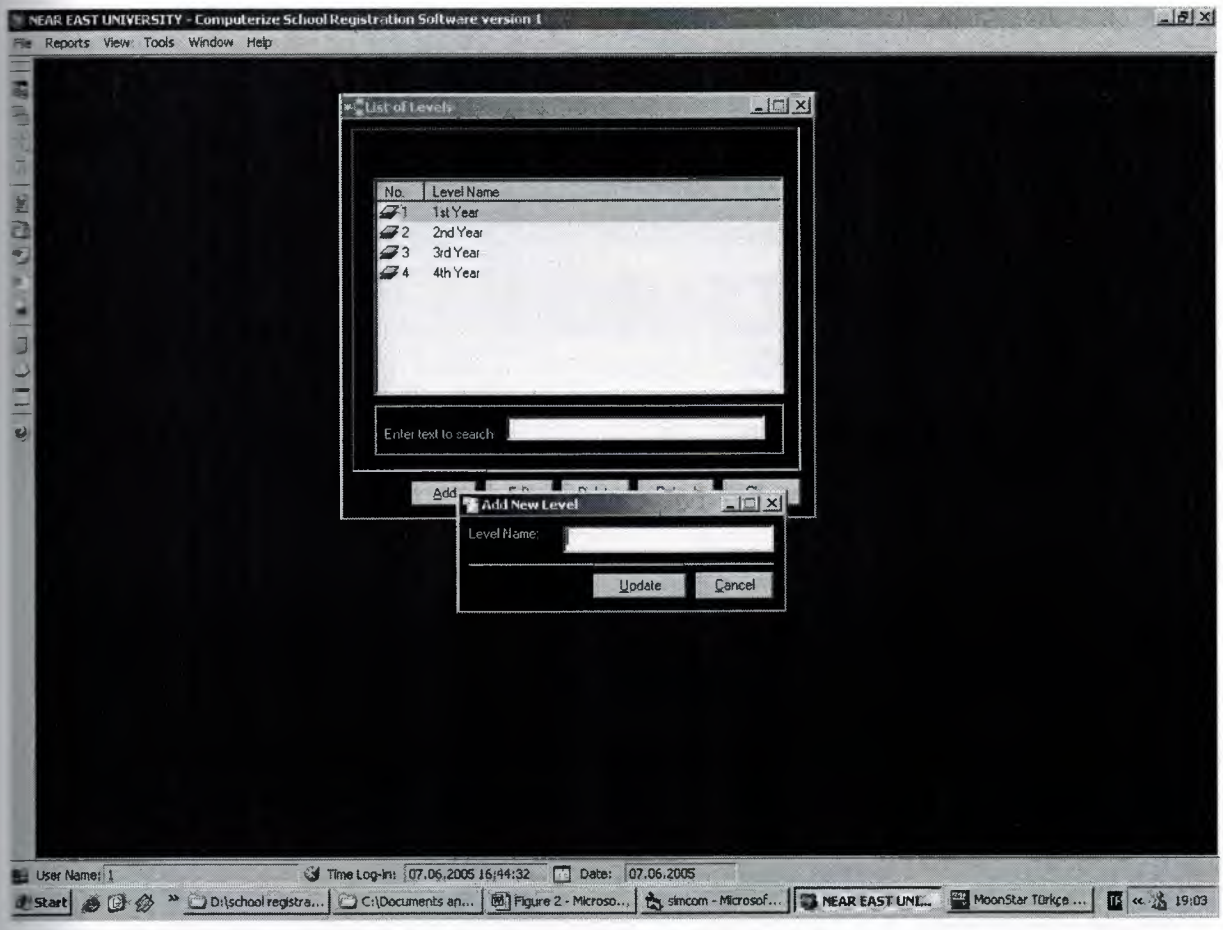

Figure 3.13 Add New Level Form

Add New Level Form makes registers new level.

3.14 List Of School Year

Figure 3.14 List Of School Year Form

This Form Provides many options.Users can add new semester year and change old semester options,delete old register.

3.15 Print Option

Figure 3.15 Print Option For Student Record Form

Print Option For Student Record Form supply search student and printout the student information.

3.16 Regitration Slip Report.

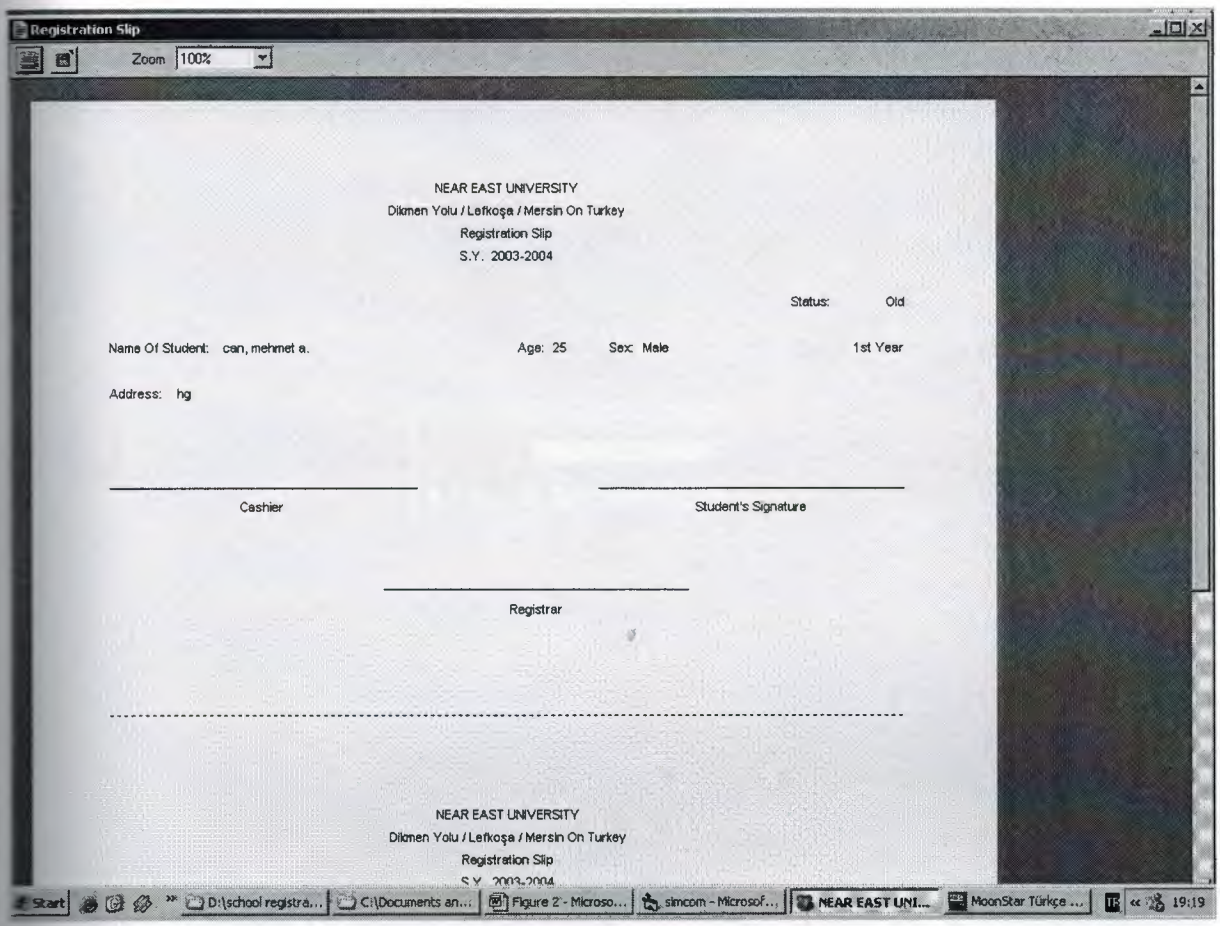

Figure 3.16 Regitration Slip Report.

This Report shows student information and makes printout.

3.17 Print Option For Student List

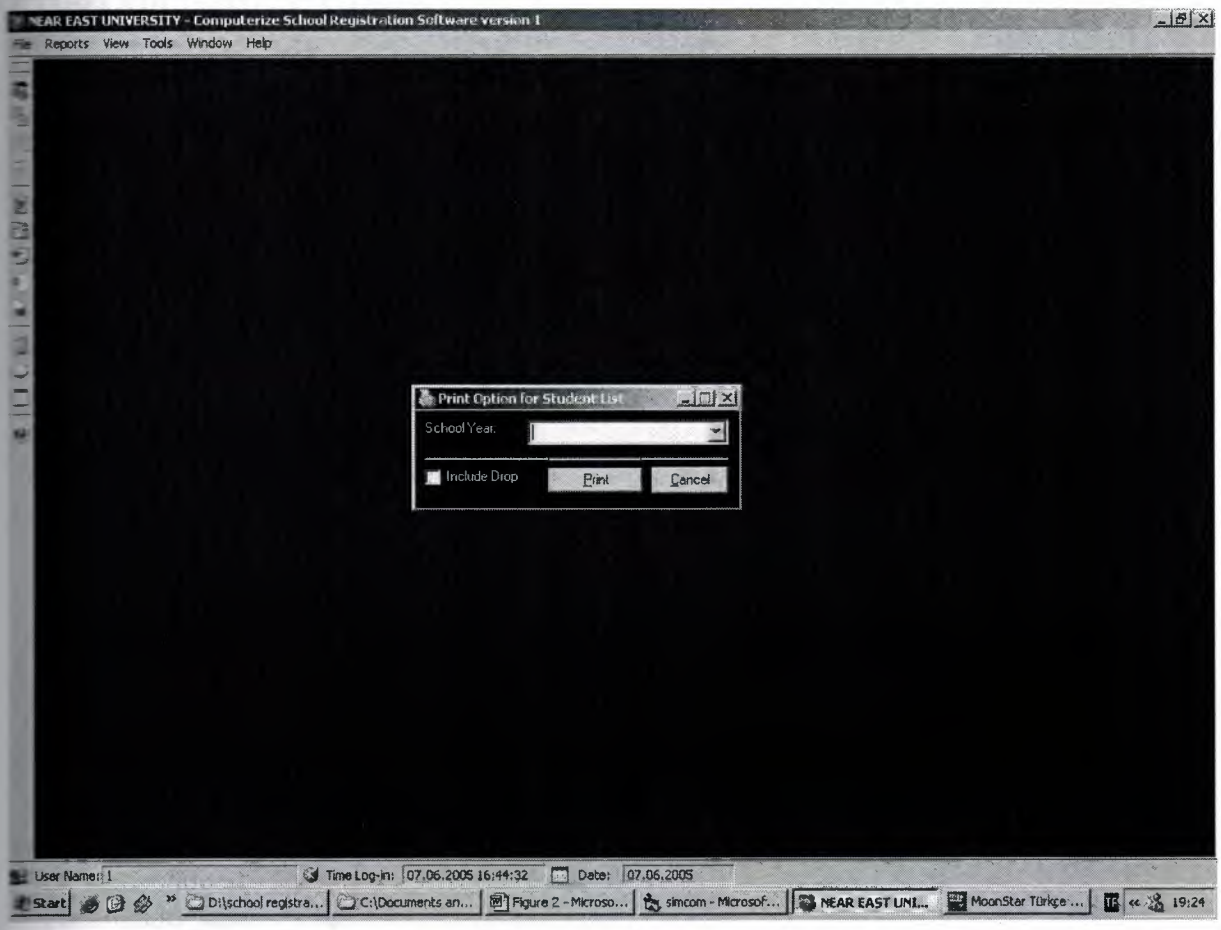

Figure 3.17 Print Option For Student List Form

By using this dialog form users can print easly all student information.

APPENDIX

Program Source Codes

```
Option Explicit

Public SY As String
Public sy_stat As String

Private Sub Combo1_Click()
If Combo1.Text = "Ascending Order" Then
   ListView1.Sorted = True
   ListView1.SortOrder = lvwAscending
ElseIf Combo1.Text = "Descending Order" Then
   ListView1.Sorted = True
   ListView1.SortOrder = lvwDescending
Else
   Combo1.SetFocus
End If
End Sub

Private Sub Combo1_KeyPress(KeyAscii As Integer)
KeyAscii = 0
End Sub

Private Sub Command1_Click()
If sy_stat = "Open" Then
   Form6.add_state = True
   Form6.Show
   Me.Enabled = False
Else
   MsgBox "You cannot record new student because the School Year " & SY & " was already closed." & vbCrLf & vbCrLf & "Note: You can re-open the School Year if you want by selecting 'Option' in the menu and select 'Re-open School Year'.", vbExclamation, "CSRS version 1"
End If End Sub
Private Sub Command10_Click()
If rs_stud.RecordCount < 1 Then MsgBox "No student in the list.Please check it!", vbExclamation, "CSRS version 1": Exit Sub
If rs_stud.Fields(12) <> "Drop" Then
   MsgBox "Please select a student that is currently dropped in the school.", vbExclamation, "CSRS version 1"
Else
   Dim rep As Integer
   rep = MsgBox("Are you sure you want to undrop the selected student?", vbQuestion +
```

```
vbYesNo, "CSRS version 1")
   If rep = vbNo Then Exit Sub:
   rep = 0
   Dim pos As Long
   With rs_stud
      pos = .AbsolutePosition
      .Fields(12) = "Old"
      .Update
       Requery
      Call fill_rec
      .AbsolutePosition = pos

      ListView1.ListItems.Item(Val(.AbsolutePosition)).EnsureVisible
      ListView1.ListItems.Item(Val(.AbsolutePosition)).Selected = True
  MsgBox "The student was sucessfully undropped in the school and change it's status to
'Old'.", vbInformation, "CSRS version 1"
      pos = 0
   End With
End If
End Sub
Private Sub Command2_Click()
If rs_stud.RecordCount < 1 Then MsgBox "No student in the list.Please check it!",
vbExclamation, "CSRS version 1": Exit Sub
Form6.add_state = False
Form6.Show
Me.Enabled = False
End Sub
Private Sub Command3_Click()
If rs_stud.RecordCount < 1 Then MsgBox "No student in the list.Please check it!",
vbExclamation, "CSRS version 1": Exit Sub
If Not rs_stud.Fields(10) = "" Then
   'Show Section
   Form3.sAssign = False
   Form3.Show: Me.Enabled = False
Else
   'Assign Section
   Form3.sAssign = True
   Form3.Show: Me.Enabled = False
End If
End Sub

Private Sub Command4_Click()
On Error GoTo Err:
With rs_stud
   'Check if there is no record
   If .RecordCount < 1 Then MsgBox "No student in the list.Please check it!", vbExclamation,
"CSRS version 1": Exit Sub
   'Confirm deletion of record
   Dim ans As Integer
   Dim pos As Integer
```

```
  ans = MsgBox("Are you sure you want to delete the selected record?", vbCritical +
vbYesNo, "Confirm Record Delete")
  Me.MousePointer = vbHourglass
  If ans = vbYes Then
    'Delete the record
    pos = Val(ListView1.SelectedItem)
    Call delete_rec(cn, "tblStudents", "StudentNo", "", True,
Val(ListView1.SelectedItem.ListSubItems(1)))
    .Requery
    If .RecordCount > 0 Then
      .AbsolutePosition = pos
      If .EOF Then .MoveFirst
      'Fill listview
      pos = .AbsolutePosition
      Screen.MousePointer = vbHourglass

      Call FillListView(ListView1, rs_stud, 6, 1, True, True)
      Label20.Caption = "of " & ListView1.ListItems.Count
      ListView1.ListItems.Item(pos).EnsureVisible

      ListView1.ListItems.Item(pos).Selected = True
      .AbsolutePosition = ListView1.SelectedItem

      Screen.MousePointer = vbDefault
      'End-fill listview

    Else
      ListView1.ListItems.Clear
      Label20.Caption = "of 0"
    End If
    MsgBox "Record has been successfully deleted.", vbInformation, "Confirm"
  End If
  ans = 0
  pos = 0
  Me.MousePointer = vbDefault
End With
Exit Sub
Err:
  prompt_err (Err.Description & vbCrLf & vbCrLf & "Error Number: " & Err.Number):
Me.MousePointer = vbDefault: Exit Sub
End Sub

Private Sub Command5_Click()
Call reload_rec
End Sub
Public Sub reload_rec()

Screen.MousePointer = vbHourglass

rs_stud.Filter = adFilterNone
```

```
rs_stud.Requery
rs_stud.Filter = "Status <>'Drop'"
Call FillListView(ListView1, rs_stud, 6, 1, True, True)
If Not rs_stud.RecordCount < 1 Then rs_stud.MoveFirst
Label20.Caption = "of " & ListView1.ListItems.Count

Screen.MousePointer = vbDefault
End Sub
Public Sub fill_rec()

Screen.MousePointer = vbHourglass

Call FillListView(ListView1, rs_stud, 6, 1, True, True)
If Not rs_stud.RecordCount < 1 Then rs_stud.MoveFirst
Label20.Caption = "of " & ListView1.ListItems.Count


Screen.MousePointer = vbDefault
End Sub

Private Sub Command6_Click()
Unload Me
End Sub

Private Sub Command7_Click()
If Text1.Text = "" Then MsgBox "Please enter some text to search.", vbExclamation, "CSRS
version 1": Text1.SetFocus: Exit Sub
Dim c As Byte
With Form7
   For c = 1 To rs_stud.Fields.Count - 1
      .Combo1.AddItem rs_stud.Fields.Item(Val(c)).Name
   Next c
   .Text1.Text = Text1.Text
   .Show
End With
Me.Enabled = False
'----------------
'Clear Variable
'----------------
c = 0
End Sub

Private Sub Command8_Click()
Form8.Show
Me.Enabled = False
End Sub

Private Sub Command9_Click()
If rs_stud.RecordCount < 1 Then MsgBox "No student in the list.Please check it!",
vbExclamation, "CSRS version 1": Exit Sub
```

```
If rs_stud.Fields(12) = "Drop" Then
   MsgBox "The student was already dropped.", vbExclamation, "CSRS version 1"
Else
   Dim rep As Integer
   rep = MsgBox("Are you sure you want to drop the selected student?", vbQuestion + vbYesNo, "CSRS version 1")
   If rep = vbNo Then Exit Sub:
   rep = 0

   Dim pos As Long
   With rs_stud
      pos = .AbsolutePosition
      .Fields(12) = "Drop"
      .Update

      .Requery
      Call fill_rec
      If pos > .RecordCount Then
         If Not .RecordCount < 1 Then .MoveFirst
      Else
         .AbsolutePosition = pos
      End If

      If Not .RecordCount < 1 Then ListView1.ListItems.Item(Val(.AbsolutePosition)).EnsureVisible
      If Not .RecordCount < 1 Then ListView1.ListItems.Item(Val(.AbsolutePosition)).Selected = True

      MsgBox "The student was sucessfully dropped in the school.", vbInformation, "CSRS version 1"
      pos = 0
   End With
End If
End Sub

Private Sub Form_Activate()
Command1.SetFocus
End Sub

Private Sub Form_Load()
Call use_pos(Me)

'----------------------------
'For student view option
'----------------------------
sds = 0
sms = 1
sfs = 1
sns = 1
sos = 1
```

```
'---------------------------
'End-For student view option
'---------------------------
Call set_rec_getData(rs_stud, cn, "Select qryStudents.* From qryStudents Where SchoolYear
='" & SY & "' Order by Sex Desc,LastName Asc,FirstName Asc")

reload_rec
bind_controls

Me.Caption = Me.Caption & " For School Year " & SY
End Sub

Private Sub Form_Unload(Cancel As Integer)
unbind_controls

Set rs_stud = Nothing

frm_stud_show = False
SY = ""
sy_stat = ""
Call save_pos(Me)
End Sub

Private Sub ListView1_ItemClick(ByVal Item As MSComctlLib.ListItem)
If Not rs_stud.RecordCount < 1 Then rs_stud.AbsolutePosition = ListView1.SelectedItem
End Sub


Private Sub Text1_Change()
If ListView1.ListItems.Count < 1 Then Exit Sub
Call search_in_listview(ListView1, Text1.Text)
End Sub

Private Sub Text1_GotFocus()
Call highlight_focus(Text1)
End Sub

Private Sub Text18_Change()
If Val(Text18.Text) > ListView1.ListItems.Count Or Val(Text18.Text) < 1 Then
Text18.SetFocus: Exit Sub
ListView1.ListItems.Item(Val(Text18.Text)).Selected = True
rs_stud.AbsolutePosition = ListView1.SelectedItem
End Sub
Sub bind_controls()
'---------------------------
'Set the datasource
'---------------------------
Set Text2.DataSource = rs_stud
Set Text3.DataSource = rs_stud
Set Text4.DataSource = rs_stud
```

```
Set Text5.DataSource = rs_stud
Set Text6.DataSource = rs_stud
Set Text7.DataSource = rs_stud
Set Text8.DataSource = rs_stud
Set Text9.DataSource = rs_stud
Set Text10.DataSource = rs_stud
Set Text11.DataSource = rs_stud
Set Text12.DataSource = rs_stud
Set Text13.DataSource = rs_stud
Set Text14.DataSource = rs_stud
Set Text15.DataSource = rs_stud
Set Text16.DataSource = rs_stud
Set Text17.DataSource = rs_stud
'------------------------------
'Set the datafield
'------------------------------
Text2.DataField = "FirstName"
Text3.DataField = "MiddleName"
Text4.DataField = "LastName"
Text5.DataField = "Sex"
Text6.DataField = "DateOfBirth"
Text7.DataField = "Age"
Text8.DataField = "PlaceOfBirth"
Text9.DataField = "Address"
Text10.DataField = "FatherName"
Text11.DataField = "Occupation1"
Text12.DataField = "MotherName"
Text13.DataField = "Occupation2"
Text14.DataField = "ParentAddress"
Text15.DataField = "SchoolLastAttend"
Text16.DataField = "Status"
Text17.DataField = "DateEnrolled"
End Sub
Sub unbind_controls()
'----------------------------------
'Set the datasource
'----------------------------------
Set Text2.DataSource = Nothing
Set Text3.DataSource = Nothing
Set Text4.DataSource = Nothing
Set Text5.DataSource = Nothing
Set Text6.DataSource = Nothing
Set Text7.DataSource = Nothing
Set Text8.DataSource = Nothing
Set Text9.DataSource = Nothing
Set Text10.DataSource = Nothing
Set Text11.DataSource = Nothing
Set Text12.DataSource = Nothing
Set Text13.DataSource = Nothing
Set Text14.DataSource = Nothing
```

```
Set Text15.DataSource = Nothing
Set Text16.DataSource = Nothing
Set Text17.DataSource = Nothing
'-----------------------------
'Set the datafield
'-----------------------------
Text2.DataField = ""
Text3.DataField = ""
Text4.DataField = ""
Text5.DataField = ""
Text6.DataField = ""
Text7.DataField = ""
Text8.DataField = ""
Text9.DataField = ""
Text10.DataField = ""
Text11.DataField = ""
Text12.DataField = ""
Text13.DataField = ""
Text14.DataField = ""
Text15.DataField = ""
Text16.DataField = ""
Text17.DataField = ""
End Sub

Private Sub Text18_GotFocus()
Call highlight_focus(Text18)
End Sub

Private Sub Text18_KeyPress(KeyAscii As Integer)
If Not ((KeyAscii >= 48 And KeyAscii <= 57) Or KeyAscii = 8) Then KeyAscii = 0
End Sub

Private Sub Text18_LostFocus()
Text18.Text = Val(Text18.Text)
End Sub

Private Sub Text2_GotFocus()
Call highlight_focus(Text2)
End Sub
Private Sub Text3_GotFocus()
Call highlight_focus(Text3)
End Sub
Private Sub Text4_GotFocus()
Call highlight_focus(Text4)
End Sub
Private Sub Text5_GotFocus()
Call highlight_focus(Text5)
End Sub
Private Sub Text6_GotFocus()
Call highlight_focus(Text6)
```

```
End Sub
Private Sub Text7_GotFocus()
Call highlight_focus(Text7)
End Sub
Private Sub Text8_GotFocus()
Call highlight_focus(Text8)
End Sub
Private Sub Text9_GotFocus()
Call highlight_focus(Text9)
End Sub
Private Sub Text10_GotFocus()
Call highlight_focus(Text10)
End Sub
Private Sub Text11_GotFocus()
Call highlight_focus(Text11)
End Sub
Private Sub Text12_GotFocus()
Call highlight_focus(Text12)
End Sub
Private Sub Text13_GotFocus()
Call highlight_focus(Text13)
End Sub
Private Sub Text14_GotFocus()
Call highlight_focus(Text14)
End Sub
Private Sub Text15_GotFocus()
Call highlight_focus(Text15)
End Sub
Private Sub Text16_GotFocus()
Call highlight_focus(Text16)
End Sub
Private Sub Text17_GotFocus()
Call highlight_focus(Text17)
End Sub


Public add_state As Boolean


Private Sub Command1_Click()
If is_empty(Text1) = True Then Exit Sub

With rs_level
   If add_state = True Then .AddNew: .Fields(0) = get_num("tblLevel", "LevelNo", cn)
      .Fields(1) = Text1.Text
   .Update
End With
'------------------------------
'Inform updates
'------------------------------
```

```
If add_state = True Then
   MsgBox "Adding of new level has been successfull.", vbInformation, "CSRS version 1"
   Dim rep As Integer
   rep = MsgBox("Do you want to add another level?", vbQuestion + vbYesNo, "CSRS
version 1")
   If rep = vbYes Then
      Text1.Text = ""
      Text1.SetFocus
      rs_level.Requery
      Form9.load_rec
   Else
      rs_level.Requery
      Form9.load_rec
      Unload Me
   End If
   rep = 0
Else
   MsgBox "Changes in record has been successfully saved.", vbInformation, "CSRS version
1"
   Dim pos As Long

   pos = rs_level.AbsolutePosition
   rs_level.Requery
   Form9.load_rec
   rs_level.AbsolutePosition = pos

   Form9.ListView1.ListItems.Item(pos).EnsureVisible
   Form9.ListView1.ListItems.Item(pos).Selected = True

   pos = 0
   Unload Me
End If
'------------------------------
'End-Inform updates
'------------------------------
End Sub

Private Sub Command2_Click()
Unload Me
End Sub

Private Sub Form_Load()
Call use_pos(Me)

If add_state = False Then
   Text1.Text = rs_level.Fields(1)
   Me.Icon = ImageList1.ListImages(1).Picture
   Me.Caption = "Edit Existing Level"
End If
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
Form9.Enabled = True

Call save_pos(Me)
End Sub
Private Sub Text1_GotFocus()
Call highlight_focus(Text1)
End Sub

Option Explicit


Private Sub Command1_Click()
Form12.add_state = True
Form12.Show
Me.Enabled = False
End Sub

Private Sub Command10_Click()
If rs_sy.RecordCount < 1 Then MsgBox "No school year in the list.Please check it!",
vbExclamation, "CSRS version 1": Exit Sub
If rs_sy.Fields(1) = "Open" Then
   MsgBox "The school year is not closed.Please select a closed school year to re-open.",
vbExclamation, "CSRS version 1"
Else
   Dim rep As Integer
   rep = MsgBox("Are you sure you want to re-open the selected school year?", vbQuestion +
vbYesNo, "CSRS version 1")
   If rep = vbNo Then Exit Sub:
   rep = 0
   Dim pos As Long
   With rs_sy
      pos = .AbsolutePosition
      .Fields(1) = "Open"
      .Update

      .Requery
      Call load_rec
      .AbsolutePosition = pos

      ListView1.ListItems.Item(Val(.AbsolutePosition)).EnsureVisible
      ListView1.ListItems.Item(Val(.AbsolutePosition)).Selected = True


      MsgBox "The school year was sucessfully re-opened.", vbInformation, "CSRS version
1"
      pos = 0
   End With
End If
```

```
End Sub

Private Sub Command2_Click()
If rs_sy.RecordCount < 1 Then MsgBox "No school year in the list.Please check it!", vbExclamation, "CSRS version 1": Exit Sub
If Not ListView1.SelectedItem = "" And Not rs_sy.RecordCount < 1 Then rs_sy.AbsolutePosition = ListView1.SelectedItem
Form12.add_state = False
Form12.Show
Me.Enabled = False
End Sub

Private Sub Command4_Click()
On Error GoTo Err:
With rs_sy
  '---------------------------------
  'Check if there is no record
  '---------------------------------
  If .RecordCount < 1 Then MsgBox "No school year in the list.Please check it!", vbExclamation, "CSRS version 1": Exit Sub
  '---------------------------------
  'Confirm deletion of record
  '---------------------------------
  Dim ans As Integer
  Dim pos As Integer
  ans = MsgBox("Are you sure you want to delete the selected record?", vbCritical + vbYesNo, "Confirm Record Delete")
  Me.MousePointer = vbHourglass
  If ans = vbYes Then
    '-----------------------------
    'Delete the record
    '-----------------------------
    pos = Val(ListView1.SelectedItem)
    Call delete_rec(cn, "tblSchoolYear", "SchoolYear", ListView1.SelectedItem.ListSubItems(1), False, 0)
    .Requery
    If .RecordCount > 0 Then
      .AbsolutePosition = pos
      If .EOF Then .MoveFirst
      '---------------------------------
      'Fill listview
      '---------------------------------
      pos = .AbsolutePosition
      load_rec
      ListView1.ListItems.Item(pos).EnsureVisible
      ListView1.ListItems.Item(pos).Selected = True
      .AbsolutePosition = ListView1.SelectedItem
      '---------------------------------
      'End-fill listview
      '---------------------------------
```

```
      Else
         ListView1.ListItems.Clear
      End If
      MsgBox "Record has been successfully deleted.", vbInformation, "Confirm"
   End If
   ans = 0
   pos = 0
   Me.MousePointer = vbDefault
End With
Exit Sub
Err:
   prompt_err (Err.Description & vbCrLf & vbCrLf & "Error Number: " & Err.Number):
Me.MousePointer = vbDefault: Exit Sub
End Sub

Private Sub Command5_Click()
rs_sy.Requery
load_rec
End Sub

Private Sub Command6_Click()
Unload Me
End Sub

Private Sub Command9_Click()
If rs_sy.RecordCount < 1 Then MsgBox "No school year in the list.Please check it!",
vbExclamation, "CSRS version 1": Exit Sub
If rs_sy.Fields(1) = "Close" Then
   MsgBox "The school year was already closed.", vbExclamation, "CSRS version 1"
Else
   Dim rep As Integer
   rep = MsgBox("Are you sure you want to close the selected school year?", vbQuestion +
vbYesNo, "CSRS version 1")
   If rep = vbNo Then Exit Sub:
   rep = 0
   Dim pos As Long
   With rs_sy
      pos = .AbsolutePosition
      .Fields(1) = "Close"
      .Update

      .Requery
      Call load_rec
      .AbsolutePosition = pos

      ListView1.ListItems.Item(Val(.AbsolutePosition)).EnsureVisible
      ListView1.ListItems.Item(Val(.AbsolutePosition)).Selected = True


      MsgBox "The school year was sucessfully closed.", vbInformation, "CSRS version 1"
```

```
    pos = 0
  End With
End If

End Sub

Private Sub Form_Activate()

If Not rs_sy.RecordCount < 1 Then rs_sy.AbsolutePosition = ListView1.SelectedItem
Command1.SetFocus
End Sub

Private Sub Form_Load()
Call use_pos(Me)

Call set_rec_getData(rs_sy, cn, "Select tblSchoolYear.* From tblSchoolYear Order by
SchoolYear Asc")
load_rec
End Sub

Private Sub Form_Unload(Cancel As Integer)
Set rs_sy = Nothing

Call save_pos(Me)
End Sub
Sub load_rec()
Screen.MousePointer = vbHourglass

Call FillListView(ListView1, rs_sy, 3, 1, True, True)

Screen.MousePointer = vbDefault
End Sub
Private Sub ListView1_Click()

If Not rs_sy.RecordCount < 1 Then rs_sy.AbsolutePosition = ListView1.SelectedItem
End Sub


Private Sub Text1_GotFocus()

End Sub


Option Explicit

Public add_state As Boolean
Dim old_sy As String

Private Sub Command1_Click()
If is_empty(Text1) = True Then Exit Sub
```

```
If Len(Text1.Text) < 9 Or Mid(Text1.Text, 5, 1) <> "-" Then MsgBox "Enry must be in this
format (ex. yyyy-yyyy).", vbExclamation, "CSRS version 1": Text1.SetFocus: Exit Sub

If old_sy <> Text1.Text Then
  If if_exist("tblSchoolYear", "SchoolYear", Text1) = True Then Exit Sub
End If

With rs_sy
   If add_state = True Then .AddNew
      .Fields(0) = Text1.Text
   .Update
End With

'------------------------------
'Inform updates
'------------------------------
If add_state = True Then
   MsgBox "Adding of new school year has been successfull.", vbInformation, "CSRS version
1"
   Dim rep As Integer
   rep = MsgBox("Do you want to add another school year?", vbQuestion + vbYesNo, "CSRS
version 1")
   If rep = vbYes Then
      Text1.Text = ""
      Text1.SetFocus
      rs_sy.Requery
      Form11.load_rec
   Else
      rs_sy.Requery
      Form11.load_rec
      Unload Me
   End If
   rep = 0
Else
   MsgBox "Changes in record has been successfully saved.", vbInformation, "CSRS version
1"
   Dim pos As Long

   pos = rs_sy.AbsolutePosition
   rs_sy.Requery
   Form11.load_rec
   rs_sy.AbsolutePosition = pos

   Form11.ListView1.ListItems.Item(pos).EnsureVisible
   Form11.ListView1.ListItems.Item(pos).Selected = True

   pos = 0
   Unload Me
End If
'------------------------------
```

```
'End-Inform updates
'----------------------------
End Sub

Private Sub Command2_Click()
Unload Me
End Sub

Private Sub Form_Load()
Call use_pos(Me)

If add_state = False Then
   Text1.Text = rs_sy.Fields(0)
   old_sy = rs_sy.Fields(0)
   Me.Icon = ImageList1.ListImages(1).Picture
   Me.Caption = "Edit Existing School Year"
End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
old_sy = ""
Form11.Enabled = True

Call save_pos(Me)
End Sub
Private Sub Text1_GotFocus()
Call highlight_focus(Text1)
End Sub

Private Sub Text1_KeyPress(KeyAscii As Integer)
If Not ((KeyAscii >= 48 And KeyAscii <= 57) Or KeyAscii = 8 Or KeyAscii = 45) Then
KeyAscii = 0
End Sub

Option Explicit

Dim rs_sel_lv_for_sec As New ADODB.Recordset

Private Sub Command1_Click()
If ListView1.ListItems.Count < 1 Then Unload Me: Exit Sub
With Form14
   .lv_no = ListView1.SelectedItem.ListSubItems(1)
   .lv_name = ListView1.SelectedItem.ListSubItems(2)
   .Show
End With
Unload Me
End Sub

Private Sub Form_Load()
Call use_pos(Me)
```

```
Call set_rec_getData(rs_sel_lv_for_sec, cn, "Select tblLevel.* From tblLevel Order by LevelNo Asc")
load_rec
End Sub

Private Sub Form_Unload(Cancel As Integer)
Set rs_sel_lv_for_sec = Nothing

Call save_pos(Me)
End Sub
Sub load_rec()
Screen.MousePointer = vbHourglass

Call FillListView(ListView1, rs_sel_lv_for_sec, 3, 1, True, True)

Screen.MousePointer = vbDefault
If ListView1.ListItems.Count < 1 Then Command1.Caption = "&Close"
End Sub

Option Explicit

Public lv_name As String
Public lv_no As Long


Private Sub Command1_Click()
Form15.add_state = True
Form15.Show
Me.Enabled = False
End Sub

Private Sub Command2_Click()
If rs_sec.RecordCount < 1 Then MsgBox "No section in the list.Please check it!", vbExclamation, "CSRS version 1": Exit Sub
If Not ListView1.SelectedItem = "" And Not rs_sec.RecordCount < 1 Then rs_sec.AbsolutePosition = ListView1.SelectedItem
Form15.add_state = False
Form15.Show
Me.Enabled = False
End Sub

Private Sub Command4_Click()
On Error GoTo Err:
With rs_sec
   '---------------------------------
   'Check if there is no record
   '---------------------------------
   If .RecordCount < 1 Then MsgBox "No section in the list.Please check it!", vbExclamation, "CSRS version 1": Exit Sub
```

```
    '--------------------------------
    'Confirm deletion of record
    '--------------------------------
    Dim ans As Integer
    Dim pos As Integer
    ans = MsgBox("Are you sure you want to delete the selected record?", vbCritical +
vbYesNo, "Confirm Record Delete")
    Me.MousePointer = vbHourglass
    If ans = vbYes Then
      '-----------------------------
      'Delete the record
      '-----------------------------
      pos = Val(ListView1.SelectedItem)
      Call delete_rec(cn, "tblSections", "SectionNo", "", True,
Val(ListView1.SelectedItem.ListSubItems(1)))
      .Requery
      If .RecordCount > 0 Then
        .AbsolutePosition = pos
        If .EOF Then .MoveFirst
        '--------------------------------
        'Fill listview
        '--------------------------------
        pos = .AbsolutePosition
        load_rec
        ListView1.ListItems.Item(pos).EnsureVisible
        ListView1.ListItems.Item(pos).Selected = True
        .AbsolutePosition = ListView1.SelectedItem
        '--------------------------------
        'End-fill listview
        '--------------------------------
      Else
        ListView1.ListItems.Clear
      End If
      MsgBox "Record has been successfully deleted.", vbInformation, "Confirm"
    End If
    ans = 0
    pos = 0
    Me.MousePointer = vbDefault
End With
Exit Sub
Err:
    prompt_err (Err.Description & vbCrLf & vbCrLf & "Error Number: " & Err.Number):
Me.MousePointer = vbDefault: Exit Sub
End Sub

Private Sub Command5_Click()
rs_sec.Requery
load_rec
End Sub
```

```
Private Sub Command6_Click()
Unload Me
End Sub

Private Sub Form_Activate()

If Not rs_sec.RecordCount < 1 Then rs_sec.AbsolutePosition = ListView1.SelectedItem
Command1.SetFocus
End Sub

Private Sub Form_Load()
Call use_pos(Me)

Me.Caption = Me.Caption & lv_name
Call set_rec_getData(rs_sec, cn, "Select tblSections.* From tblSections Where LevelNo =" &
lv_no & " Order by SectionName Asc")
load_rec
End Sub

Private Sub Form_Unload(Cancel As Integer)
Set rs_sec = Nothing

Call save_pos(Me)
End Sub
Sub load_rec()
Screen.MousePointer = vbHourglass

Call FillListView(ListView1, rs_sec, 7, 1, True, True)

Screen.MousePointer = vbDefault
End Sub
Private Sub ListView1_Click()

If Not rs_sec.RecordCount < 1 Then rs_sec.AbsolutePosition = ListView1.SelectedItem
End Sub


Option Explicit

Public add_state As Boolean

Private Sub Command1_Click()
If is_empty(Text1) = True Then Exit Sub
If Val(Text2.Text) = 0 Then MsgBox "The field must not be zero value.", vbExclamation,
"CSRS version 1": Text2.SetFocus: Exit Sub
If Val(Text3.Text) = 0 Then MsgBox "The field must not be zero value.", vbExclamation,
"CSRS version 1": Text3.SetFocus: Exit Sub
If Val(Text4.Text) = 0 Then MsgBox "The field must not be zero value.", vbExclamation,
"CSRS version 1": Text4.SetFocus: Exit Sub
```

```
If Val(Text2.Text) > Val(Text3.Text) Then MsgBox "The minimum average must not be greater than to " & Val(Text3.Text) & ".", vbExclamation, "CSRS version 1": Text3.SetFocus: Exit Sub

With rs_sec
  If add_state = True Then .AddNew: .Fields(0) = get_num("tblSections", "SectionNo", cn): .Fields(1) = Form14.lv_no
     .Fields(2) = Text1.Text
     .Fields(3) = Val(Text2.Text)
     .Fields(4) = Val(Text3.Text)
     .Fields(5) = Val(Text4.Text)
  .Update
End With
'----------------------------
'Inform updates
'----------------------------
If add_state = True Then
  MsgBox "Adding of new section has been successfull.", vbInformation, "CSRS version 1"
  Dim rep As Integer
  rep = MsgBox("Do you want to add another section?", vbQuestion + vbYesNo, "CSRS version 1")
  If rep = vbYes Then
     Text1.Text = ""
     Text2.Text = "0"
     Text3.Text = "0"
     Text4.Text = "0"
     Text1.SetFocus
     rs_sec.Requery
     Form14.load_rec
  Else
     rs_sec.Requery
     Form14.load_rec
     Unload Me
  End If
  rep = 0
Else
  MsgBox "Changes in record has been successfully saved.", vbInformation, "CSRS version 1"
  Dim pos As Long

  pos = rs_sec.AbsolutePosition
  rs_sec.Requery
  Form14.load_rec
  rs_sec.AbsolutePosition = pos

  Form14.ListView1.ListItems.Item(pos).EnsureVisible
  Form14.ListView1.ListItems.Item(pos).Selected = True

  pos = 0
  Unload Me
```

```
End If
'----------------------------
'End-Inform updates
'----------------------------
End Sub

Private Sub Command2_Click()
Unload Me
End Sub

Private Sub Form_Load()
Call use_pos(Me)

If add_state = False Then
   Text1.Text = rs_sec.Fields(2)
   Text2.Text = rs_sec.Fields(3)
   Text3.Text = rs_sec.Fields(4)
   Text4.Text = rs_sec.Fields(5)
   Me.Icon = ImageList1.ListImages(1).Picture
   Me.Caption = "Edit Existing Section"
End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
Form14.Enabled = True

Call save_pos(Me)
End Sub
Private Sub Text1_GotFocus()
Call highlight_focus(Text1)
End Sub
Private Sub Text2_GotFocus()
Call highlight_focus(Text2)
End Sub

Private Sub Text2_KeyPress(KeyAscii As Integer)
If Not ((KeyAscii >= 48 And KeyAscii <= 57) Or KeyAscii = 8) Then KeyAscii = 0
End Sub
Private Sub Text3_GotFocus()
Call highlight_focus(Text3)
End Sub

Private Sub Text3_KeyPress(KeyAscii As Integer)
If Not ((KeyAscii >= 48 And KeyAscii <= 57) Or KeyAscii = 8) Then KeyAscii = 0
End Sub

Private Sub Text4_GotFocus()
Call highlight_focus(Text4)
End Sub
```

```
Private Sub Text4_KeyPress(KeyAscii As Integer)
If Not ((KeyAscii >= 48 And KeyAscii <= 57) Or KeyAscii = 8) Then KeyAscii = 0
End Sub

Private Sub Form_Load()
Call use_pos(Me)

MonthView1.Value = Date
End Sub

Private Sub Form_Unload(Cancel As Integer)
Call save_pos(Me)
End Sub

Option Explicit

Public prnt_sec As Long

Dim rs_prnt_stud As New ADODB.Recordset

Private Sub SSTab1_DblClick()

End Sub

Private Sub Command1_Click()
If ListView1.ListItems.Count < 1 Then MsgBox "No student record to print.",
vbExclamation, "CSRS version 1": Exit Sub
If Not rs_prnt_stud.RecordCount < 1 Then rs_prnt_stud.AbsolutePosition =
ListView1.SelectedItem


With rpt_header
   .SchoolAddress = school_address
   .SchoolName = school_name
   .SY = "S.Y.  " & rs_prnt_stud.Fields(11)
End With

Dim rpt_rs As New ADODB.Recordset

Call set_rec_getData(rpt_rs, cn, "Select qryStudents.* From qryStudents Where StudentNo ="
& rs_prnt_stud.Fields(0) & " Order by Sex Desc,LastName Asc")

Set DataReport1.DataSource = rpt_rs
DataReport1.Show vbModal

Set rpt_rs = Nothing
End Sub

Private Sub Command2_Click()
If ListView1.ListItems.Count < 1 Then MsgBox "No student record to print.",
```

```
vbExclamation, "CSRS version 1": Exit Sub
If rs_prnt_stud.RecordCount < 1 Then Exit Sub


With rpt_header
   .SchoolAddress = school_address
   .SchoolName = school_name
   .SY = "S.Y.  " & DataCombo1.Text
End With

Set DataReport2.DataSource = rs_prnt_stud
DataReport2.Show vbModal

End Sub

Private Sub Command3_Click()
Form18.Show: Me.Enabled = False
End Sub

Private Sub Command4_Click()
If ListView1.ListItems.Count < 1 Then MsgBox "No student record to print.",
vbExclamation, "CSRS version 1": Exit Sub
If Not ListView1.SelectedItem = "" And Not rs_prnt_stud.RecordCount < 1 Then
rs_prnt_stud.AbsolutePosition = ListView1.SelectedItem

On Error Resume Next

Kill Environ("TMP") & "\SupportDB.mdb"
FileCopy App.Path & "\SupportDB.db", Environ("TMP") & "\SupportDB.mdb"

Dim cn_tmp As New ADODB.Connection
Dim rs_tmp As New ADODB.Recordset
Dim pos, c As Long

Call set_conn_getData(cn_tmp, Environ("TMP") & "\SupportDB.mdb", True, "reg386")
Call set_rec_getData(rs_tmp, cn_tmp, "Select tblStudent.* From tblStudent")

With rs_prnt_stud
   pos = .AbsolutePosition

   .MoveFirst
   For c = 1 To .RecordCount
      rs_tmp.AddNew
         rs_tmp.Fields(0) = c
         rs_tmp.Fields(1) = .Fields(23)
      rs_tmp.Update

      .MoveNext
   Next c
```

```
    .AbsolutePosition = pos
End With

rs_tmp.Requery

With rpt_header
    .SchoolAddress = school_address
    .SchoolName = school_name
    .SY = "S.Y.  " & rs_prnt_stud.Fields(11)
    .SectionName = rs_prnt_stud.Fields("LevelName") & " - " & Text2.Text
End With

Set DataReport3.DataSource = rs_tmp
DataReport3.Show vbModal

Set rs_tmp = Nothing
Set cn_tmp = Nothing
End Sub

Private Sub Command5_Click()
If is_empty(DataCombo1) = True Then Exit Sub
If is_empty(Text2) = True Then Exit Sub
If Len(DataCombo1.Text) < 9 Or Mid(DataCombo1.Text, 5, 1) <> "-" Then MsgBox "Enry
must be in this format (ex. yyyy-yyyy).", vbExclamation, "CSRS version 1":
DataCombo1.SetFocus: Exit Sub

rs_prnt_stud.Filter = adFilterNone
rs_prnt_stud.Requery
If Text5.Text = "No" Then
    rs_prnt_stud.Filter = "SchoolYear ='" & DataCombo1.Text & "' And Status <>'Drop' And
SN =" & prnt_sec
Else
    rs_prnt_stud.Filter = "SchoolYear ='" & DataCombo1.Text & "' And SN =" & prnt_sec
End If
Call fill_rec
End Sub

Private Sub Command7_Click()
Unload Me
End Sub

Private Sub DataCombo1_Change()
Text2.Text = ""
End Sub

Private Sub DataCombo1_KeyPress(KeyAscii As Integer)
KeyAscii = 0
End Sub

Private Sub Form_Load()
```

```
Call use_pos(Me)

Dim rs_sy As New ADODB.Recordset
Call set_rec_getData(rs_sy, cn, "Select tblSchoolYear.* From tblSchoolYear Order by SchoolYear Asc")

Set DataCombo1.RowSource = rs_sy
DataCombo1.ListField = "SchoolYear"

Set rs_sy = Nothing

Call set_rec_getData(rs_prnt_stud, cn, "Select qryStudents.* From qryStudents Order by Sex Desc,LastName Asc,FirstName Asc")
End Sub

Private Sub Form_Unload(Cancel As Integer)
Set rs_prnt_stud = Nothing

Call save_pos(Me)
End Sub

Private Sub ListView1_Click()

If Not rs_prnt_stud.RecordCount < 1 Then rs_prnt_stud.AbsolutePosition = ListView1.SelectedItem
End Sub

Private Sub Text2_GotFocus()
Call highlight_focus(Text2)
End Sub

Private Sub Text2_KeyPress(KeyAscii As Integer)
Command3_Click
End Sub

Private Sub Text5_KeyPress(KeyAscii As Integer)
KeyAscii = 0
End Sub

Private Sub Text5_Validate(Cancel As Boolean)
If Text5.Text <> "No" And Text5.Text <> "Yes" Then MsgBox "Please select a valid entry in the list.", vbExclamation, "CSRS version 1": Cancel = True
End Sub
Public Sub fill_rec()

Screen.MousePointer = vbHourglass

Call FillListView(ListView1, rs_prnt_stud, 6, 1, True, True)
If Not rs_prnt_stud.RecordCount < 1 Then rs_prnt_stud.MoveFirst
```

```
Screen.MousePointer = vbDefault
End Sub

Option Explicit


Public gen_ave As Integer
Public lv_id As Long

Dim rs_sel_sec As New ADODB.Recordset
Private Sub Command1_Click()
If ListView1.ListItems.Count < 1 Then Unload Me: Exit Sub

Form17.prnt_sec = ListView1.SelectedItem.ListSubItems(1)
Form17.Text2 = ListView1.SelectedItem.ListSubItems(2)
Unload Me

End Sub

Private Sub Command3_Click()
Form19.Show: Me.Enabled = False
End Sub

Private Sub Form_Load()
Call use_pos(Me)

Call set_rec_getData(rs_sel_sec, cn, "Select qrySections.* From qrySections Order by
MinAve Desc")

End Sub

Private Sub Form_Unload(Cancel As Integer)
Set rs_sel_sec = Nothing
Form17.Enabled = True

Call save_pos(Me)
End Sub

Private Sub Label2_Click()

End Sub

Private Sub Text2_Change()

rs_sel_sec.Filter = "LevelNo = " & lv_id

load_rec
End Sub

Private Sub Text2_GotFocus()
```

```
Call highlight_focus(Text2)
End Sub

Private Sub Text2_KeyPress(KeyAscii As Integer)
Command3_Click
End Sub
Sub load_rec()
Screen.MousePointer = vbHourglass

rs_sel_sec.Requery
Call FillListView(ListView1, rs_sel_sec, 6, 1, True, True)

Screen.MousePointer = vbDefault
If ListView1.ListItems.Count < 1 Then
   Command1.Caption = "&Close"
Else
   Command1.Caption = "&Select"
End If
End Sub

Option Explicit

Dim rs_sel_lv As New ADODB.Recordset

Private Sub Command1_Click()
If ListView1.ListItems.Count < 1 Then Unload Me: Exit Sub
Form18.lv_id = ListView1.SelectedItem.ListSubItems(1)
Form18.Text2 = ListView1.SelectedItem.ListSubItems(2)
Unload Me
End Sub

Private Sub Form_Load()
Call use_pos(Me)

Call set_rec_getData(rs_sel_lv, cn, "Select tblLevel.* From tblLevel Order by LevelNo Asc")
load_rec
End Sub

Private Sub Form_Unload(Cancel As Integer)
Set rs_sel_lv = Nothing
Form18.Enabled = True

Call save_pos(Me)
End Sub
Sub load_rec()
Screen.MousePointer = vbHourglass

Call FillListView(ListView1, rs_sel_lv, 3, 1, True, True)

Screen.MousePointer = vbDefault
```

```
If ListView1.ListItems.Count < 1 Then Command1.Caption = "&Cancel"
End Sub

Option Explicit

Dim rs_ssy As New ADODB.Recordset

Private Sub Command1_Click()
With Form1
   .SY = ListView1.SelectedItem.ListSubItems(1)
   .sy_stat = ListView1.SelectedItem.ListSubItems(2)
   .Show
   frm_stud_show = True
End With
Unload Me
End Sub

Private Sub Command2_Click()
Unload Me
End Sub

Private Sub DataCombo1_Change()
Me.Caption = rs_ssy.Fields(0)
End Sub
Private Sub Form_Load()
Call use_pos(Me)

Call set_rec_getData(rs_ssy, cn, "Select tblSchoolYear.* From tblSchoolYear Order by
SchoolYear Asc")
If rs_ssy.RecordCount < 1 Then Command1.Visible = False: Exit Sub
'---------------------------------
'Fill the list view
'---------------------------------
Call FillListView(ListView1, rs_ssy, 3, 1, True, True)
End Sub

Private Sub Form_Unload(Cancel As Integer)
Set rs_ssy = Nothing

Call save_pos(Me)
End Sub

Private Sub ListView1_BeforeLabelEdit(Cancel As Integer)

End Sub

Option Explicit

Dim rs_prnt_list As New ADODB.Recordset
```

```
Private Sub Check1_Click()

End Sub

Private Sub Command1_Click()
If is_empty(DataCombo1) = True Then Exit Sub
If Len(DataCombo1.Text) < 9 Or Mid(DataCombo1.Text, 5, 1) <> "-" Then MsgBox "Enry
must be in this format (ex. yyyy-yyyy).", vbExclamation, "CSRS version 1":
DataCombo1.SetFocus: Exit Sub

If Check1.Value = 1 Then
   rs_prnt_list.Filter = "SchoolYear ='" & DataCombo1.Text & "'"
Else
   rs_prnt_list.Filter = "SchoolYear ='" & DataCombo1.Text & "' And Status <>'Drop'"
End If

If rs_prnt_list.RecordCount < 1 Then
   MsgBox "The selected school year does not have student record.", vbExclamation, "CSRS
version 1"
   DataCombo1.SetFocus
   Exit Sub
End If

With rpt_header
   .SchoolAddress = school_address
   .SchoolName = school_name
   .SY = "S.Y.  " & rs_prnt_list(11)
End With

Set DataReport4.DataSource = rs_prnt_list
DataReport4.Show vbModal

End Sub

Private Sub Command2_Click()
Unload Me
End Sub

Private Sub DataCombo1_KeyPress(KeyAscii As Integer)
KeyAscii = 0
End Sub

Private Sub Form_Load()
Call use_pos(Me)

Dim rs_sy As New ADODB.Recordset
Call set_rec_getData(rs_sy, cn, "Select tblSchoolYear.* From tblSchoolYear Order by
```

```
SchoolYear Asc")

Set DataCombo1.RowSource = rs_sy
DataCombo1.ListField = "SchoolYear"

Set rs_sy = Nothing

Call set_rec_getData(rs_prnt_list, cn, "Select qryStudents.* From qryStudents Order by Sex
Desc,LastName Asc,FirstName Asc")

End Sub

Private Sub Form_Unload(Cancel As Integer)
Set rs_prnt_list = Nothing

Call save_pos(Me)
End Sub

Option Explicit


Dim rs_prnt_list As New ADODB.Recordset




Private Sub Command1_Click()
If is_empty(DataCombo1) = True Then Exit Sub
If Len(DataCombo1.Text) < 9 Or Mid(DataCombo1.Text, 5, 1) <> "-" Then MsgBox "Enry
must be in this format (ex. yyyy-yyyy).", vbExclamation, "CSRS version 1":
DataCombo1.SetFocus: Exit Sub
On Error Resume Next

Dim cn_tmp As New ADODB.Connection
Dim rs_tmp As New ADODB.Recordset
Dim rs_lv As New ADODB.Recordset

Call set_rec_getData(rs_lv, cn, "Select tblLevel.* From tblLevel Order by LevelNo Asc")
If rs_lv.RecordCount < 1 Then MsgBox "There is no level available.", vbExclamation,
"CSRS version 1": Exit Sub

Kill Environ("TMP") & "\SupportDB.mdb"
FileCopy App.Path & "\SupportDB.db", Environ("TMP") & "\SupportDB.mdb"

Call set_conn_getData(cn_tmp, Environ("TMP") & "\SupportDB.mdb", True, "reg386")
Call set_rec_getData(rs_tmp, cn_tmp, "Select tblSummary.* From tblSummary")

rs_prnt_list.Requery
Do While Not rs_lv.EOF
   If Check1.Value = 1 Then
```

```
    rs_tmp.AddNew
      rs_tmp.Fields(0) = rs_lv.Fields(1)
      rs_prnt_list.Filter = "SchoolYear ='" & DataCombo1.Text & "' And LevelNo =" &
rs_lv.Fields(0) & " And Sex ='Male'"
      rs_tmp.Fields(1) = rs_prnt_list.RecordCount
      rs_prnt_list.Filter = "SchoolYear ='" & DataCombo1.Text & "' And LevelNo =" &
rs_lv.Fields(0) & " And Sex ='Female'"
      rs_tmp.Fields(2) = rs_prnt_list.RecordCount
      rs_prnt_list.Filter = "SchoolYear ='" & DataCombo1.Text & "' And LevelNo =" &
rs_lv.Fields(0)
      rs_tmp.Fields(3) = rs_prnt_list.RecordCount
    rs_tmp.Update
  Else
    rs_tmp.AddNew
      rs_tmp.Fields(0) = rs_lv.Fields(1)
      rs_prnt_list.Filter = "Status <>'Drop' And SchoolYear ='" & DataCombo1.Text & "'
And LevelNo =" & rs_lv.Fields(0) & " And Sex ='Male'"
      rs_tmp.Fields(1) = rs_prnt_list.RecordCount
      rs_prnt_list.Filter = "Status <>'Drop' And SchoolYear ='" & DataCombo1.Text & "'
And LevelNo =" & rs_lv.Fields(0) & " And Sex ='Female'"
      rs_tmp.Fields(2) = rs_prnt_list.RecordCount
      rs_prnt_list.Filter = "Status <>'Drop' And SchoolYear ='" & DataCombo1.Text & "'
And LevelNo =" & rs_lv.Fields(0)
      rs_tmp.Fields(3) = rs_prnt_list.RecordCount
    rs_tmp.Update
  End If

  rs_lv.MoveNext
Loop

With rpt_header
  .SchoolAddress = school_address
  .SchoolName = school_name
  .SY = "S.Y.  " & DataCombo1.Text
End With

Set DataReport5.DataSource = rs_tmp
DataReport5.Show vbModal

Set rs_lv = Nothing
Set rs_tmp = Nothing
Set cn_tmp = Nothing

End Sub

Private Sub Command2_Click()
Unload Me
End Sub

Private Sub DataCombo1_KeyPress(KeyAscii As Integer)
```

```
KeyAscii = 0
End Sub

Private Sub Form_Load()
Call use_pos(Me)

Dim rs_sy As New ADODB.Recordset
Call set_rec_getData(rs_sy, cn, "Select tblSchoolYear.* From tblSchoolYear Order by SchoolYear Asc")

Set DataCombo1.RowSource = rs_sy
DataCombo1.ListField = "SchoolYear"

Set rs_sy = Nothing

Call set_rec_getData(rs_prnt_list, cn, "Select qrySummaryReport.* From qrySummaryReport Order by LevelNo Asc")

End Sub

Private Sub Form_Unload(Cancel As Integer)
Set rs_prnt_list = Nothing

Call save_pos(Me)
End Sub

Option Explicit


Dim rs_set As New ADODB.Recordset


Private Sub Command1_Click()
If is_empty(Text1) = True Then Exit Sub
If is_empty(Text2) = True Then Exit Sub

With rs_set
  .Fields(0) = Text1.Text
  .Fields(1) = Text2.Text
  .Update
End With

school_name = Text1.Text
school_address = Text2.Text
MDIForm1.Caption = Text1.Text & " - " & "Computerize School Registration Software version 1"

MsgBox "Changes has been successfully saved.", vbInformation, "CSRS version 1"
Unload Me
End Sub
```

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

Private Sub Form_Load()
Call use_pos(Me)

Call set_rec_getData(rs_set, cn, "Select SystemInfo.* From SystemInfo")

Text1.Text = rs_set.Fields(0)
Text2.Text = rs_set.Fields(1)
End Sub

Private Sub Form_Unload(Cancel As Integer)
Set rs_set = Nothing

Call save_pos(Me)
End Sub

Private Sub Text1_GotFocus()
Call highlight_focus(Text1)
End Sub

Private Sub Text2_GotFocus()
Call highlight_focus(Text2)
End Sub


Private Sub Command1_Click()
Form26.Show
Form26.SetFocus
Form26.WindowState = 0
End Sub

Private Sub Command2_Click()
Form25.Show
Form25.SetFocus
Form25.WindowState = 0
End Sub

Private Sub Command3_Click()
Form22.Show
Form22.SetFocus
Form22.WindowState = 0
End Sub

Private Sub Command4_Click()
Unload Me
```

```
End Sub

Private Sub Form_Load()
Call use_pos(Me)
End Sub

Private Sub Form_Unload(Cancel As Integer)
Call save_pos(Me)
End Sub

Private Sub Label4_Click()

End Sub

Option Explicit

Dim c_attempt As String
Dim rs_user As New ADODB.Recordset
Dim conn_user As New ADODB.Connection

Private Sub Command1_Click()
end_app = True
Unload Me
End Sub

Private Sub Command2_Click()
'------------------------------------------------
'Verify the fields if empty
'------------------------------------------------
If Text1.Text = "" Then Text1.SetFocus: Exit Sub
If Text2.Text = "" Then Text2.SetFocus: Exit Sub
'------------------------------------------------
'Check if the User Name is valid
'------------------------------------------------
If rec_found(rs_user, "Username", Text1.Text) = False Then
   c_attempt = c_attempt - 1
   If c_attempt < 0 Then
      MsgBox "You already used all attempt." & vbCrLf & "This will terminate the application.", vbCritical, "CSRS version 1"
   Else
      MsgBox "The User Name you entered is not valid." & vbCrLf & "Please try again." & vbCrLf & vbCrLf & "Warning: You only have " & c_attempt & " attempt.", vbCritical, "CSRS version 1"
      Label7.Caption = c_attempt
   End If
   Text1.SetFocus
   Call verify_attempt
   Exit Sub
End If
'------------------------------------------------
```

```
'Check if the Password is valid
'-----------------------------------------------
If Text2.Text <> rs_user.Fields(3) Then
   c_attempt = c_attempt - 1
   Label7.Caption = c_attempt
   If c_attempt < 0 Then
      MsgBox "You already used all attempt." & vbCrLf & "This will terminate the
application.", vbCritical, "CSRS version 1"
   Else
      MsgBox "You did NOT enter the Correct Password." & vbCrLf & "Please try again." &
vbCrLf & vbCrLf & "Warning: You only have " & c_attempt & " attempt.", vbCritical,
"CSRS version 1"
      Label7.Caption = c_attempt
   End If
   Text2.SetFocus
   Call verify_attempt
   Exit Sub
End If
'-----------------------------------------------
'Copy the Username and log-time to variable
'-----------------------------------------------
user_name = Text1.Text
user_login = Now
user_type = rs_user.Fields(2)
'-----------------------------------------------
'This the Username and log-time to variable
'-----------------------------------------------
Call record_login(user_login, user_name)
With MDIForm1.StatusBar1.Panels
   .Item(3).Text = user_name
   .Item(6).Text = user_login
End With
Unload Me
End Sub

Private Sub Form_Activate()
Text1.SetFocus
End Sub

Private Sub Form_Load()
'-----------------------------------------------
'Set the variables to have connection to database
'-----------------------------------------------
Call set_conn_getData(conn_user, App.Path & "\MasterFile.mdb", True, "reg386")
Call set_rec_getData(rs_user, conn_user, "Select * From Users")
'-----------------------------------------------
'Move them form to center
'-----------------------------------------------
Call centerForm(Me, Screen.Height, Screen.Width)
'-----------------------------------------------
```

```
'Initialize the number of attempt
'-----------------------------------------------
c_attempt = 3
Label7.Caption = c_attempt
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
'-----------------------------------------------
'If the user click the close button
'-----------------------------------------------
If UnloadMode = 0 Then end_app = True
End Sub

Private Sub Form_Unload(Cancel As Integer)
'-----------------------------------------------
'Clear variable from the computer memory
'-----------------------------------------------
Set rs_user = Nothing
Set conn_user = Nothing
End Sub
Private Sub verify_attempt()
'-----------------------------------------------------
'If all attempt is used then terminate the application
'-----------------------------------------------------
If c_attempt < 0 Then end_app = True: Unload Me
End Sub

Private Sub Text1_GotFocus()
Call highlight_focus(Text1)
End Sub

Private Sub Text2_GotFocus()
Call highlight_focus(Text2)
End Sub


Private Sub Combo1_KeyPress(KeyAscii As Integer)
KeyAscii = 0
End Sub

Private Sub Command1_Click()
Unload Me
End Sub

Private Sub Command2_Click()
If is_empty(Combo1) = True Then Exit Sub
If is_empty(Text1) = True Then Exit Sub

If Len(Text1.Text) <> 4 Then MsgBox "Invalid entry. Please check it.", vbExclamation,
```

```
"CSRS version 1": Text1.SetFocus: Exit Sub

If Option1.Value = True Then
   Dim rep As Integer
   rep = MsgBox("Are you sure you want to clear all log-in details?", vbCritical + vbYesNo,
"CSRS version 1")
   If rep = vbYes Then Screen.MousePointer = vbHourglass: cn.Execute "Delete * From
UsersLog": Screen.MousePointer = vbDefault: MsgBox "All log-in details has been
successfully cleared.", vbInformation, "CSRS version 1"
Else
   If month_value(Combo1.Text) = 0 Then MsgBox "Invalid selection.", vbExclamation,
"CSRS version 1": Combo1.SetFocus: Exit Sub
   Dim rs_log As New ADODB.Recordset

   Call set_rec_getData(rs_log, cn, "Select qryUsers.* From qryUsers Where Month =" &
month_value(Combo1.Text) & " And Year =" & Val(Text1.Text) & " Order by Sort Asc")

   With rpt_header
      .SchoolAddress = school_address
      .SchoolName = school_name
   End With

   Set DataReport6.DataSource = rs_log
   DataReport6.Show vbModal

   Set rs_log = Nothing
End If
End Sub

Private Sub Frame1_DragDrop(Source As Control, X As Single, Y As Single)

End Sub

Private Sub Option1_Click()
If Option1.Value = True Then
   Frame2.Visible = False
Else
   Frame2.Visible = True
End If
End Sub

Private Sub Option2_Click()
If Option1.Value = True Then
   Frame2.Visible = False
Else
   Frame2.Visible = True
End If
End Sub
Private Sub Form_Load()
Call use_pos(Me)
```

```
Option1.Value = True
With Combo1
   .AddItem "January"
   .AddItem "February"
   .AddItem "March"
   .AddItem "April"
   .AddItem "May"
   .AddItem "June"
   .AddItem "July"
   .AddItem "August"
   .AddItem "September"
   .AddItem "October"
   .AddItem "November"
   .AddItem "December"
End With
Text1.Text = Year(Date)
End Sub

Private Sub Form_Unload(Cancel As Integer)
Call save_pos(Me)
End Sub

Function month_value(ByVal Month_Name As String) As Byte
month_value = 0
Month_Name = Left(UCase(Month_Name), 1) & Right(LCase(Month_Name),
Len(Month_Name) - 1)
Select Case Month_Name
   Case "January": month_value = 1
   Case "February": month_value = 2
   Case "March": month_value = 3
   Case "April": month_value = 4
   Case "May": month_value = 5
   Case "June": month_value = 6
   Case "July": month_value = 7
   Case "August": month_value = 8
   Case "September": month_value = 9
   Case "October": month_value = 10
   Case "November": month_value = 11
   Case "December": month_value = 12
End Select
End Function

Private Sub Text1_GotFocus()
Call highlight_focus(Text1)
End Sub

Private Sub Text1_KeyPress(KeyAscii As Integer)
If Not ((KeyAscii >= 48 And KeyAscii <= 57) Or KeyAscii = 8) Then KeyAscii = 0
End Sub
```

```
Private Sub Command1_Click()
Form27.add_state = True
Form27.Show
Me.Enabled = False
End Sub

Private Sub Command2_Click()
If rs_log.RecordCount < 1 Then MsgBox "No user in the list.Please check it!", vbExclamation, "CSRS version 1": Exit Sub
If Not rs_log.RecordCount < 1 Then rs_log.AbsolutePosition = ListView1.SelectedItem
Form27.add_state = False
Form27.Show
Me.Enabled = False
End Sub

Private Sub Command4_Click()
On Error GoTo Err:
With rs_log
   '---------------------------------
   'Check if there is no record
   '---------------------------------
   If .RecordCount < 1 Then MsgBox "No user in the list.Please check it!", vbExclamation, "CSRS version 1": Exit Sub
   If Not rs_log.RecordCount < 1 Then rs_log.AbsolutePosition = ListView1.SelectedItem
   If LCase(.Fields(1)) = LCase(user_name) Then MsgBox "Cannot delete because user is curently logged.", vbExclamation, "CSRS version 1": Exit Sub

   '---------------------------------
   'Confirm deletion of record
   '---------------------------------
   Dim ans As Integer
   Dim pos As Integer
   ans = MsgBox("Are you sure you want to delete the selected record?", vbCritical + vbYesNo, "Confirm Record Delete")
   Me.MousePointer = vbHourglass
   If ans = vbYes Then
      '-----------------------------
      'Delete the record
      '-----------------------------
      pos = Val(ListView1.SelectedItem)
      Call delete_rec(cn, "Users", "Username", ListView1.SelectedItem.ListSubItems(1), False, 0)
      .Requery
      If .RecordCount > 0 Then
         .AbsolutePosition = pos
         If .EOF Then .MoveFirst
         '---------------------------------
         'Fill listview
```

```
        '--------------------------------
        pos = .AbsolutePosition
        load_rec
        ListView1.ListItems.Item(pos).EnsureVisible
        ListView1.ListItems.Item(pos).Selected = True
        .AbsolutePosition = ListView1.SelectedItem
        '--------------------------------
        'End-fill listview
        '--------------------------------
      Else
        ListView1.ListItems.Clear
      End If
      MsgBox "Record has been successfully deleted.", vbInformation, "Confirm"
    End If
    ans = 0
    pos = 0
    Me.MousePointer = vbDefault
End With
Exit Sub
Err:
    prompt_err (Err.Description & vbCrLf & vbCrLf & "Error Number: " & Err.Number):
Me.MousePointer = vbDefault: Exit Sub
End Sub

Private Sub Command5_Click()
rs_log.Requery
load_rec
End Sub

Private Sub Command6_Click()
Unload Me
End Sub

Private Sub Form_Activate()

If Not rs_log.RecordCount < 1 Then rs_log.AbsolutePosition = ListView1.SelectedItem
Command1.SetFocus
End Sub

Private Sub Form_Load()
Call use_pos(Me)

Me.Caption = Me.Caption
Call set_rec_getData(rs_log, cn, "Select Users.* From Users Order by Username Asc")
load_rec
End Sub

Private Sub Form_Unload(Cancel As Integer)
Set rs_log = Nothing
```

```
Call save_pos(Me)
End Sub
Sub load_rec()
Screen.MousePointer = vbHourglass

Call FillListView(ListView1, rs_log, 4, 1, True, False)

Screen.MousePointer = vbDefault
End Sub

Private Sub Frame1_DragDrop(Source As Control, X As Single, Y As Single)

End Sub

Private Sub ListView1_Click()

If Not rs_log.RecordCount < 1 Then rs_log.AbsolutePosition = ListView1.SelectedItem
End Sub



Private Sub SSTab1_Click()

End Sub

Option Explicit


Public add_state As Boolean

Private Sub Command1_Click()
If is_empty(Text1) = True Then Exit Sub
If is_empty(Text3) = True Then Exit Sub

With rs_log
  If add_state = True Then .AddNew
    .Fields(1) = Text1.Text
    .Fields(2) = Text2.Text
    .Fields(3) = Text3.Text
  .Update
End With
'-----------------------------
'Inform updates
'-----------------------------
If add_state = True Then
  MsgBox "Adding of new user has been successfull.", vbInformation, "CSRS version 1"
  Dim rep As Integer
  rep = MsgBox("Do you want to add another user?", vbQuestion + vbYesNo, "CSRS
version 1")
  If rep = vbYes Then
```

```
    Text1.Text = ""
    Text2.Text = "User"
    Text3.Text = ""
    Text1.SetFocus
    rs_log.Requery
    Form26.load_rec
  Else
    rs_log.Requery
    Form26.load_rec
    Unload Me
  End If
  rep = 0
Else
  If LCase(user_name) = LCase(Text1.Text) Then
    user_name = Text1.Text
    user_type = Text2.Text
    MDIForm1.StatusBar1.Panels.Item(3).Text = user_name
  End If
  MsgBox "Changes in record has been successfully saved.", vbInformation, "CSRS version
1"
  Dim pos As Long

  pos = rs_log.AbsolutePosition
  rs_log.Requery
  Form26.load_rec
  rs_log.AbsolutePosition = pos

  Form26.ListView1.ListItems.Item(pos).EnsureVisible
  Form26.ListView1.ListItems.Item(pos).Selected = True

  pos = 0
  Unload Me
End If
'------------------------------
'End-Inform updates
'------------------------------
End Sub

Private Sub Command2_Click()
Unload Me
End Sub

Private Sub Form_Load()
Call use_pos(Me)

If add_state = False Then
  Text1.Text = rs_log.Fields(1)
  Text2.Text = rs_log.Fields(2)
  Text3.Text = rs_log.Fields(3)
  Me.Icon = ImageList1.ListImages(1).Picture
```

```
   Me.Caption = "Edit Existing User"
End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
Form26.Enabled = True

Call save_pos(Me)
End Sub

Private Sub Label1_Click()

End Sub

Private Sub Text1_GotFocus()
Call highlight_focus(Text1)
End Sub
Private Sub Text2_KeyPress(KeyAscii As Integer)
KeyAscii = 0
End Sub

Private Sub Text2_Validate(Cancel As Boolean)
If Text2.Text <> "User" And Text2.Text <> "Admin" Then MsgBox "Please select a valid
entry in the list.", vbExclamation, "CSRS version 1": Cancel = True
End Sub

Private Sub Text3_GotFocus()
Call highlight_focus(Text3)
End Sub

Option Explicit


Public sAssign As Boolean
Dim sEdit As Boolean
Public sSec_Id As Long


Private Sub Command1_Click()
If Command1.Caption = "Cancel" And sEdit = True Then
   disable_text
   Command1.Caption = "&OK"
   Command1.Default = True
   Command2.Caption = "&Edit"
   Command3.Visible = False
   Me.Caption = "Student Section"
Else
   Unload Me
End If
End Sub
```

```
Private Sub Command2_Click()
If rs_stud.Fields(12) = "Drop" Then MsgBox "Cannot Assign or Re-assign Section because the selected student was currently dropped in the school." & vbCrLf & vbCrLf & "Note: You can Undrop the Student if you want by selecting 'File' in the menu and then 'Student Record' and select 'Undrop Student'.", vbExclamation, "CSRS version 1": Exit Sub
If Command2.Caption = "&Edit" Then
   sEdit = True
   enable_text
   Text2.Locked = True
   Command1.Caption = "&Cancel"
   Command2.Caption = "&Save"
   Command2.Default = True
   Command3.Visible = True
   Me.Caption = "Re-Assign Advisor"
Else
   If is_empty(Text1) = True Then Exit Sub
   If is_empty(Text2) = True Then Exit Sub
   With rs_stud
      .Fields(9) = Text1.Text
      .Fields(10) = sSec_Id
      .Update
      Dim pos As Long
      pos = .AbsolutePosition
      .Requery
      .AbsolutePosition = pos
      pos = 0
   End With
   MsgBox "Updating of data has been successfull.", vbInformation, "CSRS version 1"
   Unload Me
End If
End Sub

Private Sub Command3_Click()
If Val(Text1.Text) < 1 Then MsgBox "Please put the general average first.", vbExclamation, "CSRS version 1": Text1.SetFocus: Exit Sub
If Text2.BorderStyle = 1 Then Form4.gen_ave = Val(Text1.Text): Form4.Show: Me.Enabled = False
End Sub

Private Sub Form_Load()
Call use_pos(Me)


If sAssign = True Then
   enable_text
   Command1.Caption = "&Cancel"
   Command2.Caption = "&Update"
   Command2.Default = True
   Command3.Visible = True
```

```
    Me.Caption = "Assign Advisor"
Else
    If Not rs_stud.RecordCount < 1 Then
        With rs_stud
            Text1.Text = .Fields(9)
            Text2.Text = .Fields(20)
            Text3.Text = .Fields(11)
            Text4.Text = .Fields(21)
            sSec_Id = .Fields(10)
        End With
    End If
    Command1.Default = True
End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
Form1.Enabled = True

Call save_pos(Me)
End Sub

Private Sub Text1_Change()
If Text2.BorderStyle = 1 Then Text2.Text = ""
End Sub

Private Sub Text1_KeyPress(KeyAscii As Integer)
If Not ((KeyAscii >= 48 And KeyAscii <= 57) Or KeyAscii = 8) Then KeyAscii = 0
End Sub

Private Sub Text2_GotFocus()
Call highlight_focus(Text2)
End Sub

Private Sub Text2_KeyPress(KeyAscii As Integer)
Command3_Click
End Sub

Private Sub Text3_GotFocus()
Call highlight_focus(Text3)
End Sub
Private Sub Text1_GotFocus()
Call highlight_focus(Text1)
End Sub
Sub disable_text()
    Text1.Locked = True
    Text1.BorderStyle = 0
    Text1.BackColor = &H8000000F
    Text2.BorderStyle = 0
    Text2.BackColor = &HE6FFFF
End Sub
```

```
Sub enable_text()
   Text1.Locked = False
   Text1.BorderStyle = 1
   Text1.BackColor = &H80000005
   Text2.BorderStyle = 1
   Text2.BackColor = &HE6FFFF
End Sub

Option Explicit


Public gen_ave As Integer
Public lv_id As Long

Dim rs_sel_sec As New ADODB.Recordset
Dim rs_cur_stud As New ADODB.Recordset


Private Sub Command1_Click()
If ListView1.ListItems.Count < 1 Then Unload Me: Exit Sub

If Val(rs_cur_stud.RecordCount) + 1 > Val(ListView1.SelectedItem.ListSubItems(5)) Then
   MsgBox "This section already have " & rs_cur_stud.RecordCount & " student in this
School Year " & Form1.SY & "." & vbCrLf & vbCrLf & "Note: This section allowed only "
& ListView1.SelectedItem.ListSubItems(5) & " students.", vbExclamation, "CSRS version 1"
Else
   Form3.sSec_Id = ListView1.SelectedItem.ListSubItems(1)
   Form3.Text2 = ListView1.SelectedItem.ListSubItems(2)
   Unload Me
End If
End Sub

Private Sub Command3_Click()
Form5.Show: Me.Enabled = False
End Sub

Private Sub Form_Activate()

If ListView1.ListItems.Count < 1 Then Exit Sub
reset
With ListView1.SelectedItem
   Text1.Text = .ListSubItems(3) & " - " & .ListSubItems(4)
   Text3.Text = .ListSubItems(5)
End With
rs_cur_stud.Filter = "Sec = " & ListView1.SelectedItem.ListSubItems(1) & " And
SchoolYear = '" & Form1.SY & "'"
Text4.Text = rs_cur_stud.RecordCount
End Sub

Private Sub Form_Load()
```

```
Call use_pos(Me)

Call set_rec_getData(rs_sel_sec, cn, "Select qrySections.* From qrySections Order by MinAve Desc")
Call set_rec_getData(rs_cur_stud, cn, "Select qryStudentAndSection.* From qryStudentAndSection")

Label1.Caption = Label1.Caption & gen_ave

End Sub

Private Sub Form_Unload(Cancel As Integer)
Set rs_sel_sec = Nothing
Set rs_cur_stud = Nothing
Form3.Enabled = True

Call save_pos(Me)
End Sub

Private Sub ListView1_ItemClick(ByVal Item As MSComctlLib.ListItem)
If ListView1.ListItems.Count < 1 Then Exit Sub
reset
With ListView1.SelectedItem
   Text1.Text = .ListSubItems(3) & " - " & .ListSubItems(4)
   Text3.Text = .ListSubItems(5)
End With
rs_cur_stud.Filter = "((Sec) = " & ListView1.SelectedItem.ListSubItems(1) & ") And ((SchoolYear) = '" & Form1.SY & "') And ((Status) <> 'Drop')"
Text4.Text = rs_cur_stud.RecordCount
End Sub
Sub reset()
Text1.Text = ""
Text3.Text = ""
Text4.Text = ""
End Sub

Private Sub Text2_Change()
If Text2.Text = "" Then Exit Sub

rs_sel_sec.Filter = "MinAve <= " & gen_ave & " And LevelNo = " & lv_id

load_rec
End Sub

Private Sub Text2_GotFocus()
Call highlight_focus(Text2)
End Sub

Private Sub Text2_KeyPress(KeyAscii As Integer)
Command3_Click
```

```
End Sub
Sub load_rec()
Screen.MousePointer = vbHourglass

rs_sel_sec.Requery
Call FillListView(ListView1, rs_sel_sec, 6, 1, True, True)

Screen.MousePointer = vbDefault
If ListView1.ListItems.Count < 1 Then
   Command1.Caption = "&Close"
Else
   Command1.Caption = "&Select"
End If
End Sub

Option Explicit


Dim rs_sel_lv As New ADODB.Recordset

Private Sub Command1_Click()
If ListView1.ListItems.Count < 1 Then Unload Me: Exit Sub
Form4.lv_id = ListView1.SelectedItem.ListSubItems(1)
Form4.Text2 = ListView1.SelectedItem.ListSubItems(2)
Unload Me
End Sub

Private Sub Form_Load()
Call use_pos(Me)

Call set_rec_getData(rs_sel_lv, cn, "Select tblLevel.* From tblLevel Order by LevelNo Asc")
load_rec
End Sub

Private Sub Form_Unload(Cancel As Integer)
Set rs_sel_lv = Nothing
Form4.Enabled = True

Call save_pos(Me)
End Sub
Sub load_rec()
Screen.MousePointer = vbHourglass

Call FillListView(ListView1, rs_sel_lv, 3, 1, True, True)

Screen.MousePointer = vbDefault
If ListView1.ListItems.Count < 1 Then Command1.Caption = "&Cancel"
End Sub

Option Explicit
```

```
Public add_state As Boolean

Private Sub Command1_Click()
'------------------------------
'Check the required field
'------------------------------
If is_empty(Text2) = True Then Exit Sub
If is_empty(Text3) = True Then Exit Sub
If is_empty(Text4) = True Then Exit Sub
If is_empty(Text5) = True Then Exit Sub
If is_empty(Text6) = True Then Exit Sub
If is_empty(Text1) = True Then Exit Sub
If is_empty(Text17) = True Then Exit Sub
If is_empty(Text7) = True Then Exit Sub
If is_empty(Text8) = True Then Exit Sub
If is_empty(Text9) = True Then Exit Sub
If is_empty(Text10) = True Then Exit Sub
If is_empty(Text11) = True Then Exit Sub
If is_empty(Text12) = True Then Exit Sub
If is_empty(Text13) = True Then Exit Sub
If is_empty(Text14) = True Then Exit Sub
If is_empty(Text15) = True Then Exit Sub
If is_empty(Text16) = True Then Exit Sub
'------------------------------
'End checking
'------------------------------

'------------------------------
'Updating Database
'------------------------------
Dim c_no As Long
With rs_stud
   If add_state = True Then .AddNew: c_no = get_num("tblStudents", "StudentNo", cn):
.Fields(0) = c_no: .Fields(11) = Form1.SY
      .Fields(1) = Text4.Text
      .Fields(2) = Text2.Text
      .Fields(3) = Text3.Text
      .Fields(4) = Text5.Text
      .Fields(5) = Format(Text6.Text & "/" & Text1.Text & "/" & Text17.Text,
"mm/dd/yyyy")
      .Fields(6) = Text7.Text
      .Fields(7) = Text9.Text
      .Fields(8) = Text15.Text

      .Fields(12) = Text16.Text
      .Fields(13) = DTPicker1.Value
      .Fields(14) = Text8.Text
      .Fields(15) = Text10.Text
```

```
        .Fields(16) = Text11.Text
        .Fields(17) = Text12.Text
        .Fields(18) = Text13.Text
        .Fields(19) = Text14.Text
    .Update
End With
'-----------------------------
'End-Updating Database
'-----------------------------

'-----------------------------
'Inform updates
'-----------------------------
If add_state = True Then
    MsgBox "Adding of New Record has been successfull.", vbInformation, "CSRS version 1"
    Dim rep As Integer
    rep = MsgBox("Do you want to Record another student?", vbQuestion + vbYesNo, "CSRS
version 1")
    If rep = vbYes Then
        Command2_Click
        Call locate_new_rec(c_no)
    Else
        Call locate_new_rec(c_no)
        Unload Me
    End If
    rep = 0
    c_no = 0
Else
    MsgBox "Changes in record has been successfully saved.", vbInformation, "CSRS version
1"
    Dim pos As Long

    pos = rs_stud.AbsolutePosition
    Form1.reload_rec
    rs_stud.AbsolutePosition = pos

    Form1.ListView1.ListItems.Item(pos).EnsureVisible
    Form1.ListView1.ListItems.Item(pos).Selected = True

    pos = 0
    Unload Me
End If
'-----------------------------
'End-Inform updates
'-----------------------------
End Sub
Private Sub locate_new_rec(ByVal no As Long)
Form1.reload_rec
rs_stud.Find "StudentNo =" & no & ""
If rs_stud.EOF Then rs_stud.MoveFirst
```

```
Form1.ListView1.ListItems.Item(rs_stud.AbsolutePosition).EnsureVisible
Form1.ListView1.ListItems.Item(rs_stud.AbsolutePosition).Selected = True
End Sub
Private Sub Command2_Click()
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = "Male"
Text6.Text = ""
Text7.Text = ""
Text8.Text = ""
Text9.Text = ""
Text10.Text = ""
Text11.Text = ""
Text12.Text = ""
Text13.Text = ""
Text14.Text = ""
Text15.Text = ""
Text16.Text = "New"
Text17.Text = ""
DTPicker1.Value = Date
Text2.SetFocus
End Sub

Private Sub Command3_Click()
Unload Me
End Sub

Private Sub Form_Load()
Call use_pos(Me)

If add_state = True Then
   Me.Caption = "Record New Student"
   DTPicker1.Value = Date
Else
   Me.Icon = ImageList1.ListImages(1).Picture
   Me.Caption = "Edit Existing Student"
   Command1.Caption = "&Save"

   '------------------------------
   'Get records
   '------------------------------
   With rs_stud
         Text4.Text = .Fields(1)
         Text2.Text = .Fields(2)
         Text3.Text = .Fields(3)
         Text5.Text = .Fields(4)

         Text6.Text = Format(.Fields(5), "mm")
```

```
        Text1.Text = Format(.Fields(5), "dd")
        Text17.Text = Format(.Fields(5), "yyyy")

        Text7.Text = .Fields(6)
        Text9.Text = .Fields(7)
        Text15.Text = .Fields(8)

        Text16.Text = .Fields(12)
        DTPicker1.Value = .Fields(13)
        Text8.Text = .Fields(14)
        Text10.Text = .Fields(15)
        Text11.Text = .Fields(16)
        Text12.Text = .Fields(17)
        Text13.Text = .Fields(18)
        Text14.Text = .Fields(19)
    End With
    '-----------------------------
    'End-Get records
    '-----------------------------

End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
Form1.Enabled = True

Call save_pos(Me)
End Sub

Private Sub Frame2_DragDrop(Source As Control, X As Single, Y As Single)

End Sub

Private Sub Text1_KeyPress(KeyAscii As Integer)
If Not ((KeyAscii >= 48 And KeyAscii <= 57) Or KeyAscii = 8) Then KeyAscii = 0
End Sub

Private Sub Text11_Validate(Cancel As Boolean)
If Not Text11.Text = "" And Len(Text11.Text) > 150 Then MsgBox "Please enter only 150
characters.", vbExclamation, "CSRS version 1": Cancel = True
End Sub

Private Sub Text13_Validate(Cancel As Boolean)
If Not Text13.Text = "" And Len(Text13.Text) > 150 Then MsgBox "Please enter only 150
characters.", vbExclamation, "CSRS version 1": Cancel = True
End Sub

Private Sub Text16_KeyPress(KeyAscii As Integer)
KeyAscii = 0
End Sub
```

```
Private Sub Text16_Validate(Cancel As Boolean)
If Text16.Text <> "Drop" And Text16.Text <> "New" And Text16.Text <> "Old" Then
MsgBox "Please select a valid entry in the list.", vbExclamation, "CSRS version 1": Cancel =
True
End Sub

Private Sub Text17_KeyPress(KeyAscii As Integer)
If Not ((KeyAscii >= 48 And KeyAscii <= 57) Or KeyAscii = 8) Then KeyAscii = 0
End Sub

Private Sub Text2_GotFocus()
Call highlight_focus(Text2)
End Sub
Private Sub Text3_GotFocus()
Call highlight_focus(Text3)
End Sub
Private Sub Text4_GotFocus()
Call highlight_focus(Text4)
End Sub
Private Sub Text1_GotFocus()
Call highlight_focus(Text1)
End Sub

Private Sub Text5_KeyPress(KeyAscii As Integer)
KeyAscii = 0
End Sub

Private Sub Text6_GotFocus()
Call highlight_focus(Text6)
End Sub

Private Sub Text6_KeyPress(KeyAscii As Integer)
If Not ((KeyAscii >= 48 And KeyAscii <= 57) Or KeyAscii = 8) Then KeyAscii = 0
End Sub

Private Sub Text7_GotFocus()
Call highlight_focus(Text7)
End Sub
Private Sub Text8_GotFocus()
Call highlight_focus(Text8)
End Sub
Private Sub Text9_GotFocus()
Call highlight_focus(Text9)
End Sub
Private Sub Text10_GotFocus()
Call highlight_focus(Text10)
End Sub
Private Sub Text12_GotFocus()
Call highlight_focus(Text12)
```

```
End Sub
Private Sub Text14_GotFocus()
Call highlight_focus(Text14)
End Sub
Private Sub Text15_GotFocus()
Call highlight_focus(Text15)
End Sub

Private Sub Text5_Validate(Cancel As Boolean)
If Text5.Text <> "Male" And Text5.Text <> "Female" Then MsgBox "Please select a valid
entry in the list.", vbExclamation, "CSRS version 1": Cancel = True
End Sub
Private Sub Text6_Validate(Cancel As Boolean)
If Text6.Text = "" Then Exit Sub
If Val(Text6.Text) = 0 Or Val(Text6.Text) > 12 Then MsgBox "Please enter a valid
dd/mm/yyyy date format.", vbExclamation, "CSRS version 1": Cancel = True: Exit Sub
If Val(Text6.Text) < 10 Then Text6.Text = "0" & Right(Text6.Text, 1)
End Sub
Private Sub Text1_Validate(Cancel As Boolean)
If Text1.Text = "" Then Exit Sub
If Val(Text1.Text) = 0 Or Val(Text1.Text) > 31 Then MsgBox "Please enter a valid
dd/mm/yyyy date format.", vbExclamation, "CSRS version 1": Cancel = True: Exit Sub
If Val(Text1.Text) < 10 Then Text1.Text = "0" & Right(Text1.Text, 1)
End Sub
Private Sub Text17_Validate(Cancel As Boolean)
If Text17.Text = "" Then Exit Sub
If Val(Text17.Text) < 1900 Or Val(Text17.Text) > 2100 Then MsgBox "Please enter a valid
dd/mm/yyyy date format.", vbExclamation, "CSRS version 1": Cancel = True
End Sub

Private Sub Text7_KeyPress(KeyAscii As Integer)
If Not ((KeyAscii >= 48 And KeyAscii <= 57) Or KeyAscii = 8) Then KeyAscii = 0
End Sub

Private Sub Text7_Validate(Cancel As Boolean)
If Not Text7.Text = "" And Val(Text7.Text) < 1 Then MsgBox "Please enter the valid age of
the student.", vbExclamation, "CSRS version 1": Cancel = True
End Sub

Option Explicit

Private Sub Combo1_KeyPress(KeyAscii As Integer)
KeyAscii = 0
End Sub

Private Sub Combo2_KeyPress(KeyAscii As Integer)
KeyAscii = 0
End Sub

Private Sub Command2_Click()
```

```
Unload Me
End Sub

Private Sub Command1_Click()
If Combo1.Text = "" Then Combo1.SetFocus: Exit Sub
If Option2.Value = True And Combo2.Text = "" Then Combo2.SetFocus: Exit Sub
Me.MousePointer = vbHourglass
On Error GoTo Err
If Option1.Value = True Then
'----------------------------
'For quick search
'----------------------------
   rs_stud.Filter = Combo1.Text & " like *" & Text1.Text & "*"
   MsgBox "There is/are " & rs_stud.RecordCount & " record found in the search for " & Text1.Text & "." & vbCrLf & "'Click' reload button in the Student Record form if you want to show all data.", vbInformation, "CSRS version 1"
Else
'----------------------------
'For custom search
'----------------------------
   rs_stud.Filter = Combo1.Text & " " & Combo2.Text & " '" & Text1.Text & "'"
   MsgBox "There is/are " & rs_stud.RecordCount & " record found in the search for " & Text1.Text & "." & vbCrLf & "'Click' reload button in the Student Record form if you want to show all data.", vbInformation, "CSRS version 1"
End If
'----------------------------
'Load search result
'----------------------------
Form1.fill_rec
Me.MousePointer = vbDefault
Unload Me
Exit Sub
'----------------------------
'Prompt if their is an error
'----------------------------
Err:
   Call prompt_err(Err.Description)
   Me.MousePointer = vbDefault
   Unload Me
End Sub

Private Sub Form_Load()
Call use_pos(Me)

Option1.Value = True
End Sub

Private Sub Form_Unload(Cancel As Integer)
Form1.Enabled = True
```

```
Call save_pos(Me)
End Sub

Private Sub Option1_Click()
Label1.Top = 1320
Combo1.Top = 1560
Label2.Visible = False
Label3.Visible = False
Text1.Visible = False
Combo2.Visible = False
End Sub

Private Sub Option2_Click()
Label1.Top = 1320
Combo1.Top = 1560
Label2.Visible = True
Label3.Visible = True
Text1.Visible = True
Combo2.Visible = True
End Sub

Private Sub Text1_GotFocus()
Call highlight_focus(Text1)
End Sub

Option Explicit


'--------------------------------------
'Create variable to connect to DB
'--------------------------------------
Dim rs1 As New ADODB.Recordset
Dim rs2 As New ADODB.Recordset
'--------------------------------------
'End-Create variable to connect to DB
'--------------------------------------

Private Sub Combo1_Click()
If Combo1.Text = "" Then Exit Sub
rs2.Filter = "LevelName ='" & Combo1.Text & "'"
Call fill_combo(Combo2, rs2)
End Sub

Private Sub Command1_Click()
Dim sql As String
'--------------------------------------
'Begin filtering of records
'--------------------------------------
If Check2.Value = 0 And Check3.Value = 0 Then
   sql = "Sex ='None'"
```

```
ElseIf Check2.Value <> Check3.Value Then
   If Check2.Value = 1 Then sql = "Sex = 'Male'"
   If Check3.Value = 1 Then sql = "Sex = 'Female'"
End If

If sql = "" Then
   If Not Combo1.Text = "" Then sql = "LevelName = '" & Combo1.Text & "'"
   If Not Combo2.Text = "" Then sql = sql & " And SectionName = '" & Combo2.Text & "'"
Else
   If Not Combo1.Text = "" Then sql = sql & " And LevelName = '" & Combo1.Text & "'"
   If Not Combo2.Text = "" Then sql = sql & " And SectionName = '" & Combo2.Text & "'"
End If

If Check1.Value = 0 And Check4.Value = 0 And Check5.Value = 0 Then
   If sql <> "" Then
      sql = sql & " And Status = 'None'"
   Else
      sql = "Status = 'None'"
   End If
ElseIf Check1.Value <> Check4.Value Or Check1.Value <> Check5.Value Or Check4.Value
<> Check5.Value Then
   If Check1.Value = 1 Then
      If sql <> "" Then
         sql = sql & " And Status = 'Drop'"
      Else
         sql = "Status = 'Drop'"
      End If
   Else
      If sql <> "" Then
         sql = sql & " And Status <> 'Drop'"
      Else
         sql = "Status <> 'Drop'"
      End If
   End If
   If Check4.Value <> Check5.Value Then '//////////////////////////////////////
   If Check4.Value = 1 Then
      If sql <> "" Then
         If Check1.Value = 1 Then
            sql = sql & " Or Status = 'New'"
         Else
            sql = sql & " And Status = 'New'"
         End If
      Else
         sql = sql & "Status = 'New'"
      End If
   End If

   If Check5.Value = 1 Then
      If sql <> "" Then
         If Check1.Value = 1 Or Check4.Value = 1 Then
```

```
            sql = sql & " Or Status = 'Old'"
        Else
            sql = sql & " And Status = 'Old'"
        End If
    Else
        sql = sql & "Status = 'Old'"
    End If
  End If
  End If '//////////////////////////////////////
End If

rs_stud.Filter = sql
'-------------------------------------
'Save settings to variable
'-------------------------------------
sds = Check1.Value
sms = Check2.Value
sfs = Check3.Value
sns = Check4.Value
sos = Check5.Value
'-------------------------------------
'End-Clear variable
'-------------------------------------
sql = ""
'-------------------------------------
'Load search result
'-------------------------------------
Form1.fill_rec
Me.MousePointer = vbDefault
Unload Me
End Sub

Private Sub Command2_Click()
Unload Me
End Sub

Private Sub Combo1_KeyPress(KeyAscii As Integer)
KeyAscii = 0
End Sub

Private Sub Combo2_KeyPress(KeyAscii As Integer)
KeyAscii = 0
End Sub

Private Sub Command3_Click()
Check1.Value = 0
Check2.Value = 1
Check3.Value = 1
Check4.Value = 1
Check5.Value = 1
```

```
Combo1.Text = ""
Combo2.Text = ""

rs1.Requery
rs2.Filter = adFilterNone '[ You can use also .Filter="" ]
rs2.Requery

Call fill_combo(Combo1, rs1)
Call fill_combo(Combo2, rs2)
End Sub

Private Sub Form_Load()
Call use_pos(Me)

Check1.Value = sds
Check2.Value = sms
Check3.Value = sfs
Check4.Value = sns
Check5.Value = sos

'--------------------------------------
'Set the variables
'--------------------------------------
Call set_rec_getData(rs1, cn, "Select tblLevel.* From tblLevel Order by LevelName Asc")
Call set_rec_getData(rs2, cn, "Select qrySections.* From qrySections Order by SectionName
Asc")
'--------------------------------------
'End-Set the variables
'--------------------------------------

'--------------------------------------
'Fill Combo control
'--------------------------------------
Call fill_combo(Combo1, rs1)
Call fill_combo(Combo2, rs2)
'--------------------------------------
'End-Fill Combo control
'--------------------------------------
End Sub

Private Sub Form_Unload(Cancel As Integer)
Form1.Enabled = True

'--------------------------------------
'Clear variable
'--------------------------------------
Set rs1 = Nothing
Set rs2 = Nothing
'--------------------------------------
```

```
'End-Clear variable
'-------------------------------------

Call save_pos(Me)
End Sub
Sub fill_combo(ByRef sCombo As ComboBox, ByRef sRS As ADODB.Recordset)
sCombo.Clear
If sRS.RecordCount < 1 Then Exit Sub
sRS.MoveFirst
Do While Not sRS.EOF
   sCombo.AddItem sRS.Fields(1)
   sRS.MoveNext
Loop
sRS.MoveFirst
End Sub

Private Sub Frame1_DragDrop(Source As Control, X As Single, Y As Single)

End Sub



Private Sub Command1_Click()
Form10.add_state = True
Form10.Show
Me.Enabled = False
End Sub

Private Sub Command2_Click()
If rs_level.RecordCount < 1 Then MsgBox "No level in the list.Please check it!",
vbExclamation, "CSRS version 1": Exit Sub
If Not ListView1.SelectedItem = "" And Not rs_level.RecordCount < 1 Then
rs_level.AbsolutePosition = ListView1.SelectedItem
Form10.add_state = False
Form10.Show
Me.Enabled = False
End Sub

Private Sub Command4_Click()
On Error GoTo Err:
With rs_level
   '---------------------------------
   'Check if there is no record
   '---------------------------------
   If .RecordCount < 1 Then MsgBox "No level in the list.Please check it!", vbExclamation,
"CSRS version 1": Exit Sub
   '---------------------------------
   'Confirm deletion of record
   '---------------------------------
   Dim ans As Integer
```

```
   Dim pos As Integer
   ans = MsgBox("Are you sure you want to delete the selected record?", vbCritical + vbYesNo, "Confirm Record Delete")
   Me.MousePointer = vbHourglass
   If ans = vbYes Then
      '----------------------------
      'Delete the record
      '----------------------------
      pos = Val(ListView1.SelectedItem)
      Call delete_rec(cn, "tblLevel", "LevelNo", "", True, Val(ListView1.SelectedItem.ListSubItems(1)))
      .Requery
      If .RecordCount > 0 Then
         .AbsolutePosition = pos
         If .EOF Then .MoveFirst
         '--------------------------------
         'Fill listview
         '--------------------------------
         pos = .AbsolutePosition
         load_rec
         ListView1.ListItems.Item(pos).EnsureVisible
         ListView1.ListItems.Item(pos).Selected = True
         .AbsolutePosition = ListView1.SelectedItem
         '--------------------------------
         'End-fill listview
         '--------------------------------
      Else
         ListView1.ListItems.Clear
      End If
      MsgBox "Record has been successfully deleted.", vbInformation, "Confirm"
   End If
   ans = 0
   pos = 0
   Me.MousePointer = vbDefault
End With
Exit Sub
Err:
   prompt_err (Err.Description & vbCrLf & vbCrLf & "Error Number: " & Err.Number): Me.MousePointer = vbDefault: Exit Sub
End Sub

Private Sub Command5_Click()
rs_level.Requery
load_rec
End Sub

Private Sub Command6_Click()
Unload Me
End Sub
```

```
Private Sub Form_Activate()

If Not rs_level.RecordCount < 1 Then rs_level.AbsolutePosition = ListView1.SelectedItem
Command1.SetFocus
End Sub

Private Sub Form_Load()
Call use_pos(Me)

Call set_rec_getData(rs_level, cn, "Select tblLevel.* From tblLevel Order by LevelNo Asc")
load_rec
End Sub

Private Sub Form_Unload(Cancel As Integer)
Set rs_level = Nothing

Call save_pos(Me)
End Sub
Sub load_rec()
Screen.MousePointer = vbHourglass

Call FillListView(ListView1, rs_level, 3, 1, True, True)

Screen.MousePointer = vbDefault
End Sub
Private Sub ListView1_Click()

If Not rs_level.RecordCount < 1 Then rs_level.AbsolutePosition = ListView1.SelectedItem
End Sub

Private Sub SSTab1_Click()

End Sub

Private Sub Text1_Change()
If ListView1.ListItems.Count < 1 Then Exit Sub
Call search_in_listview(ListView1, Text1.Text)
End Sub
Private Sub Text1_GotFocus()
Call highlight_focus(Text1)
End Sub
Option Explicit

' Reg Key Security Options...
Const READ_CONTROL = &H20000
Const KEY_QUERY_VALUE = &H1
Const KEY_SET_VALUE = &H2
Const KEY_CREATE_SUB_KEY = &H4
Const KEY_ENUMERATE_SUB_KEYS = &H8
Const KEY_NOTIFY = &H10
```

```
Const KEY_CREATE_LINK = &H20
Const KEY_ALL_ACCESS = KEY_QUERY_VALUE + KEY_SET_VALUE + _
                KEY_CREATE_SUB_KEY + KEY_ENUMERATE_SUB_KEYS + _
                KEY_NOTIFY + KEY_CREATE_LINK + READ_CONTROL

' Reg Key ROOT Types...
Const HKEY_LOCAL_MACHINE = &H80000002
Const ERROR_SUCCESS = 0
Const REG_SZ = 1                    ' Unicode nul terminated string
Const REG_DWORD = 4                 ' 32-bit number

Const gREGKEYSYSINFOLOC = "SOFTWARE\Microsoft\Shared Tools Location"
Const gREGVALSYSINFOLOC = "MSINFO"
Const gREGKEYSYSINFO = "SOFTWARE\Microsoft\Shared Tools\MSINFO"
Const gREGVALSYSINFO = "PATH"

Private Declare Function RegOpenKeyEx Lib "advapi32" Alias "RegOpenKeyExA" (ByVal
hKey As Long, ByVal lpSubKey As String, ByVal ulOptions As Long, ByVal samDesired As
Long, ByRef phkResult As Long) As Long
Private Declare Function RegQueryValueEx Lib "advapi32" Alias "RegQueryValueExA"
(ByVal hKey As Long, ByVal lpValueName As String, ByVal lpReserved As Long, ByRef
lpType As Long, ByVal lpData As String, ByRef lpcbData As Long) As Long
Private Declare Function RegCloseKey Lib "advapi32" (ByVal hKey As Long) As Long


Private Sub cmdSysInfo_Click()
  Call StartSysInfo
End Sub

Private Sub cmdOK_Click()
  Unload Me
End Sub

Public Sub StartSysInfo()
   On Error GoTo SysInfoErr

   Dim rc As Long
   Dim SysInfoPath As String

   ' Try To Get System Info Program Path\Name From Registry...
   If          GetKeyValue(HKEY_LOCAL_MACHINE,          gREGKEYSYSINFO,
gREGVALSYSINFO, SysInfoPath) Then
   ' Try To Get System Info Program Path Only From Registry...
   ElseIf      GetKeyValue(HKEY_LOCAL_MACHINE,      gREGKEYSYSINFOLOC,
gREGVALSYSINFOLOC, SysInfoPath) Then
      ' Validate Existance Of Known 32 Bit File Version
      If (Dir(SysInfoPath & "\MSINFO32.EXE") <> "") Then
         SysInfoPath = SysInfoPath & "\MSINFO32.EXE"

      ' Error - File Can Not Be Found...
```

```
    Else
      GoTo SysInfoErr
    End If
  ' Error - Registry Entry Can Not Be Found...
  Else
    GoTo SysInfoErr
  End If

  Call Shell(SysInfoPath, vbNormalFocus)

  Exit Sub
SysInfoErr:
  MsgBox "System Information Is Unavailable At This Time", vbOKOnly
End Sub

Public Function GetKeyValue(KeyRoot As Long, KeyName As String, SubKeyRef As
String, ByRef KeyVal As String) As Boolean
  Dim i As Long                           ' Loop Counter
  Dim rc As Long                          ' Return Code
  Dim hKey As Long                        ' Handle To An Open Registry Key
  Dim hDepth As Long                      '
  Dim KeyValType As Long                  ' Data Type Of A Registry Key
  Dim tmpVal As String                    ' Tempory Storage For A Registry Key Value
  Dim KeyValSize As Long                  ' Size Of Registry Key Variable
  '------------------------------------------------------------
  ' Open RegKey Under KeyRoot {HKEY_LOCAL_MACHINE...}
  '------------------------------------------------------------
  rc = RegOpenKeyEx(KeyRoot, KeyName, 0, KEY_ALL_ACCESS, hKey) ' Open Registry
Key

  If (rc <> ERROR_SUCCESS) Then GoTo GetKeyError        ' Handle Error...

  tmpVal = String$(1024, 0)                 ' Allocate Variable Space
  KeyValSize = 1024                         ' Mark Variable Size

  '------------------------------------------------------------
  ' Retrieve Registry Key Value...
  '------------------------------------------------------------
  rc = RegQueryValueEx(hKey, SubKeyRef, 0, _
                KeyValType, tmpVal, KeyValSize)    ' Get/Create Key Value

  If (rc <> ERROR_SUCCESS) Then GoTo GetKeyError        ' Handle Errors

  If (Asc(Mid(tmpVal, KeyValSize, 1)) = 0) Then           ' Win95 Adds Null Terminated
String...
    tmpVal = Left(tmpVal, KeyValSize - 1)           ' Null Found, Extract From String
  Else                                ' WinNT Does NOT Null Terminate String...
    tmpVal = Left(tmpVal, KeyValSize)               ' Null Not Found, Extract String Only
  End If
  '------------------------------------------------------------
```

```
  ' Determine Key Value Type For Conversion...
  '-------------------------------------------------------------
  Select Case KeyValType                          ' Search Data Types...
  Case REG_SZ                                     ' String Registry Key Data Type
    KeyVal = tmpVal                               ' Copy String Value
  Case REG_DWORD                                  ' Double Word Registry Key Data Type
    For i = Len(tmpVal) To 1 Step -1              ' Convert Each Bit
      KeyVal = KeyVal + Hex(Asc(Mid(tmpVal, i, 1)))   ' Build Value Char. By Char.
    Next
    KeyVal = Format$("&h" + KeyVal)               ' Convert Double Word To String
  End Select

  GetKeyValue = True                              ' Return Success
  rc = RegCloseKey(hKey)                          ' Close Registry Key
  Exit Function                                   ' Exit

GetKeyError:      ' Cleanup After An Error Has Occured...
  KeyVal = ""                                     ' Set Return Val To Empty String
  GetKeyValue = False                             ' Return Failure
  rc = RegCloseKey(hKey)                          ' Close Registry Key
End Function


Private Sub Label3_Click()

End Sub

Private Sub Form_Click()
Unload Me
End Sub

Private Sub Form_KeyPress(KeyAscii As Integer)
Unload Me
End Sub

Option Explicit

Private Sub MDIForm_Activate()
If end_app = True Then End
End Sub

Private Sub MDIForm_Load()
Call use_control_vis(Toolbar1)
Call use_control_vis(StatusBar1)
Call use_control_pos(Toolbar1)

Call set_conn_getData(cn, App.Path & "\MasterFile.mdb", True, "reg386")

Dim rs As New ADODB.Recordset
Call set_rec_getData(rs, cn, "Select SystemInfo.* From SystemInfo")
```

```
school_name = rs.Fields(0)
school_address = rs.Fields(1)

Set rs = Nothing

Me.Caption = school_name & " - " & Me.Caption

Me.Show
frmSplash.Show vbModal

Form24.Show vbModal
End Sub

Private Sub MDIForm_QueryUnload(Cancel As Integer, UnloadMode As Integer)
Dim repp As Integer
repp = MsgBox("This will terminate the application.Do you want to proceed?",
vbExclamation + vbYesNo, "CSRS version 1")
If repp = vbNo Then
   Cancel = 1
End If
End Sub

Private Sub MDIForm_Unload(Cancel As Integer)
'--------------------------------------
'Record user's logout time
'--------------------------------------
Call record_logout(user_login, user_name)

Set cn = Nothing

Call save_control_pos(Toolbar1)
Call save_control_vis(Toolbar1)
Call save_control_vis(StatusBar1)

'--------------------------------------
'Terminate the entire application
'--------------------------------------
End
End Sub

Private Sub mnuAbt_Click()
frmAbout.Show vbModal
End Sub

Private Sub mnuAI_Click()
Me.Arrange vbArrangeIcons
End Sub

Private Sub mnuAO_Click()
```

```
If user_type <> "Admin" Then MsgBox "This function is for administrator only. Please log-in as administrator to gain access.", vbCritical, "CSRS version 1": Exit Sub


Form23.Show
Form23.SetFocus
Form23.WindowState = 0
End Sub

Private Sub mnuC_Click()
Me.Arrange vbCascade
End Sub

Private Sub mnuCalc_Click()
On Error GoTo Err
Shell "calc.exe", vbNormalFocus
Exit Sub
Err:
   MsgBox "You don't have a Calculator installed in your computer.", vbExclamation, "CSRS version 1"
End Sub

Private Sub mnuCalen_Click()
Form16.Show
Form16.SetFocus
Form16.WindowState = 0
End Sub

Private Sub mnuE_Click()
Unload Me
End Sub

Private Sub mnuHSM_Click()
Toolbar1.Visible = Not Toolbar1.Visible
End Sub

Private Sub mnuHSS_Click()
StatusBar1.Visible = Not StatusBar1.Visible
End Sub

Private Sub mnuIR_Click()
Form17.Show
Form17.SetFocus
Form17.WindowState = 0
Form17.Command2.SetFocus
End Sub

Private Sub mnuL_Click()
Form9.Show
Form9.SetFocus
```

```
Form9.WindowState = 0
End Sub

Private Sub mnuMRS_Click()
Form17.Show
Form17.SetFocus
Form17.WindowState = 0
Form17.Command1.SetFocus
End Sub

Private Sub mnuMSR_Click()
If frm_stud_show = True Then Form1.SetFocus: Form1.WindowState = 0: Exit Sub
Form2.Show
End Sub

Private Sub mnuMSY_Click()
Form11.Show
Form11.SetFocus
Form11.WindowState = 0
End Sub

Private Sub mnuPR_Click()
Form21.Show
Form21.SetFocus
Form21.WindowState = 0
End Sub

Private Sub mnuS_Click()
Form13.Show
Form13.SetFocus
Form13.WindowState = 0
End Sub

Private Sub mnuSI_Click()

End Sub

Private Sub mnuSL_Click()
Form20.Show
Form20.SetFocus
Form20.WindowState = 0
End Sub

Private Sub mnuSPS_Click()
Form17.Show
Form17.SetFocus
Form17.WindowState = 0
Form17.Command4.SetFocus
End Sub
```

```
Private Sub mnuTAB_Click()
Toolbar1.Align = 2
End Sub

Private Sub mnuTAL_Click()
Toolbar1.Align = 3
End Sub

Private Sub mnuTAP_Click()
Toolbar1.Align = 1
End Sub

Private Sub mnuTAR_Click()
Toolbar1.Align = 4
End Sub

Private Sub mnuTH_Click()
Me.Arrange vbTileHorizontal
End Sub

Private Sub mnuTV_Click()
Me.Arrange vbTileVertical
End Sub

Private Sub mnuNP_Click()
On Error GoTo Err
Shell "notepad.exe", vbNormalFocus
Exit Sub
Err:
   MsgBox "You don't have a NotePad  installed in your computer.", vbExclamation, "CSRS
version 1"
End Sub

Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
Select Case Button.Index
   Case 3: mnuMSR_Click
   Case 4: mnuS_Click
   Case 5: mnuL_Click
   Case 6: mnuMSY_Click
   Case 8: mnuMRS_Click
   Case 9: mnuIR_Click
   Case 10: mnuSPS_Click
   Case 11: mnuSL_Click
   Case 12: mnuPR_Click
   Case 14: mnuCalc_Click
   Case 15: mnuNP_Click
   Case 16: mnuCalen_Click
   Case 18: mnuAbt_Click
End Select
End Sub
```

```
Option Explicit

Public Sub use_pos(ByRef sForm As Form)
On Error Resume Next
Dim t, l As String
Open App.Path & "\Settings\" & sForm.Name & ".pos" For Input As #1
   Input #1, t
   Input #1, l
Close #1
t = Trim(t)
l = Trim(l)
sForm.Top = Val(t)
sForm.Left = Val(l)
'-------------------------
'Clear variables
'-------------------------
t = ""
l = ""
End Sub
Public Sub save_pos(ByVal sForm As Form)
On Error Resume Next
Call create_save_setting_dir
Open App.Path & "\Settings\" & sForm.Name & ".pos" For Output As #1
   Print #1, sForm.Top
   Print #1, sForm.Left
Close #1
End Sub
Public Sub use_control_vis(ByRef sControl)
On Error Resume Next
Dim t As String
Open App.Path & "\Settings\" & sControl.Name & ".vis" For Input As #1
   Input #1, t
Close #1
t = Trim(t)
sControl.Visible = t
'-------------------------
'Clear variables
'-------------------------
t = ""
End Sub
Public Sub save_control_vis(ByVal sControl)
On Error Resume Next
Call create_save_setting_dir
Open App.Path & "\Settings\" & sControl.Name & ".vis" For Output As #1
   Print #1, sControl.Visible
Close #1
End Sub
Public Sub use_control_pos(ByRef sControl)
On Error Resume Next
Dim t As String
```

```
Open App.Path & "\Settings\" & sControl.Name & ".pos" For Input As #1
   Input #1, t
Close #1
t = Trim(t)
sControl.Align = Val(t)
'------------------------
'Clear variables
'------------------------
t = ""
End Sub
Public Sub save_control_pos(ByVal sControl)
On Error Resume Next
Call create_save_setting_dir
Open App.Path & "\Settings\" & sControl.Name & ".pos" For Output As #1
   Print #1, sControl.Align
Close #1
End Sub
Private Sub create_save_setting_dir()
On Error Resume Next
MkDir (App.Path & "\Settings")
End Sub
Public Sub FillListView(ByRef sListView As ListView, ByRef sRecordSource As
ADODB.Recordset, ByVal sNumOfFields As Byte, ByVal sNumIco As Byte, ByVal
with_num As Boolean, ByVal show_first_rec As Boolean)
Dim X As Variant '|Optional to be declare as variant|
Dim i As Byte
On Error Resume Next
sRecordSource.MoveFirst
sListView.ListItems.Clear
Do While Not sRecordSource.EOF
   If with_num = True Then
      Set X = sListView.ListItems.Add(, , sRecordSource.AbsolutePosition, sNumIco,
sNumIco)
   Else
      Set X = sListView.ListItems.Add(, , sRecordSource.Fields(0), sNumIco, sNumIco)
   End If
      For i = 1 To sNumOfFields - 1
         If Not sRecordSource.Fields(Val(i)) = "" Then
            If show_first_rec = True Then
               X.SubItems(i) = sRecordSource.Fields(Val(i) - 1)
            Else
               X.SubItems(i) = sRecordSource.Fields(Val(i))
            End If
         End If
      Next i
   sRecordSource.MoveNext
Loop
i = 0
Set X = Nothing
End Sub
```

```
Public Sub search_in_listview(ByRef sListView As ListView, ByVal sFindText As String)
Dim tmp_listtview As ListItem
Set tmp_listtview = sListView.FindItem(sFindText, lvwSubItem + lvwText, lvwPartial,
lvwPartial)
If Not tmp_listtview Is Nothing Then
   tmp_listtview.EnsureVisible
   tmp_listtview.Selected = True
End If
End Sub
Public Sub highlight_focus(ByRef sText As TextBox)
With sText
   .SelStart = 0
   .SelLength = Len(sText.Text)
End With
End Sub
Public Sub prompt_err(ByVal sErrorDescription As String)
MsgBox sErrorDescription & vbCrLf & vbCrLf & "*Note: Contact the programmer to learn
more about this.", vbExclamation, "CSRS version 1"
End Sub

Public Sub delete_rec(ByRef sCONN As ADODB.Connection, ByVal sTable As String,
ByVal sField As String, ByVal sString As String, ByVal isnumber As Boolean, ByVal snum
As Long)
If isnumber = True Then
   sCONN.Execute "Delete * From " & sTable & " Where " & sField & " =" & snum
Else
   sCONN.Execute "Delete * From " & sTable & " Where " & sField & " ='" & sString & "'"
End If
End Sub
Public Function is_empty(ByRef sText As Variant) As Boolean
If sText.Text = "" Then
   is_empty = True
   MsgBox "The field is required.Please check it!", vbExclamation, "CSRS version 1"
   sText.SetFocus
Else
   is_empty = False
End If
End Function
Public Function get_num(ByVal sTable As String, ByVal sField As String, ByRef sCN As
ADODB.Connection) As Long
On Error GoTo Err
Dim rs As New ADODB.Recordset
rs.Open "SELECT Max(" & sTable & "." & sField & ") AS [Number] From " & sTable & "
ORDER BY Max(" & sTable & "." & sField & ") DESC", sCN, adOpenStatic,
adLockOptimistic
get_num = rs.Fields(0) + 1

sTable = ""
sField = ""
Set rs = Nothing
```

```
Exit Function
Err:
   '---------------------------------
   'Error when incounter a null value
   '---------------------------------
   If Err.Number = 94 Then get_num = 1: Resume Next
End Function
Public Function if_exist(ByVal sTable As String, ByVal sField As String, ByRef sEntryField
As Variant) As Boolean
Dim rs As New ADODB.Recordset
if_exist = False
Call set_rec_getData(rs, cn, "Select * From " & sTable & " Where " & sField & " ='" &
sEntryField.Text & "'")
If rs.RecordCount > 0 Then
   MsgBox "The adding of new entry cannot be done because '" & sEntryField.Text & "' is
already" & vbCrLf & "exist in the record.Please check and change it." & vbCrLf & vbCrLf &
"Note: Duplication of entries is not allowed in this form.", vbExclamation, "CSRS version 1"
   sEntryField.SetFocus
   if_exist = True
End If
Set rs = Nothing
End Function
Public Sub centerForm(ByRef sForm As Form, ByVal sHeight As Integer, ByVal sWidth As
Integer)
sForm.Move (sWidth - sForm.Width) / 2, (sHeight - sForm.Height) / 2
End Sub

Option Explicit

Global school_name          As String
Global school_address       As String

Global user_type            As String
Global user_name            As String
Global user_login           As Date

Global end_app              As Boolean

Global rs_log               As New ADODB.Recordset

'---------------------
'General connection
'---------------------
Global cn                   As New ADODB.Connection
'---------------------
'For student
'---------------------
Global frm_stud_show        As Boolean

Global sds, sms, sfs, sns, sos  As Byte
```

```
Global rs_stud              As New ADODB.Recordset
'--------------------
'For level
'--------------------
Global rs_level             As New ADODB.Recordset
'--------------------
'For School Year
'--------------------
Global rs_sy                As New ADODB.Recordset
'--------------------
'For Sections
'--------------------
Global rs_sec               As New ADODB.Recordset

'--------------------
'For printing
'--------------------
Global rpt_header As report_header


Public Sub set_conn_getData(ByRef sConnection As ADODB.Connection, ByVal
sDataLocation As String, ByVal sHavePassword As Boolean, ByVal sPassword As String)
If sHavePassword = True Then
   sConnection.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & sDataLocation &
";Persist Security Info=False;Jet OLEDB:Database Password=" & sPassword
Else
   sConnection.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & sDataLocation &
";Persist Security Info=False"
End If
End Sub
Public Sub set_rec_getData(ByRef sRecordset As ADODB.Recordset, ByRef sConnection As
ADODB.Connection, ByVal sSQL As String)
With sRecordset
   .CursorLocation = adUseClient
   .Open sSQL, sConnection, adOpenKeyset, adLockOptimistic
End With
End Sub


Public Function rec_found(ByRef sRecordset As ADODB.Recordset, ByVal sField As String,
ByVal sFindText As String) As Boolean
'-----------------------------------------------
'Move the recordset to the first record
'-----------------------------------------------
sRecordset.Requery '[ Use this instead of movefirst so that new record added can be used
immediately ]
'Search the record
sRecordset.Find sField & " = '" & sFindText & "'"
'-----------------------------------------------
```

```
'Verify if the search string was found or not
'-------------------------------------------------
If sRecordset.EOF Then
   rec_found = False
Else
   rec_found = True
End If
End Function
Public Sub record_login(ByVal sTimeLogin As Date, ByVal sUserName As String)
On Error Resume Next
'-------------------------------------------------
'Declare variables
'-------------------------------------------------
Dim rs_user_log As New ADODB.Recordset
Dim conn_user_log As New ADODB.Connection
'-------------------------------------------------
'Set the variables to have connection to database
'-------------------------------------------------
Call set_conn_getData(conn_user_log, App.Path & "\MasterFile.mdb", True, "reg386")
Call set_rec_getData(rs_user_log, conn_user_log, "Select * From UsersLog")
With rs_user_log
   .AddNew
      .Fields(0) = sUserName
      .Fields(1) = sTimeLogin
   .Update
End With
'-------------------------------------------------
'Clear variables
'-------------------------------------------------
Set rs_user_log = Nothing
Set conn_user_log = Nothing
End Sub
Public Sub record_logout(ByVal sTimeLogin As Date, ByVal sUserName As String)
On Error Resume Next
'-------------------------------------------------
'Declare variables
'-------------------------------------------------
Dim rs_user_log As New ADODB.Recordset
Dim conn_user_log As New ADODB.Connection
'-------------------------------------------------
'Set the variables to have connection to database
'-------------------------------------------------
Call set_conn_getData(conn_user_log, App.Path & "\MasterFile.mdb", True, "reg386")
Call set_rec_getData(rs_user_log, conn_user_log, "SELECT UsersLog.Username,
UsersLog.[Log-in], UsersLog.[Log-out] From UsersLog WHERE (((UsersLog.Username)='"
& sUserName & "') AND ((UsersLog.[Log-in])='" & sTimeLogin & "'))")
With rs_user_log
      .Fields(2) = Now
   .Update
End With
```

```
MsgBox sUserName & " has been sucessfully log-out.", vbInformation, "Log-out Time: " &
sTimeLogin
'Clear variables

Set rs_user_log = Nothing
Set conn_user_log = Nothing
End Sub

Option Explicit
Public Type report_header
   SchoolName As String
   SchoolAddress As String
   SY As String
   SectionName As String
End Type
Private Sub DataReport_Initialize()

With Me.Sections("Section1").Controls
   .Item("Label1").Caption = rpt_header.SchoolName
   .Item("Label2").Caption = rpt_header.SchoolAddress
   .Item("Label3").Caption = rpt_header.SY

   .Item("Label18").Caption = rpt_header.SchoolName
   .Item("Label20").Caption = rpt_header.SchoolAddress
   .Item("Label21").Caption = rpt_header.SY
End With

End Sub
Private Sub DataReport_Initialize()
With Me.Sections("Section1").Controls
   .Item("Label25").Caption = rpt_header.SchoolName
   .Item("Label26").Caption = rpt_header.SchoolAddress
   .Item("Label27").Caption = rpt_header.SY
End With
End Sub
Private Sub DataReport_Initialize()
With Me.Sections("Section2").Controls
   .Item("Label25").Caption = rpt_header.SchoolName
   .Item("Label26").Caption = rpt_header.SchoolAddress
   .Item("Label27").Caption = rpt_header.SY
   .Item("Label30").Caption = rpt_header.SectionName
End With
End Sub
```

CONCLUSION

I believe that Student information systems is important because it set the "easly" for student registration and option. I collected information and notes and firstly began with page design.After I searched the visual basic program and what I will be able to do by this program

The design of database system is code phase and the conclusion of the project and testing. I perceived to use the Visual Basic programming language more effective. I began to use the data report feature more effectively.We have learnt that without standards some aspects of computing would not work, without the good programs.. At the time of visual basic programming development several other commercial companies had been developing their own protocols.

REFERENCES

[1] Adolfo Rodriguez ,John Gatrell ,John Karas ,Roland Peschke

[2] Cisco Systems, Inc

[3] Near East University Library

[4] Microsoft Corporation, Inc

http://www.microsoft.com

[5] Mr.Ümit İlhan

[6] Memik Y.[2002]Microsoft Visual Basıc for Windows 98/me/2000/xp Profesyonel Sürüm(BETA Basım Yayım Dağıtım A.Ş)

[7] Pala, Z[2004]Herkes İçin Visual Basıc (Akçağ Basım Yayım Dağıtım A.Ş)