



NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

**STUDENT REGISTRATION AND ADMINISTRATION
USING ACTIVE SERVER PAGES**

**Graduation Project
COM400**

Student: Hüseyin Ali Şahin (20001089)

Supervisor: Mr. Ümit İlhan

Nicosia-2005

ACKNOWLEDGMENTS

It is my pleasure to take this opportunity to emphasize my great gratitude to many individuals who have given me a lot of supports during my five-year Undergraduation program in the **Near East University**.

First of all, I would like to mention about my thanks to my supervisor **Mr. Ümit İlhan** for supervising me in my project. Under the guidance of him I successfully overcome many difficulties and I learned lots of various techniques about web designing. Also I thank for giving his time during the my study and my advising.

I also want to thank all my friends and specially **Fatih BULUT, Muhammed Akgün**, who supported and helped me all the time.

Finally, special thanks for my family, especially my parents for their encouragement and endless support, and for being patientfull during my undergraduate degree study. I also grateful with everybody who never hesitate about their help and suport.

ABSTRACT

The repaid increase of computer's influence in our daily life. Computer takes an important place for the people. The user can use the service from an internet cafe, from a mobile phone, or any place and device having an internet connection.

This project is a complete student registration and evaluation program for internet, we decided to write on student registration and evaluation program, running on a server and which users can use from any where in the world. The user only needs a browser and internet connection. It allows the admin to register the student, select the lectures that the student has to take, check the debt of the student. Also the student can reach to his/her information.

We made this Project on Active Server Page(ASP) with VBScript of the programming language. Also Microsoft Access is used to store the data of the student.

TABLE OF CONTENTS

| | |
|--------------------------|-----|
| ACKNOWLEDGMENT | i |
| ABSTRACT | ii |
| TABLE OF CONTENTS | iii |
| INTRODUCTION | iv |

CHAPTER ONE : WHAT IS THE WORLD WIDE WEB

| | | |
|-------|--|---|
| 1.1. | World-Wide-Web (WWW) | 1 |
| 1.2. | What is the Internet? | 1 |
| 1.3. | What's it going to do for me? | 2 |
| 1.4. | World Wide Web - What to expect: | 2 |
| 1.5. | HyperText Transport Protocol (http) | 2 |
| 1.6. | Universal Resource Locator (URL) | 3 |
| 1.7. | File Transfer Protocol (FTP) | 4 |
| 1.8. | TCP/IP | 4 |
| 1.9. | Network of Lowest Bidders | 5 |
| 1.10. | What is Online Registration? | 6 |
| 1.11. | What if a course section that I select is not available? | 7 |

CHAPTER TWO: ACTIVE SERVER PAGES & HTML

| | | |
|----------|---|----|
| 2.1. | ACTIVE SERVER PAGES (ASP) | |
| 2.1.1. | The need for ASP | 8 |
| 2.1.2. | What is ASP ? | 8 |
| 2.1.3. | What Can You Do with Active Server Pages? | 9 |
| 2.1.4. | What can ASP do for you? | 9 |
| 2.1.5. | What Do Server-Side Scripts Look Like? | 10 |
| 2.1.6. | What you need to run ASP | 10 |
| 2.1.6.1. | Internet Information Services | 10 |
| 2.1.6.2. | Personal Web Server | 10 |

| | | |
|-----------|--|----|
| 2.1.7. | Quick references before begin | 11 |
| 2.1.7.1. | Steps for Installation | 11 |
| 2.1.7.2. | Creating Virtual Directories | 11 |
| 2.1.7.3. | Accessing your webpage | 11 |
| 2.1.8. | What is localhost? | 12 |
| 2.1.9. | Basic code of asp | 12 |
| 2.1.9.1. | Outputs and Variables | 12 |
| 2.1.9.2. | Sending output to the browse | 12 |
| 2.1.9.3. | Variables | 13 |
| 2.1.9.4. | Variable Operations | 14 |
| 2.1.9.5. | The basics of IF | 16 |
| 2.1.9.6. | Common Comparisions | 16 |
| 2.1.9.7. | Other IF Options | 17 |
| 2.1.9.8. | FOR and NEXT Loops | 18 |
| 2.1.9.9. | Using The Variable | 19 |
| 2.1.9.10. | Step | 19 |
| 2.1.9.11. | While Loops | 20 |
| 2.1.9.12. | Until Loops | 20 |
| 2.2. | HYPER TEXT MARKUP LANGUAGE (HTML) | 21 |
| 2.2.1. | Document Structure | 21 |
| 2.2.2. | HTML Tags | 22 |
| 2.2.2.1. | Containers | 23 |
| 2.2.2.2. | Standalone Tags | 23 |
| 2.2.2.3. | Nesting HTML Tags | 23 |
| 2.2.2.4. | Structural HTML Tags | 24 |

CHAPTER THREE: INTERNET SECURITY

| | | |
|------|-------------------------------|----|
| 3.1. | Introduction | 29 |
| 3.2. | Overview of Internet Security | 30 |
| 3.3. | Basic Security Concepts | 31 |

| | | |
|----------|--|----|
| 3.4. | Why Care About Security? | 32 |
| 3.5 | History | 33 |
| 3.6 | Network Security Incidents | 35 |
| | 3.6.1 Sources of Incidents | 36 |
| | 3.6.2 Types of Incidents | 36 |
| | 3.6.3 Incidents and Internet Growth | 38 |
| | 3.6.4 Incident Trends | 39 |
| 3.7 | Internet Vulnerabilities | 43 |
| | 3.7.1 Why the Internet Is Vulnerable | 43 |
| | 3.7.2 Types of Technical Vulnerabilities | 44 |
| 3.8 | Improving Security | 46 |
| | 3.8.1 Security Policy, Procedures, and Practices | 47 |
| | 3.8.2 Security Technology | 49 |
| 3.9 | Information Warfare The Future | 53 |
| 3.10 | The Future | 54 |
| | 3.10.1 Internetworking Protocols | 54 |
| | 3.10.2 Intrusion Detection | 55 |
| | 3.10.3 Software Engineering and System Survivability | 56 |
| | 3.10.4 Web-Related Programming and Scripting Languages | 57 |
| | 3.10.5 Intelligent Autonomous Agents - A New Computing | 58 |
| Paradigm | | |
| 3.11 | INSTALLING IIS | 59 |
| | 3.11.1 Installing IIS on Windows XP Pro | 59 |
| | 3.11.2 Installing IIS on Windows 2000 Professional | 62 |

4. CHAPTER FOUR: MICROSOFT ACCESS DATABASE

| | | |
|------|----------------------------------|----|
| 4.1. | Introduction to Microsoft Access | 64 |
| 4.2. | The Database Window | 65 |
| 4.3. | Tables | 68 |

| | |
|--|----|
| 4.4. Queries | 68 |
| 4.4 . Brief overview of Relational Databases and Database Applications | 69 |

CHAPTER FIVE: Student Online Registration With ASP Project

| | |
|-------------------|-----|
| CONCLUSION | 150 |
|-------------------|-----|

| | |
|-------------------|-----|
| REFERENCES | 151 |
|-------------------|-----|

| | |
|----------------------------------|----|
| APPENDIX A: PROGRAM CODES | 87 |
|----------------------------------|----|

Introduction

Nowaday's the computer science both hardware and software is being developed over the previous years, programming is always providing the sciences by a systematic development. In our Project we did construct special program related to student registration from the internet. We made to write on student registration and evaluation program, running on a server and which users can use from anywhere in the world. The user only needs a browser and an internet connection.

For the implementation of the project, we used a Windows-based operation system, Windows XP; and Internet Information Server(IIS). The programming language we used was Active Server Pages(ASP) with VBScript. As tools for implementation and debugging we used Macromedia Dreamweaver MX, Internet Explorer, Microsoft Visual InterDev V6.

CHAPTER ONE

1.1 World-Wide-Web (WWW)

The WWW is usually thought of as the future of Internet. The WWW uses hypertext and multimedia and allows the user to "travel" through the net, read text documents, view images, hear sounds, see movies and animation.

The WWW has become so common that you wouldn't be surprised to hear someone say: "Hey when was your last visit to <http://www.somesite.com>"

The World Wide Web which is based on a protocol named HTTP, and it enables access to the information on the Internet, and local information, based on hypertext documents. "Surfing" through the net, using a 'browser' or 'navigator' is made possible by moving from a document or a site to another with hypertext links.

The World Wide Web is split into two parts: The clients and the servers. The servers manage the data and answer requests from the client for that data. The client's application (browser or navigator) enables this connection to the servers to collect the information.

The Web relies on three mechanisms to make these resources readily available to the widest possible audience:

1. A uniform naming scheme for locating resources on the Web
2. Protocols, for access to named resources over the Web
3. Hypertext, for easy navigation among resources

The ties between the three mechanisms are apparent throughout this specification.

1.2 What is the Internet?

The Internet is simply an international computer network (computers from all over the world linked together). The core, or "backbone" of the network consists of computers permanently linked through high-speed connections. To join the Internet, all you have to do is connect your computer to any of these computers. Once you're online (connected) your computer can talk to every other computer on the Internet whether they are in your home town or on the other side of the globe.

1.3 What's it going to do for me?

Having the Internet at your disposal is like having 30 million expert consultants on your payroll (except you don't have to pay them). You can find answers to almost every question you've ever had, send messages across the world instantly, transfer documents, shop, sample new music, visit art galleries, read books, play games, chat, read the latest news in any language, meet people with similar interests, download an almost unlimited variety software, or just "surf" mindlessly through mountains of "visual bubblegum". The Internet will soon become (to many it already has) as integral to business as the telephone and fax machine.

1.4 World Wide Web - What to expect:

The Web is the glossy, glamorous, user-friendly face of the Internet: a media-rich potpourri of virtual shopping malls, music samples, online magazines, art galleries, libraries, museums, games, job agencies, movie previews, and plenty more.

Once you're online, for the most part, it's all free. It's coverage includes over 30,000 companies, everything from Disneyland to Wall Street, and everywhere from Iceland to Johannesburg, all from the keyboard of your computer. If it's not happening on the World Wide Web, it's probably not happening.

1.5 HyperText Transport Protocol (HTTP)

The WWW organizes the information on the Internet, and local files in HYPERTEXT documents which put into use HTML.

Hypertext is a form of presenting information, text, and graphics, where specific words can be expanded to provide other information. These words are the "links" to other documents, which, again, can contain text, files, graphics, sounds, movies. Another way of using the "links" is to direct the user to a different location within the same document. There are no rules about what kind the link would be, or where it would point to. The link is anything and everything the creator of the document finds interesting.

1.6 Universal Resource Locator (URL)

A URL is a text string that holds the type of the source, the Internet address of the server, and the location of the file on that server. Uniform Resource Locators (URLs) enable you to know where any file is, anywhere on the Internet. A URL can be used for directing the browser to it, or as an anchor (link) within an HTML file. The URL provides information on resource, location, path, (and a filename), and also the type of server on which the file is.

The common server types are:

| | | |
|---------------|------------------|-----------|
| HTTP server | identified as | http:// |
| FTP server | | ftp:// |
| TELNET serve | | telnet:// |
| GOPHER server | | gopher:// |
| A local file | is identified as | file:// |

Figure1.1

As an example, the URL for the document you are reading now is:

http://home.cet.com/support/internet_whatism.htm

Where:

[http://](#) specifies the resource as an HTTP server.

home.cet.com where home is the name of the computer and cet.com is the domain (network) in which the machine (or server) is located.

/support/ is the path.

internet_whatism.htm is the name of the file.

OK.

So we know the terms, and we know how the World-Wide-Web uses HyperText, but is that all there is to the internet ??? Of course not! The Internet is most known for the WWW but there's a lot more to it:

Note: The following services were at one time a separate, but indispensable parts of the Internet, but have been replaced (or integrated) by modern Internet "browsers" such as Internet Explorer or Netscape Navigator

1.7 File Transfer Protocol (FTP)

FTP does exactly as the name (or rather the acronym) implies: accesses, and transfers files that are stored on remote computer systems. In Internet "speak", these remote computers are called "SITES". Files on FTP sites are stored within a "tree" of directories (or folders for you mac/win95 users). One of the directories at the "root" would normally be named *PUB*, and its sub-directories will commonly have names that apply to their contents.

When visiting an FTP site, the user must specify the name of the site to log into (such as ftp.cet.com). If that site is meant to be used publicly, the login-name will be anonymous, with your email address as the password. Once logged-in, the user can navigate his way through the directory-tree to the desired directory, select one or more files, and transfer them to your local system.

1.8 TCP/IP

Summary: TCP and IP were developed by a Department of Defense (DOD) research project to connect a number different networks designed by different vendors into a network of networks (the "Internet"). It was initially successful because it delivered a few basic services that everyone needs (file transfer, electronic mail, remote logon) across a very large number of client and server systems. Several computers in a small department can use TCP/IP (along with other protocols) on a single LAN.

The IP component provides routing from the department to the enterprise network, then to regional networks, and finally to the global Internet. On the battlefield a communications network will sustain damage, so the DOD designed TCP/IP to be robust and automatically recover from any node or phone line failure. This design allows the construction of very large networks with less central management. However, because of the automatic recovery, network problems can go undiagnosed and uncorrected for long periods of time.

As with all other communications protocol, TCP/IP is composed of layers:

.IP - is responsible for moving packet of data from node to node. IP forwards each packet based on a four byte destination address (the IP number). The Internet authorities assign ranges of numbers to different organizations. The organizations assign groups of their numbers to departments. IP operates on gateway machines that move data from department to organization to region and then around the world.

.TCP - is responsible for verifying the correct delivery of data from client to server. Data can be lost in the intermediate network. TCP adds support to detect errors or lost data and to trigger retransmission until the data is correctly and completely received.

.Sockets - is a name given to the package of subroutines that provide access to TCP/IP on most systems.

1.9 Network of Lowest Bidders

The Army puts out a bid on a computer and DEC wins the bid. The Air Force puts out a bid and IBM wins. The Navy bid is won by Unisys. Then the President decides to invade Grenada and the armed forces discover that their computers cannot talk to each other. The DOD must build a "network" out of systems each of which, by law, was delivered by the lowest bidder on a single contract.

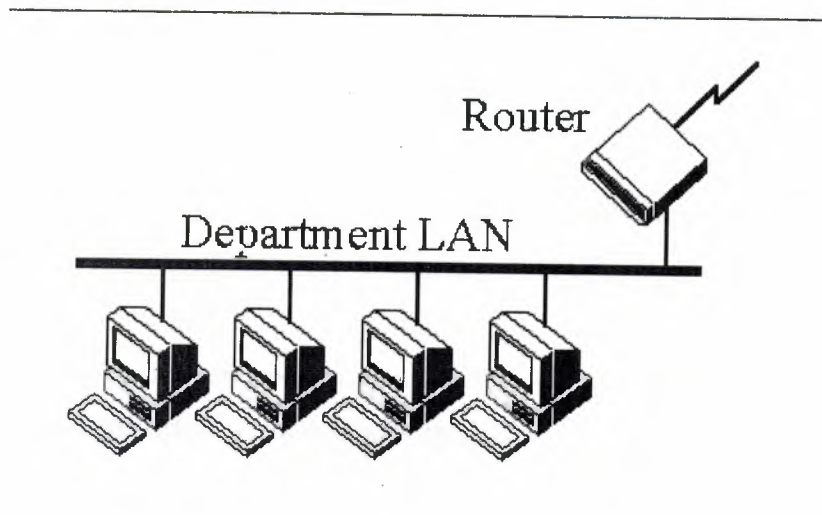


Figure 1.2

The Internet Protocol was developed to create a Network of Networks (the "Internet"). Individual machines are first connected to a LAN (Ethernet or Token Ring). TCP/IP shares the LAN with other uses (a Novell file server, Windows for Workgroups peer systems). One device provides the TCP/IP connection between the LAN and the rest of the world.

To insure that all types of systems from all vendors can communicate, TCP/IP is absolutely standardized on the LAN. However, larger networks based on long distances and phone lines are more volatile. In the US, many large corporations would wish to reuse large internal networks based on IBM's SNA. In Europe, the national phone companies traditionally standardize on X.25. However, the sudden explosion of high speed microprocessors, fiber optics, and digital phone systems has created a burst of new options: ISDN, frame relay, FDDI, Asynchronous Transfer Mode (ATM). New technologies arise and become obsolete within a few years. With cable TV and phone companies competing to build the National Information Superhighway, no single standard can govern citywide, nationwide, or worldwide communications.

The original design of TCP/IP as a Network of Networks fits nicely within the current technological uncertainty. TCP/IP data can be sent across a LAN, or it can be carried within an internal corporate SNA network, or it can piggyback on the cable TV service. Furthermore, machines connected to any of these networks can communicate to any other network through gateways supplied by the network vendor.

1.10 What is Online Registration?

Online Registration is a feature that allows eligible students to register for classes or revise their rosters through DiamondLine using a standard Touch-Tone telephone or via OWLnet a web-based system. A call to the DiamondLine may be placed from a residence hall, from home, from another state or even from another country with compatible Touch-Tone service.

Does Online Registration affect academic advising?

Although advisers' signatures are not required for Online Registration, it is important that you consult your adviser each semester before using these system. Non-matriculated undergraduates must receive approval from their advisers before they will be granted access to the phone system. First semester Freshmen must also see their advisers for similar approval. Registering for inappropriate courses (unsatisfied prerequisites or co-requisites) may result in the removal of these courses from your roster by your Dean's Office.

What will Online Registration allow me to do?

Using the buttons on your Touch-Tone telephone or using a PC, you will be able to:

- Register for the semester
- Add a course to your roster
- Substitute* one course on your roster for another
- Hear or view a list of any courses already on your roster
- Change your password
- Add the payment

Substitute is a transaction that will drop the unwanted section **ONLY** if the new desired section is open and available to you.

1.11 What if a course section that I select is not available?

If the section you select is closed, the system will search for another section of that course being offered at the same time, day, and campus (a "clone"). If a clone is available, the system will inform you that it has automatically registered you in this other section.

If the section you select is closed, and the system determines that there is no clone, there may still be other non-clone sections of that course that will not conflict with your roster. The systems will list these alternate sections, and allow you to pick one that you like. If you are using the DiamondLine you can ask the system to automatically select one of these alternate sections for you.

CHAPTER TWO

ACTIVE SERVER PAGES & HTML

2.1. ACTIVE SERVER PAGES (ASP)

2.1.1. The need for ASP.

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.

2.1.2. 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.

Translated into plain English, that reads - 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.

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.

2.1.3. What Can You 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 widgetry that you see on webpages nowadays can be easily done using ASP.

2.1.4. What can ASP do for you?

- Dynamically edit, change or add any content of a Web page
- Respond to user queries or data submitted from HTML forms
- Access any data or databases and return the results to a browser
- Customize a Web page to make it more useful for individual users
- The advantages of using ASP instead of CGI and Perl, are those of simplicity and speed
- Provides security since your ASP code can not be viewed from the browser
- Since ASP files are returned as plain HTML, they can be viewed in any browser

2.1.5. What Do Server-Side Scripts Look Like?

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.

2.1.6. What you need to run 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.

2.1.6.1. 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.

2.1.6.2. 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.

2.1.7. Quick references before begin

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.

2.1.7.1. 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.

2.1.7.2. 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.")

2.1.7.3. Accessing your webpage

Now that your server is completely configured and ready to use, why not give it a try?
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)

2.1.8. What is 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. Concept is now clear.

2.1.9. BASIC CODE OF ASP

2.1.9.1. Outputs and Variables

2.1.9.2. Sending output to the browser

In this part I will explain what is probably the most important use of ASP: output.

It is always been a tradition of programming tutorials to begin by writing the simple 'Hello World' program, so this one will not make an exception! Sending output is done using the ASP command:

```
Response.Write()
```

so to write 'Hello World' to the user's browser the complete code would be:

```
<%@ Language=VBScript %>  
<%  
Response.Write("Hello World")  
%>
```

Again, this code begins by telling the system that you are writing in VBScript. Then comes the Response.Write command. Basically this is made up of two parts. 'Response' tells the server that you want to send information to the user. There are other types of command including: Request (which gets information from the user), Session (for user session details), Server (for controlling the server) and Application (for commands relating to the application).

More about these later.

The second part, 'Write', tells the server that the type of response you would like to send is to write information to the user's browser. This doesn't just have to be text, but can include variables, which will be discussed in more depth later in this tutorial.

2.1.9.3. Variables

Probably the most important feature of a programming language is a variable. A variable is basically a way of storing text, numbers or other data, so that it can be referenced later. For example, to change the earlier 'Hello World' script:

```
<%@ Language=VBScript %>
<%
OutputText = "Hello World"
Response.Write(OutputText)
%>
```

The output of this code will be exactly the same as the first script, but it is fundamentally different as it uses variables. Basically what this code does follows:

```
OutputText = "Hello World"
```

This line sets up a variable called OutputText and stores in it the string of letters 'Hello World'. As this is now stored in a variable, you can now reference this text you have stored in any part of your script, and you can also manipulate it. The next line:

```
Response.Write(OutputText)
```

tells the server that you are sending information to the browser, and that the information to be sent is the contents of the variable called OutputText. Please note that the variable name is not enclosed in quotation marks. If you did this the browser would simply output the title of the variable as text

There is a second way of outputting the values of variables, other than using Response.Write.

The earlier code could have been written:

```
<%@ Language=VBScript %>
<%
OutputText = "Hello World"
=OutputText
%>
```

In this example, the = sign is used instead of ResponseWrite.

2.1.9.4. Variable Operations

The main benefits to storing information in variables is that you can use the text over and over again. For example, once storing "Hello World" in the variable OutputText, I can then use it in various places in my code:

```
<%@ Language=VBScript %>
<%
OutputText = "Hello World"
%>

This is my <%=OutputText %> script. <Br>

The whole reason for it is to output the text <%=OutputText %> to the browser.
```

which would display in the browser:

This is my Hello World script.

The whole reason for it is to output the text Hello World to the browser.

You can also do various operations on text stored in variables using len, left and right.

The len function simply tells you how many characters are in a string, so if you used the following code:

```
<%=len(OutputText) %>
```

The server would return to the browser the length of the text stored in OutputText, in this case "Hello World", so the browser would display the number 11 on the screen. You could also assign this value to a variable using:

```
<% StringLength = len(OutputText) %>
```

which would set the value of the variable called StringLength to 11.

You can also use the functions left and right. These will display only part of the variable. For example:

```
<% =left(OutputText, 2) %>
```

which would display:

He

and the code:

```
<% =right(OutputText, 4) %>
```

would display:

orld

Basically, these functions take the number of characters specified from the left or right of the string, so left("Some Text", 5) takes the first 5 characters of the text

2.1.9.5. The basics of IF

If statements are used to compare two values and carry out different actions based on the results of the test. If statements take the form IF, THEN, ELSE. Basically the IF part checks for a condition. If it is true, the then statement is executed. If not, the else statement is executed.

IF Structure

The structure of an IF statement is as follows:

```
If something=somethingelse Then  
Execute some code  
Else  
Execute other code  
End If
```


2.1.9.6. Common Comparisons

The ASP IF statement construction is very much like plain text, but here is a quick example of a common use of ASP. In this example the user has entered a password which has been stored in the variable EnteredPassword. The idea of this script is to check whether the user has entered the correct password:

```
<%@ Language=VBScript %>
<%
If EnteredPassword="password1" Then
Response.Write("Well done. You got the password right.")
Else
Response.Write("Sorry. That was the wrong password.")
End If
%>
```

If the user enters the correct password (password1) the text:

Well done. You got password right.

but if you get it incorrect you will be shown the text:

Sorry. That was the wrong password.

2.1.9.7. Other IF Options

There are many of different comparisons you can make with ASP, for example you can compare two variables:

```
If EnteredPassword=RealPassword Then
```

or different types of comparison:

```
If Age>13 Then
```

which will check to see if the age entered by the user is greater than 13.

You can also place HTML etc. in IF statements, as the ASP will continue executing a THEN statement until it reaches an Else or an End If, and will continue to execute Else statements until it reaches End If, for example:

```
<%  
If EnteredPassword="password1" Then  
%>  
<font face="Arial" size="3">Congratulations. You may enter.</font>  
<%  
Else  
%>  
<font face="Arial" size="5" color="Red">ERROR! You cannot enter.</font>  
<%  
End If  
%>
```

1.1.9.8. FOR and NEXT Loops

FOR/NEXT loops are used when you want to execute a piece of code a set number of times. For example, you want to output the word 'Hello' 10 times, you could either code it manually or you could use:

```
<%  
For index = 1 to 10  
Response.Write("Hello")  
Next  
%>
```

Basically, this code says:

For index = 1 to 10

Repeat the following code until variable 'index' is equal to 10, starting at 1 and going up 1 by

Next

This tells the server to return to the beginning of the loop and increment the variable.

2.1.9.9. Using The Variable

A loop isn't much use if it just does the same thing over and over again. It really offers no benefits over a simple piece of code. The real power appears when you use the counter variable in your code. If, for example, I wanted to output the numbers 1 to 10 I could use:

```
<%  
For index = 1 to 10  
Response.Write(index)  
Next  
&>
```

2.1.9.10. Step

Step is an extra part you can add on to the end of the For line of the code to change the way it counts. In the loop above, the code starts by setting index to 1, then when Next is reached it adds another 1 (2), the next time it adds another 1 (3) and so on. Using, STEP you can change this action. For example:

```
<%  
For index = 1 to 10 STEP 2  
Response.Write(index)  
Next  
%>
```

Would output:

246810

It is counting up in 2s. You can also count down:

For index 10 to 1 STEP -1

which will count down from 10 to 1.

2.1.9.11. While Loops

Another type of loop which can be used in ASP is the While loop. A While loop is written as:

```
<%  
Do While thenumber<10  
  Resonse.Write("Less than 10")  
  thenumber = thenumber + 1  
Loop  
%>
```

To explain this code:

Do While thenumber<10

This code first checks if the variable thenumber has a value which is less than 10, then if it is executes the following code until it reaches:

Loop

This tells the code to return to the Do line. Now, you may have noticed the problem here. If all the Do line does is check whether thenumber has the value of less than 10, the loop will go on forever. This is why the line:

thenumber = thenumber + 1

has to be included. This increments the value of thenumber, so that it will eventually be more than 10, and the loop will end. Of course, you aren't just limited to adding and subtracting as

you are with a For loop. You can make any changes to the variable you like in the code.

2.1.9.12. Until Loops

A third type of loop is the Until loop. This is almost exactly the same as the While loop:

```
<%  
Do Until thenumber=10  
Response.Write("Less than 10")  
thenumber = thenumber + 1  
Loop  
%>
```

The difference between this and a While loop is that the code will execute until the condition in the Do line is met, unlike a While loop where it will only execute while the condition is met. As with the While loop you must increment the variable yourself.

2.2. HYPER TEXT MARKUP LANGUAGE (HTML)

HTML, or HyperText Markup Language is designed to specify the logical organisation of a document, with important hypertext extensions. It is not designed to be the language of a WYSIWYG word processor such as Word or WordPerfect. This choice was made because the same HTML document may be viewed by many different "browsers", of very different abilities. Thus, for example, HTML allows you to mark selections of text as titles or paragraphs, and then leaves the interpretation of these marked elements up to the browser. For example one browser may indent the beginning of a paragraph, while another may only leave a blank line.

HTML instructions divide the text of a document into blocks called elements. These can be divided into two broad categories -- those that define how the BODY of the document is to be displayed by the browser, and those that define information 'about' the document, such as the title or relationships to other documents.

When you save an HTML file, you can use either the .htm or the .html extension. We have used .htm in our examples. It might be a bad habit inherited from the past when some of the commonly used software only allowed three letter extensions.

2.2.1. Document Structure

An HTML document contains text (the contents of the page) with embedded tags, which provide instructions for the structure, appearance, and function of the contents.

An HTML document is divided into two major portions: the head and the body. The head contains information about the document, such as its title and “meta” information describing the contents. The body contains the actual contents of the document (the part that is displayed in the browser window).

The following example shows the tags that make up the standard skeletal structure of an HTML document:

```
<HTML>
<HEAD>
<TITLE>Document Title</TITLE>
</HEAD>
<BODY>
  Contents of Document
</BODY>
</HTML>
```

2.2.2. HTML Tags

Every HTML tag is made up of a tag name, sometimes followed by an optional list of attributes, all of which appears between angle brackets < >. Nothing within the brackets will be displayed in the browser. The tag name is generally an abbreviation

of the tag's function (this makes them fairly simple to learn). Attributes are properties that extend or refine the tag's function.

The name and attributes within a tag are not case sensitive. `<BODY BGCOLOR=white>` will work the same as `<body bgcolor=white>`. However, values for particular attributes may be case sensitive, particularly URLs and filenames.

2.2.2.1. Containers

Most HTML tags are containers, meaning they have a beginning (also called “opener” or “start”) tag and an end tag. The text enclosed within the tags will follow the tag's instructions, as in the following example:

The weather is `<I>`gorgeous`</I>` today.

Result: The weather is gorgeous today.

An end tag contains the same name as the start tag, but it is preceded by a slash (/). You can think of it as an “off” switch for the tag. End tags never contain attributes.

For some tags, the end tag is optional and the browser determines when the tag ends by context. This practice is most common with the `<p>` (paragraph) tag. Browsers have supported the `<p>` tag without its end tag, so many web authors take advantage of the shortcut. Not all tags allow this, however, and not all browsers are forgiving, so when in doubt include the end tag. This is especially important when using Cascading Style Sheets with your document.

In the HTML charts that appear in this book, container tags are indicated with the syntax `< >...</>`. If the end tag is optional, it will be so noted in the tag's explanation.

2.2.2.2. Standalone Tags

A few tags do not have end tags because they are used to place standalone elements on the page. The image tag () is such a tag and it simply plops a graphic into the flow of the page. Other standalone tags include the linebreak (
), horizontal rule (<hr>), and tags that provide information about a document and don't affect its displayed content, such as the <meta> and <base> tags.

Attributes

Attributes are added within a tag to extend or modify the tag's actions. You can add multiple attributes within a single tag. Tag attributes, if any, belong after the tag name, each separated by one or more spaces. Their order of appearance is not important.

Most attributes take values, which follow an equal sign (=) after the attribute's name. Values are limited to 1024 characters in length and may be case sensitive. Sometimes the value needs to appear in quotation marks (double or single). Here's how to determine if you need quotation marks around a value:

- If the value is a single word or number, and contains only letters (a-z), numbers (0-9), or the special characters period (.) or hyphen (-), then it is OK to place it directly after the equal sign without quotation marks.
- If the value contains several words separated by commas or spaces, or if it contains any special characters besides a period or hyphen, then it needs to be contained within quotation marks. For example, URLs require quotation marks because they contain the characters "://". Likewise, quotation marks are required around color specifications that take the syntax "#rrggbb".

2.2.2.3. Nesting HTML Tags

HTML tags can be applied to content containing other HTML tags for multiple tag effects on a single element. This is called nesting, and to do it properly, both the beginning and end tags

of the enclosed tag must be completely contained within the beginning and end tags of the applied tag, as follows:

The weather is `<I>gorgeous</I>` today.

Result: The weather is gorgeous today.

This links to `a really coolpage`.

Result: This links to a really cool page.

2.2.2.4. Structural HTML Tags

`<base>`

Specifies the base URL for all relative URLs in the document. Place this within the `<head>` of the document.

Attributes

- `href=url` Specifies the URL to be used.
- `target=name` Defines the default target window for all links in the document. Often used to target frames.

`<body>...</body>`

Defines the beginning and the end of the document body. The body contains the content of the document (the part that is displayed in the browser window).

Attributes to the `<body>` tag affect the entire document.

Attributes

- `alink="#rrggbb"` or color name
Sets the color of active links (i.e., the color while the mouse button is held down during a "click"). Color is specified in hexadecimal RGB values or by standard web color name.

- `background=url`
Provides the URL to a graphic file to be used as a tiling graphic in the background of the document.
- `bgcolor="#rrggb" or color name`
Sets the color of the background for the document. Color is specified in hexadecimal RGB values or by standard web color name.
- `link="#rrggb" or color name`
Sets the default color for all the links in the document. Color is specified in hexadecimal RGB values or by standard web color name.
- `text="#rrggb" or color name`
Sets the default color for all the text in the document. Color is specified in hexadecimal RGB values or by standard web color name.
- `vlink="#rrggb" or color name`
Sets the color of the visited links for the document. Color is specified in hexadecimal RGB values or by standard web color name.

Netscape Navigator 4.0 only

- `marginwidth=number`
Specifies the distance (in number of pixels) between the left browser edge and the beginning of the text and graphics in the window.
- `marginheight=number`
Specifies the distance (in number of pixels) between the top edge of the browser and the top edge of text or graphics in the window.

Internet Explorer only

- `bgproperties="fixed"`

When set to “fixed,” the background image does not scroll with the document content.

- `leftmargin=number`

Specifies the distance (in number of pixels) between the left browser edge and the beginning of the text and graphics in the window.

- `topmargin=number`

Specifies the distance (in number of pixels) between the top edge of the browser and the top edge of text or graphics in the window.

`<head>...</head>`

Defines the head (also called “header”) portion of the document that contains information about the document. The `<head>` tag has no attributes, but serves only as a container for the other header tags, such as `<base>`, `<meta>`, and `<title>`.

`<html>...</html>`

Placed at the beginning and end of the document, this tag tells the browser that the entire document is composed in HTML.

`<link>`

Defines a relationship between the current document and another document. This tag goes within the `<head>` portion of the document. It is often used to refer to an external stylesheet.

Attributes

- `href=url`

Identifies the target document.

- `methods=list`

Specifies a browser-dependent list of comma-separated display methods for this link. It is not commonly used.

- `rev=relation`

Specifies the relationship from the target document to the source.

- rel=relation

Specifies the relationship from the current source document to the target.

- rel=stylesheet

This attribute is used within the <link> tag to create a relationship with an external stylesheet.

- title=text

Provides a title for the target document.

- type=resource

Shows the type of an outside link. The value text/css indicates that the linked document is an external cascading style sheet

- urn=urn

Defines a location-independent Universal Resource Name (URN) for the referenced document. The actual syntax of the URN has not been defined, making this more of a placeholder for future versions of HTML.

<meta>

Provides additional information about the document. It should be placed within the <head> tags at the beginning of the document. It is commonly used for making documents searchable (by adding keywords) and may be used for clientpull functions.

Attributes

- content=text

Specifies the value of the meta tag and is always used in conjunction with name= or http-equiv=.

- http-equiv=text

Specifies information to be included in the HTTP header that the server appends to the document. It is used in conjunction with the name attribute.

- name=text

Specifies a name for the meta information.

- scheme=text

Provides additional information for the interpretation of meta data. This is a new attribute introduced in HTML 4.0<title>...</title>

Specifies the title of the document. The title generally appears in the top bar of the browser window.

CHAPTER THREE

3.INTERNET SECURITY

3.1. Introduction

The vast majority of worms and other successful cyber attacks are made possible by vulnerabilities in a small number of common operating system services. Attackers are opportunistic. They take the easiest and most convenient route and exploit the best-known flaws with the most effective and widely available attack tools. They count on organizations not fixing the problems, and they often attack indiscriminately, scanning the Internet for any vulnerable systems. The easy and destructive spread of worms, such as Blaster, Slammer, and Code Red, can be traced directly to exploitation of unpatched vulnerabilities.

Four years ago, the SANS Institute and the National Infrastructure Protection Center (NIPC) at the FBI released a document summarizing the Ten Most Critical Internet Security Vulnerabilities. Thousands of organizations used that list, and the expanded Top-20 lists that followed one, two, and three years later, to prioritize their efforts so they could close the most dangerous holes first. The vulnerable services that led to worms like Blaster, Slammer, and Code Red, as well as NIMDA worms - are on that list.

This SANS Top-20 2004 is actually two Top Ten lists: the ten most commonly exploited vulnerable services in Windows and the ten most commonly exploited elements in UNIX and Linux environments. Although there are thousands of security incidents each year affecting

these operating systems, the overwhelming majority of successful attacks target one or more of these twenty vulnerable services.

The Top-20 is a consensus list of vulnerabilities that require immediate remediation. It is the result of a process that brought together dozens of leading security experts. They come from the most security-conscious government agencies in the UK, US, and Singapore; the leading security software vendors and consulting firms; the top university-based security programs; many other user organizations; and the SANS Institute. A list of participants may be found at the end of this document.

The SANS Top-20 is a living document. It includes step-by-step instructions and pointers to additional information useful for correcting the security flaws. We will update the list and the instructions as more critical threats and more current or convenient methods of protection are identified, and we welcome your input along the way. This is a community consensus document -- your experience in fighting attackers and in eliminating the vulnerabilities can help others who come after you.

3.2 Overview of Internet Security

As of 1996, the Internet connected an estimated 13 million computers in 195 countries on every continent, even Antarctica (1). The Internet is not a single network, but a worldwide collection of loosely connected networks that are accessible by individual computer hosts in a variety of ways, including gateways, routers, dial-up connections, and Internet service providers. The Internet is easily accessible to anyone with a computer and a network connection. Individuals and organizations worldwide can reach any point on the network without regard to national or geographic boundaries or time of day.

However, along with the convenience and easy access to information come new risks. Among them are the risks that valuable information will be lost, stolen, corrupted, or misused and that the computer systems will be corrupted. If information is recorded electronically and is available on networked computers, it is more vulnerable than if the same information is printed on paper and locked in a file cabinet. Intruders do not need to enter an office or home,

and may not even be in the same country. They can steal or tamper with information without touching a piece of paper or a photocopier. They can create new electronic files, run their own programs, and hide evidence of their unauthorized activity.

3.3 Basic Security Concepts

Three basic security concepts important to information on the Internet are confidentiality, integrity, and availability. Concepts relating to the people who use that information are authentication, authorization, and nonrepudiation.

When information is read or copied by someone not authorized to do so, the result is known as loss of confidentiality. For some types of information, confidentiality is a very important attribute. Examples include research data, medical and insurance records, new product specifications, and corporate investment strategies. In some locations, there may be a legal obligation to protect the privacy of individuals. This is particularly true for banks and loan companies; debt collectors; businesses that extend credit to their customers or issue credit cards; hospitals, doctors' offices, and medical testing laboratories; individuals or agencies that offer services such as psychological counseling or drug treatment; and agencies that collect taxes.

Information can be corrupted when it is available on an insecure network. When information is modified in unexpected ways, the result is known as loss of integrity. This means that unauthorized changes are made to information, whether by human error or intentional tampering. Integrity is particularly important for critical safety and financial data used for activities such as electronic funds transfers, air traffic control, and financial accounting.

Information can be erased or become inaccessible, resulting in loss of availability. This means that people who are authorized to get information cannot get what they need.

Availability is often the most important attribute in service-oriented businesses that depend on information (e.g., airline schedules and online inventory systems). Availability of the network

itself is important to anyone whose business or education relies on a network connection. When a user cannot get access to the network or specific services provided on the network, they experience a denial of service.

To make information available to those who need it and who can be trusted with it, organizations use authentication and authorization. Authentication is proving that a user is whom he or she claims to be. That proof may involve something the user knows (such as a password), something the user has (such as a "smartcard"), or something about the user that proves the person's identity (such as a fingerprint). Authorization is the act of determining whether a particular user (or computer system) has the right to carry out a certain activity, such as reading a file or running a program. Authentication and authorization go hand in hand. Users must be authenticated before carrying out the activity they are authorized to perform. Security is strong when the means of authentication cannot later be refuted - the user cannot later deny that he or she performed the activity. This is known as nonrepudiation.

3.4 Why Care About Security?

It is remarkably easy to gain unauthorized access to information in an insecure networked environment, and it is hard to catch the intruders. Even if users have nothing stored on their computer that they consider important, that computer can be a "weak link", allowing unauthorized access to the organization's systems and information.

Seemingly innocuous information can expose a computer system to compromise. Information that intruders find useful includes which hardware and software are being used, system configuration, type of network connections, phone numbers, and access and authentication procedures. Security-related information can enable unauthorized individuals to get access to important files and programs, thus compromising the security of the system. Examples of important information are passwords, access control files and keys, personnel information, and encryption algorithms.

Judging from CERT[®] Coordination Center (CERT/CC) data and the computer abuse reported in the media, no one on the Internet is immune. Those affected include banks and financial companies, insurance companies, brokerage houses, consultants, government contractors,

government agencies, hospitals and medical laboratories, network service providers, utility companies, the textile business, universities, and wholesale and retail trades.

The consequences of a break-in cover a broad range of possibilities: a minor loss of time in recovering from the problem, a decrease in productivity, a significant loss of money or staff-hours, a devastating loss of credibility or market opportunity, a business no longer able to compete, legal liability, and the loss of life.

3.5 History

The Internet began in 1969 as the ARPANET, a project funded by the Advanced Research Projects Agency (ARPA) of the U.S. Department of Defense. One of the original goals of the project was to create a network that would continue to function even if major sections of the network failed or were attacked. The ARPANET was designed to reroute network traffic automatically around problems in connecting systems or in passing along the necessary information to keep the network functioning. Thus, from the beginning, the Internet was designed to be robust against denial-of-service attacks, which are described in a section below on denial of service.

The ARPANET protocols (the rules of syntax that enable computers to communicate on a network) were originally designed for openness and flexibility, not for security. The ARPA researchers needed to share information easily, so everyone needed to be an unrestricted "insider" on the network. Although the approach was appropriate at the time, it is not one that lends itself to today's commercial and government use.

As more locations with computers (known as sites in Internet parlance) joined the ARPANET, the usefulness of the network grew. The ARPANET consisted primarily of university and government computers, and the applications supported on this network were simple: electronic mail (E-mail), electronic news groups, and remote connection to other computers. By 1971, the Internet linked about two dozen research and government sites, and researchers had begun to use it to exchange information not directly related to the ARPANET itself. The network was becoming an important tool for collaborative research.

During these years, researchers also played "practical jokes" on each other using the ARPANET. These jokes usually involved joke messages, annoying messages, and other minor security violations. Some of these are described in Steven Levy's *Hackers: Heroes of the Computer Revolution*. It was rare that a connection from a remote system was considered an attack, however, because ARPANET users comprised a small group of people who generally knew and trusted each other.

In 1986, the first well-publicized international security incident was identified by Cliff Stoll, then of Lawrence Berkeley National Laboratory in northern California. A simple accounting error in the computer records of systems connected to the ARPANET led Stoll to uncover an international effort, using the network, to connect to computers in the United States and copy information from them. These U.S. computers were not only at universities, but at military and government sites all over the country. When Stoll published his experience in a 1989 book, *The Cuckoo's Egg*, he raised awareness that the ARPANET could be used for destructive purposes.

In 1988, the ARPANET had its first automated network security incident, usually referred to as "the Morris worm" (4). A student at Cornell University (Ithaca, NY), Robert T. Morris, wrote a program that would connect to another computer, find and use one of several vulnerabilities to copy itself to that second computer, and begin to run the copy of itself at the new location. Both the original code and the copy would then repeat these actions in an infinite loop to other computers on the ARPANET. This "self-replicating automated network attack tool" caused a geometric explosion of copies to be started at computers all around the ARPANET. The worm used so many system resources that the attacked computers could no longer function. As a result, 10% of the U.S. computers connected to the ARPANET effectively stopped at about the same time.

By that time, the ARPANET had grown to more than 88,000 computers and was the primary means of communication among network security experts. With the ARPANET effectively down, it was difficult to coordinate a response to the worm. Many sites removed themselves from the ARPANET altogether, further hampering communication and the transmission of the solution that would stop the worm.

The Morris worm prompted the Defense Advanced Research Projects Agency (DARPA, the new name for ARPA) to fund a computer emergency response team, now the CERT® Coordination Center, to give experts a central point for coordinating responses to network emergencies. Other teams quickly sprang up to address computer security incidents in specific organizations or geographic regions. Within a year of their formation, these incident response teams created an informal organization now known as the Forum of Incident Response and Security Teams (FIRST). These teams and the FIRST organization exist to coordinate responses to computer security incidents, assist sites in handling attacks, and educate network users about computer security threats and preventive practices.

In 1989, the ARPANET officially became the Internet and moved from a government research project to an operational network; by then it had grown to more than 100,000 computers. Security problems continued, with both aggressive and defensive technologies becoming more sophisticated. Among the major security incidents (5) were the 1989 WANK/OILZ worm, an automated attack on VMS systems attached to the Internet, and exploitation of vulnerabilities in widely distributed programs such as the sendmail program, a complicated program commonly found on UNIX-based systems for sending and receiving electronic mail. In 1994, intruder tools were created to "sniff" packets from the network easily, resulting in the widespread disclosure of user names and password information. In 1995, the method that Internet computers use to name and authenticate each other was exploited by a new set of attack tools that allowed widespread Internet attacks on computers that have trust relationships (see the section on exploitation of trust, below) with any other computer, even one in the same room. Today the use of the World Wide Web and Web-related programming languages create new opportunities for network attacks.

Although the Internet was originally conceived of and designed as a research and education network, usage patterns have radically changed. The Internet has become a home for private and commercial communication, and at this writing it is still expanding into important areas of commerce, medicine, and public service. Increased reliance on the Internet is expected over the next five years, along with increased attention to its security.

3.6 Network Security Incidents

A network security incident is any network-related activity with negative security implications. This usually means that the activity violates an explicit or implicit security policy (see the section on security policy). Incidents come in all shapes and sizes. They can come from anywhere on the Internet, although some attacks must be launched from specific systems or networks and some require access to special accounts. An intrusion may be a comparatively minor event involving a single site or a major event in which tens of thousands of sites are compromised. (When reading accounts of incidents, note that different groups may use different criteria for determining the bounds of an incident.)

A typical attack pattern consists of gaining access to a user's account, gaining privileged access, and using the victim's system as a launch platform for attacks on other sites. It is possible to accomplish all these steps manually in as little as 45 seconds; with automation, the time decreases further.

3.6.1 Sources of Incidents

It is difficult to characterize the people who cause incidents. An intruder may be an adolescent who is curious about what he or she can do on the Internet, a college student who has created a new software tool, an individual seeking personal gain, or a paid "spy" seeking information for the economic advantage of a corporation or foreign country. An incident may also be caused by a disgruntled former employee or a consultant who gained network information while working with a company. An intruder may seek entertainment, intellectual challenge, a sense of power, political attention, or financial gain.

One characteristic of the intruder community as a whole is its communication. There are electronic newsgroups and print publications on the latest intrusion techniques, as well as conferences on the topic. Intruders identify and publicize misconfigured systems; they use those systems to exchange pirated software, credit card numbers, exploitation programs, and the identity of sites that have been compromised, including account names and passwords. By sharing knowledge and easy-to-use software tools, successful intruders increase their number and their impact.

3.6.2 Types of Incidents

Incidents can be broadly classified into several kinds: the probe, scan, account compromise, root compromise, packet sniffer, denial of service, exploitation of trust, malicious code, and Internet infrastructure attacks.

Probe

A probe is characterized by unusual attempts to gain access to a system or to discover information about the system. One example is an attempt to log in to an unused account. Probing is the electronic equivalent of testing doorknobs to find an unlocked door for easy entry. Probes are sometimes followed by a more serious security event, but they are often the result of curiosity or confusion.

Scan

A scan is simply a large number of probes done using an automated tool. Scans can sometimes be the result of a misconfiguration or other error, but they are often a prelude to a more directed attack on systems that the intruder has found to be vulnerable.

Account Compromise

An account compromise is the unauthorized use of a computer account by someone other than the account owner, without involving system-level or root-level privileges (privileges a system administrator or network manager has). An account compromise might expose the victim to serious data loss, data theft, or theft of services. The lack of root-level access means that the damage can usually be contained, but a user-level account is often an entry point for greater access to the system.

Root Compromise

A root compromise is similar to an account compromise, except that the account that has been compromised has special privileges on the system. The term root is derived from an account on UNIX systems that typically has unlimited, or "superuser", privileges. Intruders who succeed in a root compromise can do just about anything on the victim's system, including run their own programs, change how the system works, and hide traces of their intrusion.

Packet Sniffer

A packet sniffer is a program that captures data from information packets as they travel over the network. That data may include user names, passwords, and proprietary information that travels over the network in clear text. With perhaps hundreds or thousands of passwords captured by the sniffer, intruders can launch widespread attacks on systems. Installing a packet sniffer does not necessarily require privileged access. For most multi-user systems, however, the presence of a packet sniffer implies there has been a root compromise.

Denial of Service

The goal of denial-of-service attacks is not to gain unauthorized access to machines or data, but to prevent legitimate users of a service from using it. A denial-of-service attack can come in many forms. Attackers may "flood" a network with large volumes of data or deliberately consume a scarce or limited resource, such as process control blocks or pending network connections. They may also disrupt physical components of the network or manipulate data in transit, including encrypted data.

Exploitation of Trust

Computers on networks often have trust relationships with one another. For example, before executing some commands, the computer checks a set of files that specify which other computers on the network are permitted to use those commands. If attackers can forge their identity, appearing to be using the trusted computer, they may be able to gain unauthorized access to other computers.

Malicious Code

Malicious code is a general term for programs that, when executed, would cause undesired results on a system. Users of the system usually are not aware of the program until they discover the damage. Malicious code includes Trojan horses, viruses, and worms. Trojan horses and viruses are usually hidden in legitimate programs or files that attackers have altered to do more than what is expected. Worms are self-replicating programs that spread with no human intervention after they are started. Viruses are also self-replicating programs, but usually require some action on the part of the user to spread inadvertently to other programs or systems. These sorts of programs can lead to serious data loss, downtime, denial of service, and other types of security incidents.

Internet Infrastructure Attacks

These rare but serious attacks involve key components of the Internet infrastructure rather than specific systems on the Internet. Examples are network name servers, network access providers, and large archive sites on which many users depend. Widespread automated attacks can also threaten the infrastructure. Infrastructure attacks affect a large portion of the Internet and can seriously hinder the day-to-day operation of many sites.

3.6.3 Incidents and Internet Growth

Since the CERT[®] Coordination Center began operating in 1988, the number of security incidents reported to the center has grown dramatically, from less than 100 in 1988 to almost 2,500 in 1995, the last year for which complete statistics are available as of this writing. Through 1994, the increase in incident reports roughly parallels the growth of the size of the Internet during that time. Figure 1 shows the growth of the Internet and the corresponding growth of reported security incidents.

Growth in Security Incidents

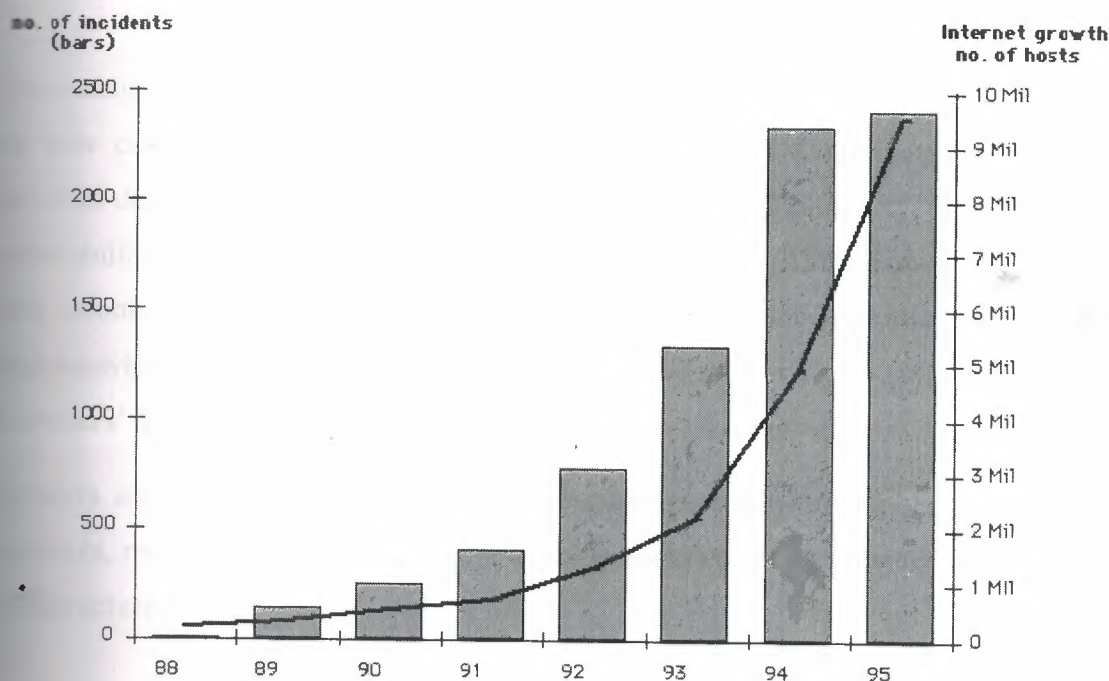


Figure 3.1

The data for 1995 and partial data for 1996 show a slowing of the rate at which incidents are reported to the CERT/CC (perhaps because of sites' increased security efforts or the significant increase in other response teams formed to handle incidents). However, the rate continues to increase for serious incidents, such as root compromises, services outages, and packet sniffers.

3.6.4 Incident Trends

In the late 1980s and early 1990s, the typical intrusion was fairly straightforward. Intruders most often exploited relatively simple weaknesses, such as poor passwords and misconfigured

systems, that allowed greater access to the system than was intended. Once on a system, the intruders exploited one or another well-known, but usually unfixed, vulnerability to gain privileged access, enabling them to use the system as they wished.

There was little need to be more sophisticated because these simple techniques were effective. Vendors delivered systems with default settings that made it easy to break into systems. Configuring systems in a secure manner was not straightforward, and many system administrators did not have the time, expertise, or tools to monitor their systems adequately for intruder activity.

Unfortunately, all these activities continue in 1996; however, more sophisticated intrusions are now common. In eight years of operation, the CERT Coordination Center has seen intruders demonstrate increased technical knowledge, develop new ways to exploit system vulnerabilities, and create software tools to automate attacks. At the same time, intruders with little technical knowledge are becoming more effective as the sophisticated intruders share their knowledge and tools.

Intruders' Technical Knowledge

Intruders are demonstrating increased understanding of network topology, operations, and protocols, resulting in the infrastructure attacks described in the previous section on Internet infrastructure attacks.

Instead of simply exploiting well-known vulnerabilities, intruders examine source code to discover weaknesses in certain programs, such as those used for electronic mail. Much source code is easy to obtain from programmers who make their work freely available on the Internet. Programs written for research purposes (with little thought for security) or written by naive programmers become widely used, with source code available to all. Moreover, the targets of many computer intrusions are organizations that maintain copies of proprietary source code (often the source code to computer operating systems or key software utilities). Once intruders gain access, they can examine this code to discover weaknesses.

Intruders keep up with new technology. For example, intruders now exploit vulnerabilities associated with the World Wide Web to gain unauthorized access to systems.

Other aspects of the new sophistication of intruders include the targeting of the network infrastructure (such as network routers and firewalls) and the ability to cloak their behavior. Intruders use Trojan horses to hide their activity from network administrators; for example, intruders alter authentication and logging programs so that they can log in without the activity showing up in the system logs. Intruders also encrypt output from their activity, such as the information captured by packet sniffers. Even if the victim finds the sniffer logs, it is difficult or impossible to determine what information was compromised.

Techniques to Exploit Vulnerabilities

As intruders become more sophisticated, they identify new and increasingly complex methods of attack. For example, intruders are developing sophisticated techniques to monitor the Internet for new connections. Newly connected systems are often not fully configured from a security perspective and are, therefore, vulnerable to attacks.

The most widely publicized of the newer types of intrusion is the use of the packet sniffers described in the section above on packet sniffers. Other tools are used to construct packets with forged addresses; one use of these tools is to mount a denial-of-service attack in a way that obscures the source of the attack. Intruders also "spoof" computer addresses, masking their real identity and successfully making connections that would not otherwise be permitted. In this way, they exploit trust relationships between computers.

With their sophisticated technical knowledge and understanding of the network, intruders are increasingly exploiting network interconnections. They move through the Internet infrastructure, attacking areas on which many people and systems depend. Infrastructure attacks are even more threatening because legitimate network managers and administrators typically think about protecting systems and parts of the infrastructure rather than the infrastructure as a whole.

In the first quarter of 1996, 7.5% of 346 incidents handled by the CERT Coordination Center involved these new and sophisticated methods, including packet sniffers, spoofing, and infrastructure attacks. A full 20% involved the total compromise of systems, in which intruders gain system-level, or root, privileges. This represents a significant increase in such attacks over previous years' attacks, and the numbers are still rising. Of 341 incidents in the

third quarter of 1996, nearly 9% involved sophisticated attacks, and root compromises accounted for 33%.

Intruders' Use of Software Tools

The tools available to launch an attack have become more effective, easier to use, and more accessible to people without an in-depth knowledge of computer systems. Often a sophisticated intruder embeds an attack procedure in a program and widely distributes it to the intruder community. Thus, people who have the desire but not the technical skill are able to break into systems. Indeed, there have been instances of intruders breaking into a UNIX system using a relatively sophisticated attack and then attempting to run DOS commands (commands that apply to an entirely different operating system).

Tools are available to examine programs for vulnerabilities even in the absence of source code. Though these tools can help system administrators identify problems, they also help intruders find new ways to break into systems.

As in many areas of computing, the tools used by intruders have become more automated, allowing intruders to gather information about thousands of Internet hosts quickly and with minimum effort. These tools can scan entire networks from a remote location and identify individual hosts with specific weaknesses. Intruders may catalog the information for later exploitation, share or trade with other intruders, or attack immediately. The increased availability and usability of scanning tools means that even technically naive, would-be intruders can find new sites and particular vulnerabilities.

Some tools automate multiphase attacks in which several small components are combined to achieve a particular end. For example, intruders can use a tool to mount a denial-of-service attack on a machine and spoof that machine's address to subvert the intended victim's machine. A second example is using a packet sniffer to get router or firewall passwords, logging in to the firewall to disable filters, then using a network file service to read data on an otherwise secure server.

The trend toward automation can be seen in the distribution of software packages containing a variety of tools to exploit vulnerabilities. These packages are often maintained by competent programmers and are distributed complete with version numbers and documentation.

A typical tool package might include the following:

- network scanner
- password cracking tool and large dictionaries
- packet sniffer
- variety of Trojan horse programs and libraries
- tools for selectively modifying system log files
- tools to conceal current activity
- tools for automatically modifying system configuration files
- tools for reporting bogus checksums

3.7 Internet Vulnerabilities

A vulnerability is a weakness that a person can exploit to accomplish something that is not authorized or intended as legitimate use of a network or system. When a vulnerability is exploited to compromise the security of systems or information on those systems, the result is a security incident. Vulnerabilities may be caused by engineering or design errors, or faulty implementation.

3.7.1 Why the Internet Is Vulnerable

Many early network protocols that now form part of the Internet infrastructure were designed without security in mind. Without a fundamentally secure infrastructure, network defense becomes more difficult. Furthermore, the Internet is an extremely dynamic environment, in terms of both topology and emerging technology.

Because of the inherent openness of the Internet and the original design of the protocols, Internet attacks in general are quick, easy, inexpensive, and may be hard to detect or trace. An attacker does not have to be physically present to carry out the attack. In fact, many attacks can be launched readily from anywhere in the world - and the location of the attacker can easily be hidden. Nor is it always necessary to "break in" to a site (gain privileges on it) to compromise confidentiality, integrity, or availability of its information or service.

Even so, many sites place unwarranted trust in the Internet. It is common for sites to be unaware of the risks or unconcerned about the amount of trust they place in the Internet. They may not be aware of what can happen to their information and systems. They may believe that their site will not be a target or that precautions they have taken are sufficient. Because the technology is constantly changing and intruders are constantly developing new tools and techniques, solutions do not remain effective indefinitely.

Since much of the traffic on the Internet is not encrypted, confidentiality and integrity are difficult to achieve. This situation undermines not only applications (such as financial applications that are network-based) but also more fundamental mechanisms such as authentication and nonrepudiation (see the section on basic security concepts for definitions).

As a result, sites may be affected by a security compromise at another site over which they have no control. An example of this is a packet sniffer that is installed at one site but allows the intruder to gather information about other domains (possibly in other countries).

Another factor that contributes to the vulnerability of the Internet is the rapid growth and use of the network, accompanied by rapid deployment of network services involving complex applications. Often, these services are not designed, configured, or maintained securely. In the rush to get new products to market, developers do not adequately ensure that they do not repeat previous mistakes or introduce new vulnerabilities.

Compounding the problem, operating system security is rarely a purchase criterion. Commercial operating system vendors often report that sales are driven by customer demand for performance, price, ease of use, maintenance, and support. As a result, off-the-shelf operating systems are shipped in an easy-to-use but insecure configuration that allows sites to use the system soon after installation. These hosts/sites are often not fully configured from a security perspective before connecting. This lack of secure configuration makes them vulnerable to attacks, which sometimes occur within minutes of connection.

Finally, the explosive growth of the Internet has expanded the need for well-trained and experienced people to engineer and manage the network in a secure manner. Because the need for network security experts far exceeds the supply, inexperienced people are called upon to secure systems, opening windows of opportunity for the intruder community.

3.7.2 Types of Technical Vulnerabilities

The following taxonomy is useful in understanding the technical causes behind successful intrusion techniques, and helps experts identify general solutions for addressing each type of problem.

Flaws in Software or Protocol Designs

Protocols define the rules and conventions for computers to communicate on a network. If a protocol has a fundamental design flaw, it is vulnerable to exploitation no matter how well it is implemented. An example of this is the Network File System (NFS), which allows systems to share files. This protocol does not include a provision for authentication; that is, there is no way of verifying that a person logging in really is whom he or she claims to be. NFS servers are targets for the intruder community.

When software is designed or specified, often security is left out of the initial description and is later "added on" to the system. Because the additional components were not part of the original design, the software may not behave as planned and unexpected vulnerabilities may be present.

Weaknesses in How Protocols and Software Are Implemented

Even when a protocol is well designed, it can be vulnerable because of the way it is implemented. For example, a protocol for electronic mail may be implemented in a way that permits intruders to connect to the mail port of the victim's machine and fool the machine into performing a task not intended by the service. If intruders supply certain data for the "To:" field instead of a correct E-mail address, they may be able to fool the machine into sending them user and password information or granting them access to the victim's machine with privileges to read protected files or run programs on the system. This type of vulnerability enables intruders to attack the victim's machine from remote sites without access to an account on the victim's system. This type of attack often is just a first step, leading to the exploitation of flaws in system or application software.

Software may be vulnerable because of flaws that were not identified before the software was released. This type of vulnerability has a wide range of subclasses, which intruders often exploit using their own attack tools. For readers who are familiar with software design, the

following examples of subclasses are included:

- race conditions in file access
- non-existent checking of data content and size
- non-existent checking for success or failure
- inability to adapt to resource exhaustion
- incomplete checking of operating environment
- inappropriate use of system calls
- re-use of software modules for purposes other than their intended ones

By exploiting program weaknesses, intruders at a remote site can gain access to a victim's system. Even if they have access to a nonprivileged user account on the victim's system, they can often gain additional, unauthorized privileges.

Weaknesses in System and Network Configurations

Vulnerabilities in the category of system and network configurations are not caused by problems inherent in protocols or software programs. Rather, the vulnerabilities are a result of the way these components are set up and used. Products may be delivered with default settings that intruders can exploit. System administrators and users may neglect to change the default settings, or they may simply set up their system to operate in a way that leaves the network vulnerable.

An example of a faulty configuration that has been exploited is anonymous File Transfer Protocol (FTP) service. Secure configuration guidelines for this service stress the need to ensure that the password file, archive tree, and ancillary software are separate from the rest of the operating system, and that the operating system cannot be reached from this staging area.

When sites misconfigure their anonymous FTP archives, unauthorized users can get authentication information and use it to compromise the system.

3.8 Improving Security

In the face of the vulnerabilities and incident trends discussed above, a robust defense requires a flexible strategy that allows adaptation to the changing environment, well-defined policies and procedures, the use of robust tools, and constant vigilance.

It is helpful to begin a security improvement program by determining the current state of security at the site. Methods for making this determination in a reliable way are becoming available. Integral to a security program are documented policies and procedures, and technology that supports their implementation.

3.8.1 Security Policy, Procedures, and Practices

Security Policy

A policy is a documented high-level plan for organization-wide computer and information security. It provides a framework for making specific decisions, such as which defense mechanisms to use and how to configure services, and is the basis for developing secure programming guidelines and procedures for users and system administrators to follow. Because a security policy is a long-term document, the contents avoid technology-specific issues.

A security policy covers the following (among other topics appropriate to the organization): high-level description of the technical environment of the site, the legal environment (governing laws), the authority of the policy, and the basic philosophy to be used when interpreting the policy risk analysis that identifies the site's assets, the threats that exist against those assets, and the costs of asset loss guidelines for system administrators on how to manage systems definition of acceptable use for users guidelines for reacting to a site compromise (e.g., how to deal with the media and law enforcement, and whether to trace the intruder or shutdown and rebuild the system)

Factors that contribute to the success of a security policy include management commitment, technological support for enforcing the policy, effective dissemination of the policy, and the

security awareness of all users. Management assigns responsibility for security, provides training for security personnel, and allocates funds to security. Technological support for the security policy moves some responsibility for enforcement from individuals to technology.

The result is an automatic and consistent enforcement of policies, such as those for access and authentication. Technical options that support policy include (but are not limited to) challenge/response systems for authentication auditing systems for accountability and event reconstruction encryption systems for the confidential storage and transmission of data network tools such as firewalls and proxy servers There are many books and papers devoted to site security policies, including requests for comments RFC 1244 (6) and RFC 1281 (7), guidelines written by the Internet Engineering Task Force.

Security-Related Procedures

Procedures are specific steps to follow that are based on the computer security policy. Procedures address such topics as retrieving programs from the network, connecting to the site's system from home or while traveling, using encryption, authentication for issuing accounts, configuration, and monitoring.

Security Practices

System administration practices play a key role in network security. Checklists and general advice on good security practices are readily available. Below are examples of commonly recommended practices:

Ensure all accounts have a password and that the passwords are difficult to guess. A one-time password system is preferable.

Use tools such as MD5 checksums (8), a strong cryptographic technique, to ensure the integrity of system software on a regular basis.

Use secure programming techniques when writing software. These can be found at security-related sites on the World Wide Web.

Be vigilant in network use and configuration, making changes as vulnerabilities become known.

Regularly check with vendors for the latest available fixes and keep systems current with upgrades and patches.

Regularly check on-line security archives, such as those maintained by incident response teams, for security alerts and technical advice.

Audit systems and networks, and regularly check logs. Many sites that suffer computer security incidents report that insufficient audit data is collected, so detecting and tracing an intrusion is difficult.

3.8.2 Security Technology

A variety of technologies have been developed to help organizations secure their systems and information against intruders. These technologies help protect systems and information against attacks, detect unusual or suspicious activities, and respond to events that affect security. In this section, the focus is on two core areas: operational technology and cryptography. The purpose of operational technology is to maintain and defend the availability of data resources in a secure manner. The purpose of cryptography is to secure the confidentiality, integrity, and authenticity of data resources.

Operational Technology

Intruders actively seek ways to access networks and hosts. Armed with knowledge about specific vulnerabilities, social engineering techniques, and tools to automate information gathering and systems infiltration, intruders can often gain entry into systems with disconcerting ease. System administrators face the dilemma of maximizing the availability of system services to valid users while minimizing the susceptibility of complex network infrastructures to attack. Unfortunately, services often depend on the same characteristics of systems and network protocols that make them susceptible to compromise by intruders. In response, technologies have evolved to reduce the impact of such threats. No single technology addresses all the problems. Nevertheless, organizations can significantly improve their resistance to attack by carefully preparing and strategically deploying personnel and operational technologies. Data resources and assets can be protected, suspicious activity can be detected and assessed, and appropriate responses can be made to security events as they occur.



One-Time Passwords

Intruders often install packet sniffers to capture passwords as they traverse networks during remote log-in processes. Therefore, all passwords should at least be encrypted as they traverse networks. A better solution is to use one-time passwords because there are times when a password is required to initiate a connection before confidentiality can be protected.

One common example occurs in remote dial-up connections. Remote users, such as those traveling on business, dial in to their organization's modem pool to access network and data resources. To identify and authenticate themselves to the dial-up server, they must enter a user ID and password. Because this initial exchange between the user and server may be monitored by intruders, it is essential that the passwords are not reusable. In other words, intruders should not be able to gain access by masquerading as a legitimate user using a password they have captured.

One-time password technologies address this problem. Remote users carry a device synchronized with software and hardware on the dial-up server. The device displays random passwords, each of which remains in effect for a limited time period (typically 60 seconds).

These passwords are never repeated and are valid only for a specific user during the period that each is displayed. In addition, users are often limited to one successful use of any given password. One-time password technologies significantly reduce unauthorized entry at gateways requiring an initial password.

Firewalls

Intruders often attempt to gain access to networked systems by pretending to initiate connections from trusted hosts. They squash the emissions of the genuine host using a denial-of-service attack and then attempt to connect to a target system using the address of the genuine host. To counter these address-spoofing attacks and enforce limitations on authorized connections into the organization's network, it is necessary to filter all incoming and outgoing network traffic.

A firewall is a collection of hardware and software designed to examine a stream of network traffic and service requests. Its purpose is to eliminate from the stream those packets or requests that fail to meet the security criteria established by the organization. A simple firewall may consist of a filtering router, configured to discard packets that arrive from unauthorized addresses or that represent attempts to connect to unauthorized service ports.

More sophisticated implementations may include bastion hosts, on which proxy mechanisms operate on behalf of services. These mechanisms authenticate requests, verify their form and content, and relay approved service requests to the appropriate service hosts. Because firewalls are typically the first line of defense against intruders, their configuration must be carefully implemented and tested before connections are established between internal networks and the Internet.

Monitoring Tools

Continuous monitoring of network activity is required if a site is to maintain confidence in the security of its network and data resources. Network monitors may be installed at strategic locations to collect and examine information continuously that may indicate suspicious activity. It is possible to have automatic notifications alert system administrators when the monitor detects anomalous readings, such as a burst of activity that may indicate a denial-of-service attempt. Such notifications may use a variety of channels, including electronic mail and mobile paging. Sophisticated systems capable of reacting to questionable network activity may be implemented to disconnect and block suspect connections, limit or disable affected services, isolate affected systems, and collect evidence for subsequent analysis.

Tools to scan, monitor, and eradicate viruses can identify and destroy malicious programs that may have inadvertently been transmitted onto host systems. The damage potential of viruses ranges from mere annoyance (e.g., an unexpected "Happy Holidays" jingle without further effect) to the obliteration of critical data resources. To ensure continued protection, the virus identification data on which such tools depend must be kept up to date. Most virus tool vendors provide subscription services or other distribution facilities to help customers keep up to date with the latest viral strains.

Security Analysis Tools

Because of the increasing sophistication of intruder methods and the vulnerabilities present in commonly used applications, it is essential to assess periodically network susceptibility to compromise. A variety of vulnerability identification tools are available, which have garnered

both praise and criticism. System administrators find these tools useful in identifying weaknesses in their systems. Critics argue that such tools, especially those freely available to the Internet community, pose a threat if acquired and misused by intruders.

Cryptography

One of the primary reasons that intruders can be successful is that most of the information they acquire from a system is in a form that they can read and comprehend. When you consider the millions of electronic messages that traverse the Internet each day, it is easy to see how a well-placed network sniffer might capture a wealth of information that users would not like to have disclosed to unintended readers. Intruders may reveal the information to others, modify it to misrepresent an individual or organization, or use it to launch an attack. One solution to this problem is, through the use of cryptography, to prevent intruders from being able to use the information that they capture.

Encryption is the process of translating information from its original form (called plaintext) into an encoded, incomprehensible form (called ciphertext). Decryption refers to the process of taking ciphertext and translating it back into plaintext. Any type of data may be encrypted, including digitized images and sounds.

Cryptography secures information by protecting its confidentiality. Cryptography can also be used to protect information about the integrity and authenticity of data. For example, checksums are often used to verify the integrity of a block of information. A checksum, which is a number calculated from the contents of a file, can be used to determine if the contents are correct. An intruder, however, may be able to forge the checksum after modifying the block of information. Unless the checksum is protected, such modification might not be detected. Cryptographic checksums (also called message digests) help prevent undetected modification of information by encrypting the checksum in a way that makes the checksum unique.

The authenticity of data can be protected in a similar way. For example, to transmit information to a colleague by E-mail, the sender first encrypts the information to protect its confidentiality and then attaches an encrypted digital signature to the message. When the colleague receives the message, he or she checks the origin of the message by using a key to

verify the sender's digital signature and decrypts the information using the corresponding decryption key. To protect against the chance of intruders modifying or forging the information in transit, digital signatures are formed by encrypting a combination of a checksum of the information and the author's unique private key. A side effect of such authentication is the concept of nonrepudiation. A person who places their cryptographic digital signature on an electronic document cannot later claim that they did not sign it, since in theory they are the only one who could have created the correct signature.

Current laws in several countries, including the United States, restrict cryptographic technology from export or import across national borders. In the era of the Internet, it is particularly important to be aware of all applicable local and foreign regulations governing the use of cryptography.

3.9 Information Warfare

Extensive and widespread dependence on the Internet has called new attention to the importance of information to national security. The term information warfare refers to the act of war against the information resources of an adversary. Like warfare on land or in the air, information warfare is one component of a range of attack strategies for dominating an adversary in order to gain or maintain an objective.

Information warfare is divided into two categories: offensive and defensive. The purpose of offensive information warfare is to attack the information resources of an adversary to gain dominance. Defensive information warfare is the protection of your information assets against attack.

Information assets can take many forms, from messages sent by courier in diplomatic bags to the computers used to analyze enemy positions based on satellite data. In computer security, information assets include digital information, the computers that process them, and the networks that transmit the digital information from place to place. Computer security is a key element for protecting the availability, integrity, and confidentiality of all these information assets.

Internet security protects information assets consisting of computers, information, and networks that are part of the Internet. Internet security is related to information warfare when the Internet contains information assets that are important to the information warfare

objective. For example, if an adversary can use the Internet to access battle plans, the Internet is being used for information warfare.

Internet security is important to both offensive and defensive information warfare because the Internet is a global and dependable resource on which many countries rely. Historically, military networks and computers were unreachable by nonmilitary participants. The Internet, however, provides a cost-effective way for military and government units to communicate and participate in achieving objectives. Use of the Internet means that individuals, multinational companies, and terrorist organizations all can gain access to important information resources of governments and military forces. Thus, it is important to address Internet security concerns as a key component of defensive information warfare.

Because the Internet is global, it can be an avenue of attack for offensive information warfare by many governments. One of the battlefields for a future military offensive could very well involve the Internet. Intruder technology (as described in a separate section above) could be used by a government as a weapon against information resources, or used randomly by a terrorist organization against civilian targets.

In the study of information warfare, there are many new problems to solve that are not evident in other forms of warfare. These problems include identifying the enemy, responding without making your systems vulnerable to attack, and gathering intelligence on the Internet about preparations for a military exercise. These and other problems are likely to be the subject of discussion and investigation for some time to come.

3.10 The Future

Research and development efforts are underway to allow critical applications to operate in the future in a more secure environment than exists today.

3.10.1 Internetworking Protocols

Most of the network protocols currently in use have changed little since the early definitions of the ARPA research and education network when trust was the norm. To have a secure foundation for the critical Internet applications of the future, severe weaknesses must be addressed: lack of encryption to preserve privacy, lack of cryptographic authentication to identify the source of information, and lack of cryptographic checksums to preserve the integrity of data (and the integrity of the packet routing information itself). New internetworking protocols are under development which use cryptography to authenticate the originator of a packet and to protect the integrity and confidentiality of data.

The IETF (Internet Engineering Task Force) Proposed Standard for the Next Generation Internet Protocol (IPng) is being designed to cope with the vastly increased addressing and routing needs associated with the exponential growth of the Internet. IPng provides integral support for authenticating hosts and protecting the integrity and confidentiality of data.

The first release of IPng is officially termed IPv6 (Internet Protocol version 6). Since it is impractical to replace the existing protocol instantly and simultaneously throughout the Internet, IPv6 is designed to coexist with the current version of IP, allowing for a gradual transition over the course of years. Implementations of IPv6 for many routers and host operating systems are underway.

In the future, authentication protocols will increasingly be supported by technology that authenticates individuals (in the context of their organizational or personal roles) through the use of smart cards, fingerprint readers, voice recognition, retina scans, and so forth.

Protocol design, analysis, and implementation will be the subject of continued research. A primary goal is 100% verifiably secure protocols (that is, protocols as provably secure as the cryptographic algorithms supporting them), but researchers are nowhere near attaining this goal.

3.10.2 Intrusion Detection

Research is underway to improve the ability of networked systems and their managers to determine that they are, or have been, under attack. Intrusion detection is recognized as a problematic area of research that is still in its infancy. There are two major areas of research in intrusion detection: anomaly detection and pattern recognition.

Research in anomaly detection is based on determining patterns of "normal" behavior for networks, hosts, and users and then detecting behavior that is significantly different (anomalous). Patterns of normal behavior are frequently determined through data collection over a period of time sufficient to obtain a good sample of the typical behavior of authorized users and processes. The basic difficulty facing researchers is that normal behavior is highly variable based on a wide variety of innocuous factors. Many of the activities of intruders are indistinguishable from the benign actions of an authorized user.

The second major area of intrusion detection research is pattern recognition. The goal here is to detect patterns of network, host, and user activity that match known intruder attack scenarios. One problem with this approach is the variability that is possible within a single overall attack strategy. A second problem is that new attacks, with new attack patterns, cannot be detected by this approach.

Finally, to support the needs of the future Internet, intrusion detection tools and techniques that can identify coordinated distributed attacks are critically needed, as are better protocols to support traceability.

3.10.3 Software Engineering and System Survivability

Current software engineering methods and practice have had only limited success in managing the intellectual complexity of designing and implementing software. Moreover, in the design of software systems, security concerns are typically an afterthought (addressed through add-ons and software patches) rather than being an integral part of the overall design.

This means that software systems of any significant size and complexity are likely to have exploitable security flaws. Because managing the intellectual complexity of software is difficult, up-front security design in products is rare, and detailed knowledge about systems is widespread, systems will be breached in spite of our best efforts to make them invulnerable.

Therefore, the concept of information systems security must encompass the specification of systems that exhibit behaviors that contribute to survivability in spite of intrusions. Only then can systems be developed that are robust in the presence of attack and are able to survive attacks that cannot be completely repelled.

System survivability is the capacity of a system to continue performing critical functions in a timely manner even if significant portions of the system are incapacitated by attack or accident. We use the term system in the broadest possible sense, which includes networks and large-scale "systems of systems".

Although the concepts and practices associated with system survivability are embryonic, they include (but are not limited to) traditional areas of software engineering and computer science such as reliability, testing, dependability, fault tolerance, verification of correctness, performance, and information system security. Promising research in survivability encompasses a wide variety of research methods in software engineering. Inoculation tools may be developed that will automate the distribution of security fixes, throughout an entire network infrastructure, to provide comprehensive protection from a newly discovered security flaw. The concept of inoculation may be further generalized to encompass adaptive networks, which consist of distributed cooperative network elements that exchange information on security problems and actively change and adjust in response to security threats.

3.10.4 Web-Related Programming and Scripting Languages

Downloading interesting, informative, or entertaining "content" from a remote site to a user's local machine is central to the activity of Web browsing (or "net surfing"). The content getting the most attention from Web users and the greatest concern from security experts is executable content, code to be executed on the local machine on download. This executable content may provide live audio of a conference in progress, a jazz tune, three-dimensional (3-D) animation effects, or hostile code that destroys the local file system. Executable code is authored using one or more Web-related programming or scripting languages designed specifically for the production of platform-independent executable content. Languages in this category include JAVA and ActiveX. Executable content is called an "applet" in JAVA and a "control panel" in ActiveX.

Web-related programming languages pose new security challenges and concerns because code is downloaded, installed, and run on a user's machine without a review of source code (the recommended practice for secure use of publicly available software). These activities can be triggered by following any hypertext link or opening any page while browsing. A user may not even be aware that code has been downloaded and executed. Some Web-related programming languages, most notably JAVA, have built-in security features, but security experts are concerned about the adequacy of these features.

As executable content makes Web browsing even more alluring, further research in software engineering and greater user awareness will be necessary to counter security risks. Presently, the security of executable content depends upon the correctness of multiple vendors' implementations, the inherent security of platform-independent "virtual machines," and the safety of the source code that is executed. In the foreseeable future, users need to be educated about the risks so they can make informed choices about where to place their trust.

3.10.5 Intelligent Autonomous Agents - A New Computing Paradigm

The future Internet environment is likely to be increasingly dependent on an agent-based model of computing, with significant implications for Internet security. Agents are executable software objects with executions that are not tied to any specific host or computing resource or to any geographical or logical network location. Agents perform computation and communication defined by a user, but the execution platforms are typically outside the user's administrative control (and outside the administrative control of the user's organization). The conceptual model of agent operation is one in which an intelligent agent, at the request of a user, goes to one or more remote hosts to perform a computation or gather information and then returns to the user with the result. An agent's mode of operation may range from partially to fully autonomous, and the degree to which an agent is autonomous may vary throughout the life of that agent.

A future agent-based computing environment may include features such as these:

Agents share information and cooperate to complete the user's task.

Agents protect themselves with intrinsic security mechanisms but also depend on some measure of extrinsic security provided by the infrastructure and cooperating agents.

Since most of an agent's activity takes place outside the user's domain of administrative control (and hence outside any firewall designed to protect the user), the traditional firewall has little to contribute to security.

Replication and agent diversity provide increased survivability while under attack and under conditions of degraded or uncertain infrastructure support.

Agents communicate to enhance the detection of threats. Specialized sensor agents are specifically designed to detect particular types of threats, and groups of diverse sensor agents provide the entire agent "collective" with a comprehensive profile of current threats.

The agent-supported infrastructure protects itself and takes defensive action without user intervention.

3.11 INSTALLING IIS

3.11.1 Installing IIS on Windows XP Pro

If you are running Windows XP Professional on your computer you can install Microsoft's web server, Internet Information Server 5.1 (IIS) for free from the Windows XP Pro installation CD and configure it to run on your system by following the instructions below:

1. Place the Windows XP Professional CD-Rom into your CD-Rom Drive.
2. Open 'Add/Remove Windows Components' found in 'Add/Remove Programs' in the 'Control Panel'.
3. Place a tick in the check box for 'Internet Information Services (IIS)' leaving all the default installation settings intact.
4. Once IIS is installed on your machine you can view your home page in a web browser by typing 'http://localhost' (you can substitute 'localhost' for the name of your computer) into the address bar of your web browser. If you have not placed your web site into the default directory you should now be looking at the IIS documentation.
5. If you are not sure of the name of your computer right-click on the 'My Computer' icon on your desktop, select 'Properties' from the shortcut menu, and click on the 'Computer Name' tab.

6. Your default web directory to place your web site in is 'C:\Inetpub\wwwroot', but if you don't want to over write the IIS documentation found in this directory you can set up your own virtual directory through the 'Internet Information Services' console.
7. The 'Internet Information Services' console can be found in the 'Administration Tools' in the 'Control Panel' under 'Performance and Maintenance', if you do not have the control panel in Classic View.

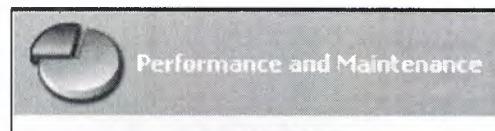


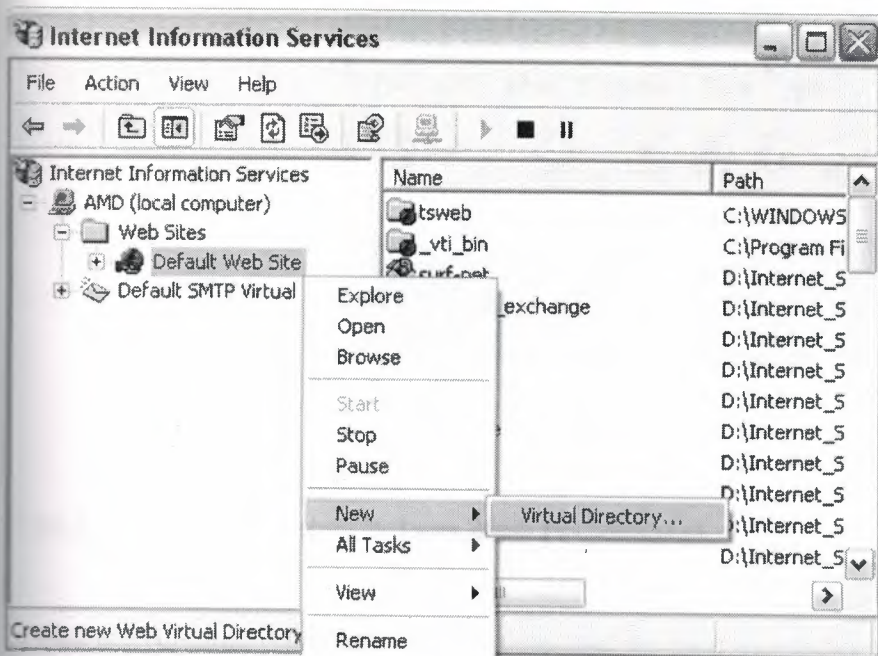
Figure 3.2

8. Double-click on the 'Internet Information Services' icon.



Figure 3.3

9. Once the 'Internet Information Services' console is open you will see any IIS web services you have running on your machine including the SMTP server and FTP server, if you chose to install them with IIS.
10. To add a new virtual directory right click on 'Default Web Site' and select 'New', followed by 'Virtual Directory', from the drop down list.



11. Next you will see the 'Virtual Directory Creation Wizard' from the first screen click the 'next' button.
12. You will then be asked to type in an 'Alias' by which you will access the virtual directory from your web browser (this is the name you will type into your web browser after 'localhost' to view any web pages you place in the directory).
13. Next you will see a 'Browse...' button, click on this to select the directory your web site pages are in on your computer, after which click on the 'next' button to continue.
14. On the final part of the wizard you will see a series of boxes, if you are not worried about security then select them all, if you are and want to run ASP scripts then check the first two, followed by the 'next' button.
15. Once the virtual directory is created you can view the web pages in the folder by typing 'http://localhost/aliasName' (where 'aliasName' is, place the alias you called the virtual directory) into the address bar of your web browser (you can substitute 'localhost' for the name of your computer if you wish).



Figure 3.4

3.11.2 Installing IIS on Windows 2000 Professional

If you are running Windows 2000 Professional on your computer you can install Microsoft's web server, Internet Information Server (IIS) for free from the Windows 2000 Pro installation CD and configure it to run on your system by following the instructions below: -

1. Place the Windows 2000 Professional CD-Rom into your CD-Rom Drive.
2. Open 'Add/Remove Windows Components' found in 'Add/Remove Programs' in the 'Control Panel'.
3. Place a tick in the check box for 'Internet Information Services (IIS)' leaving all the default installation settings intact.
4. Once IIS is installed on your machine you can configure IIS through the 'Personal Web Manager' found in the 'Administration Tools' in the 'Control Panel'.
5. Double-click on the 'Personal Web Manager' icon.



Figure 3.5

6. Once the Personal Web Manager is open you will see the Main dialog box where it will show your home page and home directory default values. Where the home page is shown below as 'http://My_Computer', will be 'http://' followed by the name of your computer. Clicking on each of these values will open your home page in your default web browser or open the default home directory in Windows Explorer.

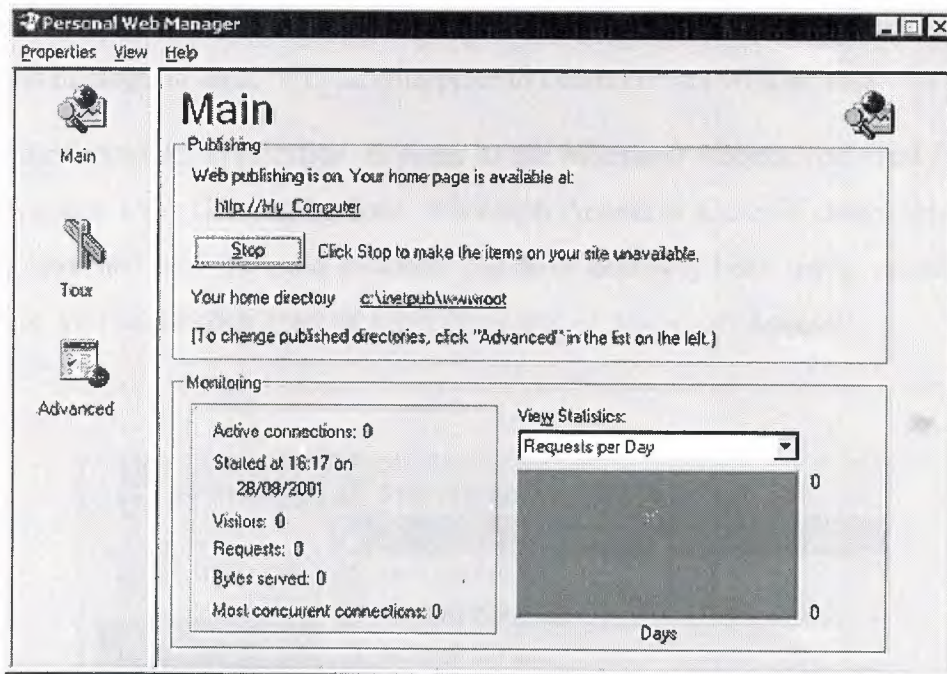


Figure 3.6

7. To view your home page in a web browser type 'http://localhost' (you can substitute 'localhost' for the name of your computer) into the address bar of your web browser. If you are not sure of the name of your computer right-click on the 'My Computer' icon on your desktop, select 'Properties' from the shortcut menu, and click on the 'Network Identification' tab.
8. Until you place your own web site in the default directory for the web servers home page you should now be looking at the documentation for IIS.
9. To place your own web site in place of the IIS documentation in your home page you need to place your own web page in the 'c:\inetpub\wwwroot' directory making sure the page is called Default.htm or Default.asp. Now when you type 'http://' followed by your computer name', into your web browser you should see your own home page.

CHAPTER FOUR

4. MICROSOFT ACCESS DATABASE

4.1. Introduction to Microsoft Access

Microsoft Access is a computer application used to create and manage computer-based databases on desktop computers and/or on connected computers (a network). Microsoft Access can be used for personal information management (PIM), in a small business to organize and manage all data, or in an enterprise to communicate with servers.

Like any other computer application, in order to use Microsoft Access, you must first open it. There are various ways this can be done. Microsoft Access is a classic computer application and it gets launched like the usual products you have probably been using. As such, to start this program, you could click Start → (All) Programs → Microsoft Access:

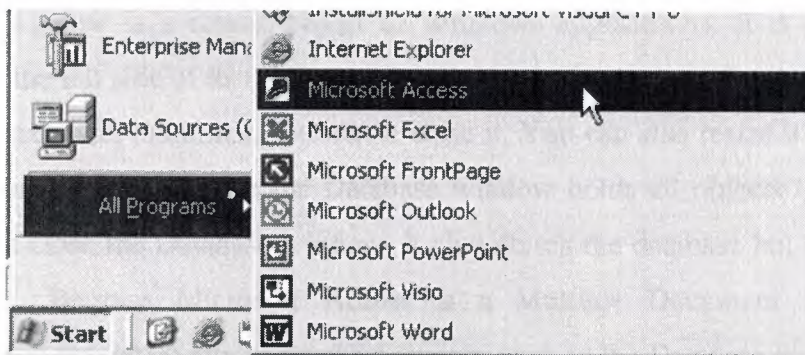


Figure 4.1

As a regular member of the Microsoft Office suite of applications, if your installation created a sub-menu on the Start menu, you could click Start → New Office Application and proceed from the New dialog box.

Although Microsoft Office 97 and Microsoft Office 2000 get installed in the C:\Program Files\Microsoft Office folder, they treat the shortcuts that launch them differently. The applications that are part of Microsoft Office 97 designate their shortcuts with full names and these are installed in the Microsoft Office folder. Microsoft Office 2000 (Premium) uses shortcut names to designate its shortcuts and they are installed in the Microsoft Office\Office folder. This means that you could launch an application from Windows Explorer or My Computer.

You can also launch Microsoft Access from a shortcut. If you happen to use the software on a regular basis, you can create a shortcut on your desktop or on the Quick Launch area. Many users also take advantage of the Microsoft Office Shortcut Bar. Sometimes, the icon you need will not be there; in that case you should insert it manually.

If you are working on a network of related computers, your database may be located in another computer. In this case the network or database administrator would create a link or shortcut to the drive that is hosting the database. You can then click or double-click this link or shortcut to open the database and, as a result, launch Microsoft Access.

4.2. The Database Window

After creating or opening a database, unless the product is setup otherwise, the first object that appears is a rectangular box named the Database window. It is different on Microsoft Access 97 and Microsoft Access 2000 (and later versions).

The Database window is a classic object of Windows applications. It is equipped with a system icon on the left side of its title bar and three system buttons on the right side. Based on this, you can maximize, minimize, restore, or close it. You can also resize it by dragging one of its borders or corners. Because the Database window holds all objects that are part of a database, if you close the Database window, it also closes the database but leaves Microsoft Access opened. Because Microsoft Access is a Multiple Document Interface (MDI) application, if you maximize any of its child objects, such as the Database windows, the other objects that you subsequently open would be maximized also. In all versions of Microsoft Access, objects are organized in categories.

In Microsoft Access 97, each category is represented by a property page. Therefore, to select a category, you can click its tab.

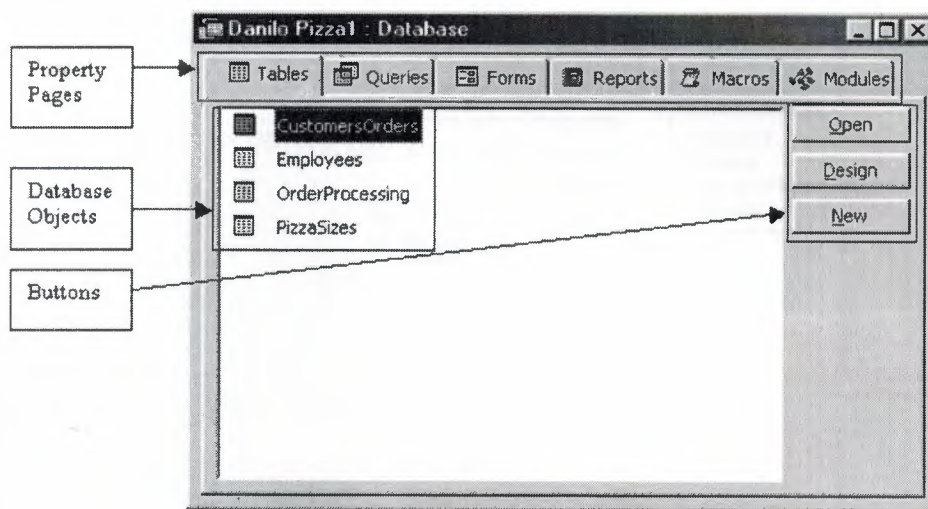


Figure 4.2

Once in the property page of a category, to open an object:

- You can double-click it
- You can click it to select it, then click the Open button on the right side
- You can also right-click an object and click Open

One of the biggest changes that Microsoft Access 2000 brought was on the Database window. It got completely redesigned and highly improved. Like all classic windows, it is equipped with a title bar similar to the Database window of the 97 version as we described above. Under the title bar, the Database window is equipped with a contextual toolbar. This means that the toolbar responds according to the object that is selected in the Database window.

Like all releases, objects in Microsoft Access are organized in categories. In the 2000 and later versions, objects are represented by the Objects Bar. Therefore, to select a category, you can click its button. Besides the buttons that represent categories, when you click a button, one, two, or three links allow you to create objects of that category. For example, you can create a table by double-clicking the Create Table By Using Wizard link.

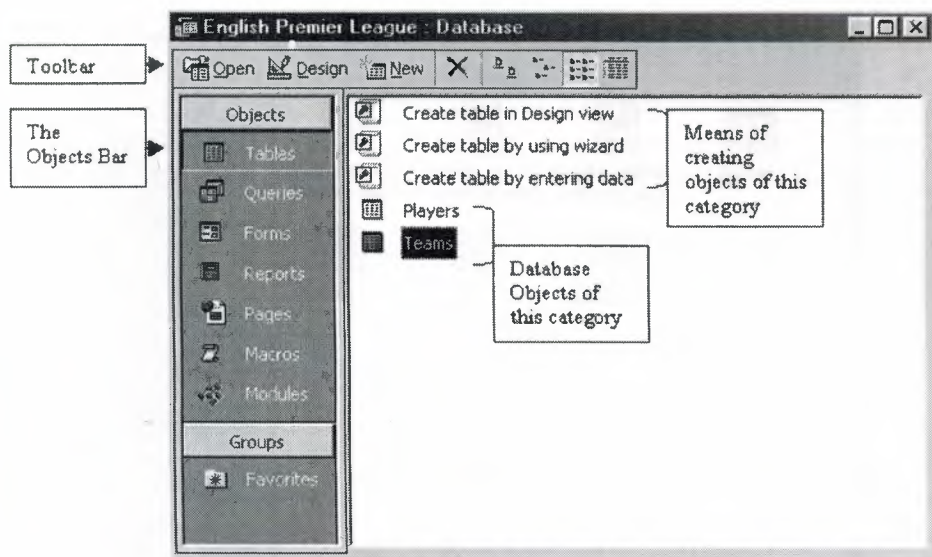


Figure 4.3

To open an object:

- Double-click it
- Click it to select it and click the *Open* button on the Database window's toolbar
- Right-click it and click *Open*

Besides providing the ability to create a new object or open an existing one, you can also delete an object using the Database window's toolbar. To do this, you can click the object to select it. Then, on this toolbar, click the *Delete* button. The Database window's toolbar also provides four view buttons that allow you to change the way the list displays in the right side of the view. If you have used Windows Explorer, My Computer, or My Documents, you are probably familiar with these buttons. For example, here is the Database window that displays its list in Large Icons:

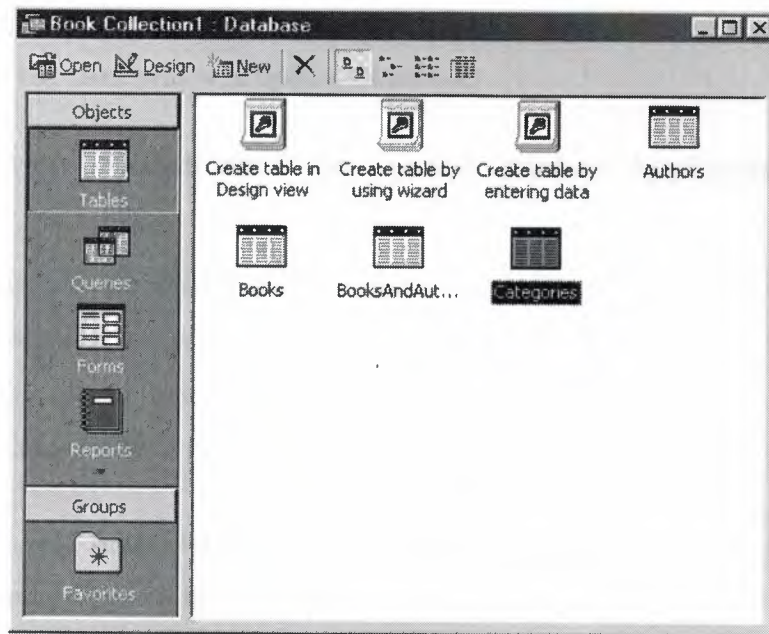


Figure 4.4

4.3. Tables

A Microsoft Access database is a file made of various internal objects: tables, queries, forms, reports, etc. All these are managed from an object called the Database Window. The objects are kept in categories. To access an object, you click the button that corresponds to its category.

A table is the central point of a database, because all data is stored in tables. For better organization, you will have various tables in your database, each for a different purpose.

Each table is recognized by its name. To open a particular table, you can double-click it. You can also right-click a table's name and click Open. If the desired table is already selected on the Database Window, you can click the Open button to open it.

4.4. Queries

A table can be large depending on the information it holds. To further organize your data, you should be able to retrieve necessary information for a specific purpose. The solution is to create a query (or queries) in order to limit part of the data in a table for a specific goal, for better managing or searching. That's the role of a query.

Just like tables, queries are managed from the Database Window in their own category. You can open a query the same way you open a table.

4.5. Brief overview of Relational Databases and Database Applications

The first databases implemented during the 1960s and 1970s were based upon either flat data files or the hierarchical or networked data models. These methods of storing data were relatively inflexible due to their rigid structure and heavy reliance on applications programs to perform even the most routine processing.

In the late 1970s, the relational database model which originated in the academic research community became available in commercial implementations such as IBM DB2 and Oracle. The relational data model specifies data stored in relations that have some relationships among them (hence the name relational).

In relational databases such as Sybase, Oracle, IBM DB2, MS SQL Server and MS Access, data is stored in tables made up of one or more columns (Access calls a column a field).

The data stored in each column must be of a single data type such as Character, Number or Date. A collection of values from each column of a table is called a record or a row in the table.

Different tables can have the same column in common. This feature is used to explicitly specify a relationship between two tables. Values appearing in column A in one table are shared with another table. Below are two examples of tables in a relational database for a local bank:

| | CustomerID | Name | Address | City | State | Zip |
|---|------------|--------|----------------|---------|-------|--------|
| ▶ | 1 | can | Dikmen | ankara | IAB | 060010 |
| | 2 | canan | Fatih Cad. | alanya | AB | 070182 |
| | 3 | candan | karanfil sokak | antalya | AB | 070258 |
| * | 0 | | | | | |

Figure 4.5

| | CustomerID | Account_number | Account_type | Date_opened | Balance |
|---|------------|----------------|--------------|-------------|--------------|
| ▶ | 1 | 20103 | Saving | 26.04.2005 | 4.000,00 TL |
| | 1 | 20101 | Checking | 28.04.2005 | 2.500,00 TL |
| | 2 | 20102 | Saving | 26.04.2005 | 30,00 TL |
| | 3 | 20105 | Saving | 25.05.2005 | 10.000,00 TL |
| | 3 | 20104 | Checking | 01.05.2005 | 3.500,00 TL |
| * | 0 | 0 | | | 0,00 TL |

Figure 4.6

The Customer table has 6 columns (CustomerID, Name, Address, City, State and Zip) and 3 rows (or records) of data. The Accounts table has 5 columns (CustomerID, AccountNumber, AccountType, DateOpened and Balance) with 5 rows of data.

Each of the columns conforms to one of three basic data types: Character, Number or Date. The data type for a column indicates the type of data values that may be stored in that column.

- Number - may only store numbers, possibly with a decimal point.
- Character - may store numbers, letters and punctuation. Access calls this data type Text.
- Date - may only store date and time data.

In some database implementations other data types exist such as Images (for pictures or other data). However, the above three data types are most commonly used.

Notice that the two tables share the column CustomerID and that the values of the CustomerID column in the Customer table are the same the values in the CustomerID column in the Accounts table. This relationship allows us to specify that the Customer Mr. Axe has both a Checking and a Savings account that were both opened on the same day: December 1, 1994.

Another name given to such a relationship is Master/Detail. In a master/detail relationship, a single master record (such as Customer 1003, Mr. Axe) can have many details records (the two accounts) associated with it.

In a Master/Detail relationship, it is possible for a Master record to exist without any Details. However, it is impossible to have a Detail record without a matching Master record. For

example, a Customer may not necessarily have any account information at all. However, any account information must be associated with a single Customer.

Each table also must have a special column called the Key that is used to uniquely identify rows or records in the table. Values in a key column (or columns) may never be duplicated. In the above tables, the CustomerID is the key for the Customer table while the AccountNumber is the key for the Accounts table.

CHAPTER FIVE

Student Online Registration With ASP Project

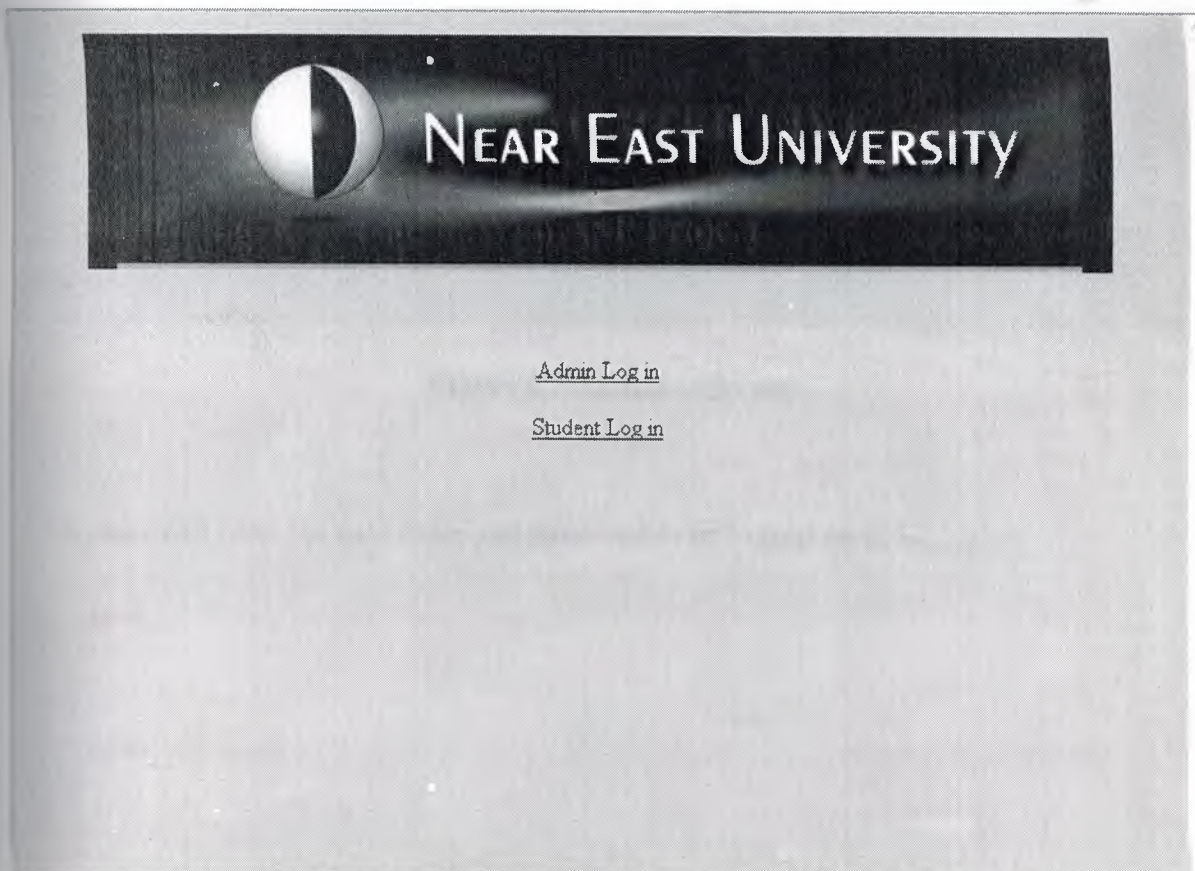
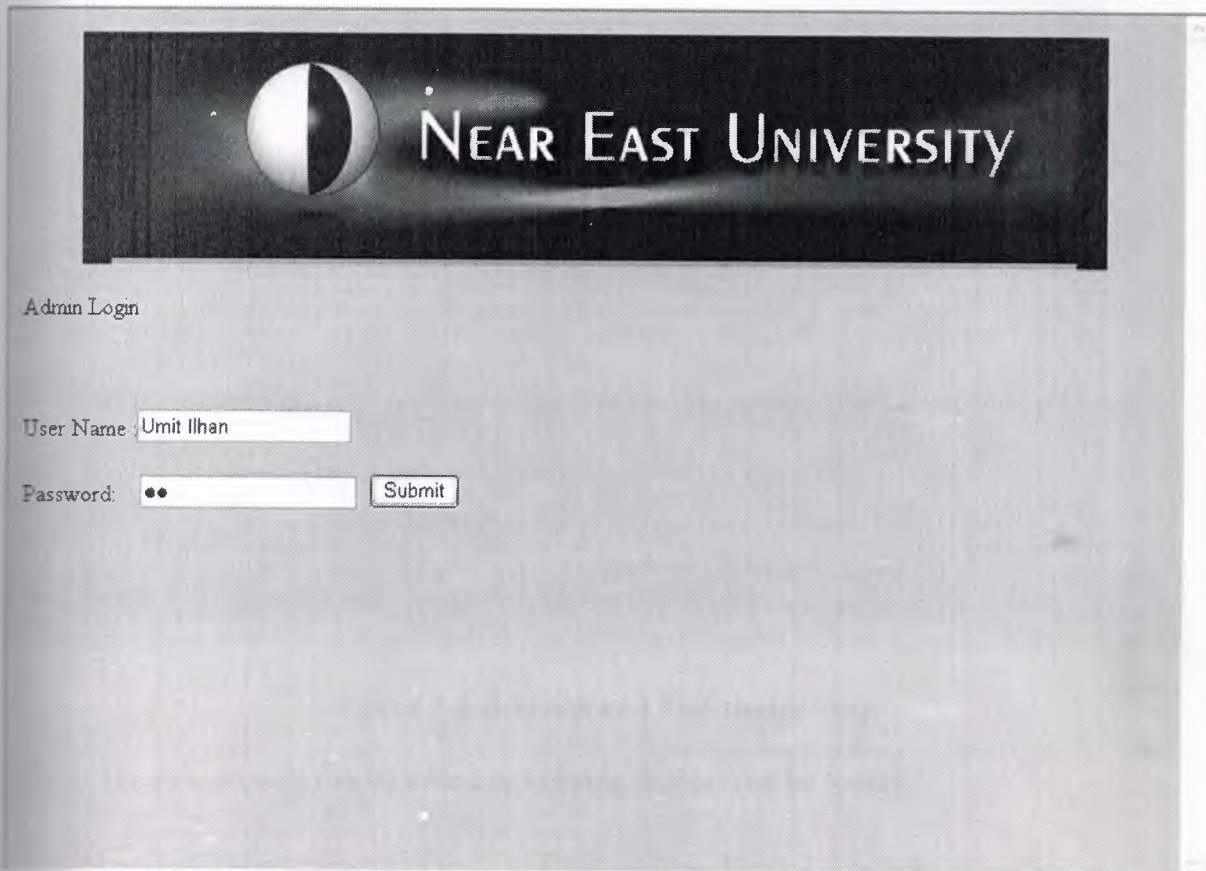


Figure 5.1 Default (Homepage) Page

The above web page (Figure 5.1) is the homepage of “student online registration” web site. This page in student and user has to be registered.



The screenshot shows a web browser window displaying the 'Admin Login' page of the Near East University. At the top, there is a dark banner with a logo consisting of a sphere with a vertical line through its center, and the text 'NEAR EAST UNIVERSITY' in white capital letters. Below the banner, the text 'Admin Login' is visible. The login form includes a 'User Name' field with the text 'Umit Ilhan' entered, a 'Password' field with two black dots indicating masked text, and a 'Submit' button to the right of the password field.

Figure 5.1 Admin login.asp

The user will enter the user name and password to be logged on as admin.

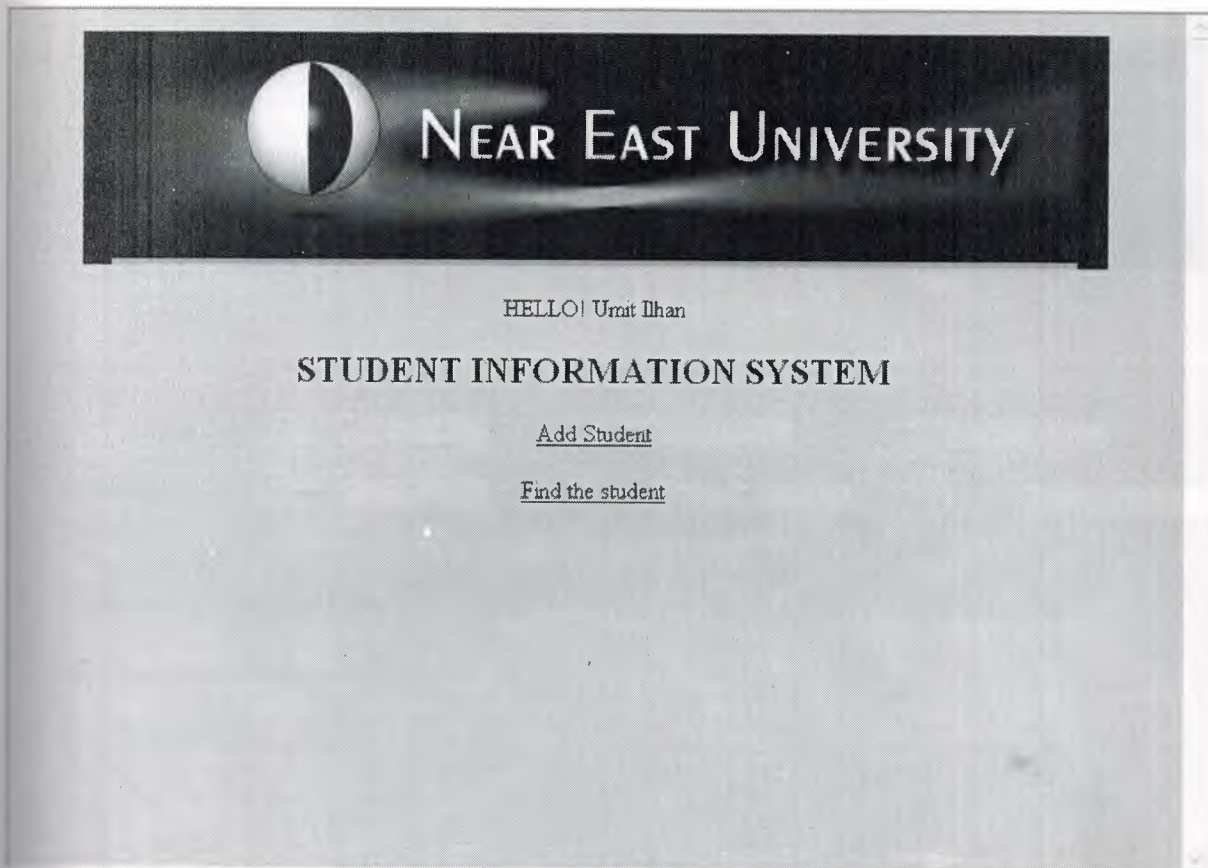



Figure 5.3 Added&and find student.asp

The new student can be added or existing student can be found..



NEAR EAST UNIVERSITY

COURSE REGISTRATION FORM

| | | | | | |
|---------------|----------|---------------|---------|------------|-------|
| Student No. : | 20001089 | Name : | Huseyin | Surname: | şahin |
| Mother Name : | Senem | Father Name : | Kazım | Dep.Name : | 1 |

[New Register Lesson](#)

[Change Registration Lesson](#)

[Enter Grade](#)


[Delete the student](#)

[Show Paying](#)

[Add Paying](#)

Figure 5.4 Admin.newregister.asp

The user can change register lesson, enter grade, delete the student, show paying,add paying by clicking the links above



NEAR EAST UNIVERSITY

Student Login

Student No:

Password:

Figure 5.5 StudentInfo.asp

The user will enter the user name and password to be logged on as student.

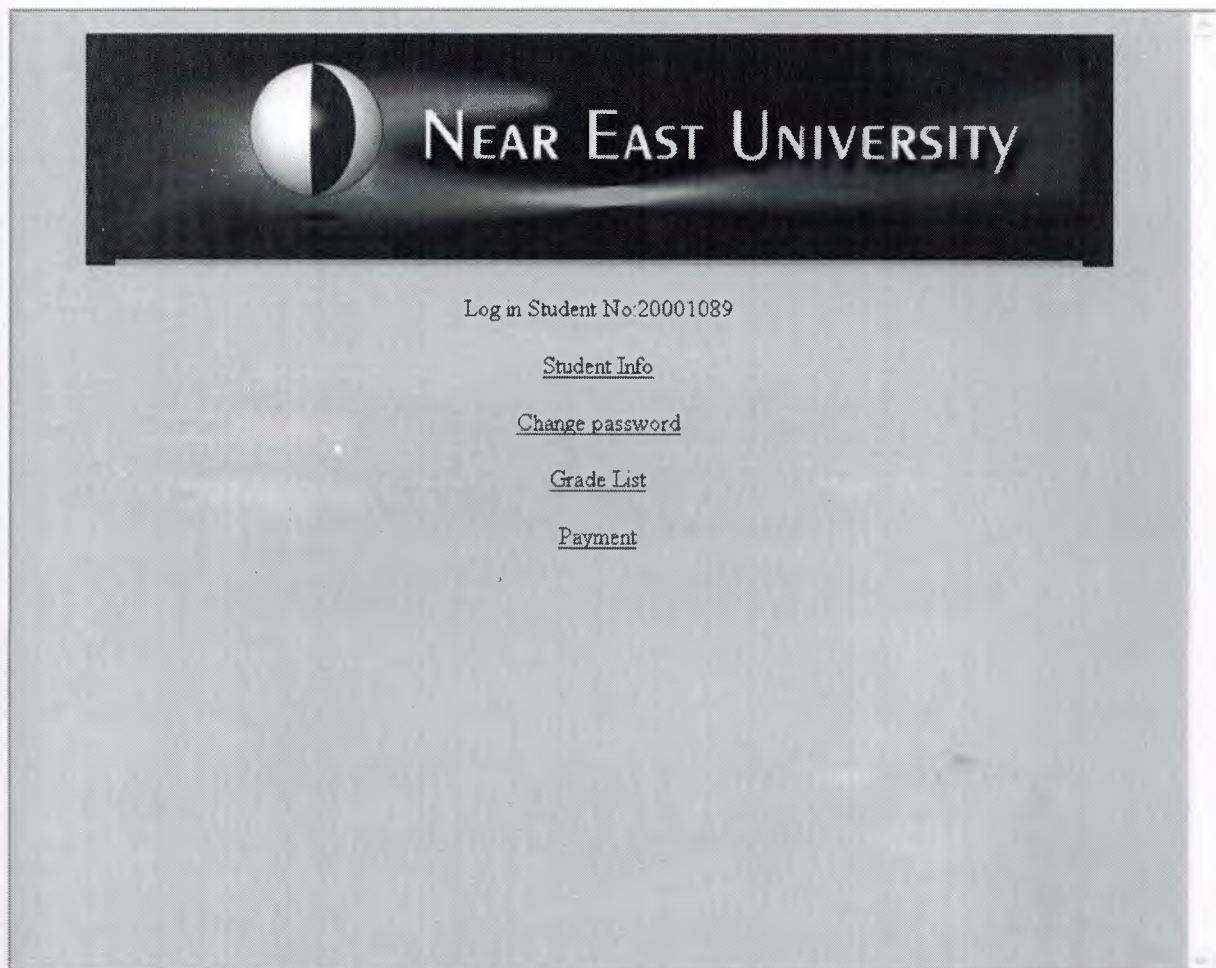
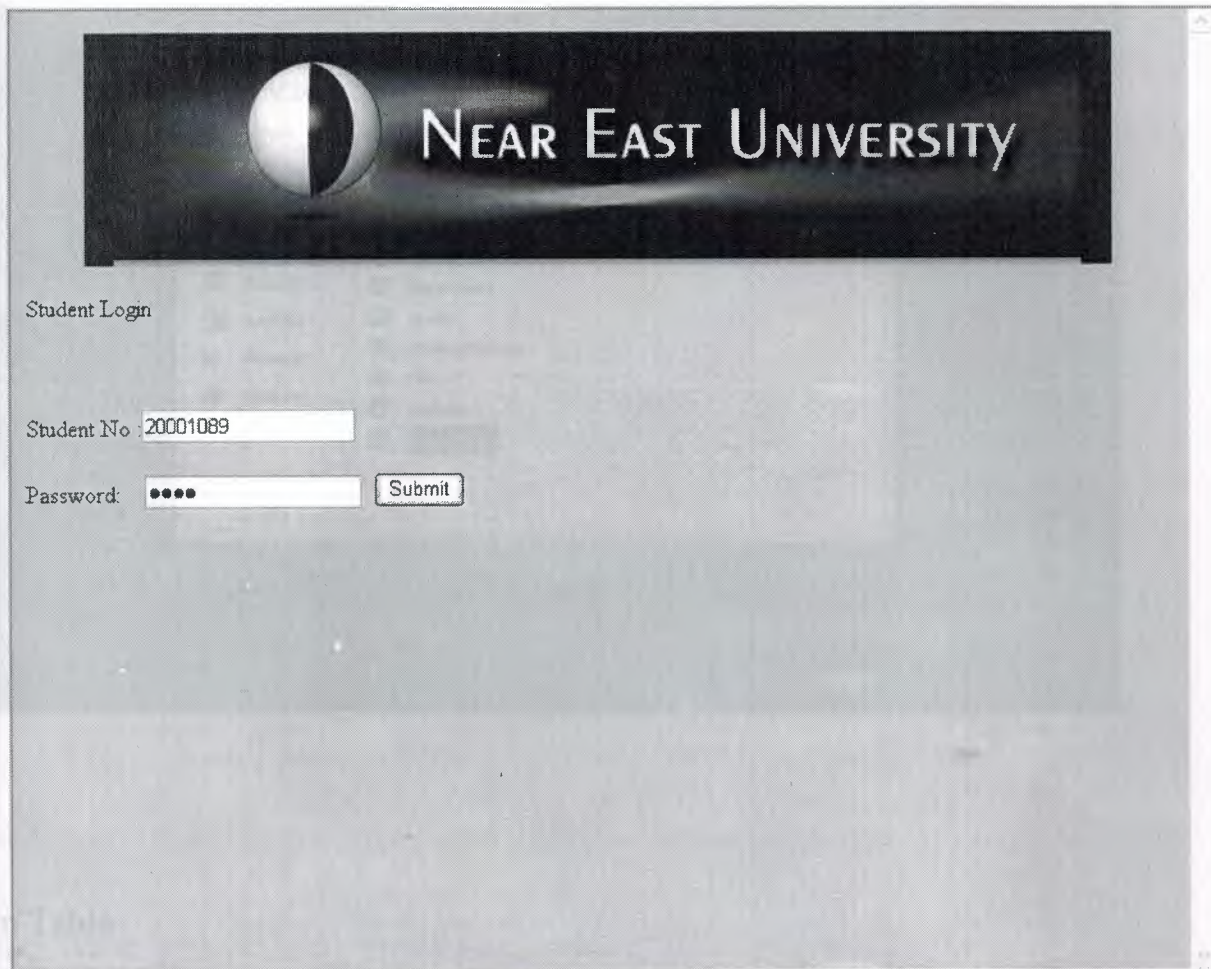


Figure 5.6 Student log in.asp

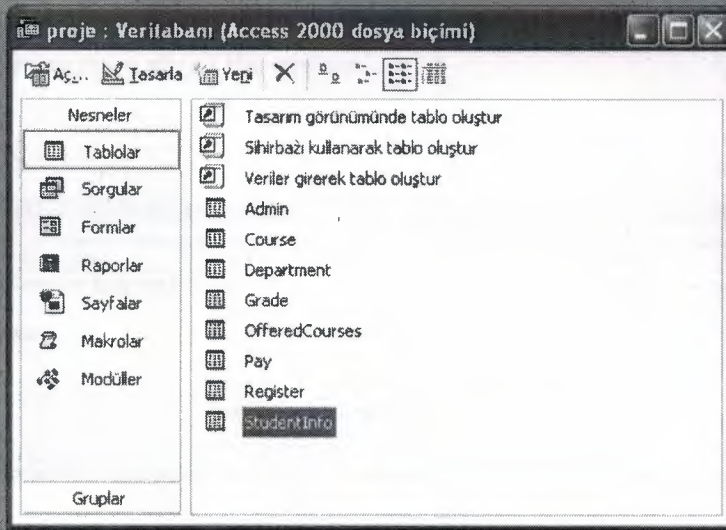
The user can student info ,change password ,grade list, delete the student, and payment by clicking the links above



The image shows a web-based student login interface for Near East University. At the top, there is a dark banner with the university's logo (a sphere with a vertical line) and the name "NEAR EAST UNIVERSITY" in white capital letters. Below the banner, the text "Student Login" is displayed. The login form consists of two input fields: "Student No" with the value "20001089" entered, and "Password" with four dots representing masked characters. To the right of the password field is a "Submit" button. The entire form is set against a light gray background.

DATABASE TABLES

In this project we used Microsoft Access 2000 for the database. The name of the data base is db.mdb. There are 8 tables in this database file(Admin, Course, Department, Grade, Offeredcourse, Student info, Register,Pay)



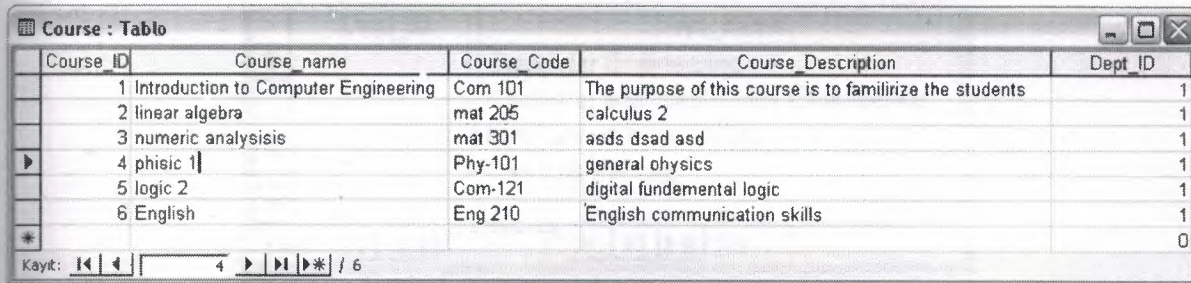
Admin Table

The information for the Admin are stored on this table.

| Admin : Tablo | | |
|----------------------|-------------------|-----------|
| | AdminName | Password |
| | Huseyin Ali Sahin | 02 |
| ▶ | Kenan Sahin | 2946 |
| | Muhammed Akgun | zxc123456 |
| | Numan | a13a12 |
| | Umit Ilhan | 01 |
| * | | |
| Kayıt: 1 2 3 4 5 / 5 | | |

Course Table

This table has got information about the course.



| Course ID | Course_name | Course_Code | Course_Description | Dept ID |
|-----------|--------------------------------------|-------------|--|---------|
| 1 | Introduction to Computer Engineering | Com 101 | The purpose of this course is to familirize the students | 1 |
| 2 | linear algebra | mat 205 | calculus 2 | 1 |
| 3 | numeric analysis | mat 301 | asds dsad asd | 1 |
| 4 | phisic 1 | Phy-101 | general ophysics | 1 |
| 5 | logic 2 | Com-121 | digital fundamental logic | 1 |
| 6 | English | Eng 210 | English communication skills | 1 |
| | | | | 0 |

Kayit: 4 / 6

The course_id field is used to store the regularly course. It's variable type is

AutoNumber.

The course_name field is used to store course name. It's variable type is Text.

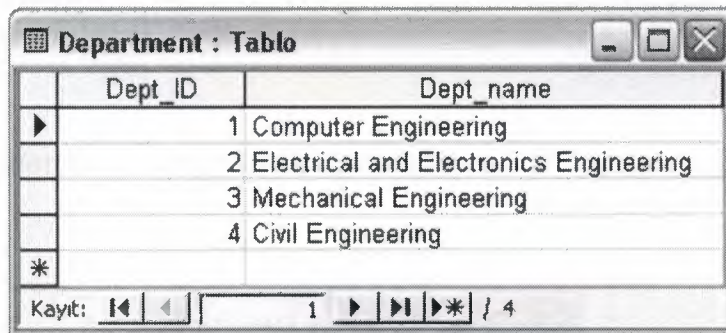
The course_code field is used to store course code. It's variable type is Text.

The course_desc field is used to store information about the course. It's variable type is Memo.

The dept_id field is used to store which department choose. It's variable type is Number.

Department Table

This table has got information about the department.



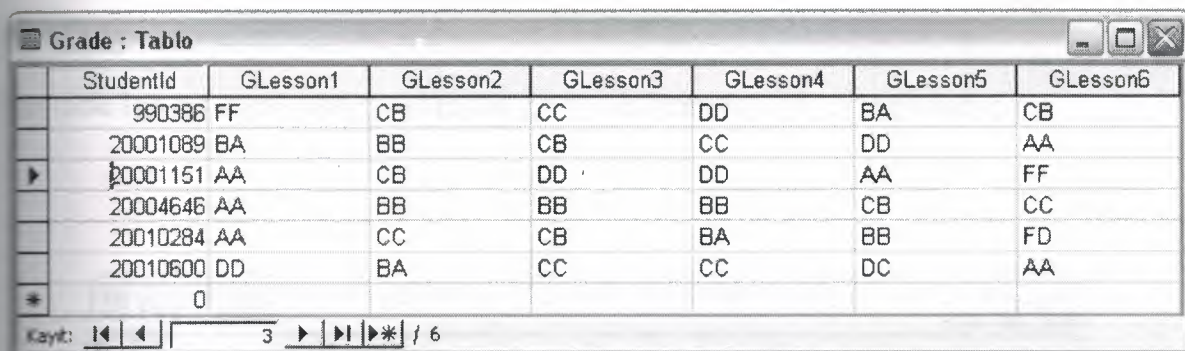
| Dept_ID | Dept_name |
|---------|--|
| 1 | Computer Engineering |
| 2 | Electrical and Electronics Engineering |
| 3 | Mechanical Engineering |
| 4 | Civil Engineering |

The dept_id fields is used to store the regularly department. It's variable type is AutoNumber.

The dept_name field is used to store department name. It's variable type is Text.

Grade Table

This table is used to calculate GPA and give information.



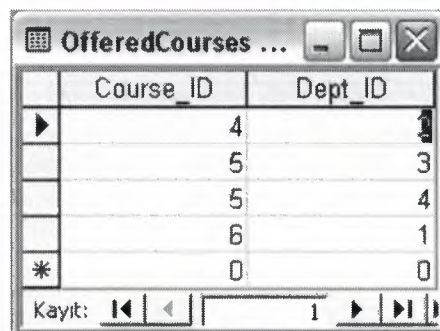
| StudentId | GLesson1 | GLesson2 | GLesson3 | GLesson4 | GLesson5 | GLesson6 |
|-----------|----------|----------|----------|----------|----------|----------|
| 990386 | FF | CB | CC | DD | BA | CB |
| 20001089 | BA | BB | CB | CC | DD | AA |
| 20001151 | AA | CB | DD | DD | AA | FF |
| 20004646 | AA | BB | BB | BB | CB | CC |
| 20010284 | AA | CC | CB | BA | BB | FD |
| 20010600 | DD | BA | CC | CC | DC | AA |

The student_id field is used to give information about students number. It's variable type is Text.

Gradelesson field is used to give information about grade. It's variable type is Number

Offered course table

This table is give information about which course is open which course is close.



| | Course_ID | Dept_ID |
|---|-----------|---------|
| ▶ | 4 | 1 |
| | 5 | 3 |
| | 5 | 4 |
| | 6 | 1 |
| * | 0 | 0 |

Kayit: 1

The course_id file is used to give information about which course is open. It's

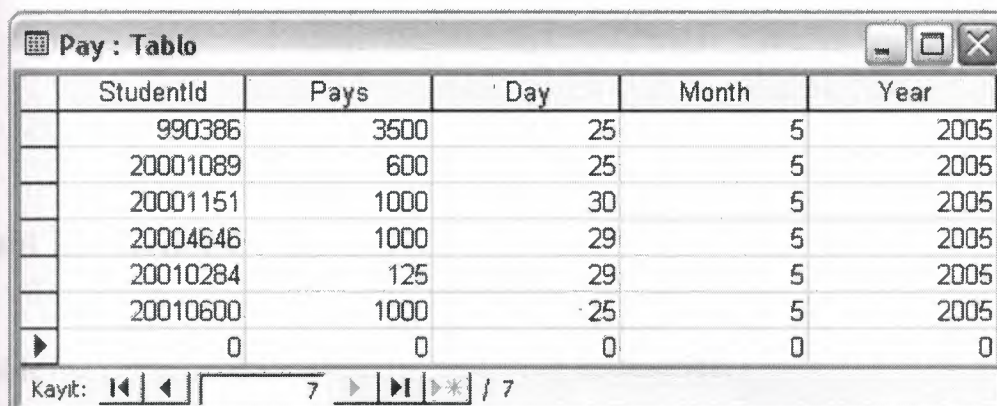
variable type is Number

The dept_id fields is used to store the regularly department. It's variable type is

AutoNumber.

Pay table

This table is used to calculate paying and give as information of paying balance



| | StudentId | Pays | Day | Month | Year |
|---|-----------|------|-----|-------|------|
| | 990386 | 3500 | 25 | 5 | 2005 |
| | 20001089 | 600 | 25 | 5 | 2005 |
| | 20001151 | 1000 | 30 | 5 | 2005 |
| | 20004646 | 1000 | 29 | 5 | 2005 |
| | 20010284 | 125 | 29 | 5 | 2005 |
| | 20010600 | 1000 | 25 | 5 | 2005 |
| ▶ | 0 | 0 | 0 | 0 | 0 |

Kayit: 7

Register table

This table has got information about the student register or not.

| Register : Tablo | | | | | | | |
|------------------|---------|---------|---------|---------|---------|---------|-----------|
| | Lesson1 | Lesson2 | Lesson3 | Lesson4 | Lesson5 | Lesson6 | StudentID |
| ▶ | 0 | 2 | 0 | 1 | 2 | 0 | 990386 |
| | 1 | 2 | 6 | 0 | 0 | 0 | 20001089 |
| | 1 | 1 | 2 | 0 | 3 | 1 | 20001151 |
| | 2 | 4 | 6 | 0 | 4 | 0 | 20004646 |
| | 0 | 0 | 2 | 0 | 0 | 6 | 20010284 |
| | 0 | 2 | 1 | 4 | 0 | 0 | 20010600 |
| * | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Kayıt: 1 / 6 | | | | | | | |

Student Information table

This table has got information about the student all information.

| StudentInfo : Tablo | | | | | | | | | |
|---------------------|------------|--------------|-----------------|---------------|----------------|------|------|-------------|------|
| | Student ID | Student_name | Student_surname | Student_Mothe | Student_Father | Stud | Stud | Student_pas | Harc |
| ▶ | 20001089 | Huseyin Ali | şahin | Senem | Kazim | 1 | 1 | 2946 | 3500 |
| | 20001151 | john | davidson | claudia | james | 1 | 0 | 123456 | 3500 |
| | 20004646 | numan | şahin | gölçin | mevlut | 1 | 1 | 46 | 3000 |
| | 20010284 | kadir | şahin | senem | kazim | 1 | 1 | 2946 | 3000 |
| | 20010600 | Abdi | ipekçi | ayten | rüştü | 3 | 1 | 33 | 2500 |
| | 990386 | Ayşe | Bubikoglu | Selvi | Barbaros | 2 | 1 | 11 | 2000 |
| * | | | | | | 0 | 0 | | 0 |

Kayit: 1 / 6

The st_id field is used to store student number. It's variable type is Text.

The st_name field is used to store student name. It's variable type is Text.

The student_Surname field is used to store student surnam. It's variable type is text.

The st_mother name field is used to store students mother name. It's variable type is Text.

The st_father name field is used to store students father name. It's variable type is Text.

The st_password field is used to store student password. It's variable type is Text.

The st_dept_id field is used to store which department you choose its number.

It's variable type is Number.

The st_active field is used to give information student active or not. It's variable type is Yes/No.

The Pay field is give information about harc balance. It's variable number.

SOURCE CODES

Index.html

```
<html>

<head>
<meta http-equiv="Content-Language" content="tr">
<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-1254">
<title>NEAR EAST UNIVERSITY</title>
</head>

<body bgcolor=ForestGreen>

<p align="center"></p>
<p align="center">&nbsp;</p>
<p align="center"><a href="AdminLogin.htm">Admin Log in</a></p>
<p align="center"><a href="Stdlogin.htm">Student Log in</a></p>

</body>

</html>
```

Admin.asp

```
<% language="VBScript"

Set bag = Server.CreateObject("ADODB.Connection")
bag.Open "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" &
Server.MapPath("proje.mdb")
Set RsBas=Server.CreateObject("ADODB.Recordset")
```



```

admin=Request("adminname")
password=Request("password")
SqlQ ="select * from Admin where AdminName='" & admin & "' and password='" &
password & "'"

RsBas.open SqlQ,Bag,1,3

if rsbas.RecordCount =1 then
    Session("adminId")=admin
    Response.Redirect ("AdminMenu.asp")
else
    Response.Write("Wrong admin name or password")
end if

bag.Close
set bag=nothing
%>

```

AdminLogin.html

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD>
<META http-equiv=Content-Type content="text/html; charset=unicode">
<META content="MSHTML 6.00.2900.2180" name=GENERATOR></HEAD>
<BODY>
<P align=center><FONT size=5>Near East University</FONT></P>
<P align=left>Admin Log In</P>
<form method="post" action="Admin.asp">
<P>User name :<INPUT name=adminname></P>
<P>Password:&nbsp;   <INPUT type=password
name=password><INPUT id=submit1 type=submit value=Submit name=submit1></P>
</form>
</BODY></HTML>

```

StudentLogin.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD>
<META http-equiv=Content-Type content="text/html; charset=unicode">
<META content="MSHTML 6.00.2900.2180" name=GENERATOR></HEAD>
<BODY>
<P align=center>Near East University</P>
<P>Student Login</P>
<form method="post" action="Student.asp">
<P>&nbsp;</P>
<P>Student No&nbsp;<INPUT name=S1 style="WIDTH: 143px; HEIGHT: 22px"
size=18></P>
<P>Password:&nbsp;&nbsp;&nbsp;<INPUT type=password
name=P1>&nbsp;&nbsp;<INPUT id=submit1 type=submit value=Submit name=submit1
style="LEFT: 220px; TOP: 131px"></P>
</form>
</BODY></HTML>
```

AdminMenu.asp

```
<html>
<head>
<meta http-equiv="Content-Language" content="tr">
<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-1254">
<title>NEAR EAST UNIVERSITY</title>
</head>
<body bgColor=ForestGreen>
<p align="center"></p>
```



```
<INPUT type=reset value=Reset> </FORM>
</BODY>
</HTML>
```

AdminAddStd.asp

```
<% language="VBScript"

stdNo=Request("StudentNo")
stdNm=request("StudentName")
stdSnm=request("StudentSurname")
stdMnm=request("MotherName")
stdFnm=request("FatherName")
stdDepNm=request("DepName")
stdPass=request("Password")
stdPayment=request("Payment")
if stdNo="" or stdNm="" or stdSnm="" then
    Response.Write("bütün kutuları doldurmalısınız")
    Response.End
else

Set bag = Server.CreateObject("ADODB.Connection")
bag.Open "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" &
Server.MapPath("proje.mdb")
Set RsBas=Server.CreateObject("ADODB.Recordset")

SqlQ ="insert into studentInfo values('" & stdNo & "','" & stdNm & "','" & stdSnm & "','" &
stdMnm & "','" _
    & StdFnm & "','" & StdDepNm & "','" & StdPass & "','"&stdpayment&')"
Response.Write(sqlq)
Response.End

RsBas.open SqlQ,Bag,1,3
```

```
Response.Write("Kaydınız başarıyla tamamlandı")
```

```
Response.Redirect "listele.asp"
```

```
bag.Close
```

```
set bag=nothing
```

```
end if
```

```
%>
```

AdminStdFinded.asp

```
<% language="VBScript"%>
```

```
<html>
```

```
<head>
```

```
<title>Course Registration Form</title>
```

```
</head>
```

```
<body bgcolor="yellow">
```

```
<p align="center"><font size="4">NEAR EAST UNIVERSITY</font></p>
```

```
<p><IMG height=200 src="ciu-8[1].gif" width=500 border=0></p>
```

```
COURSE REGISTRATION FORM</font></p>
```

```
<%
```

```
if session("AdminId")="" then
```

```
    Response.End
```

```
end if
```

```
studentno=request("StudentNo")
```

```
if studentno<>"" then
```

```
    Session("StudentNo")=studentno
```

```
else
```

```
    Response.End
```

```
end if
```

```
Set bag = Server.CreateObject("ADODB.Connection")
```

```
bag.Open "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" &
```

```
Server.MapPath("proje.mdb")
```

```
Set RsBas=Server.CreateObject("ADODB.Recordset")
```

```
SqlQ ="select * from StudentInfo where Student_ID=" & studentno & ""
```

Response.Write(sqlq)

Response.End

```
RsBas.open SqlQ,Bag,1,3
```

```
if rsbas.RecordCount=0 then
```

Response.Write("Kayıt bulunamadı")

Response.End

end if

 $\frac{1}{2}$

| | |
|--|---|
| <p>  </p> | <p>  </p> |
|--|---|

```
id="AutoNumber1" height="0">
```

|
 <td width="100%"> | |[illegible]

```
<INPUT name=StudentNo value=%%rsbas.Fields("Student_ID") %> size="20">&nbsp;   
```

Name :

```
<INPUT name=StudentName value=<%=rsbas.Fields("Student_Name") %>
```

size="20"> Surname:

```
<INPUT name=StudentSurname value=<%=rsbas.Fields("Student Surname")%>
```

size="20"></p>

 Mother Name :

```
<INPUT name=MotherName value=<%=rsbas.Fields("Student Mother Name")%>
```

`size="20"> Father Name :`

```
<INPUT name=FatherName value= <%=rsbas.Fields("Student Father Name")%>
```

```
size="20">&nbsp;&nbsp;  Dep.Name &nbsp;&nbsp;  &nbsp;   :
```

```
<INPUT name=DepName value= <%=rsbas.Fields("Student Dept ID") %>
```

size="20"></P>

| | 1980 | 1981 | 1982 | 1983 | 1984 | 1985 | 1986 | 1987 | 1988 | 1989 | 1990 | 1991 | 1992 | 1993 | 1994 | 1995 | 1996 | 1997 | 1998 | 1999 | 2000 | 2001 | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 | 2012 | 2013 | 2014 | 2015 | 2016 | 2017 | 2018 | 2019 | 2020 | 2021 | 2022 | 2023 | 2024 | 2025 | 2026 | 2027 | 2028 | 2029 | 2030 | 2031 | 2032 | 2033 | 2034 | 2035 | 2036 | 2037 | 2038 | 2039 | 2040 | 2041 | 2042 | 2043 | 2044 | 2045 | 2046 | 2047 | 2048 | 2049 | 2050 | 2051 | 2052 | 2053 | 2054 | 2055 | 2056 | 2057 | 2058 | 2059 | 2060 | 2061 | 2062 | 2063 | 2064 | 2065 | 2066 | 2067 | 2068 | 2069 | 2070 | 2071 | 2072 | 2073 | 2074 | 2075 | 2076 | 2077 | 2078 | 2079 | 2080 | 2081 | 2082 | 2083 | 2084 | 2085 | 2086 | 2087 | 2088 | 2089 | 2090 | 2091 | 2092 | 2093 | 2094 | 2095 | 2096 | 2097 | 2098 | 2099 | 2100 | 2101 | 2102 | 2103 | 2104 | 2105 | 2106 | 2107 | 2108 | 2109 | 2110 | 2111 | 2112 | 2113 | 2114 | 2115 | 2116 | 2117 | 2118 | 2119 | 2120 | 2121 | 2122 | 2123 | 2124 | 2125 | 2126 | 2127 | 2128 | 2129 | 2130 | 2131 | 2132 | 2133 | 2134 | 2135 | 2136 | 2137 | 2138 | 2139 | 2140 | 2141 | 2142 | 2143 | 2144 | 2145 | 2146 | 2147 | 2148 | 2149 | 2150 | 2151 | 2152 | 2153 | 2154 | 2155 | 2156 | 2157 | 2158 | 2159 | 2160 | 2161 | 2162 | 2163 | 2164 | 2165 | 2166 | 2167 | 2168 | 2169 | 2170 | 2171 | 2172 | 2173 | 2174 | 2175 | 2176 | 2177 | 2178 | 2179 | 2180 | 2181 | 2182 | 2183 | 2184 | 2185 | 2186 | 2187 | 2188 | 2189 | 2190 | 2191 | 2192 | 2193 | 2194 | 2195 | 2196 | 2197 | 2198 | 2199 | 2200 | 2201 | 2202 | 2203 | 2204 | 2205 | 2206 | 2207 | 2208 | 2209 | 2210 | 2211 | 2212 | 2213 | 2214 | 2215 | 2216 | 2217 | 2218 | 2219 | 2220 | 2221 | 2222 | 2223 | 2224 | 2225 | 2226 | 2227 | 2228 | 2229 | 2230 | 2231 | 2232 | 2233 | 2234 | 2235 | 2236 | 2237 | 2238 | 2239 | 2240 | 2241 | 2242 | 2243 | 2244 | 2245 | 2246 | 2247 | 2248 | 2249 | 2250 | 2251 | 2252 | 2253 | 2254 | 2255 | 2256 | 2257 | 2258 | 2259 | 2260 | 2261 | 2262 | 2263 | 2264 | 2265 | 2266 | 2267 | 2268 | 2269 | 2270 | 2271 | 2272 | 2273 | 2274 | 2275 | 2276 | 2277 | 2278 | 2279 | 2280 | 2281 | 2282 | 2283 | 2284 | 2285 | 2286 | 2287 | 2288 | 2289 | 2290 | 2291 | 2292 | 2293 | 2294 | 2295 | 2296 | 2297 | 2298 | 2299 | 2300 | 2301 | 2302 | 2303 | 2304 | 2305 | 2306 | 2307 | 2308 | 2309 | 2310 | 2311 | 2312 | 2313 | 2314 | 2315 | 2316 | 2317 | 2318 | 2319 | 2320 | 2321 | 2322 | 2323 | 2324 | 2325 | 2326 | 2327 | 2328 | 2329 | 2330 | 2331 | 2332 | 2333 | 2334 | 2335 | 2336 | 2337 | 2338 | 2339 | 2340 | 2341 | 2342 | 2343 | 2344 | 2345 | 2346 | 2347 | 2348 | 2349 | 2350 | 2351 | 2352 | 2353 | 2354 | 2355 | 2356 | 2357 | 2358 | 2359 | 2360 | 2361 | 2362 | 2363 | 2364 | 2365 | 2366 | 2367 | 2368 | 2369 | 2370 | 2371 | 2372 | 2373 | 2374 | 2375 | 2376 | 2377 | 2378 | 2379 | 2380 | 2381 | 2382 | 2383 | 2384 | 2385 | 2386 | 2387 | 2388 | 2389 | 2390 | 2391 | 2392 | 2393 | 2394 | 2395 | 2396 | 2397 | 2398 | 2399 | 2400 | 2401 | 2402 | 2403 | 2404 | 2405 | 2406 | 2407 | 2408 | 2409 | 2410 | 2411 | 2412 | 2413 | 2414 | 2415 | 2416 | 2417 | 2418 | 2419 | 2420 | 2421 | 2422 | 2423 | 2424 | 2425 | 2426 | 2427 | 2428 | 2429 | 2430 | 2431 | 2432 | 2 |
|--|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|---|
|--|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|---|


```

<p>

<p>
<%if rsbas.Fields("Student_active")=0 then%>
    <a href="AdminStdRegForm.asp">
    <%end if%>New Register Lesson</a></p>

<p>
<%if rsbas.Fields("Student_active")=1 then%>
    <a href="AdminTheStdChangeRegisterForm.asp">
    <%end if%>Change Registeritition Lesson</a></p>

<p>
<a href="AdminTheStdChangeGradeForm.asp">Enter Grade</a></p>
<p><a href="AdminTheStdDelete.asp">Delete the student</a></p>
<p><a href="AdminTheStdPayingViewwe.asp">Show Paying</a></p>
<p><a href="AdminTheStdPayingPage.asp">Add Paying</a></p>
<%
session("StudentNo")=studentno
bag.Close
set bag=nothing
%>
</body>
</html>

```

AdminFindStd.asp

```

<%
if Session("AdminId")="" then Response.Redirect("index.htm")
%>
<HTML><HEAD>
</HEAD>
<BODY>
<P align=center><FONT size=5>Near East University</FONT></P>
<P align=left> Student find</P>
<form method="post" action="AdminStdFinded.asp">

```

</BODY>

<head>

</head>

NEAR EAST UNIVERSITY

<p align="left"><font

COURSE REGISTRATION FORM

```
if session("AdminId")="" or Session("StudentNo")="" then
```

end if

```
select1=request("select1")
```

```
select3=request("select3")
```

```
select5=request("select5")
```

```
Set bag = Server.CreateObject("ADODB.Connection")
```

```

bag.Open "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" &
Server.MapPath("proje.mdb")
Set RsBas=Server.CreateObject("ADODB.Recordset")

SqlQ ="insert into register
values("&select1&","&select2&","&select3&","&select4&","&select5&","&select6&","&stu
dentno&")"
RsBas.open SqlQ,Bag,1,3

SqlQ ="update StudentInfo set student_active=1 where Student_Id="&studentno&""
RsBas.open SqlQ,Bag,1,3

SqlQ ="insert into grade(StudentId)Values("&studentno&")"
RsBas.open SqlQ,Bag,1,3

Response.Write("kayıt yapıldı")

Response.Redirect "listele.asp"
bag.Close
set bag=nothing
%>
</body>
</html>

```

AdminStdRegForm.asp

```

<html>
<head>
<title>Course Registration Form</title>
</head>
<body bgcolor="yellow">
<p align="center"><font size="4">NEAR EAST UNIVERSITY</font></p>
<p><IMG height=200 src="ciu-8[1].gif" width=500 border=0></p>

```



```

<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING:
0px">
<SELECT name="select4">
<option selected value="0">Lessons Codes
<%rsbas.MoveFirst%>
<%for i=0 to rsbas.RecordCount-1
Response.Write("<option
value="&chr(34)&rsbas.Fields("course_Id")&chr(34)&">"&rsbas.Fields("course_code")&"-
"&rsbas.Fields("course_name"))
rsbas.MoveNext
next%>
</OPTION>
</SELECT></p>
<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING:
0px">
<SELECT name="select5">
<option selected value="0">Lessons Codes
<%rsbas.MoveFirst%>
<%for i=0 to rsbas.RecordCount-1
Response.Write("<option
value="&chr(34)&rsbas.Fields("course_Id")&chr(34)&">"&rsbas.Fields("course_code")&"-
"&rsbas.Fields("course_name"))
rsbas.MoveNext
next%>
</OPTION>
</SELECT></p>
<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING:
0px">
<SELECT name="select6">
<option selected value="0">Lessons Codes
<%rsbas.MoveFirst%>
<%for i=0 to rsbas.RecordCount-1

```



```

studentno=session("StudentNo")
Set bag = Server.CreateObject("ADODB.Connection")
bag.Open "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" &
Server.MapPath("proje.mdb")
Set RsBas=Server.CreateObject("ADODB.Recordset")

SqlQ ="select * from register where studentID=" & studentno
rsbas.Close
RsBas.open SqlQ,Bag,1,3
Lesson1=rsbas.Fields("Lesson1")
Lesson2=rsbas.Fields("Lesson2")
Lesson3=rsbas.Fields("Lesson3")
Lesson4=rsbas.Fields("Lesson4")
Lesson5=rsbas.Fields("Lesson5")
Lesson6=rsbas.Fields("Lesson6")

SqlQ ="select * from grade where studentID=" & studentno
rsbas.Close
RsBas.open SqlQ,Bag,1,3
GLesson1=rsbas.Fields("GLesson1")
GLesson2=rsbas.Fields("GLesson2")
GLesson3=rsbas.Fields("GLesson3")
GLesson4=rsbas.Fields("GLesson4")
GLesson5=rsbas.Fields("GLesson5")
GLesson6=rsbas.Fields("GLesson6")

rsbas.Close

SqlQ ="select * from StudentInfo where Student_ID=" & studentno & ""
Response.Write(sqlq)
Response.End
RsBas.open SqlQ,Bag,1,3
%>
<TD width="100%">
<table border="2" cellpadding="0" cellspacing="0" style="border-collapse: collapse"
id="AutoNumber1" height="0">

```



```

<p><INPUT type="text" id=L1 name=Lesson1<%
if Lesson1<>0 then
    Response.Write(" value="&chr(34)&rsbas.Fields("course_code")&" -
"&rsbas.Fields("course_name")&chr(34)&">")
else
    Response.Write(" value="&chr(34)&"Empty"&chr(34)&">")
end if
%>
<SELECT name="Grade1">
<option value="0">Marks
<%for i=0 to 8
Response.Write("<option ")
select case i
case 0:
Grade="AA"
if GLesson1=Grade then
    Response.Write("selected ")
end if
case 1:
Grade="BA"
if GLesson1=Grade then
    Response.Write("selected ")
end if
case 2:
Grade="BB"
if GLesson1=Grade then
    Response.Write("selected ")
end if
case 3:
Grade="CB"
if GLesson1=Grade then
    Response.Write("selected ")
end if
case 4:

```

```

Grade="CC"
if GLesson1=Grade then
    Response.Write("selected ")
end if
case 5:
Grade="DC"
if GLesson1=Grade then
    Response.Write("selected ")
end if
case 6:
Grade="DD"
if GLesson1=Grade then
    Response.Write("selected ")
end if
case 7:
Grade="FD"
if GLesson1=Grade then
    Response.Write("selected ")
end if
case 8:
Grade="FF"
if GLesson1=Grade then
    Response.Write("selected ")
end if
end select
Response.Write("value=" & chr(34) & Grade & chr(34) & ">" & Grade)
next%>
</OPTION>
</SELECT></p>
<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING:
0px">
<%
SqlQ ="select * from course where course_ID=" & Lesson2 & " and Dept_ID=" & DeptId
rsbas.Close

```

```

RsBas.open SqlQ,Bag,1,3
%>
<INPUT type="text" id=L2 name=Lesson2<%
if Lesson2<>0 then
    Response.Write(" value="&chr(34)&rsbas.Fields("course_code")&" -
    "&rsbas.Fields("course_name")&chr(34)&">")
else
    Response.Write(" value="&chr(34)&"Empty"&chr(34)&">")
end if
%>
<SELECT name="Grade2">
<option value="0">Marks
<%for i=0 to 8
Response.Write("<option ")
select case i
case 0:
Grade="AA"
if GLesson2=Grade then
    Response.Write("selected ")
end if
case 1:
Grade="BA"
if GLesson2=Grade then
    Response.Write("selected ")
end if
case 2:
Grade="BB"
if GLesson2=Grade then
    Response.Write("selected ")
end if
case 3:
Grade="CB"
if GLesson2=Grade then
    Response.Write("selected ")

```



```

        end if
case 4:
Grade="CC"
if GLesson2=Grade then
    Response.Write("selected ")
    end if
case 5:
Grade="DC"
if GLesson2=Grade then
    Response.Write("selected ")
    end if
case 6:
Grade="DD"
if GLesson2=Grade then
    Response.Write("selected ")
    end if
case 7:
Grade="FD"
if GLesson2=Grade then
    Response.Write("selected ")
    end if
case 8:
Grade="FF"
if GLesson2=Grade then
    Response.Write("selected ")
    end if
end select
Response.Write("value=" & chr(34) & Grade & chr(34) & ">" & Grade)
next%>
</OPTION>
</SELECT></p>
<p align="left">
<%
SqlQ ="select * from course where course_ID=" & Lesson3 & " and Dept_ID=" & DeptId

```

```

rsbas.Close
RsBas.open SqlQ,Bag,1,3
%>
<INPUT type="text" id=L3 name=Lesson3<%
if Lesson3<>0 then
    Response.Write(" value=" &chr(34)&rsbas.Fields("course_code")&" -
    "&rsbas.Fields("course_name")&chr(34)&">")
else
    Response.Write(" value=" &chr(34)&"Empty"&chr(34)&">")
end if
%>
<SELECT name="Grade3">
<option value="0">Marks
<%for i=0 to 8
Response.Write("<option ")
select case i
case 0:
Grade="AA"
if GLesson3=Grade then
    Response.Write("selected ")
end if
case 1:
Grade="BA"
if GLesson3=Grade then
    Response.Write("selected ")
end if
case 2:
Grade="BB"
if GLesson3=Grade then
    Response.Write("selected ")
end if
case 3:
Grade="CB"
if GLesson3=Grade then

```

```

        Response.Write("selected ")
    end if
case 4:
Grade="CC"
if GLesson3=Grade then
    Response.Write("selected ")
end if
case 5:
Grade="DC"
if GLesson3=Grade then
    Response.Write("selected ")
end if
case 6:
Grade="DD"
if GLesson3=Grade then
    Response.Write("selected ")
end if
case 7:
Grade="FD"
if GLesson3=Grade then
    Response.Write("selected ")
end if
case 8:
Grade="FF"
if GLesson3=Grade then
    Response.Write("selected ")
end if
end select
Response.Write("value=" & chr(34) & Grade & chr(34) & ">" & Grade)
next %>
</OPTION>
</SELECT></p>
<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING:
0px">

```



```

<%
SqlQ="select * from course where course_ID=" & Lesson4 & " and Dept_ID=" & DeptId
rsbas.Close
RsBas.open SqlQ,Bag,1,3
%>
<INPUT type="text" id=L4 name=Lesson4<%
if Lesson4<>0 then
    Response.Write(" value=" & chr(34) & rsbas.Fields("course_code") & " - " & rsbas.Fields("course_name") & chr(34) & ">")
else
    Response.Write(" value=" & chr(34) & "Empty" & chr(34) & ">")
end if
%>
<SELECT name="Grade4">
<option value="0">Marks
<%for i=0 to 8
Response.Write("<option ")
select case i
case 0:
Grade="AA"
if GLesson4=Grade then
    Response.Write("selected ")
end if
case 1:
Grade="BA"
if GLesson4=Grade then
    Response.Write("selected ")
end if
case 2:
Grade="BB"
if GLesson4=Grade then
    Response.Write("selected ")
end if
case 3:

```

```

Grade="CB"
if GLesson4=Grade then
    Response.Write("selected ")
end if
case 4:
Grade="CC"
if GLesson4=Grade then
    Response.Write("selected ")
end if
case 5:
Grade="DC"
if GLesson4=Grade then
    Response.Write("selected ")
end if
case 6:
Grade="DD"
if GLesson4=Grade then
    Response.Write("selected ")
end if
case 7:
Grade="FD"
if GLesson4=Grade then
    Response.Write("selected ")
end if
case 8:
Grade="FF"
if GLesson4=Grade then
    Response.Write("selected ")
end if
end select
Response.Write("value=" & chr(34) & Grade & chr(34) & ">" & Grade)
next%>
</OPTION>
</SELECT></p>

```

```

<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING:
0px">
<%
SqlQ ="select * from course where course_ID=" & Lesson5 & " and Dept_ID=" & DeptId
rsbas.Close
RsBas.open SqlQ,Bag,1,3
%>
<INPUT type="text" id=L5 name=Lesson5<%
if Lesson5<>0 then
    Response.Write(" value=" & chr(34) & rsbas.Fields("course_code") & " -
" & rsbas.Fields("course_name") & chr(34) & ">")
else
    Response.Write(" value=" & chr(34) & "Empty" & chr(34) & ">")
end if
%>
<SELECT name="Grade5">
<option value="0">Marks
<%for i=0 to 8
Response.Write("<option ")
select case i
case 0:
Grade="AA"
if GLesson5=Grade then
    Response.Write("selected ")
end if
case 1:
Grade="BA"
if GLesson5=Grade then
    Response.Write("selected ")
end if
case 2:
Grade="BB"
if GLesson5=Grade then
    Response.Write("selected ")

```



```

end if
case 3:
Grade="CB"
if GLesson5=Grade then
    Response.Write("selected ")
end if
case 4:
Grade="CC"
if GLesson5=Grade then
    Response.Write("selected ")
end if
case 5:
Grade="DC"
if GLesson5=Grade then
    Response.Write("selected ")
end if
case 6:
Grade="DD"
if GLesson5=Grade then
    Response.Write("selected ")
end if
case 7:
Grade="FD"
if GLesson5=Grade then
    Response.Write("selected ")
end if
case 8:
Grade="FF"
if GLesson5=Grade then
    Response.Write("selected ")
end if
end select
Response.Write("value=" & chr(34) & Grade & chr(34) & ">" & Grade)
next%>

```

```

</OPTION>
</SELECT></p>
<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING:
0px">
<%
SqlQ ="select * from course where course_ID=" & Lesson6 & " and Dept_ID=" & DeptId
rsbas.Close
RsBas.open SqlQ,Bag,1,3
%>
<INPUT type="text" id=L6 name=Lesson6<%
if Lesson6<>0 then
    Response.Write(" value=" & chr(34) & rsbas.Fields("course_code") & " -
" & rsbas.Fields("course_name") & chr(34) & ">")
else
    Response.Write(" value=" & chr(34) & "Empty" & chr(34) & ">")
end if
%>
<SELECT name="Grade6">
<option value="0">Marks
<%for i=0 to 8
Response.Write("<option ")
select case i
case 0:
Grade="AA"
if GLesson6=Grade then
    Response.Write("selected ")
end if
case 1:
Grade="BA"
if GLesson6=Grade then
    Response.Write("selected ")
end if
case 2:
Grade="BB"

```

```
if GLesson6=Grade then
    Response.Write("selected ")
end if
```

case 3:

Grade="CB"

```
if GLesson6=Grade then
    Response.Write("selected ")
end if
```

case 4:

Grade="CC"

```
if GLesson6=Grade then
    Response.Write("selected ")
end if
```

case 5:

Grade="DC"

```
if GLesson6=Grade then
    Response.Write("selected ")
end if
```

case 6:

Grade="DD"

```
if GLesson6=Grade then
    Response.Write("selected ")
end if
```

case 7:

Grade="FD"

```
if GLesson6=Grade then
    Response.Write("selected ")
end if
```

case 8:

Grade="FF"

```
if GLesson6=Grade then
    Response.Write("selected ")
end if
```

end select

</html>

ntInfo where Student_ID=" & studentno & ""

dding="0" cellspacing="0" style="border-collapse: collapse"

$$= "0">$$

size=3> </p>

Student No. :

```
tNo value=<%=rsbas.Fields("Student_ID") %> size="20">&nbsp;
```

```
tName value=<%=rsbas.Fields("Student_Name") %>
```

```
tSurname value=<%=rsbas.Fields("Student_Surname")%>
```

Mother Name :

```
rName value=<%=rsbas.Fields("Student_Mother_Name")%>
```

Name : _____


```
rsbas.MoveNext
next%>
</OPTION>
</SELECT>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~</p>
<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING: 0px">
<SELECT name="select2">
<option selected value="0">Lessons Codes
<%rsbas.MoveFirst%>
<%for i=0 to rsbas.RecordCount-1
Response.Write("<option ")
if Lesson2=rsbas.Fields("course_Id")then
    Response.Write("selected ")
end if
Response.Write("value=" &chr(34)&rsbas.Fields("course_Id")&chr(34)&" "&rsbas.Fields("c
ourse_code")&"-"&rsbas.Fields("course_name"))
rsbas.MoveNext
next%>
</OPTION>
</SELECT></p>
<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING: 0px">
<SELECT name="select3">
<option selected value="0">Lessons Codes
<%rsbas.MoveFirst%>
<%for i=0 to rsbas.RecordCount-1
Response.Write("<option ")
if Lesson3=rsbas.Fields("course_Id")then
    Response.Write("selected ")
end if
Response.Write("value=" &chr(34)&rsbas.Fields("course_Id")&chr(34)&" "&rsbas.Fields("c
ourse_code")&"-"&rsbas.Fields("course_name"))
rsbas.MoveNext
```



```

next%>
</OPTION>
</SELECT></p>
<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING:
0px">
<SELECT name="select4">
<option selected value="0">Lessons Codes
<%rsbas.MoveFirst%>
<%for i=0 to rsbas.RecordCount-1
Response.Write("<option ")
if Lesson4=rsbas.Fields("course_Id")then
    Response.Write("selected ")
end if
Response.Write("value="&chr(34)&rsbas.Fields("course_Id")&chr(34)&">"&rsbas.Fields("c
ourse_code")&"-"&rsbas.Fields("course_name"))
rsbas.MoveNext
next%>
</OPTION>
</SELECT></p>
<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING:
0px">
<SELECT name="select5">
<option selected value="0">Lessons Codes
<%rsbas.MoveFirst%>
<%for i=0 to rsbas.RecordCount-1
Response.Write("<option ")
if Lesson5=rsbas.Fields("course_Id")then
    Response.Write("selected ")
end if
Response.Write("value="&chr(34)&rsbas.Fields("course_Id")&chr(34)&">"&rsbas.Fields("c
ourse_code")&"-"&rsbas.Fields("course_name"))
rsbas.MoveNext
next%>
</OPTION>

```

```

</SELECT></p>
<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING:
0px">
<SELECT name="select6">
<option selected value="0">Lessons Codes
<%rsbas.MoveFirst%>
<%for i=0 to rsbas.RecordCount-1
Response.Write("<option ")
if Lesson6=rsbas.Fields("course_Id")then
    Response.Write("selected ")
    end if
Response.Write("value="&chr(34)&rsbas.Fields("course_Id")&chr(34)&" "&rsbas.Fields("c
ourse_code")&" "&rsbas.Fields("course_name"))
rsbas.MoveNext
next%>
</OPTION>
</SELECT>&nbsp;</p>
<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING:
0px">
</p>
<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING:
0px"><font size="4"><b>
-----
</b></font>
</p>
<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING:
0px">&nbsp;</p>
<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING:
0px">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
<input type="submit" value="SAVE" name="B1"><input type="reset" value="Reset"
name="B2"></p>
</form>

```



```

        Response.End
    end if

    studentno=Session("StudentNo")
    stdpayment=request("payment")

    Set bag = Server.CreateObject("ADODB.Connection")
    bag.Open "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" &
    Server.MapPath("proje.mdb")
    Set RsBas=Server.CreateObject("ADODB.Recordset")

    SqlQ ="insert into pay
    values("&studentno&","&stdpayment&","&day(date())&","&month(date())&","&year(date())
    &")"
    RsBas.open SqlQ,Bag,1,3

    SqlQ ="select * from StudentInfo where Student_Id="&studentno&"
    RsBas.open SqlQ,Bag,1,3
    harci=rsbas.Fields("Harc")

    SqlQ ="select * from Pay where studentId="&studentno&" and Year="&year(date())
    'Response.Write(sqlq)
    'Response.End
    rsbas.Close
    RsBas.open SqlQ,Bag,1,3
    TotalPayment=0
    Response.Write("<p>")
    for i=0 to rsbas.RecordCount-1
        totalpayment=totalpayment+rsbas.Fields("Pays")
        Response.Write("Ödeme:"&rsbas.Fields("Day")&"/"&rsbas.Fields("Month")&"/"&rsb
as.Fields("Year")&" : "&rsbas.Fields("Pays"))
        Response.Write("<p>")
        rsbas.MoveNext
    next
    if harci>totalpayment then
        Response.Write((harci-totalpayment)&" borcu kalmıştır.")
    
```

```

else
    Response.Write("Borcu bulunmamaktadır.")
end if

Response.Redirect "listele.asp"

bag.Close

set bag=nothing

%>

</body></html>

```

AdminTheStdPayingPage.asp

```

<%
if Session("AdminId")="" or Session("StudentNo")="" then Response.Redirect("index.htm")
%>

<HTML><HEAD>

</HEAD>

<BODY>

<P align=center><FONT size=5>Near East University</FONT></P>

<P align=left> Student Payment</P>

<form method="post" action="AdminTheStdPaying.asp">

<P> Payment&nbsp;  <INPUT name=Payment>&nbsp;  <INPUT id=submit1
type=submit value=Submit name=submit1></P>

</form>

</BODY>

</HTML>

```

AdminTheStdPayingViewe.asp

```

<%
if Session("AdminId")="" or Session("StudentNo")="" then Response.Redirect("index.htm")
%>

<HTML><HEAD>

</HEAD>

<BODY>

```



```
SqlQ ="select * from StudentInfo where Student_Id=" &studentno& ""
```

```
RsBas.open SqlQ,Bag,1,3
```

```
harci=rsbas.Fields("Harc")
```

```
SqlQ ="select * from Pay where studentId=" &studentno& " and Year=" &year(date())
```

```
'Response.Write(sqlq)
```

```
'Response.End
```

```
rsbas.Close
```

```
RsBas.open SqlQ,Bag,1,3
```

```
TotalPayment=0
```

```
Response.Write("<p>")
```

```
for i=0 to rsbas.RecordCount-1
```

```
    totalpayment=totalpayment+rsbas.Fields("Pays")
```

```
    Response.Write("Ödeme:
```

```
"&rsbas.Fields("Day")&"/"&rsbas.Fields("Month")&"/"&rsbas.Fields("Year")&" Miktar:
```

```
"&rsbas.Fields("Pays"))
```

```
    Response.Write("<p>")
```

```
    rsbas.MoveNext
```

```
next
```

```
Response.Write("<p>")
```

```
if harci>totalpayment then
```

```
    Response.Write((harci-totalpayment)&" borcu kalmıştır.")
```

```
else
```

```
    Response.Write("Borcu bulunmamaktadır.")
```

```
end if
```

```
'Response.Redirect "listele.asp"
```

```
bag.Close
```

```
set bag=nothing
```

```
%>
```

```
</body>
```

```
</html>
```

Change.asp

```
<%@ Language=VBScript %>
<%session("StudentNo")%>
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
</HEAD>
<BODY>
<form action="change1.asp" method="post">
Mother name:<input type="text" name="anne"><br>
Father name:<input type="text" name="baba"><br>
Old password:<input type="text" name="oldpass"><br>
New password:<input type="text" name="newpass"><br>
<input type="submit" value="change">
</form>
</BODY>
</HTML>
```

change1.asp

```
<%@ Language=VBScript %>
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
</HEAD>
<BODY>
<%
if Session("StudentNo")="" then
    Response.End
end if
anne=Request.Form("anne")
```



```

baba=Request.Form("baba")
oldpass=Request.Form("oldpass")
newpass=Request.Form("newpass")

Set bag = Server.CreateObject("ADODB.Connection")
bag.Open "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" &
Server.MapPath("proje.mdb")
Set rrk=Server.CreateObject("ADODB.Recordset")
s="select * from StudentInfo"
set rrk=bag.Execute(s)
dim c
c=0

while not rrk.eof
if anne=rrk("Student_Mother_name") and baba=rrk("Student_Father_name") and
oldpass=rrk("Student_password") then
sl="update StudentInfo set Student_password=" & newpass & " where
Student_ID=" & session("StudentNo") & ""
bag.execute(sl)
c=1
end if
rrk.MoveNext
wend
if c=1 then
Response.Write "Password changed"

elseif c=0 then
Response.Write "Control your values"
end if
%>
</BODY>
</HTML>

```

courseE.asp

[illegible]

"check_checkb(3,this.value)"><input name="T3" ><input name="T47" size="55" ><input name="T91" size="5" ><input name="T135" size="5" ><input name="T179" size="5" DISABLED></p>

<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING: 0px"><input type="checkbox" name="C" value="ON" onclick = "check_checkb(4,this.value)"><input name="T4" ><input name="T48" size="55" ><input name="T92" size="5" ><input name="T136" size="5" ><input name="T180" size="5" DISABLED></p>

<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING: 0px"><input type="checkbox" name="C" value="ON" onclick = "check_checkb(5,this.value)"><input name="T5" ><input name="T49" size="55" ><input name="T93" size="5" ><input name="T137" size="5" ><input name="T181" size="5" DISABLED></p>

<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING: 0px"><input type="checkbox" name="C" value="ON" onclick = "check_checkb(6,this.value)"><input name="T6" ><input name="T50" size="55" ><input name="T94" size="5" ><input name="T138" size="5" ><input name="T182" size="5" DISABLED></p>

<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING: 0px"><input type="checkbox" name="C" value="ON" onclick = "check_checkb(7,this.value)"><input name="T7" ><input name="T51" size="55" ><input name="T95" size="5" ><input name="T139" size="5" ><input name="T183" size="5" DISABLED></p>

<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING: 0px"><input type="checkbox" name="C" value="ON" onclick = "check_checkb(8,this.value)"><input name="T8" ><input name="T52" size="55" ><input name="T96" size="5" ><input name="T140" size="5" ><input name="T184" size="5" DISABLED></p>

<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING: 0px">

 </p>

<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING: 0px">

```
<input type="submit" value="SAVE" name="B1"><input type="reset" value="CANCEL"
name="B2"><input type="reset" value="PRINT" name="B3"></p>
</form>
</body>
</html>
```

kayıt son.html

```
<? session_register("login");
session_register("std_no");
session_register("std_name");
session_register("std_surname");
session_register("std_dep");
session_register("std_regdate");
session_register("ac_year");
session_register("sem");

session_register("course_code");
session_register("course_name");
session_register("course_credit");
include "admin.php";?>
<html>

<head>
<meta http-equiv="Content-Language" content="tr">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<meta name="GENERATOR" content="Microsoft FrontPage 5.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<title>Course Registration Form</title>
</head>

<body bgcolor="yellow">
```


[illegible]

[illegible]


```

Response.Write(" value=" & GLesson1 & ">")%>
</p>
<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING:
0px">
<%
SqlQ ="select * from course where course_ID=" & Lesson2 & " and Dept_ID=" & DeptId
rsbas.Close
RsBas.open SqlQ,Bag,1,3
%>
<INPUT type="text" id=L2 name=Lesson2<%
if Lesson2<>0 then
    Response.Write(" value=" & chr(34) & rsbas.Fields("course_code") & " -
    " & rsbas.Fields("course_name") & chr(34) & ">")
else
    Response.Write(" value=" & chr(34) & "Empty" & chr(34) & ">")
end if
%>
<INPUT type="text" id=GL2 name=GLesson2<%
Response.Write(" value=" & GLesson2 & ">")%>
</p>
<p align="left">
<%
SqlQ ="select * from course where course_ID=" & Lesson3 & " and Dept_ID=" & DeptId
rsbas.Close
RsBas.open SqlQ,Bag,1,3
%>
<INPUT type="text" id=L3 name=Lesson3<%
if Lesson3<>0 then
    Response.Write(" value=" & chr(34) & rsbas.Fields("course_code") & " -
    " & rsbas.Fields("course_name") & chr(34) & ">")
else
    Response.Write(" value=" & chr(34) & "Empty" & chr(34) & ">")
end if
%>

```



```

<INPUT type="text" id=GL3 name=GLesson3<%
Response.Write(" value=" & GLesson3 & ">")%>
</p>
<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING:
0px">
<%
SqlQ ="select * from course where course_ID=" & Lesson4 & " and Dept_ID=" & DeptId
rsbas.Close
RsBas.open SqlQ,Bag,1,3
%>
<INPUT type="text" id=L4 name=Lesson4<%
if Lesson4<>0 then
    Response.Write(" value=" & chr(34) & rsbas.Fields("course_code") & " -
" & rsbas.Fields("course_name") & chr(34) & ">")
else
    Response.Write(" value=" & chr(34) & "Empty" & chr(34) & ">")
end if
%>
<INPUT type="text" id=GL4 name=GLesson4<%
Response.Write(" value=" & GLesson4 & ">")%>
</p>
<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING:
0px">
<%
SqlQ ="select * from course where course_ID=" & Lesson5 & " and Dept_ID=" & DeptId
rsbas.Close
RsBas.open SqlQ,Bag,1,3
%>
<INPUT type="text" id=L5 name=Lesson5<%
if Lesson5<>0 then
    Response.Write(" value=" & chr(34) & rsbas.Fields("course_code") & " -
" & rsbas.Fields("course_name") & chr(34) & ">")
else
    Response.Write(" value=" & chr(34) & "Empty" & chr(34) & ">")

```

```

        end if
    %>
    <INPUT type="text" id=GL5 name=GLesson5<%
    Response.Write(" value=" & GLesson5 & ">")%>
</p>
<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING:
0px">
<%
SqlQ ="select * from course where course_ID=" & Lesson6 & " and Dept_ID=" & DeptId
rsbas.Close
RsBas.open SqlQ,Bag,1,3
%>
<INPUT type="text" id=L6 name=Lesson6<%
if Lesson6<>0 then
    Response.Write(" value=" & chr(34) & rsbas.Fields("course_code") & " -
" & rsbas.Fields("course_name") & chr(34) & ">")
else
    Response.Write(" value=" & chr(34) & "Empty" & chr(34) & ">")
end if
%>
<INPUT type="text" id=GL6 name=GLesson6<%
Response.Write(" value=" & GLesson6 & ">")%>
</p>
<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING:
0px">
</p>
<p align="left" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; WORD-SPACING:
0px"><font size="4"><b>
-----
</b></font>
</p>
<%
bag.Close
set bag=nothing

```

```
%>
</body>
</html>
```

StudentMenu.asp

```
<html>

<head>
<meta http-equiv="Content-Language" content="tr">
<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-1254">
<title>NEAR EAST UNIVERSITY</title>
</head>

<body>

<p align="center"><b><font size="6">NEAR EAST UNIVERSITY</font></b></p>
<p align="center">Log in Student No:<%=Session("StudentNo")%></p>
<p align="center"><a href="StdInfo.asp">Student Info</a></p>
<P align=center><a href="change.asp">Change password</a></P>
<P align=center><a href="StudentGradeList.asp">Grade List</P>
<P align=center><a href="StudentPayingViewe.asp">Payment</P>

</body>
</html>
```

StudentPayingViewe.asp

```
<html>
<head>
```


[illegible]

```
Response.Write("Ödeme:
"&rsbas.Fields("Day")&"/"&rsbas.Fields("Month")&"/"&rsbas.Fields("Year")&" Miktar:
"&rsbas.Fields("Pays"))
Response.Write("<p>")
rsbas.MoveNext
next
Response.Write("<p>")
if harci>totalpayment then
    Response.Write((harci-totalpayment)&" borcu kalmıştır.")
else
    Response.Write("Borcu bulunmamaktadır.")
end if
Response.Redirect "listele.asp"
bag.Close
set bag=nothing
%>
</body>
</html>
```

CONCLUSION

We have used ASP technology in order to accomplish this project. The most important aspect of ASP is database management. All the information of the students have been saved to a database. Also this project guided us to improved scripting and HTML knowledge.

While designing web interfaces with third party programs such as FrontPage and Dreamweaver that necessarily do not need programming background, but integrating ASP to the HTML codes requires a scripting and background knowledge that leaded us to improve scripting knowledge in web programming.

The most important reason that ASP appealed us that it is a key to the future while Internet is spreading in every segment of life and millions of people are getting online everyday.

In this project, we have established the fundamentals of ASP and we will be happy to use it in our professional life.

REFERENCES

- 1.<http://www.aspxnet.de/>
- 2.<http://www.upu.int/security>
- 3.<http://www.neu.edu.tr>
4. <http://msdn.microsoft.com>
<http://www.devguru.com>
5. Microsoft Access
<http://www.microsoft.com/office/access/default.htm>
6. ASP: Learning by Example , Robert B. Mellor ,2001
7. Beginning ASP Databases, John Kauffman, Kevin Spencer, Thearon Willis, John Kauffman
, 1999