



NEAR EAST UNIVERSITY

**INSTITUTE OF APPLIED
AND SOCIAL SCIENCES**

**STUDENT MANAGEMENT SYSTEM USING
VISUAL BASIC AND ASP**

Ibrahim Elnour

Final Project

Department of Computer Engineering

Nicosia- 2003

NEAR EAST UNIVERSITY

**INSTITUTE OF APPLIED
AND SOCIAL SCIENCES**

**STUDENT MANAGEMENT SYSTEM USING
VISUAL BASIC AND ASP**

IBRAHIM ELNOUR

Final Project

Department of Computer Engineering

Nicosia- 2003

ACKNOWLEDGEMENTS

*“First, thank **Allah** for what ever he gives.*

*Second I would like to thank my supervisor Mr. UMIT ILAHAN for his valuable advice,
encouragement and endless support.*

*Third, I would like to acknowledge special thank to the Near East University for
offering me a suitable environment during my study.*

*I would like to dedicate my research to my family who are always motivating me with
all their love.*

*I gratefully acknowledge the role of my friends those who sat behind me during the hard
times.*

*Finally, Special thank to the Jury members who offering me this great
opportunity to present my project”*

ABSTRACT

In the practice many database applications have been implemented through the internet, the internet is reducing the time taken to reach your database and made it available on demand.

The main aim here is to make an easy to use interface for student and courses relationship and to make the information available in the internet.

Ente

TABLE OF CONTENTS

ACKNOWLEDGEMENT.....	i
ABSTRACT.....	ii
TABLE OF CONTENTS.....	iii
CHAPTER ONE	
INTRODUCTION TO DATA DESING.....	1
1.1 Overview.....	1
1.2 Why design.....	1
1.3 Problems Resulting from Poor Design.....	1
1.4 Domain/Key Normal Form	2
1.5 Dependency.....	2
1.6 Domain.....	2
1.7 Restriction.....	2
1.8 The Normalizing Process.....	3
1.9 A Method of Database Design.....	3
1.10 Design of the Database Model.....	3
1.11 Summary.....	8
CHAPTER TWO	
DATABASES USING VISUAL BASIC.....	9
2.1 Introduction.....	10
2.2 The Data Control.....	10
2.3 Steps to creating a data aware application	12
2.4 The Record Set	12
2.5 Microsoft Visual Data Tools	16

CHAPTER THREE

DATABASES USING ASP.....	19
3.1 The need for ASP.....	19
3.2 What is ASP ?.....	20
3.3 Active Server Pages 3.0.....	21
3.4Running ASP.....	22
3.5 Beginning ASP	23
3.6 Accessing your webpage	23
3.7 The Database.....	25
3.8 Summary.....	26

CHAPTER FOUR

STUDENT DATABASE SYSTEM.....	27
4.1 Overview.....	27
4.2 Designing the Tables.....	28
4.3 The User Interface.....	32
4.4 The Internet View.....	43
4.5 Summary.....	47
 REFERENCES.....	 48
APPENDIX A.....	49
APPENDIX B.....	65

CHAPTER ONE

INTRODUCTION TO DATABASE DESIGN

1.1 Overview

Database design is a complex subject, no matter how easy some people think it is. A properly designed database is a model of a business, or some other in the real world. Like their physical model counterparts, data models enable you to get answers about the facts that make up the objects being modeled. It's the questions that need answers that determine which facts need to be stored in the data model.

In the relational model, data is organized in tables that have the following characteristics: every record has the same number of facts; every field contains the same type of facts in each record; there is only one entry for each fact; no two records are exactly the same; the order of the records and fields is not important.

At the end of this chapter, we have a basic understanding of problems resulting from poor database design, be familiar with the Domain/Key normal form, understand a process for designing a relational database, and be aware of the tools used in Microsoft Access® to support integrity constraints in a database.

1.2 Why Design?

Accurate design is crucial to the operation of a reliable and efficient information system. Microcomputer technology is now so advanced that the impact of a poor design may not show up as early as in the past; however, when the problems appear they can be severe.

Although Microsoft Access and Microsoft® FoxPro® are powerful and easy to use, they have historically not lent themselves well to ad hoc design.

The design of a database has to do with the way data is stored and how that data is related. The design process is performed after you determine exactly what information needs to be stored and how it is to be retrieved.

The more carefully you design, the better the physical database meets users' needs. In the process of designing a complete system, you must consider user needs from a variety of viewpoints.

1.3 Problems Resulting from Poor Design

A number of problems can be shown as a result of poor database design:

- The database and/or application may not function properly.
- Data may be unreliable or inaccurate.
- Performance may be degraded.
- Flexibility may be lost.

1.4 Domain/Key Normal Form

Relational theorists have classified database schemas that have inconsistencies based on the anomalies to which they are susceptible. You may have encountered discussions about different forms, such as first normal form or third normal form. One of the unique normalization forms was proposed by R. Fagin, in 1981, and is used as the basis of this presentation. Fagin ascertained that if the tables in your database are in Domain/Key Normal Form (D/KNF), then they are free of modification anomalies.

To understand D/KNF there are four terms that must be understood: dependency, key, domain, and restriction.

1.5 Dependency

A dependency is a relationship that may exist between two columns. Given the value of one column, we are able to determine the value of another column.

Most tables should have a column or a combination of columns that uniquely identifies a row of data. A column is key if all other columns in a row are dependent on it.

1.6 Domain

A domain is the set of values a column can have. Every column has a domain, which has both physical and logical properties.

Physical Description. The physical part of a domain is the type of information about that column.

Logical Description. The logical part of the domain is the set of information associated with that fact.

1.7 Restriction

A restriction is a limitation of some type on the values in a table. A dependency is a type of restriction. When a column is a key, it means that all other columns in that table are dependent on the key. Remember that a key can be a combination of columns.

A domain is another type of restriction. When defining the physical and logical properties of a column, we restrict the data in that column.

Restriction is a general term. There are many other ways to restrict data in a table. Below are some examples:

1.8 The Normalizing Process

Normalizing the database ensures that the structure of the database allows changes to be made without incurring unexpected consequences. The role of normalization is to maintain stable, reliable data through good database design.

The goal of good database design is to ensure that all restrictions are logical consequences of domain and key restrictions.

Tables, like paragraphs, should have a single theme.

1.9 A Method of Database Design

As we have seen, database design plays a major role in the stability and the reliability of the data. In this section, we see the process of designing a database.

Although there are a number of rules that can be followed in designing a database structure, the design process is as much an art as it is a science. Follow these rules when at all possible, but not to the point where the database loses the functionality that is so important to the user.

Doing a paper design first has several advantages:

- Saves time, money, and problems
- Makes system more reliable; avoids potential data-modification problems
- Serves as a blueprint for discussion
- Helps in estimating costs and size

A good design should have the following objectives:

- Meet the users' needs
- Solve the problem
- Be free of modification anomalies
- Have a reliable and stable database, where the tables are as independent as possible
- Be easy to use

1.10 Design of the Database Model

The design of the database structure requires the following steps:

1. List the objects.
2. List the facts about the objects.
3. Turn the objects and facts into tables and columns.

4. Determine the relationship among objects.
5. Determine the key columns.
6. Determine the linking columns.
7. Determine the constraints.
8. Evaluate the design model.
9. Implement the database.

Step 1: List the Objects

Make a list of all objects. An object is a single theme, similar to a paragraph. At Student database the objects are:

Students
Courses
Term or
semester

Step 2: List the Facts about the Objects

There is a great deal of information associated with every object. In this step, you should list the facts about an object and then eliminate the facts that are not important to the solution of the problem. The Student object, for example, can have many facts associated with it: student name, address, nationality, date of birth, passport no and others.

Step 3: Turn the Objects and Facts into Tables and Columns

Objects automatically become tables, and facts become columns once the column domains are determined. Recall that a domain is a set of values that a column can have. Every column has a domain, which has both physical and logical properties. For example, the column for student name is defined as TEXT 15. TEXT 15 is the physical property of the column. Because of this definition, its domain is the set of all student names with 15 characters or less.

If a column is used to link two or more tables, the domains must be the same and the columns should be given the same name. If the logical description differs (for example, student name and course name), the columns are not the same and should not share the same name.

Often it helps in the design stages to draw boxes to represent the tables. In later steps you can then fill in key columns and draw the relationships among the tables.

Step 4: Determine the Relationship Among Objects

To determine the relationship among the objects, take each object and look at how that object may be related to another. Keep in mind that not every relationship existing between objects is important. The relationships that are important are those that allow to model the database after the real-world situation that the database represents.

One-to-one relationships. For any given row in Table A, there is only one row in Table B. For any given row in Table B, there is only one row in Table A. There are no one-to-one relationships in the student database. An example of a one-to-one relationship is that of employee data and private employee data. General information, such as employee name, address, and start date, is kept in one table, and to ensure privacy, personal information, such as salary, is kept in another table.

One-to-many relationships. For any given row in Table A, there are many rows in Table B. For any given row in Table B, there is only one row in Table A. The relationship between an employee and an employee's dependents is one-to-many, because one employee may have many dependents, but a dependent is related to only one employee.

Many-to-many relationships. For any given row in Table A, there are many rows in Table B. For any given row in Table B, there are many rows in Table A. There is a many-to-many relationship between the student table and the course table. A student can take many different courses and a course can be associated with many different students.

The first step in determining the type of relationship between tables is to list every table and to see how it relates to any others:

An effective method to find the type of relationship is to ask whether a specific record in Table A can point to (is linked to) one or to many rows in Table B, and then reverse the tables and ask the question again.

Step 5: Determine the Key Columns

A key can be an account number, social security number, part number, license number, or any other numeric value or combination of characters that are unique. A complex key is one that is derived from more than one column. Microsoft Access supports complex keys directly.

No other row in the table can have the value of the key column(s). Other tables may share the same set of key information. If a company name is universally unique, it is used as a unique row identifier. However, if there is any possibility another company could have the same name, then it is not unique and must not be employed as a key column. Do not use any column as a key where the possibility exists for a duplicate. A key column cannot contain null values.

By definition, all key columns should be indexed.

Because text names are usually not unique and cannot be used in math operations, it is useful to make key columns a sequential numeric value. In many cases, it is easier to develop your own unique row identifier.

Step 6: Determine the Linking Columns

If you have been careful about designating key columns, you also have determined the linking columns. Links provide a way to tie information (rows) in one table to another table. If a table has a key column, that column can generally serve as the link. Tables are linked together through their key columns. However, the placement of the key is important, and where the link is placed depends on the type of relationship between the tables.

To determine the placement of the links, you must first know the type of relationship among the objects or tables. Once you know the type of relationship among tables, it is much easier to determine where to place the linking column to tie two tables together.

Note that not all tables need to be linked relationally.

Linking in a one-to-one relationship. In one-to-one relationships the link should be the most stable column or should be from the table where the key column is created. The most stable is the column least likely to change. If an automatic numbering system is being used, then use that column as the linking column.

Linking in a one-to-many relationship. In one-to-many relationships the linking column should come from the one table. The key column from the student table (one side) should be placed in the details table (much side). When the key stno is placed in the details table, it is referred to as a foreign key in the details table.

Linking in a many-to-many relationship. The many-to-many relationship causes problems when attempting to retrieve data and when relating a value in one table to its corresponding value in the other table. It is important to understand this relationship to be able to recognize and control this situation when it arises.

A classic many-to-many relationship is students and courses. A student can be an item on many different courses and a course can have many students associated with it.

But which key will we use for a link? If stno is placed in the course table, then all of the course data would have to be repeated for each course that have been taken by course. If cno is placed in the student table, then the student information has to be repeated for each course taken by the student. This leads to redundant data, and the potential for invalid data is increased. Performance may suffer.

The solution to many-to-many relationships is to create an intersection table. This table should contain the key columns from both tables.

Step 7: Determine the Relationship Constraints

Often the information we get from a database comes from more than one table. For example, if we want to know who the parent of a particular dependent is, the name is determined by using the value in stno to look up the correct row in the student table. The question of who the parent is can be answered only if there is a row in the student table with an stno value corresponding to that in the dependent table.

To ensure the integrity of the data in our database, our model should require, for example, that no row can be added to the dependent table, unless there is already a corresponding row in the master table. This requirement is known as a relationship constraint. In this case, a constraint must exist on the dependent table that ensures that the master (parent) exists.

There are at least four methods to implement relationship constraints:

- Built-in controls in the DBMS
- Data entry and access procedures
- Programming
- Implementation of rules

Microsoft Access has certain referential integrity constraint mechanisms built into the engine.

In Microsoft Access, rules at the database or form level can be employed to enforce column domains (for example, accept values less than 200, or text value must be F or M) or in any other operation where you want a data entry test to be performed.

Step 8: Evaluate the Design

The next step in the design process is the evaluation of the design. In this step, you should look for any design flaws that could cause the data to be unreliable, unstable, or redundant.

Every table should be evaluated by asking the following questions:

1. Does each table have a single theme? It should. Each column should be a fact about the key.
2. Does each table have a key column(s)? It should.
3. Are there any dependencies? Only logical consequences of the key should exist.
4. Are the domains unique among tables? Do not mix domains unless the column is common between tables.
5. Are the restrictions domain or key?

6. Is the table easy to use?

Step 9: Implement the Design

Once the database had been designed on paper, the next step is to implement the design in Microsoft Access. When defining tables in Microsoft Access, it is extremely important to keep your paper design in mind. Designing a database on the fly can cause problems that may be quite difficult to recover from. (Remember the anomalies earlier in this chapter.)

In Microsoft Access, there are two tools that will help you complete the implementation of your design. The Table Wizard can be used to generate a variety of common tables. The graphical system relationships window can be used to set up relationships and key dependencies. Note, that while using the Table Wizard ensures proper relational design, the eight steps prior to implementation remain important and should be completed prior to constructing the tables.

1.11 Summary

By following the nine-step design process, the problems of data redundancy, changing multiple occurrences of data, and deletion and insertion anomalies can be avoided. It is well worth the time spent in the design process to ensure a reliable and flexible system.

Design to the point where redundancy is eliminated or controlled. As we design our database, we keep in mind the following list of common database errors to avoid:

- Trash-table-putting everything in the same table
- No unique row identifier (key column or columns)
- No linking or common columns
- Mixing logical and physical descriptions of domains
- Putting the linking column in the wrong table
- Restrictions not enforced
- Many-to-many relationships without intersecting tables.

CHAPTER TWO

DATABASES USING VISUAL BASIC

2.1 Introduction

Visual Basic incorporates the same database engine that powers Microsoft Access.

This enables Visual Basic to access databases in a number of standard formats in a relatively simple manner.

Database formats supported are:

- Microsoft Access
- dBASE
- Microsoft FoxPro
- Paradox
- Btrieve

In addition, the Professional Edition of VB 6.0 can use ODBC to access even more types of database, including client/server DBMSs such as Oracle and INGRES.

With ODBC databases you can also pass SQL commands through to the external server for processing.

In all of the examples we will see, the database will be a Microsoft Access database.

In all of the examples we will see, the database will be accessed via a Visual Basic control known as the "data control".

In the VB 6.0 Learning Edition, the data control is the **ONLY** way to access a database.

In the Professional Edition of VB we can access databases directly from code, without using a data control.

2.2 The Data Control

The data control is available on the standard toolbar.

After placing a data control on your form, you can edit its properties, just like any other control. You can also set them from code at run time.

The important properties of the data control, from the point of view of data access, are:

- Connect - (not used when accessing Access databases)
- DatabaseName
- Exclusive
- Options
- ReadOnly
- RecordSource

DatabaseName:

Determines the path and filename of the database file (.MDB file for Microsoft Access).

Exclusive:

If this property is set to True, then when you open the database you will have exclusive access to it. If set to False (the default) other users (or programs) can access the database while you are working on it. If you are working on a single user database, set this property to true - it will improve performance.

Options:

Determines access constraints for the tables you are working on.

ReadOnly:

When True, you cannot update the database (default is False).

RecordSource:

Determines the source of the set of records (the RecordSet) retrieved by the data control. This property will be set either to the name of one of the tables in the database, to the text of a valid SQL query, or to the name of a predefined query in the Access database.

2.2.1 The RecordSet

When you run the program, the data control retrieves a set of records from the database.

The set of records retrieved is determined by the RecordSource property.

This set of records is placed in an object called the RecordSet, attached to the data control.

The buttons on the data control allow you to browse forwards and backwards through the records in the RecordSet.

You can also refer to the RecordSet in your code, but more of that later.

2.2.2 Viewing the data with the Bound Controls

So far we have seen how to open a database and retrieve some data, but we haven't seen how to display the data yet!

The easiest way of viewing the data is by using one or more of the following 'bound controls' (available in the learning edition).

- Check box - Use for True/False data
- Image - Use for image data
- Label - Use for "non user-editable" data
- Picture box - Use for bitmaps
- Text box - Use for "user-editable" data

For now we will just use the Text box control - the others work in a similar way.

The bound controls all have the properties DataSource, DataField and DataChanged

The DataSource property should be set to the name of the data control.

The DataField property should be set to the name of the field whose data you want to see.

DataChanged (available at run time only) is set to True by VB when a value displayed in the control has changed.

2.3 Steps to creating a data aware application

1. Draw a data control on your Form
2. Set its DatabaseName property to the path & filename of your database
3. Set the RecordSource property to the name of the database Table you want to access
4. Draw One TextBox on your Form for each Field you want to access
5. For each TextBox
 - o set its DataSource property to the name of the data control
 - o set its DataField property to the name of the field you want that TextBox to access
6. Run the program

2.4 The RecordSet

Recall that the RecordSet is simply the table created by the data control, based on the contents of its RecordSource property.

Depending on the RecordSource property, this table may be:

- one of the tables from the database
- a subset of records from such a table
- a subset of the fields from such a table
- a join of two or more tables from the database
- in fact, anything which can be returned by an SQL query

Just like other tables in a relational database, each row of the RecordSet is referred to as a record, and each column is referred to as a field.

The RecordSet can be referred to in your code as a property (available at run time only) of the data control, as follows:

```
data1.RecordSet
```

However, you never refer to the RecordSet without either getting one of its properties - e.g.

```
data1.RecordSet.BOF
```

or applying one of its methods - e.g.

```
data1.RecordSet.MoveFirst
```

2.4.1 Navigating the RecordSet

The data control maintains a pointer to one record from the RecordSet. The record being pointed to at any given moment is called the "current record". There are four methods which you can apply to the RecordSet to move the current record pointer, these are:

- MoveFirst - makes the first record the current record
- MoveLast - makes the last record the current record
- MoveNext - moves the current record pointer forwards
- MovePrevious - moves the current record pointer backwards

e.g.,

```
data1.RecordSet.MoveFirst
```

When the current record pointer moves, any bound controls will be updated with field values from the new current record.

When the current record pointer is moved beyond the last record in the RecordSet, the EOF (End Of File) property is set to True

When the current record pointer is moved before the first record in the RecordSet, the BOF (Beginning Of File) property is set to True

It follows that when either EOF or BOF is True, the current record is invalid. When this happens the bound controls will be cleared, and you should not try to access any fields of the (invalid) current record in your code.

Typical code for working through each record in the RecordSet would look like this:

```
data1.RecordSet.MoveFirst
Do While Not (data1.RecordSet.EOF)
    '
    'code to process the current record
    '
    data1.RecordSet.MoveNext
Loop
```

Another useful property of the RecordSet is Bookmark. Bookmark is, in effect, the current record pointer. You can save the value of the Bookmark property, and use it later to move directly to that record, as follows:

```
Dim saveBookmark As String
saveBookmark = data1.RecordSet.Bookmark
' do processing which moves the current record pointer
data1.RecordSet.Bookmark = saveBookmark 'Restore the current record pointer
```

2.4.2 Modifying the database through the RecordSet

The RecordSet has a method - *AddNew* - which appends a new, empty record to the end of the RecordSet.

```
data1.RecordSet.AddNew
```


When this is executed, any bound controls are cleared.

Data can be entered into the record via the bound controls, or from code.

The database will be updated with the new data when either the current record pointer is moved, or the RecordSet's Update method is invoked.

```
data1.RecordSet.Update
```

Note that when attempting to update the database, an error will occur if the data is invalid in any way, according to rules specified in the database.

Records can be deleted from the RecordSet using the Delete method

```
data1.RecordSet.Delete
```

When this statement is executed the record is deleted immediately - it does not require the use of the Update method.

The current record pointer is not changed however, and the Bound Controls are not cleared. The current record is therefore now invalid, and any attempt to access it will cause an error.

It is therefore a good idea always to follow the Delete method with a method to move the current record pointer - usually MoveNext.

```
data1.RecordSet.Delete
```

```
data1.RecordSet.MoveNext
```

2.4.3 Accessing field values via the RecordSet object

If you want to update field values from code, rather than by the user entering the data into bound controls, you can do this with the RecordSet object.

e.g.,

```
data1.RecordSet("Surname")
```

refers to the field called Surname of the current Record.

You can therefore retrieve the value of the field with code such as

```
Dim lastName As String  
lastName = data1.RecordSet("Surname")
```

and you can edit it with code such as

```
data1.RecordSet.Edit  
data1.RecordSet("Surname") = lastName  
data1.RecordSet.Update
```

Note that you must first invoke the *Edit* method.

Note also that changes made in this way are not written to the database until the the update method is invoked.

Fields can also be referred to by their position within the table, with the first field being given the number 0 (zero).

e.g., if Surname was the first field in the table ...

```
data1.RecordSet(0) = lastName
```

would have the same effect as the previous example.

2.5 Microsoft Visual Data Tools

Using Visual Basic 6.0 you can create components that encapsulate every step in a data access system. Beginning with the data source, Microsoft Visual Data Tools (accessible through the Data View window) give you the ability to view and manipulate tables, views, stored procedures, and database schemas on SQL Server and Oracle systems.

2.5.1 Middle Tier Components and Microsoft Transaction Server

The power of Visual Basic is also leveraged to create the middle tier components in your application, as you make your own ActiveX DLLs and EXEs. Visual Basic now includes enhancements that tailor applications to work with Microsoft Transaction Server.

2.5.2 ActiveX Data Objects (ADO)

The bridge between the data providers and data consumers is through data sources created using Microsoft ActiveX Data Objects (ADO), which is the primary method in Visual Basic to access data in any data source, both relational and non-relational. For backward compatibility and project maintenance, Remote Data Objects (RDO) and Data Access Objects (DAO) are still supported.

2.5.3 Data Sources and Data Controls

On the client side, several new data sources are available, including the Data Environment, a graphical designer that allows you to quickly create ADO Connections and Commands to access your data. The Data Environment designer provides a dynamic programmatic interface to the data access objects in your project. In addition, the Data Environment provides advanced data shaping services — the ability to create hierarchies of related data, aggregates, and automatic groupings, all without code.

The new ADO Data control is similar to the intrinsic data control and Remote Data control, except that it uses ADO to access data. You can now use an ADO Recordset as a data source for your controls and objects in Visual Basic.

In Visual Basic you can now create your own data sources either as user controls or classes, to encapsulate business rules or proprietary data structures. The class module now features the DataSourceBehavior property and the GetDataMember event, which allow you to configure a class as a data source.

2.5.4 Dynamic Data Binding

The ability to dynamically bind a data source to a data consumer is now possible in Visual Basic. At run time, you can now set the DataSource property of a data consumer (such as the DataGrid control) to a data source (such as the ADO Data control). This capability, unavailable in previous versions of Visual Basic, allows you to create applications, which can access a multitude of data sources.

2.5.5 Presenting Data to the End User

Visual Basic offers a variety of rich ways to present data to your end users. ADO/OLE DB-based versions of all the data bound controls are included in Visual Basic:

- The DataList and DataCombo controls are the ADO/OLE DB equivalents of DBList and DBCombo controls.
- The DataGrid is the successor to DBGrid.
- The Chart control is now data bound.
- A new version of the FlexGrid control, called the Hierarchical FlexGrid, supports the hierarchical abilities of the Data Environment.
- The new DataRepeater control functions as a scrolling container of data bound user controls where each control views a single record.

The Data Report is a new ActiveX designer that creates reports from any data source, including the Data Environment. With the Data Report designer, formatted reports can be viewed online, printed, or exported to text or HTML pages.

2.6 Summary

In this chapter we are considering the Visual Basic environment for database programming and the use of Visual Basic to connect, enter and manipulate data using the data control, data environment and data report. The use of other Active X components is presented.

CHAPTER THREE

DATABASES USING ASP

3.1 The need for ASP

Microsoft's Active Server Pages (ASP) with IIS 3.0 offers the web developer flexible, easy to use, scalable methods to interact with ODBC compliant databases for an Internet site or Intranet application. In this article the basic methods that are needed to interact with a database are illustrated - namely, adding, editing and deleting records.

Using ASP highly interactive pages can be developed independent of the type of browser that will be used to access these pages - from Lynx to Internet Explorer 3.0. ASP encompasses the capabilities of both JavaScript and VBScript with the added bonus that components can be easily added to extend the Internet/intranet application. Using ASP as part of your development not only means that you can initially develop in Microsoft's Access and scale up to a Microsoft SQL Server 6.5 database; but that you can access other vendor databases that are ODBC compliant. Its faster than using Visual Basic and the WinCGI interface - it will be interesting to compare performance with IDC and the use of an ISAPI filter to access a ODBC database. Needless to say, anyone who likes programming in Visual Basic is going to have a ball using ASP.

In addition using Chili!ASP the functional equivalent of Microsoft's Active Server engine, can be used on a range of NT based Web servers, including Netscape, Lotus, O'Reilly's, and some UNIX servers.

On the other hand ASP lacks the platform portability that PERL (Note that with advent of Chili!ASP its not true anymore), enjoys along with vast resources available to PERL programmers on the Internet but its is much easier to learn and develop in. When this The exception handling in VBScript leaves a bit to be desired - which would be critical if say there was an error inserting data into a database. I did not use the debugger in the development of the code and found that most of the run time errors were due to the fact

that I had variables spelled wrongly or I did not include the "=" sign as part of a variable when it was embedded in HTML.

The code is to be used as a reference example, not a robust application. Conditions such as trying to delete or edit records when there are no records in the database have not been dealt with. The code was developed on Windows NT 4.0, with MS Access 7 as the database. You will need the 32 Bit ODBC Drivers for Microsoft Access 7.

To illustrate how you can put ASP to work on your web pages I am going to show you how to use ASP to interact with a database that contains user information. The example covers the basic methods that would be needed by anyone working with a database. You will be able to add, edit and delete entries into this database.

Why bother with ASP at all, when HTML can serve your needs? If you want to display information, all you have to do is fire up your favorite text editor, type in a few HTML tags, and save it as an HTML file. Bingo, you're done! But wait – what if you want to display information *that changes*? Supposing you're writing a page that provides constantly changing information to your visitors, for example, weather reports, stock quotes, a list of your girlfriends, etc, HTML can no longer keep up with the pace. What you need is a system that can present dynamic information. And ASP fits the bill perfectly.

3.2 What is ASP ?

In the language of Microsoft, Active Server Pages is an open, compile-free application environment in which you can combine HTML, scripts, and reusable ActiveX server components to create dynamic and powerful Web-based business solutions. Active Server Pages enables server side scripting for IIS with native support for both VBScript and JScript.

Active Server Pages (ASPs) are Web pages that contain *server-side scripts* in

addition to the usual mixture of text and HTML tags. Server-side scripts are special commands you put in Web pages that are processed before the pages are sent from the server to the web-browser of someone who's visiting your website. When you type a URL in the Address box or click a link on a webpage, you're asking a web-server on a computer somewhere to send a file to the web-browser (also called a "client") on your computer. If that file is a normal HTML file, it looks the same when your web-browser receives it as it did before the server sent it. After receiving the file, your web-browser displays its contents as a combination of text, images, and sounds. In the case of an Active Server Page, the process is similar, except there's an extra processing step that takes place just before the server sends the file.

3.3 Active Server Pages 3.0

Before the server sends the Active Server Page to the browser, it *runs* all server-side scripts contained in the page. Some of these scripts display the current date, time, and other information. Others process information the user has just typed into a form, such as a page in the website's guestbook. And you can write your own code to put in whatever dynamic information you want. To distinguish Active Server Pages from normal HTML pages, Active Server Pages are given the ".asp" extension.

3.3.1 What Can We Do with Active Server Pages?

There are many things you can do with Active Server Pages.

- You can display date, time, and other information in different ways.
- You can make a survey form and ask people who visit your site to fill it out, send emails, save the information to a file, etc
- You can have a database which people can access via the web.
- People can get information from the database as well as update or insert information into it.

- You can password-protect certain sections of your site, and make sure that only authorized users can see that information.
- The possibilities are virtually endless. Most of that you see on web pages nowadays can be easily done using ASP.

3.3.2 Server-Side Scripts

Server-side scripts typically start with `<%` and end with `%>`. The `<%` is called an *opening tag*, and the `%>` is called a *closing tag*. In between these tags are the server-side scripts. You can insert server-side scripts anywhere in your webpage - even inside HTML tags.

3.4 Running ASP

Since the server must do additional processing on the ASP scripts, it must have the ability to do so. The only servers which support this facility are Microsoft Internet Information Services & Microsoft Personal Web Server. Let us look at both in detail, so that you can decide which one is most suitable for you.

3.4.1 Internet Information Services

This is Microsoft's web server designed for the Windows NT platform. It can only run on Microsoft Windows NT 4.0, Windows 2000 Professional, & Windows 2000 Server. The current version is 5.0, and it ships as a part of the Windows 2000 operating system.

3.4.2 Personal Web Server

This is a stripped-down version of IIS and supports most of the features of ASP. It can run on all Windows platforms, including Windows 95, Windows 98 & Windows Me. Typically, ASP developers use PWS to develop their sites

on their own machines and later upload their files to a server running IIS. If you are running Windows 9x or Me, your only option is to use Personal Web Server 4.0.

3.5 Beginning ASP

Here a few quick tips before you begin your ASP session!

Unlike normal HTML pages, you cannot view Active Server Pages without running a web-server. To test your own pages, you should save your pages in a directory mapped as a virtual directory, and then use your web-browser to view the page.

3.5.1 Steps for Installation

- From the CD, run the SETUP.EXE program for starting the web-server installation.
- After the installation is complete, go to Start > Programs > Microsoft PWS > Personal Web Manager. and click the "Start" button under Publishing.
- Now your web-server is up & running.

3.5.2 Creating Virtual Directories

After you have installed the web-server, you can create virtual directories as follows:

- Right-Click on the folder that you wish to add as a virtual directory.
- Select "Properties" from the context-menu.
- In the second tab titled "Web Sharing," click "Share this folder," then "Add Alias". (If you do not see these options enabled, your web-server is not properly running. Please see the steps above under "Installation.")

3.6 Accessing your webpage

Now that your server is completely configured and ready to use.

Start your web-browser, and enter the following address into the address-bar.

`http://localhost/`

You should see a page come up that tells you more about Microsoft IIS (or PWS, as the case may be)

3.6.1 Localhost

Let us first see, what we mean by a hostname. Whenever you connect to a remote computer using it's URL, you are in effect calling it by its hostname.

For example, when you type in

`http://www.google.com/`

you are really asking the network to connect to a computer named `www.google.com`. It is called the "hostname" of that computer.

`localhost` is a special hostname. It always references your own machine. So what you just did, was to try to access a webpage on your own machine (which is what you wanted to do anyway.) For testing all your pages, you will need to use `localhost` as the hostname. By the way, there is also a special IP address associated with `localhost`, that is

`127.0.0.1`

So you could as well have typed:

`http://127.0.0.1/`

and would have received the same page.

To access pages in a virtual directory called `myscripts` for example, you should type in:

`http://localhost/myscripts/`

in the address bar. I hope the concept is now clear ...

3.7 The Database

So lets start with the database - I used MS Access to develop the database .

3.7.1 DSN

Once you have designed you database the next step will be to create a DSN entry, UserDB1. To do this:

- Click on your "Start" Button, and go to Control Panel under Settings.
- Click on "32 ODBC", select "System DSN"
- Click "Add" to add a DSN entry, and then on "Microsoft Access Drive". If "Microsoft Access Driver" does not appear on the list, you possibly have not installed Microsoft Access 7's, 32 bit ODBC drivers.

3.7.2 Connecting to the Database

So far we have developed a basic database and added a DSN entry in order that the database be accessed using ODBC - nothing to really write home about. ASP offers two methods to access the database. In the first each access to the database would have first connect to the database; once the connection has been established SQL statements can be used to manipulate data; once completed, all related objects are closed. There are a number of illustrations using this technique in the samples provided with the Active Server Pages. The snippet of code illustrates a connection to a database with "ADOSamples" as the DSN, obtaining a recordset based on a SQL query. Once the script has done with the data, the recordset and the connection to the database are closed.

The first post-startup request is made to the web server for any *.asp file in an application causes the Global.asa to be read. So the moment a request is made to any *.asp in the directory in which the intranet application is stored a connection is established with the DSN UserDB1. Following that the default document, in this case default.asp is processed.

In ASP based applications the programming logic, variables and HTML can be maintained in a single file. Commonly used functions across an ASP application can be in one file, that be included in different pages using the "include" statement. With regards to the logic of the example I have used a simple state space model to determine the state of the ASP page - i.e. is an addition, deletion or update taking place or not. The information of the current state of the page is dictated by the contents of the form element

named "Action". The value the element "Action" is obtained from the form in VBScripts with the statement in..

For developers familiar with Microsoft's Internet related products ASP will possibly the way to go to develop intranet applications - especially if you are a Visual Basic Programmer. In a couple of months visual tools will be available that will give ASP a more robust development environment.

3.8 Summary

In this chapter the role of using ASP for web development of databases is described and the use of ASP for connecting, manipulating and managing records through the internet is shown.

CHAPTER FOUR

STUDENT DATABASE SYSTEM

4.1 Overview

After taking a look at the database design at first stages and the steps considering the usage of Visual Basic as a language of implementation and the way that ASP is used to view the database from the web we are going to demonstrate this in the performing of the student courses relationship governing the attitude of the system.

We are going to use Microsoft Access to design the tables of the system and define the relationships and for the querying. Visual Basic is the main system handling tool used for entering, viewing, searching, assigning and reporting the data used in this system.

ASP is used for serving students who want to view their grades in certain courses or their transcripts.

The functionality of this project is to:

- Enter the Students information
- Enter the Courses information.
- Assign Courses to Students taking into account the semester and year.
- Assign Grades to Students per Course.
- Search for Students.
- Search for Courses.
- Report the list of all students.
- Report the list of all the courses.
- Report the transcript.
- Viewing the transcript.
- View Grade through the Internet.
- View Transcript through the Internet.

4.2 Designing the Tables:

For the design of tables we are going to use Microsoft Access as a container for data we are using table design view to assign fields.

In our project we have 3 main tables, one query and a table used to store authentication data.

The first table is the Students table containing the information about student as follows

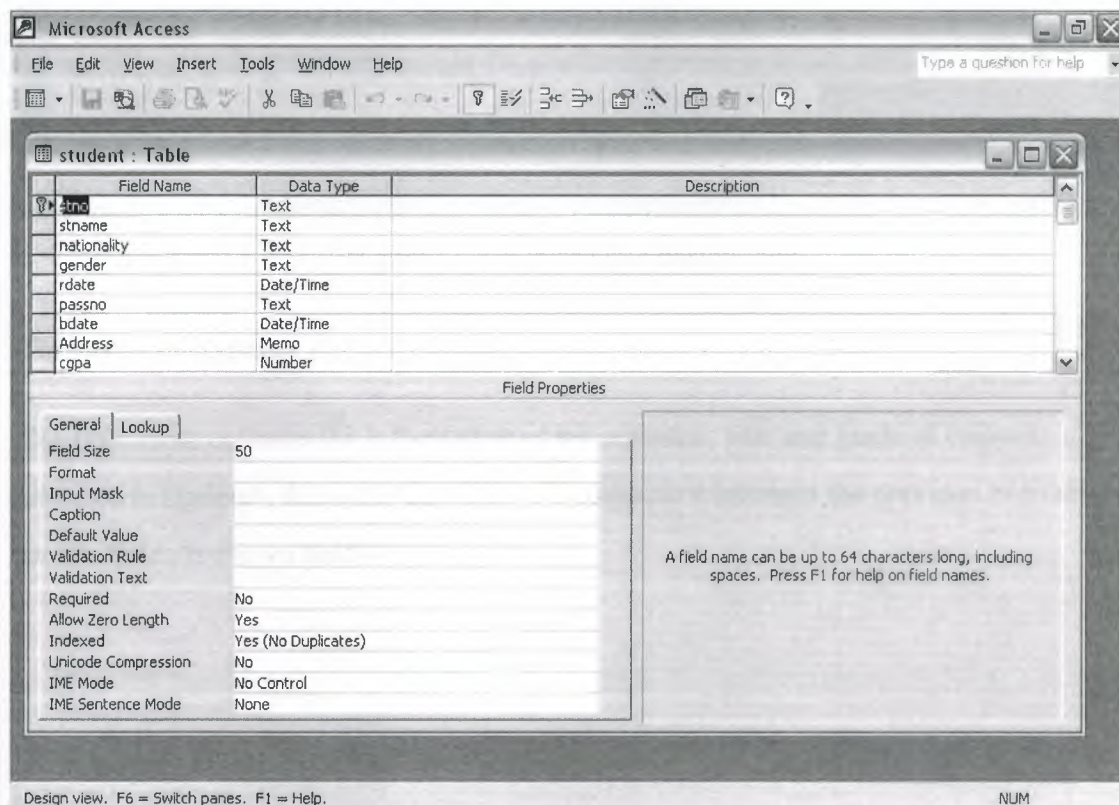


Figure 4.1 Student table design view

The primary key is **stno**.

The second table contains the information about Course as shown below:

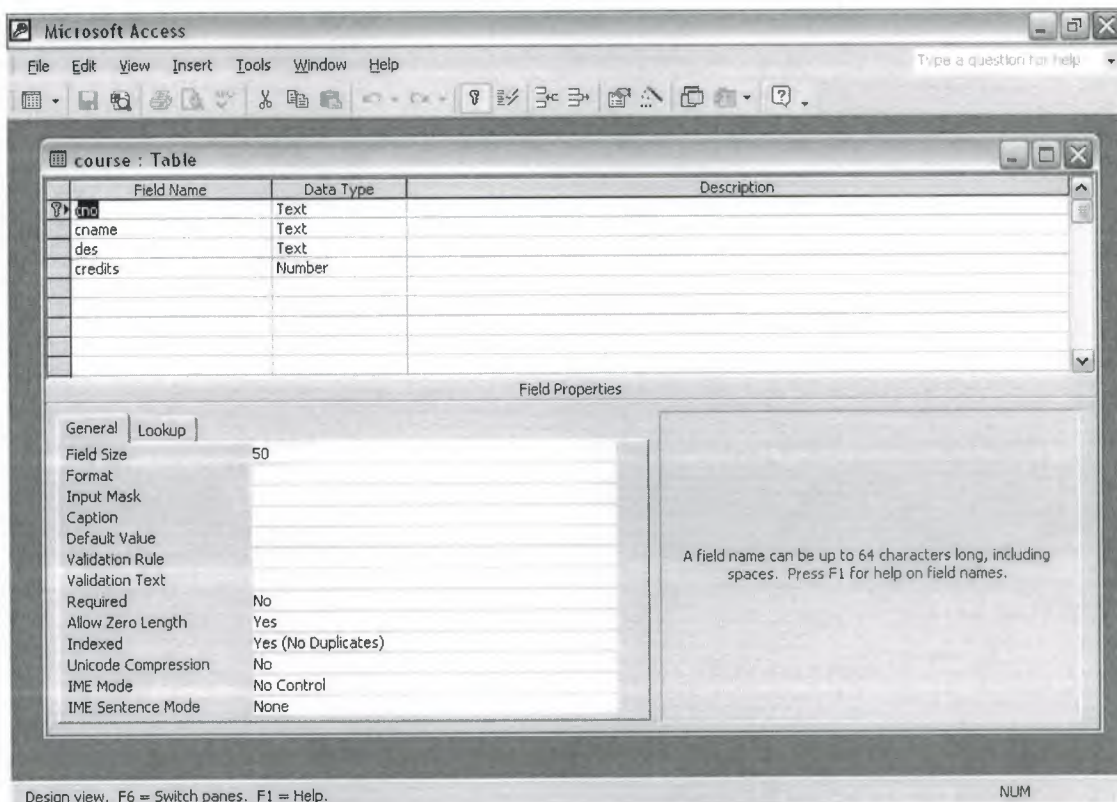


Figure 4.2 Courses table design view

The **cno** is the primary key of this table.

The third table contains the information of the semester, year and grade of courses assigned to students, this table **det** acts as a connection between the previous two tables and its fields is shown below:

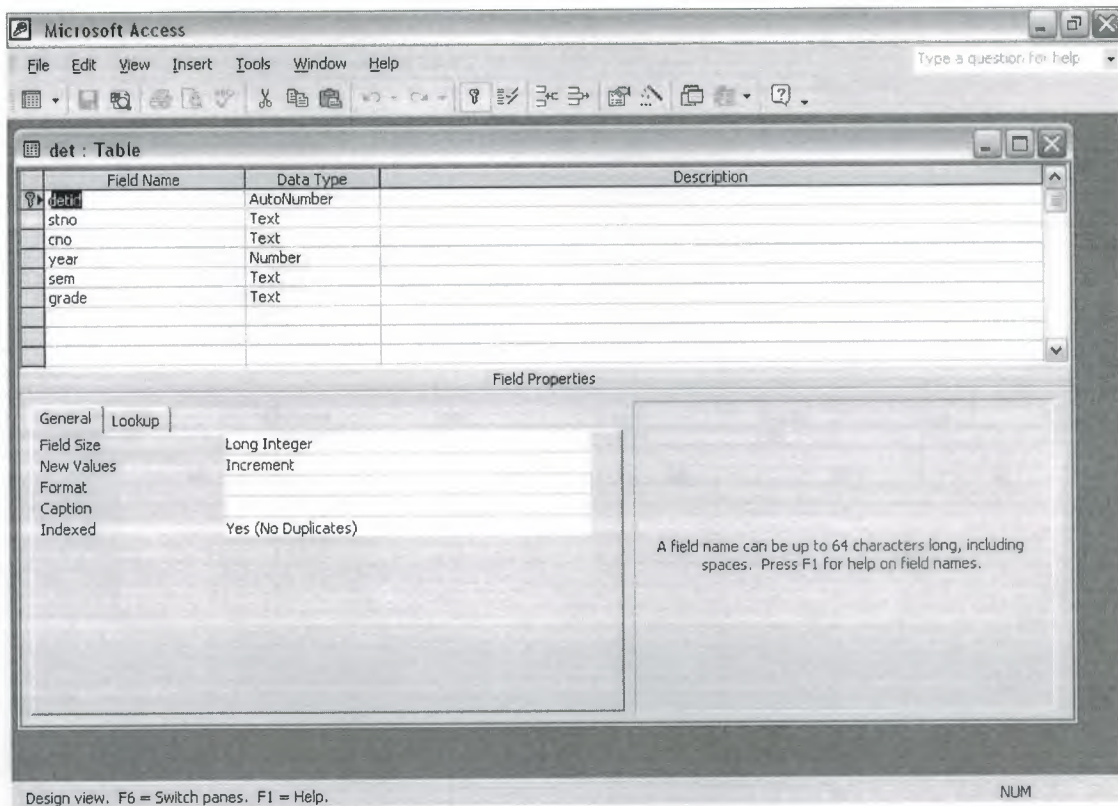


Figure 4.3 Details table design view

The primary key here is **detid** a unique random number generating automatically.

The query that is used here to control the transaction forming the transcript is called **trans** and its design is shown below:

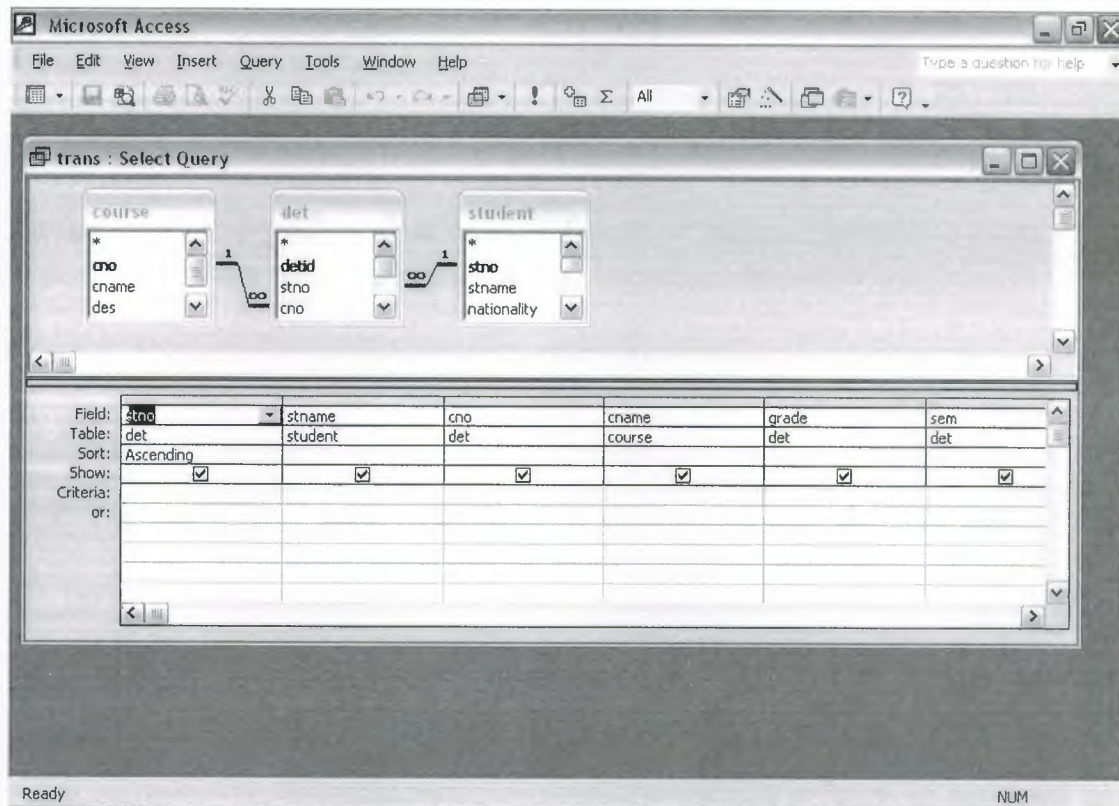


Figure 4.4 Trans Query design view

The sources of data in this query as we can see is the combination of all three previously mentioned tables with the primary key being the **stno** of the **det** table.

The relationship governing these transactions is shown in the figure below:

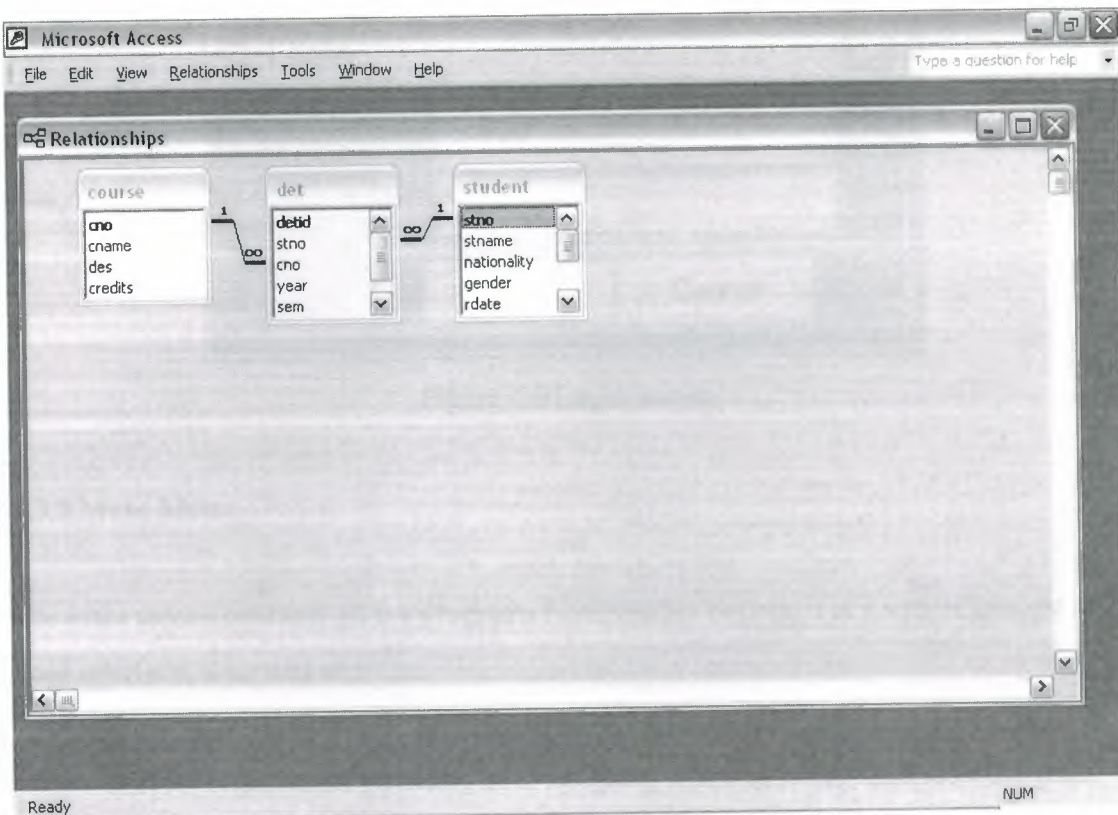


Figure 4.5 Relationship

4.3 The User Interface

The interface of our system is designed using Microsoft Visual Basic 6.0. We are to explain functionality of the system in a user like description beginning from first screen to the Exit.

4.3.1 Login

First we have to be authorized to use the system. The authentication procedure has to be assigned from inside the program. So first enter ID and Password in order to get inside, the screen shot for login dialogue is shown below:

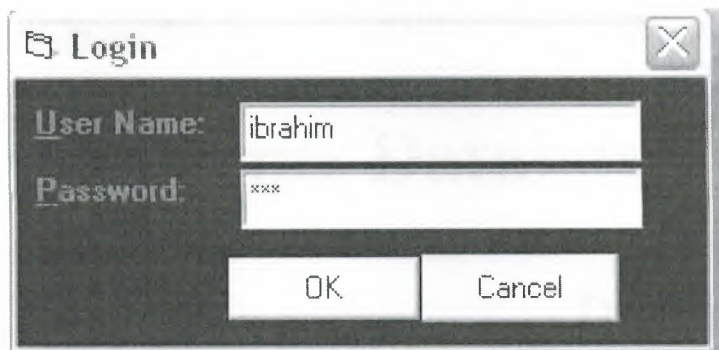


Figure 4.6 Login screen

4.3.2 Main Menu

The main screen contains all the programs functionality outlets, it is a simple straight a head interface, it consist of :

- Entering information for both students and courses.
- Student registration including both assigning a course to take and after the grade issued for that course.
- The searching section includes searching for both students and courses.
- The reports for Students, Courses and Transcript.
- The transcript view.
- The program section contains the about and the user assignment window.

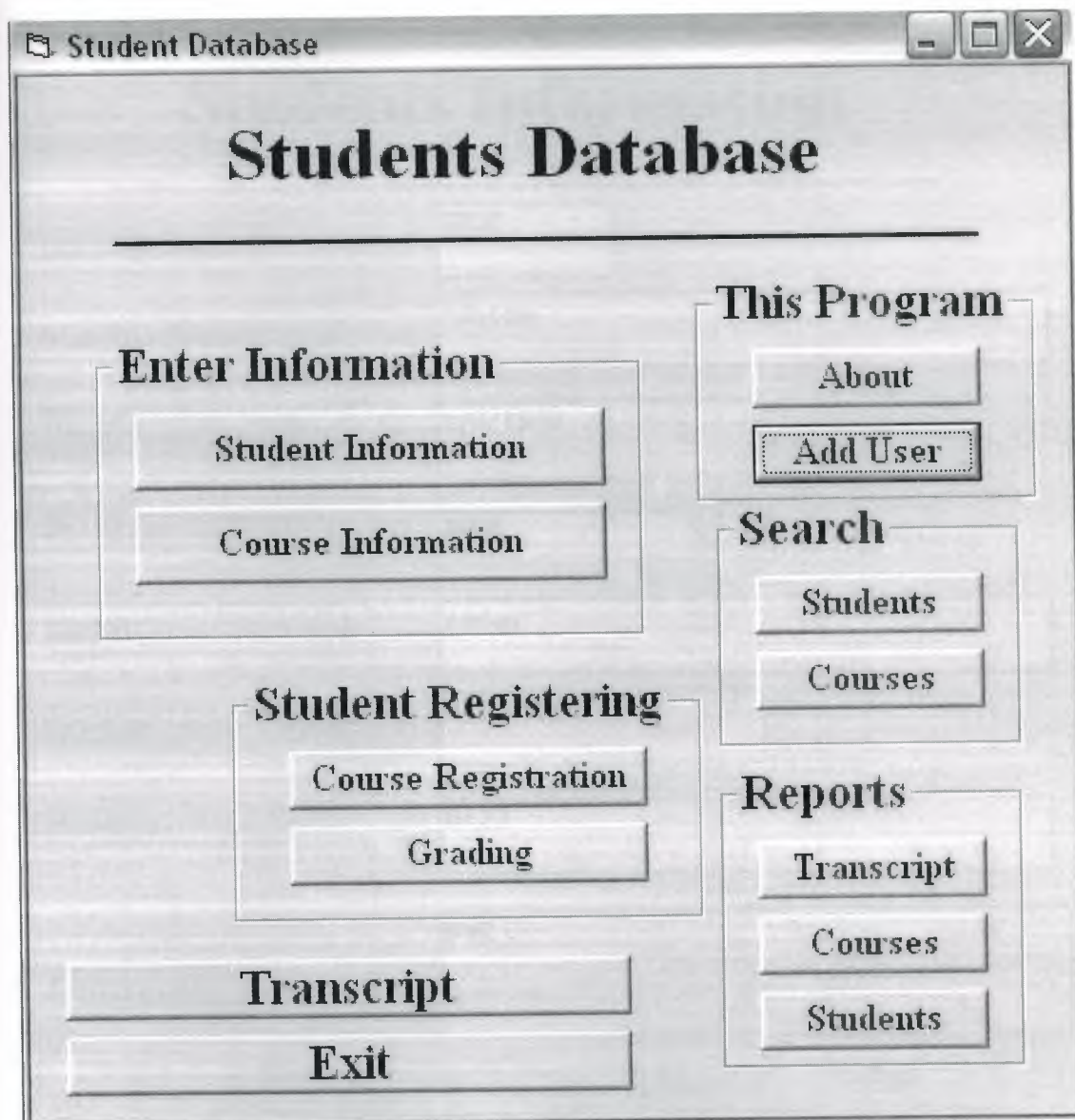


Figure 4.7 Main screen

4.3.3 Entering Information

The data entering consist of main two categories, assigning students and courses. The student assignment screen gives the user the ability to add and edit student information but no deleting capabilities (there are no student records deleted). There is a capability of going around the records, the screen is shown below:

The screenshot shows a window titled 'Students' with a standard Windows-style title bar (minimize, maximize, close buttons). The main content area is titled 'Students Information' in a large, bold font. Below the title is a horizontal line. The form contains several input fields and a dropdown menu, each with a label to its left:

- Student No:** A text box containing the value '3232'.
- Student Name:** A text box containing the value 'hussam'.
- Registration Date:** A text box containing the value '5/8/1998'.
- Gender:** A dropdown menu with 'Male' selected.
- Country:** A text box containing the value 'sudan'.
- Birth Date:** A text box containing the value '6/10/1976'.
- Passport NO.:** A text box containing the value '0123'.
- Address:** A large text box containing the value 'jraif'.

At the bottom of the form, there are three buttons: 'Back', 'Add', and 'Edit'. Below these buttons is a status bar with a left arrow, a double left arrow, the text 'Student Data', a double right arrow, and a right arrow.

Figure 4.8 Student Entering screen

The Courses form just the same as the students has the capability of editing an existing record and adding a new record as shown below:

Course Information

Course Information

Course No: 15

Course Name: Office 2000

Description: computers

Credits: 5

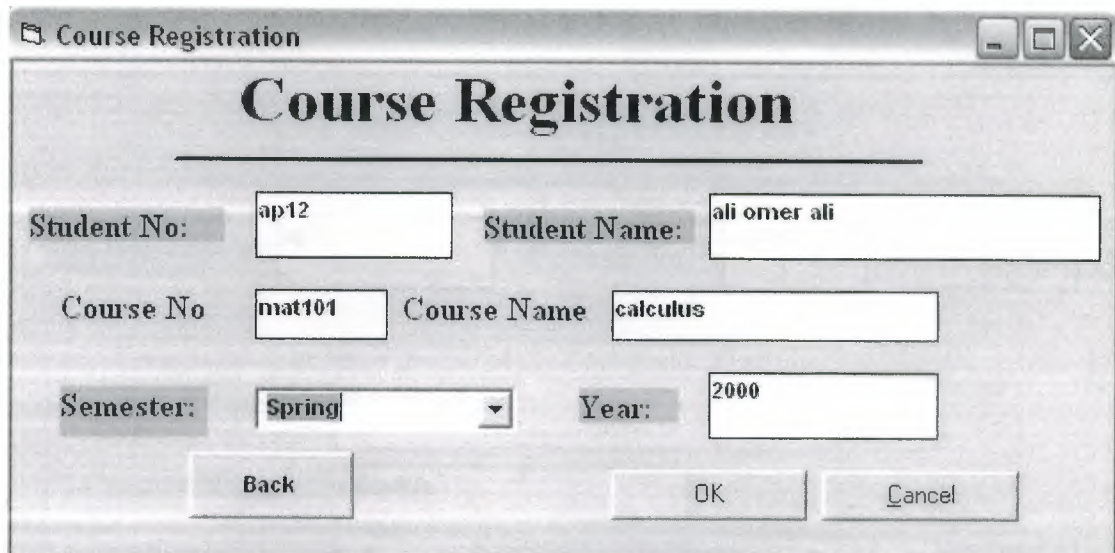
Back Add Edit

Course Data

Figure 4.9 Courses Entering screen

4.3.4 Student Registering

This contains two different screens the first assigns the courses to students for taking during a specified semester and year. So you enter the student number (stno), the course number (cno), the semester and year and click OK to be stored in the database. The student and course names are supplies automatically when pressing enter after you finish entering each corresponding field. The screen is shown below:

A screenshot of a Windows-style application window titled "Course Registration". The window has a title bar with standard minimize, maximize, and close buttons. The main content area has a large heading "Course Registration" followed by a horizontal line. Below the line, there are several input fields and labels. "Student No:" is followed by a text box containing "ap12". "Student Name:" is followed by a text box containing "ali omer ali". "Course No" is followed by a text box containing "mat101". "Course Name" is followed by a text box containing "calculus". "Semester:" is followed by a dropdown menu showing "Spring". "Year:" is followed by a text box containing "2000". At the bottom, there are three buttons: "Back", "OK", and "Cancel".

Course Registration

Course Registration

Student No: ap12 Student Name: ali omer ali

Course No mat101 Course Name calculus

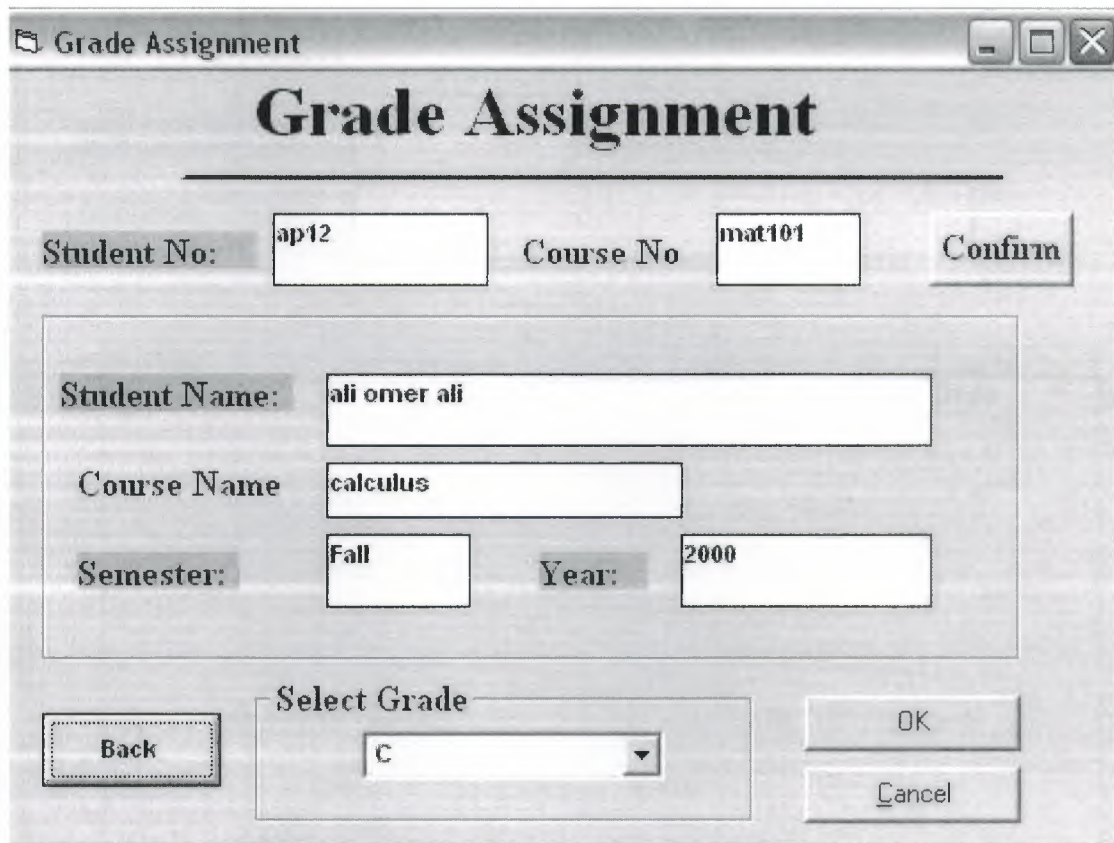
Semester: Spring Year: 2000

Back OK Cancel

Figure 4.10 Course Registration screen

The second screen is for assigning grade to the course that have been taken by the student. So you supply the student number (stno) and course number (cno) and the other information is viewed automatically if they are not that mean the student and the course are not assigned to each other. After that the grade combo box containing available grades which are in our case A, B, C, D and F.

The screen is shown below:



The image shows a graphical user interface window titled "Grade Assignment". At the top, there is a title bar with the text "Grade Assignment" and standard window control buttons (minimize, maximize, close). Below the title bar, the main title "Grade Assignment" is displayed in a large, bold font. The interface is divided into several sections. The top section contains two input fields: "Student No:" with the value "ap12" and "Course No" with the value "mat101". To the right of these fields is a "Confirm" button. Below this, there is a larger rectangular area containing three more input fields: "Student Name:" with the value "ali omer ali", "Course Name" with the value "calculus", and "Semester:" with the value "Fall". To the right of the "Semester:" field is a "Year:" field with the value "2000". At the bottom of the window, there is a "Select Grade" section. It includes a "Back" button on the left, a dropdown menu in the center showing the letter "C", and "OK" and "Cancel" buttons on the right.

Grade Assignment

Grade Assignment

Student No: ap12 Course No mat101 Confirm

Student Name: ali omer ali

Course Name calculus

Semester: Fall Year: 2000

Back Select Grade C OK Cancel

Figure 4.11 Grading screen

4.3.5 Searching

We are searching for either students or courses. The procedure for searching is similar to each other just print the student no or name and during printing there are online filtering of records until you find your target.

The screens for both students and courses:

Search For Student

Student No:

Student Name:

stno	stname	nationality	gender
▶ ap12	ali omer ali	sudan	Male

Back

Figure 4.12 Student Search screen.

cno	cname	des	credits
com10	internet	computer	4

Figure 4.13 Courses search screen

4.3.5 Reports

In our project we have three reports:

- Report the list of all students.
- Report the list of all the courses.
- Report the transcript.

The first two reports are just lists but the third one has a filtering procedure by the student number (stno). The reports are shown below:

DataReport1

Zoom 100%

NEAR EAST UNIVERSITY

LIST OF ALL STUDENTS 7/1/2003

St No:	St Name:	Nationality:	Gender:
5232	hussam	sudan	Male
444	ibo	sudan	male
50	Mohaiad	sudan	Male
55	Rejesh	pakistan	male
5121	Ametab	india	female
58	Vanbasten	holand	female
59	ozgur	turkey	male
78	Lijo chang	china	female
an1	darir	somal	Male
ap12	ali omer ali	sudan	Male

Figure 4.14 Students Report screen

DataReport2

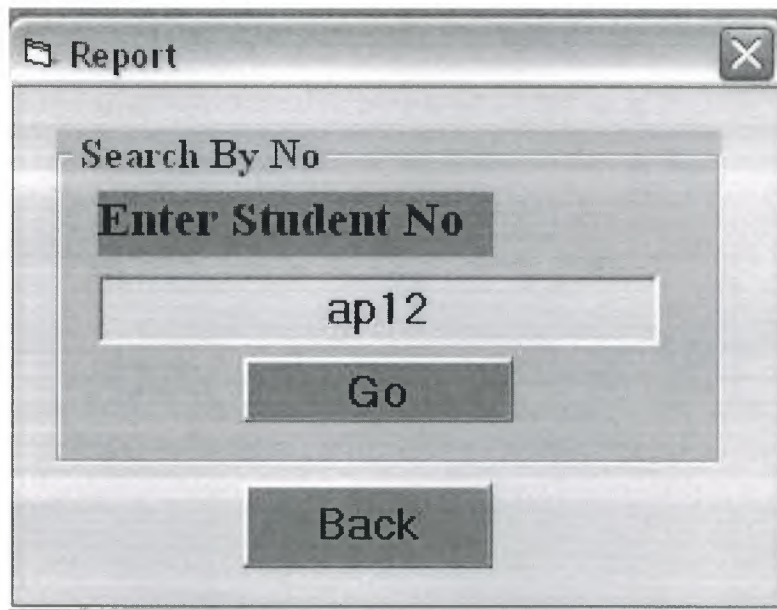
Zoom 100%

NEAR EAST UNIVERSITY

LIST OF ALL COURSES 7/1/2003

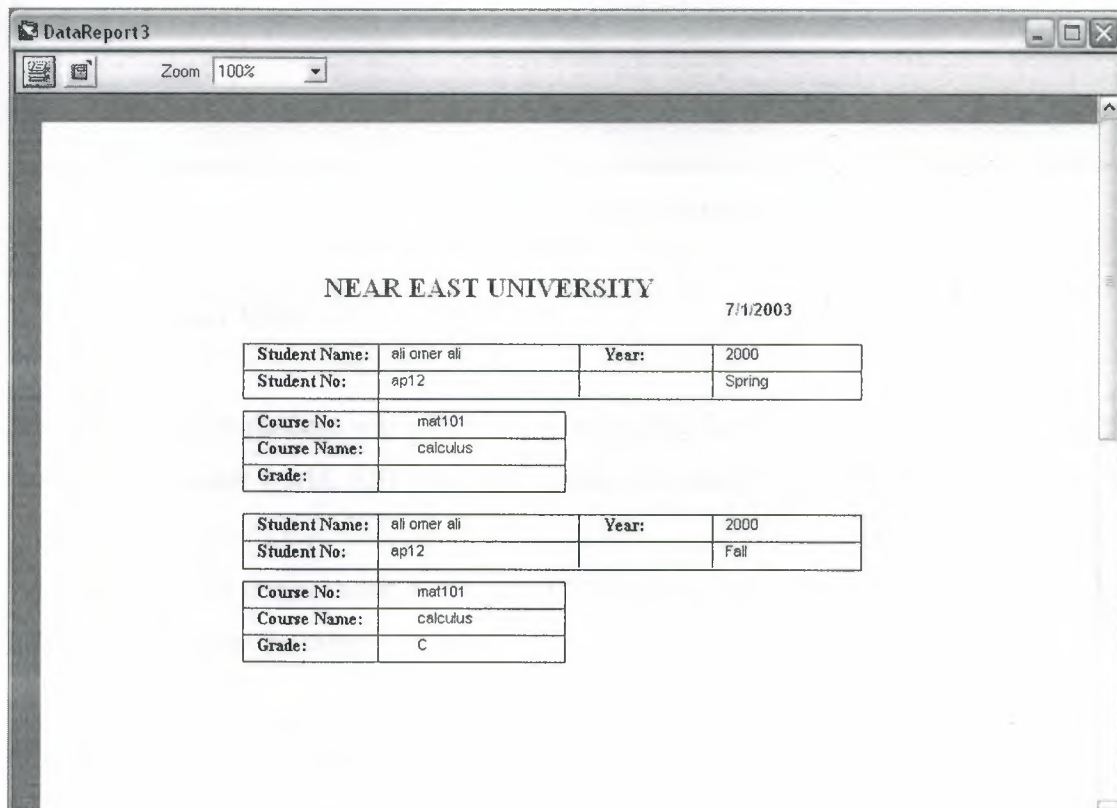
Course No:	Course Name:	Description:	Credits:
15	Office 2000	computers	5
16	Functional	ABST	3
18	Visual Basic	Computers	4
21	Auditing	BA	3
22	Basic of Computers	BA	4
23	Data structure	Computers	3
24	Management	ABST	3
COM 101	computer	introduction to	4
man	management	management for	4
mat101	calculus	integration,derivativ	3
mat102	descrete maths	mathmatics	4
ph1	physics	light and heat	3
com10	internet	computer	4

Figure 4.15 Course Report screen



A dialog box titled "Report" with a close button (X) in the top right corner. Inside the dialog, there is a section titled "Search By No" which contains a label "Enter Student No" above a text input field. The input field contains the text "ap12". Below the input field is a "Go" button. At the bottom of the dialog is a "Back" button.

Figure 4.16 Report for transcript screen



A window titled "DataReport3" with a zoom control set to 100%. The report content is as follows:

NEAR EAST UNIVERSITY

7/1/2003

Student Name:	ali omer ali	Year:	2000
Student No:	ap12		Spring

Course No:	mat101
Course Name:	calculus
Grade:	

Student Name:	ali omer ali	Year:	2000
Student No:	ap12		Fall

Course No:	mat101
Course Name:	calculus
Grade:	C

Figure 4.17 Transcript Report screen

There is a transcript view screen used to view by either student number or name as shown below:

The screenshot shows a window titled "Transcript". It contains two input fields: "Student No:" with the value "ap12" and "Student Name:" with the value "ali omer ali". To the right of these fields are two buttons: "Search" and "Clear". Below the input fields is a table with the following data:

Course No	Course Name	Grade	Semester	Year
mat101	calculus		Spring	2000
mat101	calculus	C	Fall	2000

Below the table is a large empty rectangular area, likely for a detailed report or additional data. At the bottom left of the window is a "Back" button.

Figure 4.18 Transcript screen

4.4 The Internet View

The internet is the modern way of any thing, managing databases using the internet is a key role in the new world, ASP made this option very easily implemented.

Using ASP in our project made us capable of using the student number to view the grade of a course or even the course report:

The main screen is shown below:

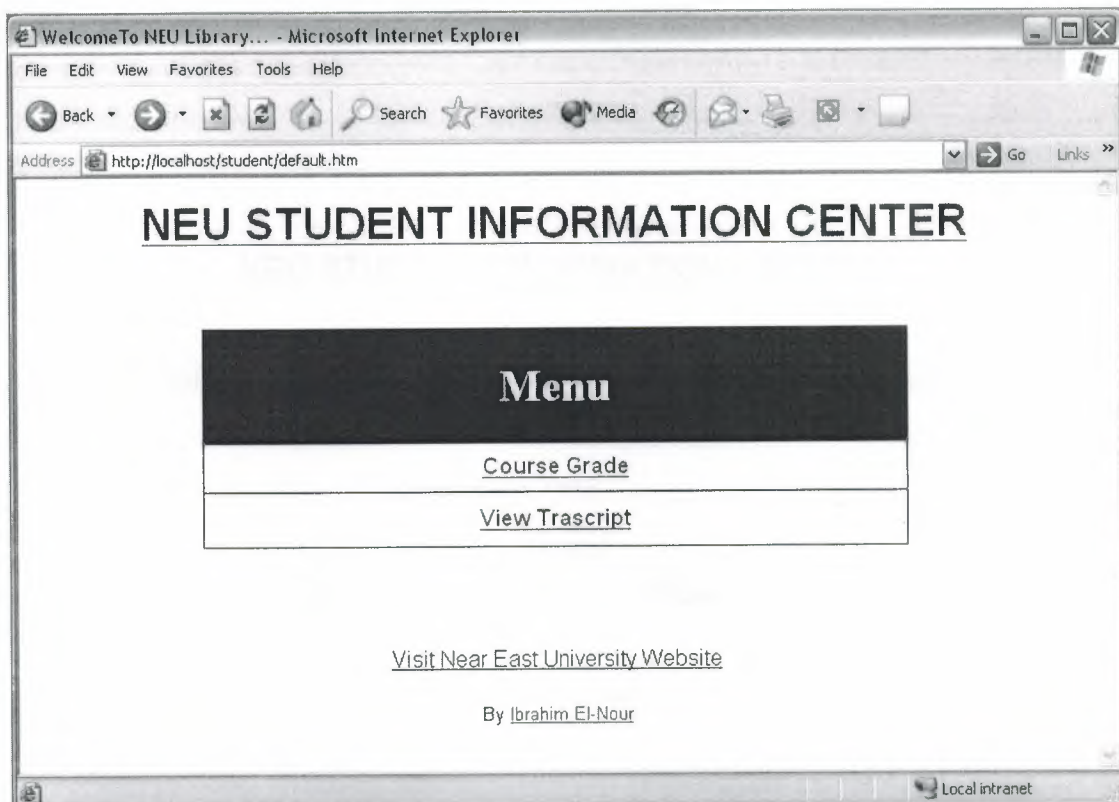


Figure 4.19 Main Web Page

If you choose to go to the course grade view we can see the:

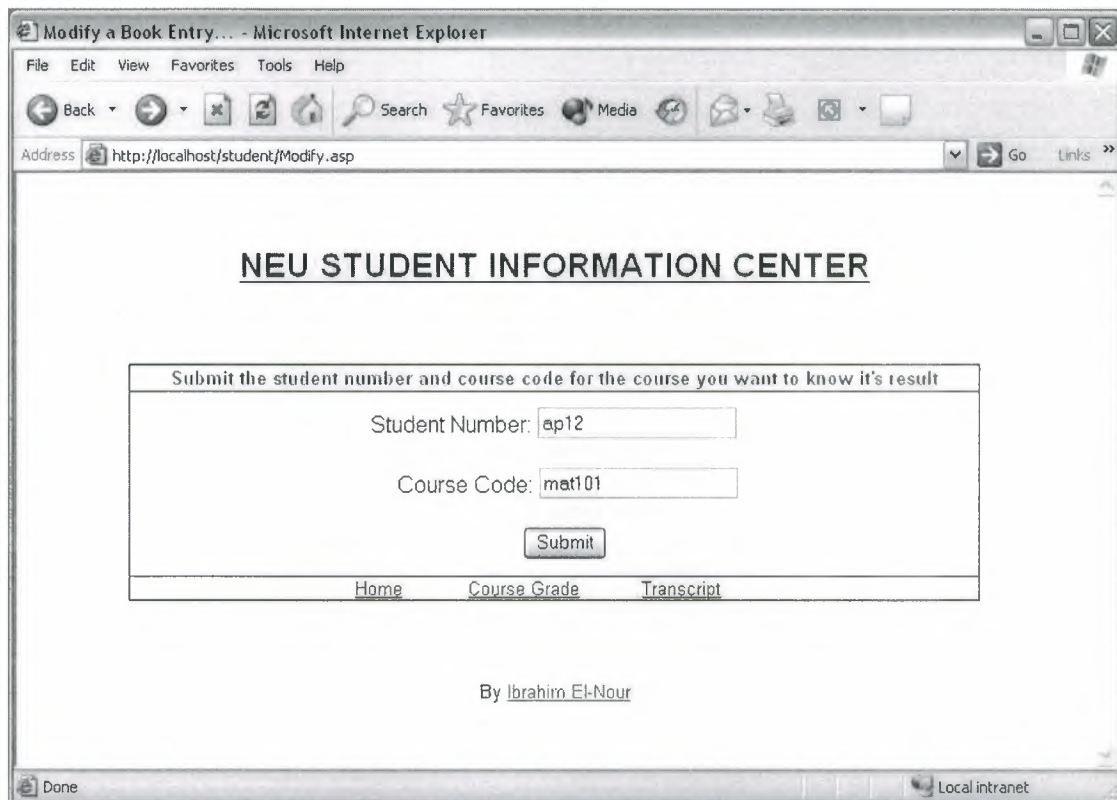


Figure 4.20 Grade View Entering screen

In this screen you enter the student number and course code you wish to find grade for.

After that the result screen with the name of the course and the grade is shown below:

.....

The other option from the main screen is the path to transcript view as shown below:

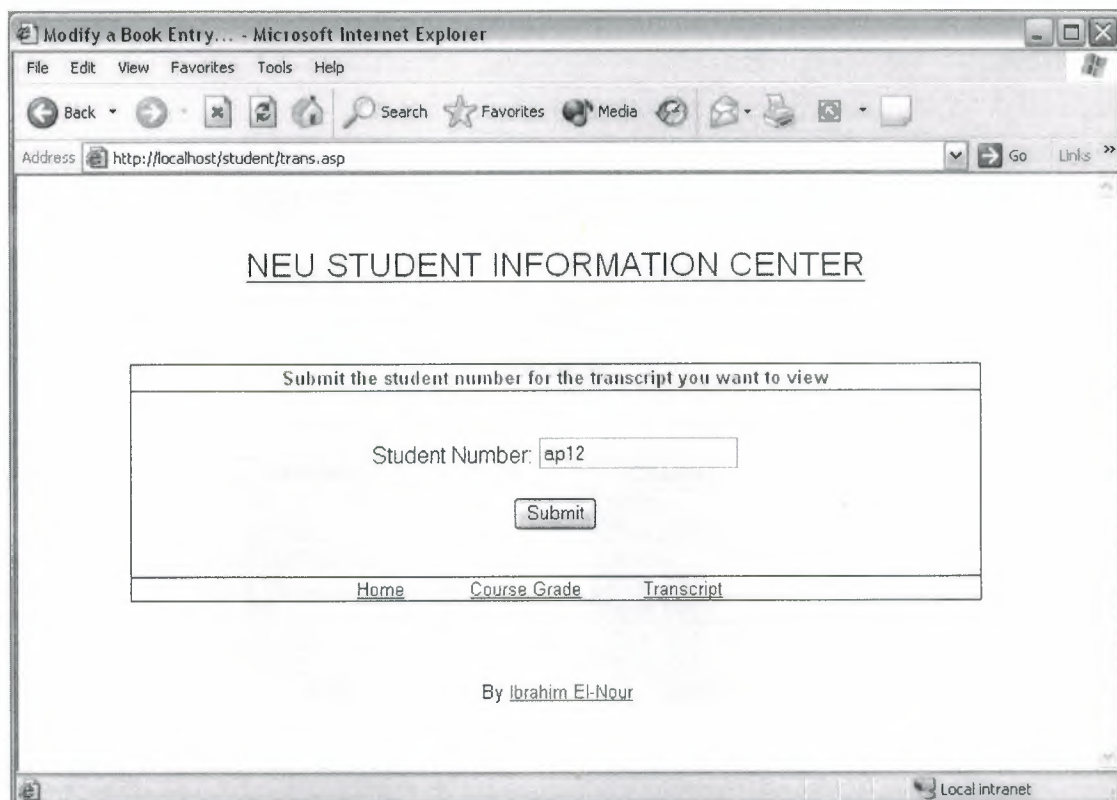


Figure 4.21 The Transcript view Entering page

This next page is the transcript result page filtered through the student number as shown below:

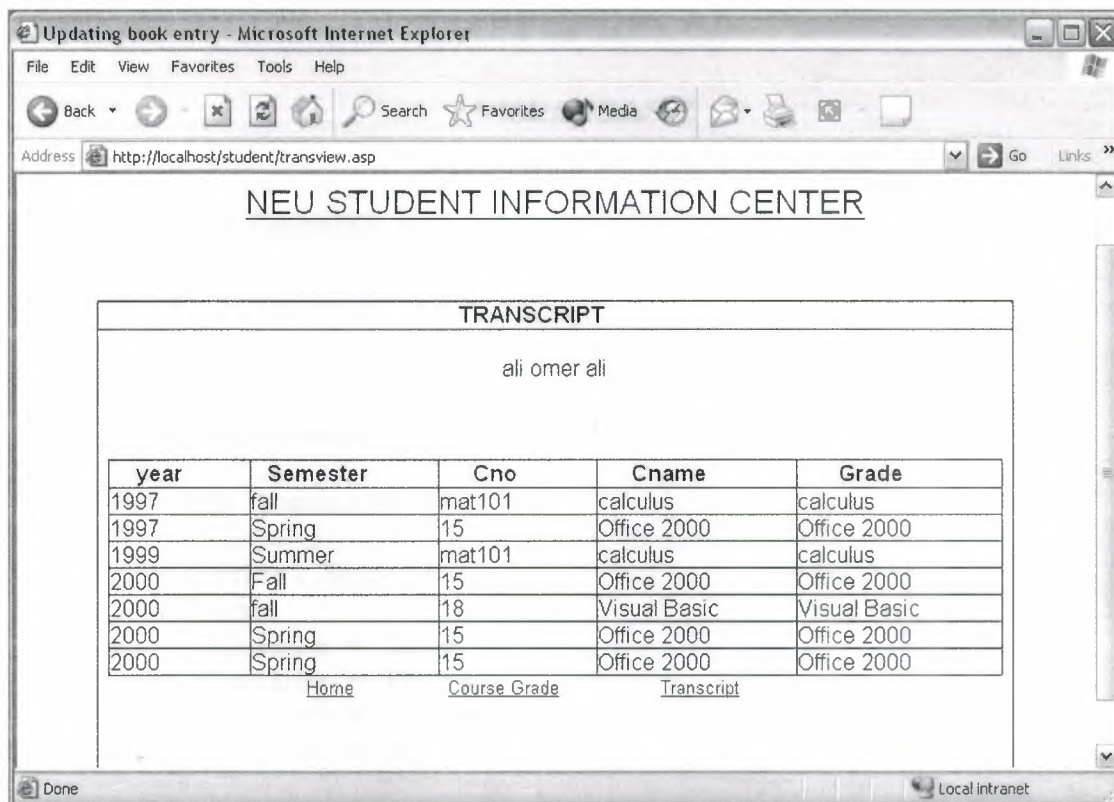


Figure 4.22 Transcript Result Page

4.5 Summary

This chapter showed the database application of the students management system using Microsoft Access for table design, Visual Basic for system interfacing and management and ASP for managing the database in the Internet.

All the screen shots are supplied along with the description of functionality of the system.



APPENDIX A

VISUAL BASIC

```
Private Sub Cmdquit_Click()  
Form6.Show  
Unload Me  
End Sub  
  
Private Sub Form_Load()  
Data1.DatabaseName = App.Path & "\stdb.mdb"  
End Sub  
Private Sub Command2_Click()  
  
Data1.Recordset.Update  
Data1.Refresh  
Command6.Visible = True  
Command2.Visible = False  
Command3.Visible = False  
Command4.Visible = True  
  
End Sub  
  
Private Sub Command3_Click()  
Data1.Recordset.CancelUpdate  
Data1.Refresh  
  
Command6.Visible = True  
Command2.Visible = False  
Command3.Visible = False  
Command4.Visible = True  
  
End Sub  
  
Private Sub Command4_Click()  
Data1.Recordset.Edit  
  
Command2.Visible = True  
Command4.Visible = False  
Command3.Visible = True  
Command6.Visible = False  
  
End Sub
```


////////////////////
 //////////////////

```

bb(0) = 0
ss(0) = 0
ss1(0) = 0
cgpav1 = 0
With Form4.Data1.Recordset
    .MoveFirst

    While .EOF = False
        If .Fields("stno") = Form4.Text1.Text And
        .Fields("cno") = Form4.Text3.Text Then
            cc(i) = .Fields("credits")
            str1 = .Fields("grade")
            vv(i) = grad(str1)
            ss(i) = cc(i) + cc(i - 1)
            bb(i) = vv(i) * cc(i)
            ss1(i) = bb(i) + bb(i - 1)
            i = i + 1
            .MoveNext
        Else
            .MoveNext
        End If
    Wend

    cgpav1 = ss1(i) / ss(i)

End With
cgpav = cgpav1

End Function
Private Function loata()
Dim cgpaf As Double
cgpaf = cgpav()

With Form1.Data1.Recordset
    .MoveFirst

    While .EOF = False
        If .Fields("stno") = Text1.Text Then
            .Fields("cgpa") = cgpaf
            Exit Sub
        Else
            .MoveNext
        End If
    Wend

```

```
End Sub
```

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

```
Private Sub Command2_Click()  
cmdOk.Caption = "Change"  
End Sub
```

```
Private Sub Form_Load()  
Data1.DatabaseName = App.Path & "\stdb.mdb"  
txtusername.Text = ""  
txtUserpsswd.Text = ""  
txtUserid.Text = ""
```

```
txtusername.Enabled = False  
txtUserpsswd.Enabled = False  
txtUserid.Enabled = False  
End Sub
```

```
////////////////////////////////////  
////////////////////////////////////
```

```
Public conndyn As Connection  
Public rsdyn As Recordset  
Public Function grad(gr As String) As Double  
Select Case gr  
Case "A"  
grad = 4  
Case "B"  
grad = 3  
Case "C"  
grad = 2  
Case "D"  
grad = 1  
Case "F"  
grad = 0  
End Select  
End Function
```

```
Public Function cgpav() As Double  
Dim cc(30), vv(30), bb(30), ss(30), ss1(30), cgpav1 As  
Double  
Dim i, j As Integer  
Dim str1 As String  
i = 1  
cc(0) = 0  
vv(0) = 0
```

```

Private Sub CmdChange_Click()
cmdOk.Caption = "Delete"
txtusername.Enabled = False
txtUserpsswd.Enabled = False
txtUsercpsswd.Enabled = False
txtUserid.Enabled = True
txtUserid.SetFocus

End Sub

Private Sub cmdOK_Click()
On Error GoTo vip
If cmdOk.Caption = "Create" Then
    If txtUserpsswd.Text = txtUsercpsswd.Text Then
        Datal.Recordset.Fields("name") = txtusername.Text
        Datal.Recordset.Fields("userid") = txtUserid.Text
        Datal.Recordset.Fields("pass") =
txtUserpsswd.Text
        Datal.Recordset.Update
        MsgBox "User Created."
        txtusername.Text = ""
        txtUserpsswd.Text = ""
        txtUserid.Text = ""
        txtUsercpsswd.Text = ""
        txtusername.Enabled = False
        txtUserpsswd.Enabled = False
        txtUserid.Enabled = False
    Else
        MsgBox "confirm password."
    End If
vip:
    If Err.Number = 3022 Then
        MsgBox "User already exists with the name " &
txtusername.Text, vbOKOnly
        txtUserid.SetFocus
    End If
Else
    Datal.Recordset.FindFirst ("userid='" & txtUserid.Text
& "'")
    If Datal.Recordset.NoMatch = True Then
        MsgBox "no user found."
    Else
        Datal.Recordset.Delete
        MsgBox "User deleted."
    End If
End If

```



```

Private Sub cmdOK_Click()
logindata.Recordset.FindFirst ("userid='" +
txtUserName.Text + "'")
    If Not logindata.Recordset.NoMatch Then
        If txtPassword.Text =
logindata.Recordset.Fields("pass") Then
            Me.Visible = False
            Form6.Show
        Else
            i = MsgBox("Enter correct password.", vbOKOnly,
"Incorrect Password")
        End If
    Else
        MsgBox "Either User name or Password is incorrect
or ask your system administrator for further details."
    End If
End Sub

Private Sub Form_Load()
logindata.DatabaseName = App.Path & "\stdb.mdb"

End Sub

Private Sub Form_Unload(Cancel As Integer)

If Cancel = 0 Then
    i = MsgBox("You want to exit this Application",
vbYesNo, "Exit Application")
    If i = vbYes Then
        End
    Else
        Cancel = 1
    End If
End If
End Sub

////////////////////////////////////
////////////////////////////////////
Private Sub CmdAdd_Click()
cmdOk.Caption = "Create"
Data1.Recordset.AddNew
txtusername.Enabled = True
txtUserpsswd.Enabled = True
txtUserid.Enabled = True
txtUsercpsswd.Enabled = True
txtusername.SetFocus
End Sub

```

```

End With

End Sub
Private Sub cmdprint1_Click()
On Error GoTo trap
    If Text1.Text = "" Then
MsgBox "Enter Student No!... "

Else:

query1 ("SELECT trans.stno, trans.stname, trans.cno,
trans.cname, trans.grade, trans.year, trans.sem FROM trans
WHERE trans.stno ='" & Trim(Text1) & "'")

End If
trap:

    With rptdynamic1
        Set .DataSource = Nothing

        Set .DataSource = rsdyn1.DataSource
        .DataMember = rsdyn1.DataMember

    .Show
End With

End Sub
////////////////////////////////////
////////////////////////////////////
Private Sub cmdOK_Click()
    Form6.Show
Me.Hide
End Sub
////////////////////////////////////
////////////////////////////////////
Private Sub cmdCancel_Click()
Dim i As Byte
Beep
i = MsgBox("You want to exit this Application", vbYesNo,
"Exit Application")

    If i = vbYes Then
        End
    End If
End Sub

```

```

conndyn.Open "Provider=Microsoft.Jet.OLEDB.3.51;Persist
Security Info=False;User ID=Admin;Data Source=" & App.Path
& "\stdb.mdb;Mode=Share Deny None;Extended
Properties=';COUNTRY=0;CP=1252;LANGID=0x0409';Jet
OLEDB:System database='';Jet OLEDB:Registry Path='';Jet
OLEDB:Database Password='';Jet OLEDB:Global Partial Bulk
Ops=2"

```

```

End Sub

```

```

Private Sub query(sql1 As String)
    Set rsdyn = New Recordset
    rsdyn.CursorLocation = adUseClient
    rsdyn.Open sql1, conndyn, adOpenForwardOnly,
adLockReadOnly

```

```

    With rptdynamic1

```

```

        Set .DataSource = Nothing

```

```

        .DataMember = trans

```

```

        Set .DataSource = rsdyn.DataSource

```

```

        .DataMember = rsdyn.DataMember

```

```

    End With

```

```

End Sub

```

```

Private Sub Form_Activate()

```

```

    query ("SELECT trans.stno, trans.stname,
trans.cno,trans.cname, trans.grade,trans.year,trans.sem
From trans")

```

```

End Sub

```

```

Private Sub cmdprint_Click()

```

```

    With rptdynamic1

```

```

        Set .DataSource = Nothing

```

```

        Set .DataSource = rsdyn.DataSource

```

```

        .DataMember = rsdyn.DataMember

```

```

        .Show

```

```

    End With

```

```

End Sub

```

```

Private Sub query1(sql2 As String)

```

```

    Set rsdyn1 = New Recordset

```

```

    rsdyn1.CursorLocation = adUseClient

```

```

    rsdyn1.Open sql2, conndyn, adOpenForwardOnly,
adLockReadOnly

```

```

    With rptdynamic1

```

```

        Set .DataSource = Nothing

```

```

        .DataMember = trans

```

```

        Set .DataSource = rsdyn1.DataSource

```

```

        .DataMember = rsdyn1.DataMember

```

```

Dim mname As String
mname = Trim(Text4.Text)
If mname <> "" Then
    DataEnvironment1.rsstudent.Filter = "stname like '" &
mname & "%'"

Else
    DataEnvironment1.rsstudent.Filter = "stname <> '123'"

End If
End Sub
////////////////////////////////////
////////////////////////////////////
Private Sub Text1_Change()
Dim mname As String
mname = Trim(Text1.Text)
If mname <> "" Then
    DataEnvironment1.rscourse.Filter = "cno like '" & mname &
"%'"

Else
    DataEnvironment1.rscourse.Filter = "cno <> '123'"

End If
End Sub

Private Sub Text4_Change()
Dim mname As String
mname = Trim(Text4.Text)
If mname <> "" Then
    DataEnvironment1.rscourse.Filter = "cname like '" & mname
& "%'"

Else
    DataEnvironment1.rscourse.Filter = "cname <> '123'"

End If
End Sub
////////////////////////////////////
////////////////////////////////////
Public rsdyn1 As Recordset
Private Sub Command1_Click()
Form6.Show
Unload Me
End Sub
Private Sub Form_Load()
Set conndyn = New Connection

```



```
Private Sub Command6_Click()
frmAbout.Show
Me.Hide
End Sub
```

```
Private Sub Command7_Click()
DataReport1.Show
End Sub
```

```
Private Sub Command8_Click()
Form7.Show
Me.Hide
End Sub
```

```
Private Sub Command9_Click()
Form8.Show
Me.Hide
End Sub
```

```
Private Sub Form_Load()
DataEnvironment1.Connection1.ConnectionString =
"Provider=Microsoft.Jet.OLEDB.3.51;Persist Security
Info=False;User ID=Admin;Data Source=" & App.Path &
"\stdb.mdb;Mode=Share Deny None;Extended
Properties=';COUNTRY=0;CP=1252;LANGID=0x0409';Jet
OLEDB:System database='';Jet OLEDB:Registry Path='';Jet
OLEDB:Database Password='';Jet OLEDB:Global Partial Bulk
Ops=2"
```

```
End Sub
////////////////////////////////////
////////////////////////////////////
```

```
Private Sub Text1_Change()
Dim mname As String
mname = Trim(Text1.Text)
If mname <> "" Then
DataEnvironment1.rsstudent.Filter = "stno like '" & mname
& "%'"
Else
DataEnvironment1.rsstudent.Filter = "stno <> '123'"
End If
End Sub
```

```
Private Sub Text4_Change()
```

```

End Sub
////////////////////////////////////
////////////////////////////////////
Private Sub Command1_Click()
Form1.Show
Me.Hide
End Sub

Private Sub Command10_Click()
End
End Sub

Private Sub Command11_Click()
DataReport2.Show

End Sub

Private Sub Command12_Click()
freportcars.Show
Me.Hide

End Sub

Private Sub Command13_Click()
Useroption.Show
End Sub

Private Sub Command2_Click()
Form2.Show
Me.Hide
End Sub

Private Sub Command3_Click()
Form3.Show
Me.Hide
End Sub

Private Sub Command4_Click()
Form4.Show
Me.Hide
End Sub

Private Sub Command5_Click()
Form5.Show
Me.Hide
End Sub

```

```

Private Sub Form_Load()
Text1.Enabled = True
Text4.Enabled = True
DataGrid1.Visible = False
End Sub

Private Sub Text1_Change()
Dim mname As String
mname = Trim(Text1.Text)
If mname <> "" Then
    DataEnvironment1.rstrans.Filter = "stno like '" & mname &
"%'"
Else
    DataEnvironment1.rstrans.Filter = "stno <> '123'"

End If
End Sub

Private Sub Text4_Change()
Dim mname As String
mname = Trim(Text4.Text)
If mname <> "" Then
    DataEnvironment1.rstrans.Filter = "stname like '" & mname
& "%'"
Else
    DataEnvironment1.rstrans.Filter = "stname <> '123'"

End If
End Sub

Private Sub Text1_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
With Form1.Data1.Recordset
.MoveFirst

    While .EOF = False
        If .Fields("stno") = Text1.Text Then
            Text4.Text = .Fields("stname")
            Exit Sub
        Else
            .MoveNext
        End If
    Wend
End With
End If

```

```

Data1.Recordset.Update
Data1.Refresh
Command2.Visible = False
Command3.Visible = False
Text1.Text = " "
Text3.Text = " "
Text2.Text = " "
Text5.Text = " "
Text4.Text = " "
Text13.Text = " "
Combo1.Text = " "
xx11:
MsgBox "Please do not leave an Empty field", , "System"
End Sub

```

```

Private Sub Command3_Click()

```

```

Data1.Refresh
Command2.Visible = False
Command3.Visible = False
End Sub

```

```

////////////////////////////////////
////////////////////////////////////

```

```

Private Sub Cmdquit_Click()
Form6.Show
Unload Me
End Sub

```

```

Private Sub Command1_Click()
Text1.Enabled = False
Text4.Enabled = False
DataGrid1.Visible = True

```

```

End Sub

```

```

Private Sub Command2_Click()
Text1.Text = ""
Text4.Text = ""
Text1.Enabled = True
Text4.Enabled = True
DataGrid1.Visible = False
End Sub

```



```

Private Sub Command4_Click()
Data1.Recordset.Edit
Command2.Visible = True
Command4.Visible = False
Command3.Visible = True

```

```

End Sub

```

```

////////////////////////////////////
////////////////////////////////////

```

```

Private Sub Cmdquit_Click()
Form6.Show
Unload Me
End Sub

```

```

Private Sub Command1_Click()
Dim cgpaf As Double
'*****
'*****
With Data1.Recordset
.MoveFirst
.FindFirst ("stno = " & "'" & Text1.Text & "'" & "and
cno =" & "'" & Text3.Text & "'")
.Requery

End With
'*****
'*****
Command2.Visible = True
Command3.Visible = True
End Sub

```

```

Private Sub Form_Load()
Data1.DatabaseName = App.Path & "\stddb.mdb"
Command2.Visible = True
Command3.Visible = True
Text2.Text = " "
Text5.Text = " "
Text4.Text = " "
Text13.Text = " "
End Sub

```

```

Private Sub Command2_Click()
On Error GoTo xx11

Data1.Recordset.Edit

```

```

If KeyAscii = 13 Then
With Form2.Data1.Recordset
    .MoveFirst

    While .EOF = False
        If .Fields("cno") = Text3.Text Then
            Text13.Text = .Fields("cname")
            Exit Sub
        Else
            .MoveNext
        End If
    Wend
End With
End If
End Sub
Private Sub Command2_Click()
On Error GoTo xx10

Data1.Recordset.Update
Data1.Refresh
Command2.Visible = False
Command3.Visible = False
Command4.Visible = True
Text1.Text = " "
Text3.Text = " "
Text2.Text = " "
Text4.Text = " "
Text13.Text = " "
Combo1.Text = " "
xx10:
MsgBox "Please do not leave an Empty field", , "System"
End Sub

Private Sub Command3_Click()
On Error GoTo xx12

Data1.Recordset.CancelUpdate
Data1.Refresh
xx12:

Command2.Visible = False
Command3.Visible = False
Command4.Visible = True

End Sub

```

End Sub

```
Command2.Visible = True
Command6.Visible = False
Command3.Visible = True
Command4.Visible = False
```

////////////////////
////////////////////

```
Private Sub Form_Load()  
Data1.DatabaseName = App.Path & "\stdb.mdb"  
Command2.Visible = True  
Command3.Visible = True  
Command4.Visible = False
```

```
While .EOF = False
    If .Fields("stno") = Text1.Text Then
        Text4.Text = .Fields("stname")
        Exit Sub
    Else
        .MoveNext
    End If
End While
```

```
Private Sub Text3_KeyPress(KeyAscii As Integer)
```

```
Private Sub Command6_Click()  
Data1.Recordset.AddNew
```

```
Command2.Visible = True  
Command6.Visible = False  
Command3.Visible = True  
Command4.Visible = False
```

```
End Sub
```

```
Private Sub Cmdquit_Click()
```

```
Form6.Show  
Unload Me  
End Sub
```

```
Private Sub Form_Load()
```

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

```
End Sub
```

```
Private Sub Command2_Click()
```

```
Data1.Recordset.Update  
Data1.Refresh  
Command6.Visible = True  
Command2.Visible = False  
Command3.Visible = False  
Command4.Visible = True
```

```
End Sub
```

```
Private Sub Command3_Click()
```

```
Data1.Recordset.CancelUpdate  
Data1.Refresh
```

```
Command6.Visible = True  
Command2.Visible = False  
Command3.Visible = False  
Command4.Visible = True
```

```
End Sub
```

```
Private Sub Command4_Click()
```

```
Data1.Recordset.Edit
```

```
Command2.Visible = True  
Command4.Visible = False  
Command3.Visible = True
```


APPENDIX B

ASP

```

<html dir="ltr">

<head><h1 align="center"><u><font face="Arial" color="#800000">NEU
STUDENT
INFORMATION CENTER</font></u></h1>
<p align="center">&nbsp;</p>
<div align="center">
  <center>
    <table border="1" cellpadding="0" cellspacing="0" style="border-
collapse: collapse; border: 1px solid #000000" bordercolor="#111111"
id="AutoNumber1" width="504" height="156">
      <tr>
        <td width="502" style="border: 1px solid #000000" height="53"
bgcolor="#800000">
          <p align="center"><b>
            <font color="#FFFFFF" face="Times New Roman"
size="6">Menu</font></b></td>
          </tr>
          <tr>
            <td width="502" style="border: 1px solid #000000" height="23"
align="center">
              <b>
                <font face="Arial"><a href="Modify.asp"><font
color="#FF0000">Course Grade</font></a></font></b></td>
              </tr>
              <tr>
                <td width="502" style="border: 1px solid #000000" height="26"
align="center">
                  <b>
                    <font face="Arial"><a href="trans.asp"><font color="#FF0000">View
Transcript</font></a></font></b></td>
                  </tr>
                </table>
              </center>
            </div>
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
            <meta name="ProgId" content="FrontPage.Editor.Document">
            <meta http-equiv="Content-Type" content="text/html; charset=windows-
1252">
            <title>WelcomeTo NEU Library...</title>
          </head>

          <body>

          <p align="left">&nbsp;</p>

          <div align="center">
            <center>
              <table border="0" cellpadding="0" cellspacing="0" style="border-
collapse: collapse" bordercolor="#111111" id="AutoNumber2" width="303"
height="79">

```

```
|  |  | | | | | |
|---|---|---|---|---|---|---|
| <font face="Arial"> |  | | --- | | <font size="2" face="Arial">By   </center> </div> | |

```

```

</body>

```

```

</html>
////////////////////////////////////
////////////////////////////////////
<html>

```

```

<head>
<p align="center">&nbsp;</p>
<p align="center"><b><u><font face="Arial" color="#800000" size="5">NEU STUDENT INFORMATION CENTER</font></u></b></p>
<p align="center">&nbsp;</p>
<meta name="GENERATOR" content="Microsoft FrontPage 5.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Updating book entry</title>
</head>

```

```

<body style="font-family: Arial">

```

```

<div align="center">
<center>
<table border="1" cellpadding="0" cellspacing="0" style="border-collapse: collapse" bordercolor="#111111" id="AutoNumber1" width="622" height="170">
|  |  | | | | | |
|---|---|---|---|---|---|---|
| |  | | --- | |  | |

```

```

<td width="580" height="18" align="center">

<%sn=request.form("stno")
cc=request.form("cno")
set b=server.createobject ("ADODB.connection")
b.open "Driver={Microsoft Access Driver (*.mdb)};DBQ=" & "c:\stdb.mdb"
c="SELECT grade,cname FROM trans WHERE stno="&sn&" AND cno="&cc&"
set d=b.execute(c)
if d.eof then%>
<p>The submitted <font color="#FF0000"><u><b>Student Number or Course code you
Entered are not correct </b></u></font>, please enter another Student Number or Course
code</p></td>
</tr>
<tr>
<td width="580" height="133" align="center"><%else%>
<p>your grade on <%=d("cname") %> is <%=d("grade") %> :</p>
<p>&nbsp;</p>
<%end if
b.close
set b=Nothing
%> </tr>
</table>
</td>
</tr>
<tr>
<td width="620" height="1" align="center">
<table border="0" cellpadding="0" cellspacing="0" style="border-collapse: collapse"
bordercolor="#111111" id="AutoNumber3" width="387">
<tr>
<td width="65" align="center">
<a href="default.htm" title="Back to Home Page"><font size="2"
face="Arial">Home</font></a></td>
<td width="87" align="center">
<font size="2">&nbsp;<a href="Modify.asp">Course Grade</a></font></td>
<td width="73" align="center">
<font size="2"><a href="trans.asp">Transcript</a></font></td>
</tr>
</table>
</td>
</tr>
</table>
</center>
</div>
<p align="center">&nbsp;</p>

<div align="center">

```



```

<center>
<table border="0" cellpadding="0" cellspacing="0" style="border-collapse: collapse"
bordercolor="#111111" id="AutoNumber5" width="189">
<tr>
<td width="189" align="center"><font face="Arial" size="2">By
<a href="mailto:pansudan9@hotmail.com"><font color="#FF0000">Ibrahim
El-Nour</font></a></font></td>
</tr>
</table>
</center>
</div>

</body>

</html>
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
<html>

<head>
<p align="center">&nbsp;</p>
<p align="center"><b><u><font face="Arial" color="#800000" size="5">NEU
STUDENT
INFORMATION CENTER</font></u></b></p>
<p align="center">&nbsp;</p>
<meta name="GENERATOR" content="Microsoft FrontPage 5.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Modify a Book Entry...</title>
</head>

<body style="font-family: Arial">

<div align="center">
<center>
<table border="1" cellpadding="0" cellspacing="0" style="border-collapse: collapse"
bordercolor="#111111" id="AutoNumber1" width="608" height="168">
<tr>
<td width="606" height="18" align="center"><b>
<font color="#FF0000" size="2">Submit the student number and course code for the
course you
want to know it's result</font></b></td>
</tr>
<tr>
<td width="606" height="132" align="center"><form method="post"
action="M.asp">

```



```

<head>
<p align="center">&nbsp;</p>
<p align="center"><u><font face="Arial" color="#800000" size="5">NEU STUDENT
INFORMATION CENTER</font></u></p>
<p align="center">&nbsp;</p>
<meta name="GENERATOR" content="Microsoft FrontPage 5.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Modify a Book Entry...</title>
</head>

<body style="font-family: Arial">

<div align="center">
<center>
<table border="1" cellpadding="0" cellspacing="0" style="border-collapse: collapse"
bordercolor="#111111" id="AutoNumber1" width="608" height="168">
<tr>
<td width="606" height="18" align="center"><b>
<font color="#FF0000" size="2">Submit the student number for the
transcript you want to view</font></b></td>
</tr>
<tr>
<td width="606" height="132" align="center"><form method="post"
action="transview.asp">
<p>Student Number: <input type="text" name="stno" size="20"></p>
<input type="submit" value="Submit" size="20"></td>
</tr>
<tr>
<td width="606" height="16" align="center">
<table border="0" cellpadding="0" cellspacing="0" style="border-collapse: collapse"
bordercolor="#111111" id="AutoNumber2" width="315">
<tr>
<td width="65" align="center">
<a href="default.htm" title="Back to Home Page"><font size="2"
face="Arial">Home</font></a></td>
<td width="87" align="center">
<font size="2">&nbsp;<a href="Modify.asp">Course Grade</a></font></td>
<td width="73" align="center">
<font size="2"><a href="trans.asp">Transcript</a></font></td>
</tr>
</table>
</td>
</tr>
</table>

```

```

</center>
</div>

<p>&nbsp;</p>
<div align="center">
  <center>
    <table border="0" cellpadding="0" cellspacing="0" style="border-collapse: collapse"
bordercolor="#111111" id="AutoNumber5" width="189">
      <tr>
        <td width="189" align="center"><font face="Arial" size="2">By
          <a href="mailto:pansudan9@hotmail.com"><font color="#FF0000">Ibrahim
            El-Nour</font></a></font></td>
      </tr>
    </table>
  </center>
</div>

</body>

</html>
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
<html>

<head>
<p align="center">&nbsp;</p>
<p align="center"><u><font face="Arial" color="#800000" size="5">NEU STUDENT
INFORMATION CENTER</font></u></p>
<p align="center">&nbsp;</p>
<meta name="GENERATOR" content="Microsoft FrontPage 5.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Updating book entry</title>
</head>

<body style="font-family: Arial">

<div align="center">
  <center>
    <table border="1" cellpadding="0" cellspacing="0" style="border-collapse: collapse"
bordercolor="#111111" id="AutoNumber1" width="622" height="170">
      <tr>
        <td width="620" height="18" align="center"><b>TRANSCRIPT</b></td>
      </tr>
      <tr>
        <td width="620" height="185" align="center">

```



```

        &nbsp;<table border="0" cellpadding="0" cellspacing="0" style="border-collapse:
collapse" bordercolor="#111111" id="AutoNumber2" width="653" height="151">

```

```

    <tr>

```

```

        <td width="653" height="18" align="center">

```

```

<%sn=request.form("stno")

```

```

    set b=server.createobject ("ADODB.connection")

```

```

    b.open "Driver={Microsoft Access Driver (*.mdb)};DBQ=" & "c:\stdb.mdb"

```

```

    c="SELECT * FROM trans WHERE stno=" & sn & " ORDER BY year,sem"

```

```

    set d=b.execute(c)

```

```

    if d.eof then%>

```

```

<p>The submitted <font color="#FF0000"><u><b>This Student does not exist
</b></u></font>, please enter another Student Number </p></td>

```

```

    </tr>

```

```

    <tr>

```

```

        <td width="653" height="133" align="center"><%else%>

```

```

<%=d("stname")%><p>&nbsp;  </p>

```

```

    <TABLE width="639" border="1" style="border-collapse: collapse"

```

```

bordercolor="#111111" cellpadding="0" cellspacing="0">

```

```

    <TR>

```

```

        <TH width="72">

```

```

            year</TH>

```

```

        <TH width="95">

```

```

            Semester</TH>

```

```

        <TH width="81">

```

```

            Cno

```

```

        </TH>

```

```

        <TH width="102">

```

```

            Cname

```

```

        </TH>

```

```

        <TH width="104">

```

```

            Grade</TH>

```

```

    </TR>

```

```

<% Do While Not d.EOF %>

```

```

    <TR>

```

```

        <TD width="72">

```

```

            <%=d("year") %> &nbsp;  </TD>

```

```

        <TD width="95">

```

```

            <%=d("sem") %> &nbsp;  </TD>

```

```

        <TD width="81">

```

```

            <%=d("cno") %> &nbsp;  </TD>

```

```

        <TD width="102">

```

```

            <%=d("cname") %> &nbsp;  </TD>

```

```

        <TD width="104">

```


</div>

</body>

</html>

//
//

REFERENCES

- David Buser, John Kauffman, Juan T Llibre, Brian Francis, David Sussman, Chris Ullman, Jon Duckett. (BEGINNING Active Server Page 3.0)
Reprinted : August 2000
- Karen Kenworthy (Visual Basic for Applications)
REVEALED!
Reprinted: 1994