NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

ONLINE FINANCIAL SYSTEM FOR A CONSTRUCTION COMPANY

Graduation Project
COM-400

Student:	Anış ÖZTEKiN (20030166)
Supervisor:	Mr Ümit SOYER

Nicosia-2008

ACKNOWLEDGMENT

I wish to thank my advisor, Mr Ümit SOYER for his great advice and recommendation for my project.

I would like to thank to Dr Ümit İLHAN. Whenever I had questions, he was always helpful.

During my education I learn lost of thing from my every teachers and I thanks to them.

I thank my family for their support.

I thank my fiance for prodiving details about his company. I dedicate this work to him.

ABSTRACT

The aim of this project was to online financial system for a construction company that contains constructions and insulations details. The program was prepared using HTML and ASP programming and Microsoft Access Database. This project consist of many pages. The first page of program is consist three links to choose how login the web page. Alternatives are admin, user and guesst. The admin and user page of the program is designed for login which are login for user id and password. There is necessary to enter the program user id and password for individual. The name of this procedure is login. After admin and user login, the program may become to active. And admin can make search, insert, update and delete the informations in the program. If user login the program make search, insert and update the informations in the program. The guesst just see the product. These are simply expressing how the program was designed to use in proper and secure way.

TABLE OF CONTENS

LIST OF ABBREVIATIONS

WWW	World Wide Web
IIS	Internet Information Service
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language
ASP	Active Server Page
SQL	Structured Query Language

INTRODUCTION

HTML (HyperText Markup Language) is the predominant markup language for web pages. It provides a means to describe the structure of text-based information in a document by denoting certain text as links, headings, paragraphs, lists, and so on. This project financial system for a construction company. So using HTML with ASP to view datas from the database.

ASP stands for Active Server Pages. ASP is a program that runs inside IIS. IIS stands for Internet Information Services. In this project used Microsoft Access Database. Steps are: Collecting to a database, adding data from database, and modifying and deleting data from the database. During the ASP programming language writen can make adding, modify and delete. The company when use this web page they work will be more easy. They see all their work what they did. This program run like localhost. Main page for enter the pages. we have chooses, if admin want to enter program, can enter with password and if details same with datas in the database entered program and admin also establish the company and enter program for search they did and what they sell. If they build new house or develop workers about insulation admin can insert the new builds and details. If some changes make for buildings or insulations admin can update. If want to delete, admin can make. Admin also can control the user. User it means workers in the company. Admin can enter (also can delete, search and update) user details and can give password for user enter the pages for make search, insert and update. And part of the balance about selling details. It is more useful for company to keep details in to database and can see which house or insulation selled, who buy it and how much money give it and what the balance, the program calculate and show. We have links to details of company. Every guesst just can see these details and make search, and can contact with the company. When the company use this web page they will be more orderly and can keep every details about what they do. And web page is the best way to advertisement and to be well known.

This document is organized as follows:

Chapter 1 describes HTML, and the history of HTML.

Chapter 2 describes ASP, and the history of ASP.

Chapter 3 describes database, and the history of database.

Chapter 4 describes SQL, and the history of SQL.

Chapter 5 description about project.

CHAPTER 1

HYPERTEXT MARKUP LANGUAGE (HTML)

1.1 Introduction to 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.[1]

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. The vocabulary of these elements and a description of the overall design of HTML documents is given in the rest of Section 2. The Last part of the section also describes standard naming schemes for HTML documents and related files.

The detailed rules for HTML (the names of the tags/elements, how they can be used) are defined using another language known as the standard generalized markup language, or SGML. SGML is wickedly difficult, and was designed for massive document collections, such as repair manuals for F-16 fighters, or maintenance plans for nuclear submarines. Fortunately, HTML is much simpler!

However, SGML has useful features that HTML lacks. For this reason, markup language and software experts have developed a new language, called XML (the eXtensible markup language) which has most of the most useful features of HTML and SGML.

1.2 What is HTML?

To publish information for global distribution, one needs a universally understood language, a kind of publishing mother tongue that all computers may potentially understand. The publishing language used by the World Wide Web is HTML (from HyperText Markup Language).

HTML gives authors the means to:

- Publish online documents with headings, text, tables, lists, photos, etc.
- Retrieve online information via hypertext links, at the click of a button.
- Design forms for conducting transactions with remote services, for use in searching for information, making reservations, ordering products, etc.
- Include spread-sheets, video clips, sound clips, and other applications directly in their documents.

1.3 History of HTML

HTML was originally developed by Tim Berners-Lee while at CERN, and popularized by the Mosaic browser developed at NCSA. During the course of the 1990s it has blossomed with the explosive growth of the Web. During this time, HTML has been extended in a number of ways. The Web depends on Web page authors and vendors sharing the same conventions for HTML. This has motivated joint work on specifications for HTML.

HTML 2.0 (November 1995) was developed under the aegis of the Internet Engineering Task Force (IETF) to codify common practice in late 1994. HTML+ (1993) and HTML 3.0 (1995) proposed much richer versions of HTML. Despite never receiving consensus in standards discussions, these drafts led to the adoption of a range of new features. The efforts of the World Wide Web Consortium's HTML Working Group to codify common practice in 1996 resulted in HTML 3.2 (January 1997).

Most people agree that HTML documents should work well across different browsers and platforms. Achieving interoperability lowers costs to content providers since they must develop only one version of a document.

If the effort is not made, there is much greater risk that the Web will devolve into a proprietary world of incompatible formats, ultimately reducing the Web's commercial potential for all participants.

Each version of HTML has attempted to reflect greater consensus among industry players so that the investment made by content providers will not be wasted and that their documents will not become unreadable in a short period of time.

HTML has been developed with the vision that all manner of devices should be able to use information on the Web: PCs with graphics displays of varying resolution and color depths, cellular telephones, hand held devices, devices for speech for output and input, computers with high or low bandwidth, and so on.

1.4 Basic HTML Data Types

1.4.1 Case Information

Each attribute definition includes information about the case-sensitivity of its values. The case information is presented with the following keys:

CS The value is case-sensitive (i.e., user agents interpret "a" and "A" differently).
CI The value is case-insensitive (i.e., user agents interpret "a" and "A" as the same).
CN The value is not subject to case changes, e.g., because it is a number or a character from the document character set.
CA The element or attribute definition itself gives case information.
CT Consult the type definition for details about case-sensitivity.

If an attribute value is a list, the keys apply to every value in the list, unless otherwise indicated.

1.4.2 SGML Basic Types

The document type definition specifies the syntax of HTML element content and attribute values using SGML tokens (e.g., PCDATA, CDATA, NAME, ID, etc.).

The following is a summary of key information:

- **CDATA** is a sequence of characters from the document character set and may include character entities. User agents should interpret attribute values as follows:
    - Replace character entities with characters,
    - Ignore line feeds,
    - Replace each carriage return or tab with a single space.

  User agents may ignore leading and trailing white space in CDATA attribute values (e.g., " myval " may be interpreted as "myval"). Authors should not declare attribute values with leading or trailing white space.

  For some HTML 4 attributes with CDATA attribute values, the specification imposes further constraints on the set of legal values for the attribute that may not be expressed by the DTD.

  Although the STYLE and SCRIPT elements use CDATA for their data model, for these elements, CDATA must be handled differently by user agents. Markup and entities must be treated as raw text and passed to the application as is. The first occurrence of the character sequence "</" (end-tag open delimiter) is treated as terminating the end of the element's content. In valid documents, this would be the end tag for the element.

- **ID** and **NAME** tokens must begin with a letter ([A-Za-z]) and may be followed by any number of letters, digits ([0-9]), hyphens ("-"), underscores ("_"), colons (":"), and periods (".").
- **IDREF** and **IDREFS** are references to ID tokens defined by other attributes. IDREF is a single token and IDREFS is a space-separated list of tokens.
- **NUMBER** tokens must contain at least one digit ([0-9]).

1.4.3 URIs

URIs are represented in the DTD by the parameter entity %URI;.

URIs in general are case-sensitive. There may be URIs, or parts of URIs, where case doesn't matter (e.g., machine names), but identifying these may not be easy. Users should always consider that URIs are case-sensitive (to be on the safe side).

Please consult the appendix for information about non-ASCII characters in URI attribute values.

1.4.4 Colors

A color value may either be a hexadecimal number (prefixed by a hash mark) or one of the following sixteen color names. The color names are case-insensitive.

Table 1.1, the color values "#800080" and "Purple" both refer to the color purple.

Color names and sRGB values

■	Black = "#000000"	▨	Green = "#008000"
□	Silver = "#C0C0C0"	□	Lime = "#00FF00"
▨	Gray = "#808080"	▨	Olive = "#808000"
□	White = "#FFFFFF"	□	Yellow = "#FFFF00"
■	Maroon = "#800000"	■	Navy = "#000080"
■	Red = "#FF0000"	■	Blue = "#0000FF"
■	Purple = "#800080"	▨	Teal = "#008080"
▨	Fuchsia = "#FF00FF"	□	Aqua = "#00FFFF"

1.4.4.1 Notes on using colors

Although colors can add significant amounts of information to documents and make them more readable, please consider the following guidelines when including color in your documents:

- The use of HTML elements and attributes for specifying color is deprecated. You are encouraged to use style sheets instead.
- Don't use color combinations that cause problems for people with color blindness in its various forms.
- If you use a background image or set the background color, then be sure to set the various text colors as well.
- Colors specified with the BODY and FONT elements and bgcolor on tables look different on different platforms (e.g., workstations, Macs, Windows, and LCD panels vs. CRTs), so you shouldn't rely entirely on a specific effect. In the future, support for the [SRGB] color model together with ICC color profiles should mitigate this problem.
- When practical, adopt common conventions to minimize user confusion.

1.4.5 Lengths

HTML specifies three types of length values for attributes:

1. **Pixels**: The value (%Pixels; in the DTD) is an integer that represents the number of pixels of the canvas (screen, paper). Thus, the value "50" means fifty pixels. For normative information about the definition of a pixel, please consult [CSS1].
2. **Length**: The value (%Length; in the DTD) may be either a %Pixel; or a percentage of the available horizontal or vertical space. Thus, the value "50%" means half of the available space.
3. **MultiLength**: The value (%MultiLength; in the DTD) may be a %Length; or a relative length. A relative length has the form "i*", where "i" is an integer. When allotting space among elements competing for that space, user agents allot pixel and percentage lengths first, then divide up remaining available space among relative lengths. Each relative length receives a portion of the available space that is proportional to the integer preceding the "*".

1.5 Elements in HTML Documents

The HTML instructions, along with the text to which the instructions apply, are called HTML elements. The HTML instructions are themselves called tags, and look like <element_name> -- that is, they are simply the element name surrounded by left and right angle brackets.

Most elements mark blocks of the document for particular purpose or formatting: the above <element_name> tag marks the beginning of such as section. The end of this section is then marked by the ending tag </element_name> -- note the leading slash character "/" that appears in front of the element name in an end tag. End, or stop tags are always indicated by this leading slash character.

For example, the heading at the top of this page is an H2 element, (a level 2 heading) which is written as:

<H2> 2.1 Elements in HTML </H2>.

Empty Elements

Some elements are empty -- that is, they do not affect a block of the document in some way. These elements do not require an ending tag. An example is the <HR> element, which draws a horizontal line across the page. This element would simply be entered as

<HR>

Upper and Lower Case

Element names are case insensitive. Thus, the the horizontal rule element can be written as any of <hr>, <Hr> or <HR>.

Elements can have Attributes

Many elements can have arguments that pass parameters to the interpreter handling this element. These arguments are called attributes of the element. For example, consider the element A, which marks a region of text as the beginning (or end) of a hypertext link. This element can have several attributes. One of them, HREF, specifies the hypertext document to which the marked piece of text is linked. To specify this in the tag for A you write:

<A HREF="http://www.somewhere.ca/file.html"> marked text </a>.

where the attribute HREF is assigned the indicated value. Note that the A element is not empty, and that it is closed by the tag </a>. Note also that end tags never take attributes -- the attributes to an element are always placed in the start tag.

1.6 HTML Document Structure

HTML documents are structured into two parts, the HEAD, and the BODY. Both of these are contained within the HTML element -- this element simply denotes this as an HTML document.

The head contains information about the document that is not generally displayed with the document, such as its TITLE. The BODY contains the body of the text, and is where you place the document material to be displayed. Elements allowed inside the HEAD, such as TITLE, are not allowed inside the BODY, and vice versa.

1.7 Naming Scheme for HTML Documents

When your HTML browser (Netscape Navigator, Internet Explorer, Opera, lynx etc.....) retrieves a file, it must know what type of data it has received in order to know what to do with it. Hypertext (that is, HTTP) servers explicitly tell the browser the type of the data being sent. In other cases, such as when the browser is using FTP to access a remote file, or when the browser is reading a file from your local disk (such as when you are editing pages prior to publishing them to a Web server), the browsers "guesses" the data type from the filename extension -- that is the part after the dot in the filename. For example, HTML files are identified by names such as name.html, where the .html extension indicates an HTML document.

Four letter extensions are common. This is not a problem with UNIX computers or Macintoshes, since these machines place no restriction on the filename. DOS and Windows 3.1 machines are unfortunately restricted to a three letter extension. Generally the extension is truncated to three letters (i.e. .html becomes .htm).

Here are some of the standard extensions, and their meanings:

.html (also .htm)
HTML document, containing text and HTML mark-up instructions.
.txt

A plain text file. The browser presents the file as a block of text and does not process it for mark-up instructions. Browsers generally treat unknown types of data as a text file.

.gif

A GIF format image file.

.xbm

An X-Bitmap (black&white) image file.

.xpm

An X-Pixmap (colour) image file.

.jpeg (also .jpg)

A jpeg-encoded image file.

.mpeg (also .mpg or .mpe)

An mpeg-encoded video file.

.qt

A (Macintosh) QuickTime-format video file

.avi

A (Microsoft) AVI-format video file

.au

An aiff-encoded audio (sound) file.

.Z

A compressed file - compressed using the adaptive Lempel-Ziv coding. This compression/decompression program are commonly found on UNIX computers.

.gz

A compressed file - compressed using the GNU gzip program. This program is common on UNIX computers and is available on PCs and Macintoshes.

1.7.1 MIME Types and File Data Formats

The World Wide Web actually uses MIME types (Multipurpose Internet Mail Extension) to define the type of a particular piece of information being sent from a Web server to a browser. A browser in turn determines, from the MIME type, how the data should be treated. Each browser has a configuration (menu or file) that maps the types of the data to particular functions. A browser can handle many types of data itself (e.g. HTML documents, GIF images) while other types are passed to auxiliary programs, such as image viewers, movie or sound players, plugins, and so on.

HTTP servers send MIME contents-types header messages ahead of every file they deliver to a browser. This header explicitly tells the browser what type of data is being sent. Thus a server must have a way of telling the type of data it is sending. Usually the server has a configuration file that relates filename extensions to the appropriate MIME type. For example, the MIME type for HTML documents is text/html. Thus, if a browser reqests that a server send the file blobs.html, the server first looks up the MIME type corresponding to the .html extension. The server then sends a message to the browser saying that data of content-type text/html is being sent, after which the server sends the actual data.

Other servers, such as FTP servers, do not send this MIME type information. In this case, the browser "guesses" the MIME type, based on the filename extension. Thus each browser must be configured with a list that relates typical extensions to the "most likely" type of data. This is also how a browser determines the type of files accessed locally of the computer.

1.8 HTML Tables

The HTML table model allows authors to arrange data -- text, preformatted text, images, links, forms, form fields, other tables, etc. -- into rows and columns of cells.[2]

Each table may have an associated caption (see the CAPTION element) that provides a short description of the table's purpose. A longer description may also be provided (via the summary attribute) for the benefit of people using speech or Braille-based user agents. Table rows may be grouped into a head, foot, and body sections, (via the THEAD, TFOOT and TBODY elements, respectively). Row groups convey additional structural information and may be rendered by user agents in ways that emphasize this structure. User agents may exploit the head/body/foot division to support scrolling of body sections independently of the head and foot sections. When long tables are printed, the head and foot information may be repeated on each page that contains table data.

Authors may also group columns to provide additional structural information that may be exploited by user agents. Furthermore, authors may declare column properties at the start of a table definition (via the COLGROUP and COL elements) in a way that enables user agents to render the table incrementally rather than having to wait for all the table data to arrive before rendering.

Table cells may either contain "header" information (see the TH element) or "data" (see the TD element). Cells may span multiple rows and columns. The HTML 4 table model allows authors to label each cell so that non-visual user agents may more easily communicate heading information about the cell to the user. Not only do these mechanisms greatly assist users with visual disabilities, they make it possible for multi-modal wireless browsers with limited display capabilities (e.g., Web-enabled pagers and phones) to handle tables.

Tables should not be used purely as a means to layout document content as this may present problems when rendering to non-visual media. Additionally, when used with graphics, these tables may force users to scroll horizontally to view a table designed on a system with a larger display. To minimize these problems, authors should use style sheets to control layout rather than tables.

1.9 Links And Anchors

HTML offers many of the conventional publishing idioms for rich text and structured documents, but what separates it from most other markup languages is its features for hypertext and interactive documents. This section introduces the link (or hyperlink, or Web link), the basic hypertext construct. A link is a connection from one Web resource to another. Although a simple concept, the link has been one of the primary forces driving the success of the Web.

A link has two ends -- called anchors -- and a direction. The link starts at the "source" anchor and points to the "destination" anchor, which may be any Web resource (e.g., an image, a video clip, a sound bite, a program, an HTML document, an element within an HTML document, etc.).

1.9.1 Visiting a linked resource

The default behavior associated with a link is the retrieval of another Web resource. This behavior is commonly and implicitly obtained by selecting the link (e.g., by clicking, through keyboard input, etc.).

The following HTML excerpt contains two links, one whose destination anchor is an HTML document named "chapter1.html" and the other whose destination anchor is a GIF image in the file "forest.gif":

```
<BODY>
...some text...
<P>You'll find a lot more in  <A href="chapter1.html">chapter one</A>.
See also this <A href="../images/forest.gif">map of the enchanted forest.</A>
</BODY>
```

By activating these links (by clicking with the mouse, through keyboard input, voice commands, etc.), users may visit these resources. Note that the href attribute in each source anchor specifies the address of the destination anchor with a URI.
The destination anchor of a link may be an element within an HTML document. The destination anchor must be given an anchor name and any URI addressing this anchor must include the name as its fragment identifier.

1.9.2 Other Link Relationships

By far the most common use of a link is to retrieve another Web resource, as illustrated in the previous examples. However, authors may insert links in their documents that express other relationships between resources than simply "activate this link to visit that related resource". Links that express other types of relationships have one or more link types specified in their source anchor.

The roles of a link defined by A or LINK are specified via the rel and rev attributes.

For instance, links defined by the LINK element may describe the position of a document within a series of documents. In the following excerpt, links within the document entitled "Chapter 2" point to the previous and next chapters:

```
<HEAD>
...other head information...
<TITLE>Chapter 2</TITLE>
<LINK rel="prev" href="chapter1.html">
<LINK rel="next" href="chapter3.html">
</HEAD>
```

The link type of the first link is "prev" and that of the second is "next" (two of several recognized link types). Links specified by LINK are **not** rendered with the document's contents, although user agents may render them in other ways (e.g., as navigation tools).

Even if they are not used for navigation, these links may be interpreted in interesting ways. For example, a user agent that prints a series of HTML documents as a single document may use this link information as the basis of forming a coherent linear document. Further information is given below on using links for the benefit of search engines.

1.10 Forms

An HTML form is a section of a document containing normal content, markup, special elements called controls (checkboxes, radio buttons, menus, etc.), and labels on those controls. Users generally "complete" a form by modifying its controls (entering text, selecting menu items, etc.), before submitting the form to an agent for processing (e.g., to a Web server, to a mail server, etc.)

Here's a simple form that includes labels, radio buttons, and push buttons (reset the form or submit it):

```
 <FORM action="http://somesite.com/prog/adduser" method="post">
</FORM>
```

1.11 Controls

Users interact with forms through named controls.

A control's "control name" is given by its name attribute. The scope of the name attribute for a control within a FORM element is the FORM element.Each control has both an initial value and a current value, both of which are character strings. Please consult the definition of each control for information about initial values and possible constraints on values imposed by the control. In general, a control's "initial value" may be specified with the control element's value attribute. However, the initial value of a TEXTAREA element is given by its contents, and the initial value of an OBJECT element in a form is determined by the object implementation (i.e., it lies outside the scope of this specification). The control's "current value" is first set to the initial value. Thereafter, the control's current value may be modified through user interaction and scripts. A control's initial value does not change. Thus, when a form is reset, each control's current value is reset to its initial value. If a control does not have an initial value, the effect of a form reset on that control is undefined. When a form is submitted for processing, some controls have their name paired with their current value and these pairs are submitted with the form. Those controls for which name/value pairs are submitted are called successful controls.

1.11.1 Control types

HTML defines the following control types:

buttons

Authors may create three types of buttons:

- submit buttons: When activated, a submit button submits a form. A form may contain more than one submit button.
- reset buttons: When activated, a reset button resets all controls to their initial values.
- push buttons: Push buttons have no default behavior. Each push button may have client-side scripts associated with the element's event attributes. When an event occurs (e.g., the user presses the button, releases it, etc.), the associated script is triggered.

  Authors should specify the scripting language of a push button script through a default script declaration (with the META element).

Authors create buttons with the BUTTON element or the INPUT element. Please consult the definitions of these elements for details about specifying different button types.

checkboxes

Checkboxes (and radio buttons) are on/off switches that may be toggled by the user. A switch is "on" when the control element's checked attribute is set. When a form is submitted, only "on" checkbox controls can become successful.

Several checkboxes in a form may share the same control name. Thus, for example, checkboxes allow users to select several values for the same property. The INPUT element is used to create a checkbox control.

radio buttons

Radio buttons are like checkboxes except that when several share the same control name, they are mutually exclusive: when one is switched "on", all others with the same name are switched "off". The INPUT element is used to create a radio button control.

If no radio button in a set sharing the same control name is initially "on", user agent behavior for choosing which control is initially "on" is undefined. At all times, exactly one of the radio buttons in a set is checked. If none of the <INPUT> elements of a set of radio buttons specifies `CHECKED', then the user agent must check the first radio button of the set initially.Since user agent behavior differs, authors should ensure that in each set of radio buttons that one is initially "on".

menus

Menus offer users options from which to choose. The SELECT element creates a menu, in combination with the OPTGROUP and OPTION elements.

text input

Authors may create two types of controls that allow users to input text. The INPUT element creates a single-line input control and the TEXTAREA element creates a multi-line input control. In both cases, the input text becomes the control's current value.

file select

This control type allows the user to select files so that their contents may be submitted with a form. The INPUT element is used to create a file select control.

hidden controls

Authors may create controls that are not rendered but whose values are submitted with a form. Authors generally use this control type to store information between client/server exchanges that would otherwise be lost due to the stateless nature of HTTP. The INPUT element is used to create a hidden control.

object controls

Authors may insert generic objects in forms such that associated values are submitted along with other controls. Authors create object controls with the OBJECT element.

The elements used to create controls generally appear inside a FORM element, but may also appear outside of a FORM element declaration when they are used to build user interfaces. This is discussed in the section on intrinsic events. Note that controls outside a form cannot be successful controls.

1.12 The FORM Element

Attribute definitions

action = uri [CT]

This attribute specifies a form processing agent. User agent behavior for a value other than an HTTP URI is undefined.

method = get|post [CI]

This attribute specifies which HTTP method will be used to submit the form data set. Possible (case-insensitive) values are "get" (the default) and "post". See the section on form submission for usage information.

enctype = content-type [CI]

This attribute specifies the content type used to submit the form to the server (when the value of method is "post"). The default value for this attribute is "application/x-www-form-urlencoded". The value "multipart/form-data" should be used in combination with the INPUT element, type="file".

accept-charset = charset list [CI]

This attribute specifies the list of character encodings for input data that is accepted by the server processing this form. The value is a space- and/or comma-delimited list of charset values. The client must interpret this list as an exclusive-or list, i.e., the server is able to accept any single character encoding per entity received.

The default value for this attribute is the reserved string "UNKNOWN". User agents may interpret this value as the character encoding that was used to transmit the document containing this FORM element.

accept = content-type-list [CI]

This attribute specifies a comma-separated list of content types that a server processing this form will handle correctly. User agents may use this information to filter out non-conforming files when prompting a user to select files to be sent to the server (cf. the INPUT element when type="file").

name = cdata [CI]

This attribute names the element so that it may be referred to from style sheets or scripts. **Note.** This attribute has been included for backwards compatibility. Applications should use the id attribute to identify elements.

1.13 The INPUT Element

Attribute definitions

type = text|password|checkbox|radio|submit|reset|file|hidden|image|button [CI]

This attribute specifies the type of control to create. The default value for this attribute is "text".

name = cdata [CI]

This attribute assigns the control name.

value = cdata [CA]

This attribute specifies the initial value of the control. It is optional except when the type attribute has the value "radio" or "checkbox".

size = cdata [CN]

This attribute tells the user agent the initial width of the control. The width is given in pixels except when type attribute has the value "text" or "password". In that case, its value refers to the (integer) number of characters.

maxlength = number [CN]

When the type attribute has the value "text" or "password", this attribute specifies the maximum number of characters the user may enter. This number may exceed the specified size, in which case the user agent should offer a scrolling mechanism. The default value for this attribute is an unlimited number.

checked [CI]

When the type attribute has the value "radio" or "checkbox", this boolean attribute specifies that the button is on. User agents must ignore this attribute for other control types.

src = uri [CT]

When the type attribute has the value "image", this attribute specifies the location of the image to be used to decorate the graphical submit button.

1.13.1 Control types created with INPUT

The control type defined by the INPUT element depends on the value of the type attribute:

text

Creates a single-line text input control.

password

Like "text", but the input text is rendered in such a way as to hide the characters (e.g., a series of asterisks). This control type is often used for sensitive input such as passwords. Note that the current value is the text entered by the user, not the text rendered by the user agent.

checkbox

Creates a checkbox.

radio

Creates a radio button.

submit

Creates a submit button.

image

Creates a graphical submit button. The value of the src attribute specifies the URI of the image that will decorate the button. For accessibility reasons, authors should provide alternate text for the image via the alt attribute.

When a pointing device is used to click on the image, the form is submitted and the click coordinates passed to the server. The x value is measured in pixels from the left of the image, and the y value in pixels from the top of the image. The submitted data includes name.x=x-value and name.y=y-value where "name" is the value of the name attribute, and x-value and y-value are the x and y coordinate values, respectively.

If the server takes different actions depending on the location clicked, users of non-graphical browsers will be disadvantaged. For this reason, authors should consider alternate approaches:

- Use multiple submit buttons (each with its own image) in place of a single graphical submit button. Authors may use style sheets to control the positioning of these buttons.
- Use a client-side image map together with scripting.

reset

Creates a reset button.

button

Creates a push button. User agents should use the value of the value attribute as the button's label.

hidden

Creates a hidden control.

file

Creates a file select control. User agents may use the value of the value attribute as the initial file name.

CHAPTER 2

ACTIVE SERVER PAGES (ASP)

2.1 Introduction to ASP

ASP stands for Active Server Pages. It is a server side technology which is used to display dynamic content on web pages. For example you could write code that would give your visitors different information, different images or even a totally different page depending on what browser version they are using.
ASP in itself isn't a language it is more a technology developed by Microsoft. ASP can be used with various scripting languages such as Perl or JScript though we will use VBscript as it is the default scripting language.

2.2 What Is ASP?

ASP is an acronym for Microsoft's **A**ctive **S**erver **P**ages. Now, ASP may seem like a bit of an odd name but, believe it or not, some real thought went into coming up with the name.
The reason that Microsoft named the technology Active Server Pages is because it very basically describes what the technology is all about:

1. The **Active** refers to the fact that the HTML is created dynamically by your ASP pages.
2. The **Server** refers to the fact that this process is done on the server-side and not the client-side. An example of a client-side technology would be JavaScript which is a language that actually runs in the browser. Server-side technologies like ASP run on the server and send out simple HTML to the browser.
3. Then there is **Pages**. I don't think this one needs any explaining.

ASP is referred to as a server technology. This means that it is technically not a programming language, however, you will probably hear ASP referred to as a language from time to time. When working in ASP you will actually write ASP in cooperation with either JavaScript or VBScript, which are the actual programming languages that are available to you.

2.3 History of ASP

ASP was one of the first web application development environments that integrated web application execution directly into the web server. This was done in order to achieve high performance compared to calling external executable programs or CGI scripts which was the most popular method for writing web applications at the time it was introduced. Today there are additional platforms for web application development that are more common on other operating systems. Both JavaServer Pages and PHP are more commonly found on webservers running non-Microsoft operating systems, with PHP currently being the more common of the two[citation needed]. Also of note is ColdFusion, a popular Java technology running on several platforms including Microsoft servers as well as other platforms.

InstantASP and ChilisoftASP are technologies that run ASP on platforms other than the Microsoft Windows Operating System. ChilisoftASP was purchased by Sun Microsystems and later renamed "Sun ONE Active Server Pages", then later renamed to "Sun Java System Active Server Pages". It appears that InstantASP is no longer available. There are large open source communities on the internet, such as ASPNuke, which produce ASP scripts, components and applications to be used free under certain license terms.

2.4 What Do I Need?

As you learned in the ASP introduction primer, ASP is a server-side technology which means that the server for your website is the sole controller of whether or not ASP is available to you. ASP is also a product of Microsoft which means it will only run on Microsoft operating systems, however, there is a third party package that will help you get around this problem.

If you are using a hosting service or getting some free personal web space with your ISP you will want to check with them to see if ASP is available to you. You can also write a simple test page to verify whether ASP works on your server if you don't want to take the time to contact your web host.

Here is a simple page for you to enter and test for ASP:

```
<%@ Language="VBScript" %>
<% Option Explicit %>

<!--- This page should display "ASP is working!" --->

<!--- if ASP is available to you. --->


<HTML>
  <HEAD>
    <TITLE>ASP Test Page</TITLE>
  </HEAD>

  <BODY>

    <% Dim TestString %>
    <% TestString = "ASP is Working!" %>

    <H1>

      <% Response.Write TestString %>

    </H1>
  </BODY>
</HTML>
```

Don't worry about how the code works for now. We'll get to all of that soon enough. Be sure to save the page with the **.asp** extension instead of the **.html** extension or you will not get the desired result. The **.asp** extension simply let's the server know that the page is an ASP page and it should be processed accordingly. If you forget and leave the **.html** extension on the end then the server will just send the page on its merry way to the browser and it will never get processed. Also, ASP pages are uploaded to your websites the same way that your HTML pages are, nothing different.

Now, if you are running your own server at home or work then you will have some configuring ahead of you. First of all, you will need to have IIS up and running. IIS is available to you in various forms on any Microsoft operating system from Windows 95 on.

For those of you that are just wanting to learn and test ASP on your home computer you can install and configure Personal Web Server (**PWS**) and run your ASP pages locally. PWS comes with your Windows operating system. This will get you started finding the right installation and set-up instructions for your particular operating system. Unfortunately, I will have to let Microsoft walk you through the installation and set-up of PWS because I simply wouldn't be able to cover all of the different operating systems in these short articles.

If you don't use a Microsoft operating system but still want to learn how to use ASP, there is a third party product out there called Chili!Soft ASP which allows you to run ASP applications on some non-Microsoft operating systems.

2.5 How Does It Work?

Now that you have a platform for running your ASP pages, hopefully, I'll bet you would like to know how the pages are actually processed. Well, it's pretty simple actually.

Each time a request is made from a browser for an ASP page the server takes a look at that ASP page to see what it (the server) needs to do. This process is referred to as interpreting. In other words, the server reads each ASP page as it is requested, interprets the code in the page and then spits out the desired result to the browser. Because of this fact, you will be the only one that can see your code on your ASP pages since only the HTML gets sent to the browser. No worries about someone "stealing" your ASP code.

While you might think that the whole process would result in a big pause while the pages are processed it really doesn't. As long as you keep your ASP pages to a reasonable length you will probably not notice any difference between an ASP page and a simple HTML page.

So, what exactly gets sent to the browser? Nothing but simple HTML. Since the browser is not a very worldly piece of software it generally only speaks a few languages, HTML and JavaScript. Therefore, ASP sends each page to the browser in HTML.Does this mean I can't use JavaScript with ASP? No, not at all. As a matter of fact you can even use JavaScript instead of VBScript to write your ASP pages. We'll discuss that later, though.

Now, don't worry if this doesn't make perfect sense to you right now. As we move on and you get a chance to create some ASP pages of your own the whole process will become much more clear to you.[3]

Have a look at the diagrams below. They are pretty simplistic and self-explanatory.

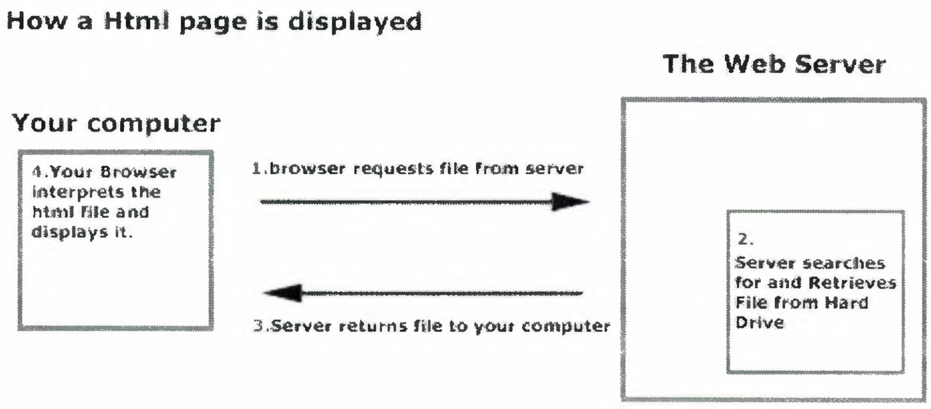

How a Html page is displayed
The Web Server
Your computer
4.Your Browser interprets the html file and displays it.
1.browser requests file from server
3.Server returns file to your computer
2. Server searches for and Retrieves File from Hard Drive

Figure 1.1 How a Htlm page is displayed

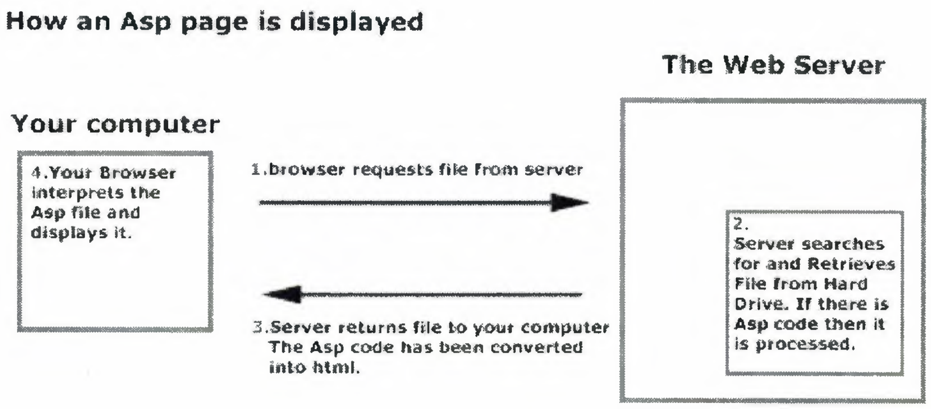

How an Asp page is displayed
The Web Server
Your computer
4.Your Browser interprets the Asp file and displays it.
1.browser requests file from server
3.Server returns file to your computer The Asp code has been converted into html.
2. Server searches for and Retrieves File from Hard Drive. If there is Asp code then it is processed.

Figure 1.2 How a Asp page is displayed

Static pages

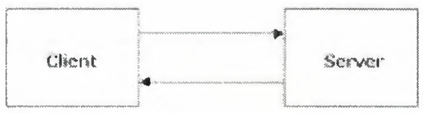


Figure 1.3 How static pages are processed

The client requests a static page like default.html and the server simply returns the page. As you can see the client can easily copy the content and html code and do whatever he wants with it. Furthermore nothing is processed and therefore a static page is returned.

Dynamic pages

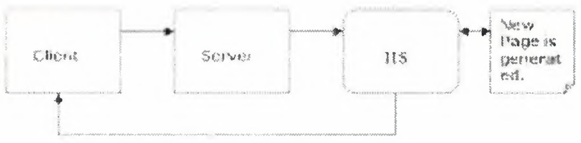


Figure 1.4 how ASP pages are processed

Here the client requests a dynamic .asp page and the server sends the page for further processing to the IIS. IIS stands for Internet Information Services. It is a web server, which processes the asp page. IIS then generates a new page with the result and returns it to the client. As you can see only the result is returned not the original asp page.

2.6 Make Your First ASP Page

Well, actually you already did. The test page code listed above is a fully functional ASP page. So, let's take a closer look at it and see what it is doing.

The first thing you probably noticed were the goofy looking <% and %> tags. Those little guys are ASP's way of telling the server that inside here is some code that needs to be looked at and processed. Also, did you notice that the ASP tags are sprinkled in

and amongst the standard HTML that you are already familiar with? You'll find that most of your ASP pages will be a mixture of ASP and HTML.

You also probably noticed the **<% Option Explicit %>** at the top of the page. Memorize this. As a matter of practice you should always put it at the top of every ASP page that you create. It forces you to declare all of the variables in your application (we'll talk about that in greater detail later). It will save you a great deal of headache when you begin to debug you pages.

Next there is **<%@ Language="VBScript" %>**. This simply tells the server what language you have chosen for your ASP pages. For the purposes of this series we will be using VBScript but if you are more familiar with JavaScript please feel free to use that instead.

2.7 VBScript

To build interactive web pages, scripting language like vbscript mostly used.

VBScript is an affspring of the popular visual basic language that runs on the web server (a server side program) for each user through their browsers. Embedded scripts (program lines) in html can't be interpreded like html tags by the browsers. Instead they are transmitted to web server to be executed. The result of execution is sent back to the user browsers.

2.8 ASP Some Basic VBScript statements

2.8.1 The DIM Statement

So, what the heck is a DIM...and what DIM-wit came up with the name? Well, DIM actually stands for dimension and is a hold over from the original BASIC language.

What DIM does is let the computer know what variables you are going to use. The computer needs to know what variables you will use in order to reserve some space for them.

DIM also has one other very important function. When DIM is used in cooperation with OPTION EXPLICIT, it forces you to declare all variable that are used in your ASP page. While that may not sound like anything important, forcing you to declare all your variables keeps you from making typos, which makes debugging your code so much easier.

Here's how DIM works:

```
DIM strName, intNumber, decNumber
```

Pretty simple, eh? Now we have three variables declared: strName, intNumber, and decNumber. The variables are now ready for use in you ASP page.

Now here are some details you need to be aware of with regard to how VBScipt handles its variables. Unlike most other programming languages which require you to declare the variable name and type, VBScript does not require the variable type.

A variable type, logically enough, refers to the type of data the variable will hold. In our example above, we declared a variable called strName. strName is our variable we will use to hold string data, in this case, a person's name.

If you are new to programming, you may want to take a look at the table below that outlines some of the most basic data types. Most programming languages have many more data types than are listed below. For the purposes of learning basic programming, though, we will mainly concentrate on the data types listed.

String	Holds a string of alphanumeric characters (numbers, letters, and symbols). An example of this would be someone's name, address, or telephone number.
Integer	Holds whole numbers. (e.g. 1, 2, 3, 4, etc.)
Decimal	Holds numbers with a decimal point (e.g. 16.42, 6.92456, etc.)

So, why even talk about data types if VBScript doesn't require me to declare them?

Well, knowing the different data types will allow you to better understand how you can manipulate the data. In VBScript there are many different built-in functions that will do a lot of your programming work for you.

In order to use those functions you will need to know what kind of data is in the variable that you are performing the function on. For example, doing a square root function on our variable strName won't get you a useful result. It's kind of hard to take the square root of someone's name.

I'm sure that you noticed that each of the variables listed above has a prefix attached to it. That is my own little system for keeping track of what the data type is for each variable. I have found it to be very handy when debugging or adding to my code. The prefix quickly tells me that strName holds a string, intNumber holds and integer and decNumber holds a decimal. There is no set rule to this technique and you can use whatever naming method you like, however, making it something intuitive is always a good idea since you may not always be the person changing or updating the code.

2.8.2 The IF .. THEN Statement

And now on to bigger and better things, the IF .. THEN statement. This will probably be one of the statements that you use the most in VBScript.

The IF .. THEN statement is what is known as a conditional statement. In other words, when some condition is met (or not met for that matter) then something happens. Boiled down to its simplest form, that is programming in a nutshell. It is simply a series of events and responses to those events.

Let's take a look at how the IF .. THEN works:

```
IF strName = "N/A" THEN

   strName = "Not Available"

END IF
```

In this case we are checking our string variable strName to see if it is set to "N/A". If it is set to "N/A" then we change it to say "Not Available" instead. You'll also notice that there is an END IF statement. That END IF tells the computer that you are done with the IF .. THEN statement and it can continue executing any other code.

The computer knows to only execute the code between the THEN and the END IF if the condition you set is met.

So, what kind of comparisons can I do?

There are quite a few comparisons that you can make. Here they are along with the data types that they apply to:

= **Equal to**. You can use this to compare any data type.

< **Less than**. You can use this to compare any numeric data type. (You can also use it for string variable types as well but you will probably find that you never need to except when comparing dates.)

<= **Less than or equal to**. You can use this to compare any numeric data type. (You can also use it for string variable types as well but you will probably find that you never need to except when comparing dates.)

> **Greater than**. You can use this to compare any numeric data type. (You can also use it for string variable types as well but you will probably find that you never need to except when comparing dates.)

>= **Greater than or equal to**. You can use this to compare any numeric data type. (You can also use it for string variable types as well but you will probably find that you never need to except when comparing dates.)

<> **Not equal to**. You can use this to compare any data type.

When using the comparison operators above always remember to put string comparisons in quotes like in the example above. Numeric comparisons, like you will see below, do not use quotes.

So, what if I wanted to make more than one comparison? Do I have to list one IF .. THEN after another?

That's where ELSEIF comes in. ELSEIF allows you to string several comparisons together.

With the short section of code above you can easily see that we are assigning a grade based on someone's score. You can also see that ELSEIF works exactly like IF in that it makes a comparison the exact same way with comparison operators.

There is one very key difference to keep in mind, though. This type of IF statement is executed in sequence. In other words, if the first condition isn't met it moves on to the second condition. If the second condition isn't met then it will move on to the third condition. If the third condition is met it will execute the appropriate code which, in our case, is strGrade = "C". Once the code has been executed the computer considers itself done and doesn't even bother looking at the fourth condition. In our case that's exactly what we want it to do since we only need to assign one grade based on the score.

But what if I wanted to make 4 comparisons and have each one be considered regardless of the result of the comparison?

In that case you will have to create 4 separate IF .. THEN statements in order to be assured that each condition will be checked.

Now there is one more piece of the puzzle. If you noticed above there was no "F" grade. That's where ELSE comes in. ELSE is a sort of catch-all. It simply means that if none of the conditions before it are met then it will execute its code.

2.8.3 The CASE Statement

Now that you have an idea of how conditional statements work we'll throw out another one to you, the CASE statement. The CASE statement is very much akin to the IF .. THEN.

For our example of the CASE statement we're going to give you a practical example. Whenever possible we'll try to give at least one practical example with each segment of our primer series. Here's your practical example for this segment:

Use the code below to create an ASP page called nav.asp:

```
<% OPTION EXPLICIT %>

<% ' This example creates a form with a drop-down %>
<% ' menu. The user selects a page from the list and %>

<% ' clicks GO! to move from page to page. Each time %>

<% ' the user clicks GO! they are sent to a page named %>

<% ' nav.asp where their selection is evaluated by a %>

<% ' CASE statement. The CASE statement then %>

<% ' redirects the user to the appropriate page. %>

<% SELECT CASE Request.Form("Category") %>

<%   CASE "Home" %>

<%     Response.Redirect "index.html" %>

<%   CASE "About Me" %>

<%     Response.Redirect "about_me.html" %>

<%   CASE "About HTML" %>

<%     Response.Redirect "about_html.html" %>

<%   CASE "About ASP" %>

<%     Response.Redirect "about_asp.html" %>

<% END SELECT %>
```

Create a new HTML page with the code below or just cut and paste the form portion of the page into an existing HTML or ASP page:

```
 <HTML>

   <HEAD>
 <TITLE>My Navigation Page</TITLE>

</HEAD>

<BODY>

 <FORM METHOD="post" ACTION="my_page.asp">

 <P>

  <FONT COLOR="#000080">

   Please select a page:</FONT>

  <SELECT NAME="Category" >
   <OPTION SELECTED>Home</OPTION>
   <OPTION>About Me</OPTION>
   <OPTION>About HTML</OPTION>
   <OPTION>About ASP</OPTION>
  </SELECT>

  <INPUT TYPE="submit" VALUE="GO!"

   NAME="Go">

  </P>
 </FORM>

</BODY>

</HTML>
```

Now here's how it all works. Once a user selects a page in the drop-down menu and clicks the GO! button they are redirected to a page called nav.asp. That page evaluates the user's request in a CASE statement and then forwards them on to the page that they requested. This all happens so quickly that the user probably will never realize they were sent to an intermediate page before arriving at the page that they requested.

Let's break down that CASE statement now. The first thing you see is SELECT CASE followed by what is to be evaluated. In this case, we are evaluating the information that is sent from the form on your HTML page. (Don't worry about the Request.Form or the Response.Redirect for now. We will get to those concepts later on in this series.)

Next we list out each possible match using CASE. In our example above we have 4 items in the drop-down menu. Therefore, we need to evaluate for those 4 items in our CASE statement. Beside each CASE is a possible match: "Home", "About Me", etc. Then below each CASE we have the instructions for what we want the computer to do. In this case we are telling our server to reroute the person to the page that they selected.

If the example above doesn't make sense to you consider this example using strGrade instead:

```
<% SELECT CASE strGrade %>

<%   CASE "A" %>

<P>Congratulations! You got an A!!!</P>

<%   CASE "B" %>

<P>Good Job! You got an B!</P>

<%   CASE "C" %>

<P>Not Bad. You got a C.</P>

<%   CASE "D" %>

<P>Well, Duh. You could do better than a D.</P>

<% END SELECT %>
```

The CASE statement above mixes VBScript and HTML on an ASP page. Depending on what grade the student received, the appropriate message is displayed on the page using standard HTML.

If the IF .. THEN and the CASE statement both get you the same result then why use CASE at all?

Well, CASE has a couple of advantages. It is a much more concise code which means less typing. It also is usually more intuitive than a very long series of IF .. THEN statements.

Feel free to plug in and adapt the code above to your pages. Depending on the circumstance, the example above can be a very handy solution for navigation problems.

2.8.4 The For .. Next Loop

For those of you that tend to be repetitive, you'll love this installment of the ASP Primer. Using loops we can thumb through data, automatically generate a list of numbers or dates, traverse an array or any number of other things.

Let's take a look at the **FOR .. NEXT** loop first. This particular loop is about as straight-forward as it gets. Its job is to start at a certain number and count until it reaches its designated stopping point.

In order to demonstrate how this works, let's take a look at our practical example for this installment of the ASP Primer. Below we will use a **FOR .. NEXT** loop to populate a drop-down list box on a form. We will be putting in the year, beginning with this year, plus 7 more years for a total of 8. This is a handy little piece of code for populating a credit card drop-down so that you will never have to go back and change the years listed as time goes by. Here it is:

```
<% OPTION EXPLICIT %>

<HTML>

<HEAD>

 <TITLE>Auto-generate Year</TITLE>

</HEAD>
```

```
<BODY>

<FORM METHOD="POST" ACTION="thispage.asp">


<P>


 <SELECT SIZE="1" NAME="year">
 <OPTION>Please select a year ...</OPTION>

 <% DIM intYearCounter, intCurrentYear %>

 <% intCurrentYear = Year(Date) %>

 <% FOR intYearCounter = intCurrentYear TO

      (intCurrentYear + 7) %>

  <% Response.Write "<OPTION>" &

      CStr(intYearCounter) & "</OPTION>" %>

 <% NEXT %>

</SELECT>


</P>


</FORM>

</BODY>

</HTML>
```

You'll notice we are using some of the elements from the last tutorial like the **DIM** statement. Next you will notice we set our newly created variable, intCurrentYear, to **Year(Date)**. This sets our variable to the current year. This is one of those nifty little functions that is built into VBScript and ASP. We'll cover this function and more later in this series.

Then we have our **FOR** statement. This is where we define where we will start our loop and how long it will continue looping. First we give it the variable we created above, intYearCounter, so that it can use that variable for storage as it counts. Next we define where we want to begin counting. In our case we will begin counting with the current year which is what is being stored in intCurrentYear. Now that the **FOR** statement knows where we want to begin we need to tell it when to stop. In our case, we want it to stop after it has run though the loop 8 times. So, we finish the **FOR** statement by using **TO (intCurrentYear + 7).**

If you are wondering why we are doing **(intCurrentYear + 7)** instead of **(intCurrentYear + 8)** imagine this: suppose our starting year is 2001. We want to display a total of 8 years starting with 2001. Therefore, our last year listed should be 2008. The difference between 2008 and 2001 is 7 which is why we add 7 to intCurrentYear.

The next line that you come to is **Response.Write "<OPTION>" & CStr(intYearCounter) & "</OPTION>"**. What this actually does is take the current number in intYearCounter and combine it with HTML to send to the browser when the page is rendered. We'll get into the particulars of CStr and Response.Write later in this series.

Then there is the NEXT statement. This simply tells the server that this is the end of our FOR .. NEXT loop. The FOR .. NEXT loop only executes the code between the FOR and the NEXT. So, when our loop runs it will only repeatedly execute the line starting with Response.Write since it is the only line inside the loop. Even though we only have one line inside our loop, you can have as many lines as you like.

2.8.5 The Do .. Loop

The **DO .. LOOP** is a combination of the IF .. THEN statement and the FOR .. NEXT loop. It's a conditional loop. In other words, it loops repeatedly (like a FOR .. NEXT loop) until some specific criteria is met (like an IF .. THEN statement).

Here is a basic example of the DO .. LOOP:

```
<% DIM intDice, intRollCounter %>

<% intRollCounter = 0 %>

<% Randomize %>

<% DO UNTIL intDice = 6 %>

  <% intRollCounter = intRollCounter + 1 %>

  <% intDice = Int(Rnd * 6) + 1 %>

<% LOOP %>

<% Response.Write "It took " & CStr(intDice)

      & " tries to roll a 6!" %>
```

This example is a simple dice game. We will keep automatically rolling a single die until we get a six and then output how many tries it took. Let's walk through the code.

The first new thing that you will encounter is the Randomize command. This command simply sets a random number seed so that you can generate random numbers. We'll cover both Randomize and Rnd in more detail later in this series.

After Randomize comes the **DO** statement. Like **FOR**, this simply tells the server that we are beginning our loop here. Then comes **UNTIL**. **UNTIL** is the conditional part and works just as you would expect it to. Each time through the loop it checks to see if our criteria is met, i.e. **intDice = 6**. When it does equal 6 the loop stops executing.

Next you will see intRollCounter be incremented by one each time we go through the loop. We are doing our incrementing by hand this time instead of using a **FOR .. NEXT** loop because we don't know how many times the loop will execute before intDice equals 6.

Then we generate a random number between 1 and 6. We'll talk about how this works when we cover Randomize and Rnd later in this series.

The last part of our loop is designated by, you guessed it, **LOOP**. This tells the server that we have reached the bottom of the loop and to try it again **UNTIL** we get it right.

Lastly, after our loop is done looping we simply write a line that says how many times we had to loop through to get a 6.

There's also another way to approach this. We could use **WHILE** instead. **WHILE** is the polar opposite of **UNTIL**. **WHILE** keeps going until a condition is **not** met. Let's change the **DO** line to use **WHILE** instead:

```
<% DO WHILE intDice <> 6 %>
```

We are accomplishing the same task as before but we are just wording it a bit different. Now we are saying keep looping as long as intDice **does not equal** 6.

Obviously, each situation will logically lend itself to either **WHILE** or **UNTIL**. You'll have to determine for yourself which makes your code easier to read and more intuitive. In our case **UNTIL** makes the most sense.

Now, there's one other little twist you will need to know. **WHILE** and **UNTIL** can be used either at the top or the bottom of the loop. That means we could move the **UNTIL** in our example down to the bottom of the loop like this:

```
<% LOOP UNTIL intDice = 6 %>
```

Now instead of doing our test at the top of the loop we are doing it at the bottom. I know it doesn't sound like a big deal but it can be very important. If you place **UNTIL** or **WHILE** at the bottom you are ensured that your loop will execute at least one time.

2.8.6 All About Arrays

And now for a new way to store information .. arrays. Arrays are a very handy storage method. Think of arrays as variables on steroids.

Imagine you wanted to store 20 different people's last names in variables. It would be quite a pain to create and keep track of strLastName1, strLastName2, etc. Not very efficient, right? That's where arrays come in. With an array you can create one variable that will have 20 slots, one for each person's name. Here is how you would declare that array:

```
<% DIM strLastNames(20) %>
```

Now you have a variable ready to receive your twenty names. Think of it as a table with 1 column and 20 rows. To refer to each slot in your array simply refer to its index number like this:

```
<% strLastNames(1) = "Smith" %>

<% strLastNames(2) = "Hatfield" %>
```

For an example of how arrays can be used let's go back to our **FOR .. NEXT** example. Change the **DIM** statement to this:

```
<% DIM intYearArray(8), intYearCounter,

    intCurrentYear %>
```

Now you have an array to store the years that you will generate in the **FOR .. NEXT** loop. Add this line **inside** the **FOR .. NEXT** loop:

```
<% intYearArray (intYearCounter - intCurrentYear + 1)

    = intYearCounter %>
```

While that looks incredibly complicated, it's not. Since our loop doesn't count from 1 to 8 we have to reduce our year counter down to single index numbers 1 through 8. That's what our little formula inside the parentheses does.

Now, if you want to get a bit more complicated you can use a two dimensional array. Think of this as a table with x number of rows **and** y number of columns. Here is how you would declare a two dimensional array:

```
<% DIM strMyArray(20,10) %>
```

Think of strMyArray as a "table" that is 20 rows by 10 columns. With a two dimensional array it is a bit more complicated to keep track of your index numbers for each item. However, if you imagine your data in a table and you remember that strMyArray(1,1) is the top left corner then it shouldn't be too difficult. If you **really** want to get complicated you could go for the three dimensional effect like this:

```
<% DIM strMyArray(5,5,5) %>
```

Now you more or less have a cube that is 5 blocks wide by 5 blocks tall by 5 blocks deep.

So, what about getting stuff back out of an array? That's pretty easy actually. All you need to do is refer to the index number of the item that you want to retrieve. If we wanted to get an item out of our intYearArray we would just refer to an index number like this:

```
intTheLastYear = intYearArray(8)
```

In a two dimensional array like this:

```
strData = strMyArray(7,2)
```

You get the idea. But if you wanted to get the data out of each element of your array you could save yourself some time by using a specialized **FOR .. NEXT** loop like this:

```
<% DIM intEachYear %>

<% FOR EACH intEachYear IN intYearArray %>

  <% Response.Write intEachYear %>

<% NEXT %>
```

You'll notice we defined our **FOR** very differently. Here we are telling the server to loop through **EACH** element in intYearArray (from the example above). Each time through the loop we want the current element to be temporarily placed in the variable intEachYear. Then within the loop we will simply print out each element as it loops through. What we will end up with is our series of years all printed on the same line.

2.9 ASP Functional VBScript

2.9.1 Do the Math

Even though you probably won't be doing much serious math in most of your ASP applications, it's still a good idea to review some of the more common math functions. Being that as it may, here are the more common VBScript mathematical functions:

Mathematics:

ABS( )	This stands for ABSolute value. An absolute value, if you remember from math class, is simply taking any number and eliminating the sign so that any number returned here is essentially a positive number.
INT( )	This function will probably get more use than any of the other mathematical functions we list here with maybe the exception of the randomizing functions below. What INT does is simply turn any decimal number into an integer (whole) number. It does this by whacking off everything after the decimal point. That means there is no rounding going on here, it simply throws away the fractional part of any number that you give it. For example, if you give **INT( )** the number 7.694, it will return then number 7.

2.9.2 Messing with Strings

Let's move on to some string functions. You will probably find yourself using string functions infinitely more than the mathematical functions above. The reason is that most of the data you interact with on the web is in the form of strings, i.e. people's names, addresses, web URL's, e-mail addresses, etc.

Here are some of the most commonly used string functions. I have tried to give an easy to understand example of each function. We'll discuss the practical applications of some of the functions later. Here they are:

Changing Strings:

LCASE( )	This function does exactly what you might think. It takes any string it gets and makes it all lower case. Example: Doing **LCASE**("Hello World!") will give you "hello world!" back.
UCASE( )	This function is, of course, the polar opposite of **LCASE( )**. **UCASE( )** changes any given string to all upper case. Example: Taking the example above, doing **UCASE**("Hello World!") will give you back "HELLO WORLD!".
LTRIM( )	This function gets rid of extra spaces on the left side of any string by just chopping them off. Example: If you give **LTRIM( )** the string " There are 5 extra spaces to the left.", it will return a string like this "There are 5 extra spaces to the left."

RTRIM ( )

This function does the opposite of **LTRIM( )**. It removes any extra spaces from the right side of any string.

Example: The string "There are 5 spaces to the right of this string.   " will be returned as "There are 5 spaces to the right of this string.".

TRIM( )

TRIM( ) then is a combination of the two above. It will chop off any extra spaces before and after a given string.

Example: The string "   There are 5 spaces before and after this string.   " will return a result of "There are 5 spaces before and after this string.".

REPLACE( )

Learn this one well. **REPLACE( )** allows you to replace any given set of characters in a string with another set of characters.

Example: This function is a bit more complicated than the rest of the string functions above. It requires more information before it can do its thing. So, with that said, here is a some code to illustrate exactly how the function works:

```
strMySentence = "I like ice cream!"
strNewSentence = REPLACE(strMySentence,"like","love")
```

The result of the code above would be that strNewSentence would now equal to "I love ice cream!". **REPLACE( )** first wants to know what string it is to perform the replace on, then what word or set of characters it is looking for, and lastly what word or set of characters it is swapping in when it finds a match. If there is more than one match within a string it will replace all of the matches it finds.

<u>All About the String</u>:

LEN( )

This is one of the few string functions that doesn't actually return a string. Instead it returns the length of any given string as an integer. In other words, it counts the number of characters, including spaces, and gives you the total.

Example: If you do **LEN**("Test String") you will get the number 11 as the result.

LEFT( )

This function allows you to get a selected section of a string. It is done by giving the function the string you want a piece of and the length of that piece. It then counts from the left to a specified length and returns that piece of the string.

Example: **LEFT**("I love ASP!",6) would return the string "I love", which is 6 characters from the left including the space.

RIGHT( )

This function does the exact same process as **LEFT( )** except from the right.

Example: **RIGHT**("I love ASP!",4) would return the string "ASP!", which is 4 characters from the right.

MID( )

As you might expect, **MID( )** allows you to take a string out of the middle of another string. For this particular function you will need a starting point <u>and</u> the number of characters that you want.

Example: **MID**("I love ASP!",3,4) will give you "love" as the result. It starts at the third character "l" and snags it plus the next 3 characters for a total of 4.

INSTR( )

This function allows you to search for a string within a string. It works much like **REPLACE( )** does in that you give it a string to search and a string to search for. The result it returns is the character number of the beginning of the match, if it finds one. If it doesn't find a match it will return a 0.

Example: **INSTR**("I love ASP!","love") will return the integer 3. If you try **INSTR**("I love ASP!","like") it will return the integer 0.

Here's an extra little tip about **INSTR( )**, **INSTR**("I love ASP!","asp",1) will ignore case-sensitivity. In other words, the number one at the end of the function tells it to find a match ignoring case. The default is to find an exact match. The result of the example above would be 8. The result of this example,
INSTR("I love ASP!","asp") would be 0, no match found.

2.10 ASP Sending a Response

If you skipped ahead to here because you already know VBScript or want to use JavaScript with ASP, then welcome! It's now time to get down to some real ASP stuff. In this tutorial you will learn some of the most used items of the Response object, **Respone.Write**, **Response.Redirect**.

2.10.1 Response.Write: Talking to the User

You will probably find yourself using **Response.Write** more than anything else in ASP. It is used to send output to the browser for the user. You will also find that you will use **Response.Write** frequently when debugging your pages and applications.

So, how does it work?

It's pretty simple really. Whatever you put in a **Response.Write** statement gets sent to the web browser. It "writes" to the browser. Here is how it works:

<% **Response.Write** "Hello!" %>

The result would be the word "Hello!" displayed in the browser. It would appear in the default style for your page.

Now, let's get a bit more creative. Let's make "Hello!" bold and blue. Add this to **Response.Write**:

<% **Response.Write**

"<b><font color="#0000FF">Hello!</font></b>" %>

Now you have mixed standard HTML into what you are outputting to the browser. Since the browser interprets the HTML that it receives, the result is "**Hello!**".

So, why would you ever want to use HTML with a **Response.Write**? Isn't that just an extra unnecessary step? Imagine this, let's say you have a password protected page. When a person logs into your page you want to display one of two messages; "Congratulations!" or "Login Incorrect". You also decide that you want a successful login to be displayed in blue while an unsuccessful login would be displayed in red.

By using **Response.Write** in and If .. Then statement you can easily make any changes to text that you would like displayed. Here is a short excerpt of some code showing how this might work:

```
<% Option Explicit %>

<% Dim strPassword %>

...

<% If strPassword = "valid" then %>

   <% Response.Write

       "<font color="#0000FF">Congratulations!</font>" %>

<% Else %>

   <% Response.Write

       "<font color="#FF0000">Login Incorrect</font>" %>

<% End If %>
```

You can also use **Response.Write** to output variable values and other data. Let's say you wanted to display the value of a variable. Here's how it works:

```
<% Option Explicit %>

<% Dim strWinner %>

<% strWinner = "Johnathan Smith" %>

<% Response.Write "Lottery winner: " & strWinner %>
```

You can use the ampersand to connect a phrase with a variable that you want to display. The result of the code above would be:

```
Lottery winner: Johnathan Smith
```

You can also output the results of some of those built-in functions that we covered in the last section. Here's an example of how to display today's date:

<% **Response.Write** "Today's date: " & Date() %>

Is that all there is to it?

Pretty much. Response.Write is about as straight-forward as it gets.

There is, however, one other very important use for **Response.Write** that I'd like to cover - debugging. Unless you are both a master typist and master coder it is very unlikely that you will ever write an ASP application that runs perfectly right out of the gate. That's where debugging and **Response.Write** come into play.

Once you have cleaned up all of the typos that you may have made when writing your code, there is still no guarantee that the code will run as planned. It may give you the wrong results in a mathematical calculation or spit out other incorrect information. When this happens, most ASP developers will drop several different **Response.Write** statements in their code to narrow down the problem. Here are some different situations where **Response.Write** might be effective:

Wrong results - Try putting a **Response.Write** after you change the value of any variable that is connected with the wrong result. This way your application will basically give you a play-by-play of what it is doing. Hopefully, by walking through the calculations step-by-step you can pinpoint where things go awry.

Skipped code - Sometimes an If .. Then or other conditional statement may not react like you expect. By placing a **Response.Write** in your conditional statements, you can determine whether or not the code inside the conditional is executed. This will allow you to trace the flow of your code and determine whether or not your conditional statements are resolving as you had expected.

2.10.2 Response.Redirect: Moving the User Around

Response.Redirect is kind of like a call forwarding button. It sends the user on to any location you designate.

The syntax is virtually the same as **Response.Write**. Instead of entering a word or phrase that you want to be sent to the browser, you enter the URL of the destination that you want the user to be redirected to. For example:

```
<% Response.Redirect

    "http://www.HTMLGoodies.com" %>
```

The code above will send the user on to the HTMLGoodies web site. You can also assign **Response.Redirect** a destination from a variable like this:

```
<% Option Explicit %>

<% Dim strDestination %>

<% strDestination = "http://www.HTMLGoodies.com" %>

<% Response.Redirect strDestination %>
```

So, why would I ever use this?

The most common usage for **Response.Redirect** is in processing user input. Let's say, for example, you have built an ASP application that allows you and your friends to exchange DVD's and keep track of them. One of your pages is a form that asks the user whether they want to add a DVD to the list, delete a DVD from the list, borrow a DVD or return a DVD. Once they have made a selection you can then send them off to the appropriate page by using **Response.Redirect** in cooperation with a Case statement.

There are, of course, other valuable uses for **Response.Redirect**, but the type of situation listed above is the most common.

2.11 ASP Forming Up

Now it's about to get real fun. From here on you'll be learning about some of the best parts of ASP. We'll be learning about forms and emails this week then it's on to connecting to and manipulating data in a database which is the real power of ASP. But first, forms ...

2.11.1 Setting Up Your Forms

This section is intended as a refresher course to make sure we are all on the same page when we get into processing the forms in the next section.

If you are a forms master then, by all means, go ahead and skip on to the Request.Form: Getting the Info section. If you have never worked with forms before in your life then I would suggest reviewing the tutorial So, You Want a Form, Huh? before going on to the next section.

For everyone else, here we go ...

There are several different basic form elements. We are just going to run through them one by one. I'll give you the HTML to create them and then a brief explanation of the code. That should be enough to jog your memory. Keep in mind, though, this is just a basics refresher and there are more attributes to the form elements than we will cover here.

The Text Box ...

Here's the code:

```
<INPUT TYPE="text" NAME="T1" SIZE="20">
```

All of your form elements start with the word **INPUT**. This let's the browser know we are wanting a **Text Box**.

Second is the **TYPE** attribute. This defines the type of **Text Box** form element we are using which, in this case is "text" for plain text.

Next is the **NAME** attribute. This attribute gives each form element its own unique name. It is very important that each element have a unique name otherwise you won't be able to get the user's input back from the form. You will find this attribute a part of every form element.

Lastly, we get to the **SIZE** attribute. This simply defines the length of the **Text Box**.

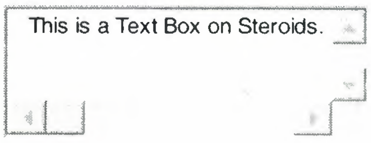
The Password Text Box ...

Here's the code:

<INPUT TYPE="password" NAME="T1" SIZE="20">

You'll notice it's exactly the same as the **Text Box** above except **TYPE** is set to "password" instead of "text". Using "password" tells the browser to mask anything typed into the **Text Box**.

The Text Area ...



Here's the code:

<TEXTAREA ROWS="3" NAME="S1" COLS="20">

This is a Text Box on Steroids.</TEXTAREA>

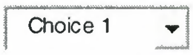
Here is your beefed up **Text Box** also known as **Text Area**. This is ideal for occasions when you need a user to enter a lot of text. It has basically the same attributes as the Text Box but with some notable additions/changes.

First is the **ROWS** attribute. I'm sure you had figured out that this defines the number of lines displayed. You can type as much as you like, however, you will only see 3 lines of text at a time.

Second, you'll notice the **COLS** attribute. This is exactly the same as **SIZE** in the **Text Box** example above. Don't ask me why they chose to use two different names for essentially the same attribute.

Lastly, you'll notice this element is defined as **TEXTAREA** with a begin and end tag. Between the begin and end tag is where you can put any default value that you want to appear in the **Text Area**.

Drop Down Lists ...



Here is the code:

```
<select name="D1">
<option>Choice 1</option>
<option>Choice 2</option>
</select>
```

This is also known as the Drop Down Menu. This element is defined by **SELECT**.

You'll also notice there is an end **SELECT** tag as well. In between these two tags is where you will define the different options that are available within the **Drop Down List**. To define an option you use the **OPTION** tag. You can list as many options as you like, just be sure each option is defined with both and begin and end **OPTION** tag.

Lastly, don't forget to end your **SELECT** element with an end select tag, </SELECT>.

The Radio Button ...

Here is the code:

```
<INPUT TYPE="radio" VALUE="V1"

CHECKED NAME="R1">
```

You'll first notice that we are back to the **INPUT** element. In this case, our **TYPE** is "radio" to define the **Radio Button**.

The new attribute here is the **VALUE**. This defines the value of your radio button. Each **Radio Button** should have its own unique value (within the group name, that is) in order to process your form later.

Next you will see **CHECKED**. This tells the browser that this element is the default element. Use this on only one element in a group to define it as the default option. By default, one button must be selected. If you don't designate a default button, the browser will designate one for you.

And now we will revisit the **NAME** attribute. It serves a little bit different role with the radio buttons. In this case, the name **should not** necessarily be a unique value. For the purposes of a set of radio buttons, the **NAME** defines the group of buttons and the **VALUE** defines the value attached to each individual button. So, if you wanted the user to make a single choice from 3 different options you would give each **Radio Button** a unique **VALUE** but they would all have the same **NAME**.

The Check Box ...

Here's the code: `<INPUT TYPE="checkbox" NAME="C3"`

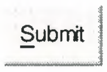
```
VALUE="ON" CHECKED>
```

And yet another **INPUT**. This time the **TYPE** is "checkbox" which defines it as a **Check Box**. Unlike radio buttons, check boxes are not intended to be grouped. The idea behind the **Check Box** is that you should be able to select as many as you like. Therefore, the **NAME** should be a unique one.

Like the **Radio Button**, the **Check Box** does have a **VALUE** attribute. This can simply store a value that tells you it was checked (we used "ON" in the example above) or it could actually store a special value like an ID number.

Lastly, there is the **CHECKED** attribute. It marks any the **Check Box** as selected by default.

The Submit Button ...



Here's the code:

<INPUT TYPE="submit" VALUE="Submit" NAME="B1">

This time our **TYPE** is "submit". This defines the element as a submit button. This button will inherently trigger the form to process when pushed.

It also has a **VALUE** which serves a bit different purpose than on most other form elements. On a button, the value indicates what will appear on the button.

2.11.2 Request.Form: Getting the Info

Now that you have had your refresher course on form elements, we'll jump into the ASP way of processing those forms.

While we discussed the different form elements in the last section, we didn't discuss how to define the form itself. To define a form on your page you will need to use the **<FORM>** and **</FORM>** tags. Within the **<FORM>** tag you will want to be sure to add these items:

<FORM METHOD="post" ACTION="myprocess.asp">

You will need the **METHOD** attribute set to "post". This sends the user's input back to the server. You will also need to define the **ACTION**. The **ACTION** tells the server what to do next once the user has hit the submit button. In the example above we will be calling the page "myprocess.asp".

When setting the action you first need to determine how you want to handle processing of the form. You can approach it in two ways. You can either call the same page that has the form on it or call a new page. Here are some advantages and disadvantages to each approach:

Same Page - Calling the same page will allow you to have all your code in one centralized location. This method is often easier to read and debug because all of your references are in the same location. It's biggest disadvantage is that it makes for a much larger page and can cause a delay in response.

New Page - This method componentizes your processes. By separating your form page from your processing page you will generally have a faster response. It also has the added benefit of being able to use the same processing page for more than one form. It's only drawback is that you might have to jump around between multiple pages to track down bugs.

So, how do you get the user's information from the form?

That's where **Request.Form** comes in. It works very similarly to Request.QueryString. To illustrate how it works I'll give you this week's practical example:

```
<% Option Explicit %>

...

<HTML>

<HEAD>
<TITLE>My Password Login Page</TITLE>
</HEAD>
```

```
<BODY>

<FORM METHOD="post" ACTION="login.asp">
<P>
Please enter your password:
<INPUT TYPE="password" NAME="Pswd" SIZE="10">
&nbsp;&nbsp;&nbsp;&nbsp;
<INPUT TYPE="submit" VALUE="Go!" NAME="SubmitButton">

</P>
</FORM>

</BODY>

</HTML>
```

Now, what you have here is your basic everyday form with one text box named "Pswd". You'll notice that we set the **ACTION** in the **FORM** tag to call the page "login.asp", which will be what we call this page. Since this will be a very small amount of code, we are going to combine our ASP and form on the same page.

If we were doing more complex or extensive coding, I would probably recommend using two pages, one for the form and one for the code. The rest of the elements in the example above are straight-forward HTML and should be familiar to you.

Now, here's our ASP code that will process the user's password:

Place this code between the <% Option Explicit %> and <HTML> above.

```
<% If Request.Form("pswd") = "password" then

       Session("access") = "yes"

       Response.Redirect "securedpage.asp"

    End If %>
```

In the code above we are checking to see if the password the user entered is correct and then sending them on to our secured section if it is. To achieve this we are using some ASP and VBScript that you have already learned in cooperation with some new stuff.

I threw you a bit of a curve on this one. You'll notice that each line doesn't have a <% and %> on it. It isn't necessary to use <% and %> on every line, so instead we use <% to begin our block of code and %> to signify the end of our code block. I waited until now to tell you about this so that you would have a chance to get used to using the ASP special delimiters (<% and %>) first.

Next, you will see **Request.Form**. This is what you will use to get the user's input from a form. You will notice that we are requesting the information in "pswd", which is the name of our password text box in our form. We are then comparing what the user typed in to our secret access password that we have created, "password". If the user entered the correct password, we will set two other things in motion. (Unlike JavaScript, you don't have to worry about people being able to view your ASP source and getting your secret password. ASP code can never be seen by the end user.)

First, we will create and set a Session variable called "access" equal to "yes". This is what we will use on our secured page(s) to see if a user is logged in or not. We'll get to that in a minute.

Second, we will send the user on to a new page called "securedpage.asp". This page can either be a single secured page or the welcome page to a whole secured section of pages.

That's really all there is to it. However, there are a few things that you need to understand about **Request.Form**. **Request.Form** will only return values on the page that was specified in the **ACTION**. If the user has moved beyond your processing page, their input on the form will be gone. In our example above, if you tried a **Request.Form("pswd")** on your securedpage.asp, you would always get an empty string returned ("").

The other very important item that you should realize is that the form elements can be very different in how they are handled. Here is a brief overview of how to handle each of the elements that we talked about earlier:

Text Boxes - Whether it's a simple or masked text box, **Request.Form** will return the user's input.

Text Areas - This works exactly the same as the Text Boxes. If you give **Request.Form** the name of the Text Area, it will return what the user input. If you set a default value for the Text Area and the user made no changes, you will get the default value back.

Radio Buttons - These are a bit different. With Radio Buttons, you will need to use the group name. When you give **Request.Form** the group name it will return the VALUE of the button the user selected. The VALUE is what will tell you which button was selected so that you can properly process the user's input.

Check Boxes - With Check Boxes, you will have to check each individual check box to see whether it has been checked. Remember, Check Boxes each have a unique name, so it can be quite cumbersome to check dozens of check boxes from a form. **Request.Form** will return the VALUE that you set for each check box. You can use the Check Box VALUE to store a simple indicator that lets you know it was checked, like the word "ON", or place a VALUE that is a bit more useful like a special ID number.

Drop Down Lists - By default, a Drop Down List will only allow for one selection. **Request.Form** will return the OPTION name that the user chose.

We will show you additional examples of these form elements in action as we progress through this series.

Now, there is one more little item to take care of to finish off our practical example we started above. For **each** page that you want to be secured, add this little bit of ASP to the top of the page:

```
<% Option Explicit %>

<% If Session("access") <> "yes" then

      Response.Redirect "login.asp"

   End If %>
```

The code above will check the Session variable that you created to see if the user is indeed logged in. If they are not, they are redirected back to your login page so that they can enter the password.

2.11.3 Sending Out Some Email

There are literally billions of emails flying around the globe almost every day. Because of this ever increasing reliance on email, ASP has the ability to generate email on the fly. This is a very useful tool especially for automated responses to subscription submissions, information requests and e-commerce applications.

Sending an email from your ASP pages is actually quite simple. There are several different software manufacturers that sell server software to handle email, however. I am going to show you how to use CDONTS, which is most commonly used with ASP. Be sure to check and see what is installed on your particular server before using the code below.

Most ASP email is handled pretty much the same way, it's just the syntax that changes. Even if you don't have CDONTS available, this example should at least give you a working understanding of how to create email from your ASP pages.

```
<% Option Explicit %>

<% Dim MyEmail

    Set MyEmail =

    Server.CreateObject("CDONTS.NewMail")

    MyEmail.From = "name@yourdomain.com"

    MyEmail.To = "name@theirdomain.com"

    MyEmail.Subject = "ASP email Test"

    MyEmail.Body = "This is my ASP email test!"          MyEmail.Send

    Set MyEmail = Nothing %>
```

Look pretty easy? Well, it is. At first glance, almost everything should look pretty logical to you. Let's see how the process works line by line.

The first line will probably be the most confusing for you. What we are doing here is basically just creating the shell for our new email. By setting MyEmail equal to Server.CreateObject("CDONTS.NewMail") we are making an instance of the email object that we are affectionately naming "MyEmail".

The next four lines should make perfect sense to you. Here we are defining all of the basic elements of the email, i.e. the from address, the to address, the subject line and the body itself. Be careful to always complete both the To and the From, otherwise you will get an error when sending.

Once you have set the contents of your email, it's a simple process to send it on its way. Just tell the object to send by using MyEmail.Send.

Lastly, we need to clear the email object when we are done with it. It is a good practice to clear all objects when you are done with them so that you don't waste valuable memory space on the server. To clear an object just set the object equal to Nothing. Also, be sure to not use quotes around Nothing, otherwise you will get an error.

So, how do you insert a line break into the body of your email?

What I always use is vbcrlf. That stands for "Visual Basic Carriage Return Line Feed" which pops the text to a new line. This only works if you are using VBScript, though. If you are using JavaScript, you will have to add in the carriage return and line feed characters. Here's an example using vbcrlf:

```
MyEmail.Body = "Hello!" & vbcrlf & vbcrlf & _

                    "This is my ASP email test!"
```

The example above will insert two returns after "Hello!" which will effectively insert a blank line between "Hello!" and the rest of the body text.

2.12 Connecting to a Database

Part of ASP's great power is its ability to interact with databases. In the next segment we will jump right in:

Connecting to a Database
Adding Data to Your Database
Modifying and Deleting Data in Your Database

One of the beauties of using VBScript and ASP for your Web database connection is that you do not have to learn anything new at least, not if you have done much database programming in Visual Basic. Any Microsoft Web server should have ActiveX Data Objects, or ADO, installed. ADO is Microsoft's new database object model, partially introduced with Visual Basic version 5 and now fully supported in version 6. (See the VDM cover story for Nov/Dec 1998: "Much ADO About Database Access" by Eric Harmon.) If you have worked with ADO at all, you know how much easier it is to use than older database technologies such as RDO. All this power is available to you in your Active Server Pages.

Well, perhaps not quite all. When creating a standalone Visual Basic database program, the easiest way to use ADO is by means of the ADO Data Control, which you place on a form and can easily program to serve as the link between your program and the data source. Since many of Visual Basic's intrinsic controls can be bound to an ADO Data Control, programming a functional database front end is a piece of cake. Unfortunately, neither the ADO Data Control nor the data bound controls can be used on a Web page. Still, even without these controls.ADO makes database access relatively easy. The programming, however, is textual rather than purely visual. At the heart of the ADO model are two objects: **Connection** and **RecordSet**. A **Connection** object represents the link between your page and the data source; that is, the database file. A **RecordSet** object represents a set of records returned from the data source in response to a query.

You have the option of using these objects directly in a regular Visual Basic program, permitting you to use ADO to access a data source without using the ADO Data Control. In an Active Server Page, this is the only way to go.

The first required step is to create a **Connection** object using the **Server** object's **CreateObject** method. Since you are dealing with an object reference you must use the **Set** keyword:

```
set con = server.createobject("adodb.connection")
```

Next, establish a link between the **Connection** object and your database. The easiest way to do this is to create a connection string that specifies the type of database and the specific database file. Here's a very simple connection string that specifies a Microsoft Access database file named mydata.mdb:

```
constr = "Provider=Microsoft.Jet.OLEDB.3.51; Data Source=mydata.mdb;"
```

Connection strings can get a lot more complicated; see the sidebar "ADO Connection Strings the Easy Way" for a time-saving tip. Once you have your connection string, you establish the connection by passing the connection string to the Connection object's Open method:

```
con.open constr
```

Once the connection is established, you next create a RecordSet that contains the desired records from a table in the data source. This requires putting together a Structured Query Language (SQL) query that selects the desired fields and records from the table. This is not the time or place to go into the details of SQL, so please take my word that the SQL query used here selects all fields and all records from a database table named sales. Here's how you would create the RecordSet:

```
set rs = con.Execute("select * from sales")
```

You can see that the Connection object's Execute method returns a reference to a RecordSet. After this script statement executes, the RecordSet contains the records from the table and you can use the **RecordSet** object's methods and properties to access and manipulate the data. Here, for example, is the VBScript code required to display all of the data in the **RecordSet**. This script creates an HTML table with the field names displayed in the first row and the data displayed, one record per row, in the remainder of the table.

CHAPTER 3
DATABASE

3.1 Database

A computer database is a structured collection of records or data that is stored in a computer system. A database relies upon software to organize the storage of data. In other words, the software models the database structure in what are known as database models (or data models). The model in most common use today is the relational model. Other models such as the hierarchical model and the network model use a more explicit representation of relationships (see below for explanation of the various database models).[5]

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

3.2 History of Database

The earliest known use of the term data base was in November 1963, when the System Development Corporation sponsored a symposium under the title Development and Management of a Computer-centered Data Base[1]. Database as a single word became common in Europe in the early 1970s and by the end of the decade it was being used in major American newspapers. (The abbreviation DB, however, survives.)

The first database management systems were developed in the 1960s. A pioneer in the field was Charles Bachman. Bachman's early papers show that his aim was to make more effective use of the new direct access storage devices becoming available: until then, data processing had been based on punched cards and magnetic tape, so that serial processing was the dominant activity. Two key data models arose at this time: CODASYL developed the network model based on Bachman's ideas, and (apparently independently) the hierarchical model was used in a system developed by North American Rockwell later adopted by IBM as the cornerstone of their IMS product.

While IMS along with the CODASYL IDMS were the big, high visibility databases developed in the 1960s, several others were also born in that decade, some of which have a significant installed base today.

Two worthy of mention are the PICK and MUMPS databases, with the former developed originally as an operating system with an embedded database and the latter as a programming language and database for the development of healthcare systems.

The relational model was proposed by E. F. Codd in 1970. He criticized existing models for confusing the abstract description of information structure with descriptions of physical access mechanisms. For a long while, however, the relational model remained of academic interest only. While CODASYL products (IDMS) and network model products (IMS) were conceived as practical engineering solutions taking account of the technology as it existed at the time, the relational model took a much more theoretical perspective, arguing (correctly) that hardware and software technology would catch up in time. Among the first implementations were Michael Stonebraker's Ingres at Berkeley, and the System R project at IBM. Both of these were research prototypes, announced during 1976. The first commercial products, Oracle and DB2, did not appear until around 1980. The first successful database product for microcomputers was dBASE for the CP/M and PC-DOS/MS-DOS operating systems.

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

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

In the 2000s, the fashionable area for innovation is the XML database. As with object databases, this has spawned a new collection of start-up companies, but at the same time the key ideas are being integrated into the established relational products.

3.3 Database Models

Various techniques are used to model data structure. Most database systems are built around one particular data model, although it is increasingly common for products to offer support for more than one model. For any one logical model various physical implementations may be possible, and most products will offer the user some level of control in tuning the physical implementation, since the choices that are made have a significant effect on performance. An example is the relational model: all serious implementations of the relational model allow the creation of indexes which provide fast access to rows in a table if the values of certain columns that are known

3.3.1 Hierarchical model

In a hierarchical model, data is organized into a tree-like structure, implying a single upward link in each node to describe the nesting, and a sort field to keep the records in a particular order in each same-level list.

3.3.2 Network model

The network model tends to store records with links to other records. Associations are tracked via "pointers". These pointers can be node numbers or disk addresses. Most network databases tend to also include some form of hierarchical model.

Examples of database engines that have network model capabilities are RDM Embedded and RDM Server.

3.3.3 Relational model

The basic data structure of the relational model can be thought of as a table, where information about a particular entity (say, an employee) is represented in columns and rows (also called tuples). Thus, the "relation" in "relational database" refers to the various tables in the database; a relation is a set of tuples. The columns enumerate the various attributes of the entity (the employee's name, address or phone number, for example), and a row is an actual instance of the entity (a specific employee) that is represented by the relation. As a result, each tuple of the employee table represents various attributes of a single employee.

All relations (and, thus, tables) in a relational database have to adhere to some basic rules to qualify as relations. First, the ordering of columns is immaterial in a table. Second, there can't be identical tuples or rows in a table. And third, each tuple will contain a single value for each of its attributes i.e. each tuple has an atomic value.

A relational database contains multiple tables, each similar to the one in the "flat" database model. One of the strengths of the relational model is that, in principle, any value occurring in two different records (belonging to the same table or to different tables), implies a relationship among those two records. Yet, in order to enforce explicit integrity constraints, relationships between records in tables can also be defined explicitly, by identifying or non-identifying parent-child relationships characterized by assigning cardinality (1:1, (0) 1:M, M:M). Tables can also have a designated single attribute or a set of attributes that can act as a "key", which can be used to uniquely identify each tuple in the table.

A key that can be used to uniquely identify a row in a table is called a primary key. Keys are commonly used to join or combine data from two or more tables. For example, an Employee table may contain a column named Location which contains a value that matches the key of a Location table. Keys are also critical in the creation of indices, which facilitate fast retrieval of data from large tables. Any suitable column can be a key, or multiple columns can be grouped together into a compound key. It is not necessary to define all the keys in advance; a column can be used as a key even if it was not originally intended to be one.

3.3.4 Relational operations

Users (or programs) request data from a relational database by sending it a query that is written in a special language, usually a dialect of SQL. Although SQL was originally intended for end-users, it is much more common for SQL queries to be embedded into software that provides an easier user interface.

In response to a query, the database returns a result set, which is the list of rows constituting the answer. The simplest query is just to return all the rows from a table, but more often, the rows are filtered in some way to return just the answer wanted. Often, data from multiple tables are combined into one, by doing a join. There are a number of relational operations in addition to join.

3.3.5 Normal forms

Relations are classified based upon the types of anomalies to which they're vulnerable. A database that's in the first normal form is vulnerable to all types of anomalies, while a database that's in the domain/key normal form has no modification anomalies. Normal forms are hierarchical in nature. That is, the lowest level is the first normal form, and the database cannot meet the requirements for higher level normal forms without first having met all the requirements of the lesser normal form.

3.3.6 Post-relational database models

Several products have been identified as post-relational because the data model incorporates relations but is not constrained by the Information Principle, requiring that all information is represented by data values in relations.

Date's information principle states:

The entire information content of the database is represented in one and only one way. Namely *as explicit values in column positions (attributes) and rows in relations (tuples)* Ergo, there are no explicit pointers between related tables.

Products using a post-relational data model typically employ a model that actually pre-dates the relational model. These might be identified as a directed graph with trees on the nodes.

Examples of models that could be classified as post-relational are PICK aka MultiValue, and MUMPS.

3.3.7 Object database models

In recent years, the object-oriented paradigm has been applied to database technology, creating a new programming model known as object databases. These databases attempt to bring the database world and the application programming world closer together, in particular by ensuring that the database uses the same type system as the application program. This aims to avoid the overhead (sometimes referred to as the impedance mismatch) of converting information between its representation in the database (for example as rows in tables) and its representation in the application program (typically as objects). At the same time, object databases attempt to introduce the key ideas of object programming, such as encapsulation and polymorphism, into the world of databases.

A variety of these ways have been tried for storing objects in a database. Some products have approached the problem from the application programming end, by making the objects manipulated by the program persistent. This also typically requires the addition of some kind of query language, since conventional programming languages do not have the ability to find objects based on their information content. Others have attacked the problem from the database end, by defining an object-oriented data model for the database, and defining a database programming language that allows full programming capabilities as well as traditional query facilities.

3.4 Database internals

3.4.1 Storage and physical database design

Database tables/indexes are typically stored in memory or on hard disk in one of many forms, ordered/unordered flat files, ISAM, heaps, hash buckets or B+ trees. These have various advantages and disadvantages discussed further in the main article on this topic. The most commonly used are B+ trees and ISAM.

Other important design choices relate to the clustering of data by category (such as grouping data by month, or location), creating pre-computed views known as materialized views, partitioning data by range or hash. As well memory management and storage topology can be important design choices for database designers.

Just as normalization is used to reduce storage requirements and improve the extensibility of the database, conversely denormalization is often used to reduce join complexity and reduce execution time for queries.

3.4.2 Indexing

All of these databases can take advantage of indexing to increase their speed, and this technology has advanced tremendously since its early uses in the 1960s and 1970s. The most common kind of index is a sorted list of the contents of some particular table column, with pointers to the row associated with the value. An index allows a set of table rows matching some criterion to be located quickly. Typically, indexes are also stored in the various forms of data-structure mentioned above (such as B-trees, hashes, and linked lists). Usually, a specific technique is chosen by the database designer to increase efficiency in the particular case of the type of index required.

Relational DBMSs have the advantage that indexes can be created or dropped without changing existing applications making use of it. The database chooses between many different strategies based on which one it estimates will run the fastest. In other words, indexes are transparent to the application or end-user querying the database; while they affect performance, any SQL command will run with or without indexes existing in the database.

Relational DBMSs utilize many different algorithms to compute the result of an SQL statement. The RDBMS will produce a plan of how to execute the query, which is generated by analyzing the run times of the different algorithms and selecting the quickest. Some of the key algorithms that deal with joins are nested loop join, sort-merge join and hash join. Which of these is chosen depends on whether an index exists, what type it is, and its cardinality.

An index speeds up access to data, but it has disadvantages as well. First, every index increases the amount of storage on the hard drive necessary for the database file, and second, the index must be updated each time the data are altered, and this costs time. (Thus an index saves time in the reading of data, but it costs time in entering and altering data. It thus depends on the use to which the data are to be put whether an index is on the whole a net plus or minus in the quest for efficiency.)

A special case of an index is a primary index, or primary key, which is distinguished in that the primary index must ensure a unique reference to a record. Often, for this purpose one simply uses a running index number (ID number). Primary indexes play a significant role in relational databases, and they can speed up access to data considerably.

3.4.3 Transactions and concurrency

In addition to their data model, most practical databases ("transactional databases") attempt to enforce a database transaction . Ideally, the database software should enforce the ACID rules, summarized here:

- Atomicity: Either all the tasks in a transaction must be done, or none of them. The transaction must be completed, or else it must be undone (rolled back).
- Consistency: Every transaction must preserve the integrity constraints — the declared consistency rules — of the database. It cannot place the data in a contradictory state.
- Isolation: Two simultaneous transactions cannot interfere with one another. Intermediate results within a transaction are not visible to other transactions.

Durability: Completed transactions cannot be aborted later or their results discarded. They must persist through (for instance) restarts of the DBMS after crashes.
In practice, many DBMS's allow most of these rules to be selectively relaxed for better performance.
Concurrency control is a method used to ensure that transactions are executed in a safe manner and follow the ACID rules. The DBMS must be able to ensure that only serializable, recoverable schedules are allowed, and that no actions of committed transactions are lost while undoing aborted transactions .

3.4.4 Replication

Replication of databases is closely related to transactions. If a database can log its individual actions, it is possible to create a duplicate of the data in real time. The duplicate can be used to improve performance or availability of the whole database system. Common replication concepts include:

- Master/Slave Replication: All write requests are performed on the master and then replicated to the slaves
- Quorum: The result of Read and Write requests are calculated by querying a "majority" of replicas.
- Multimaster: Two or more replicas sync each other via a transaction identifier.

3.4.5 Security

Database security denotes the system, processes, and procedures that protect a database from unintended activity.

In the United Kingdom legislation protecting the public from unauthorized disclosure of personal information held on databases falls under the Office of the Information Commissioner. United Kingdom based organizations holding personal data in electronic format (databases for example) are required to register with the Data Commissioner. (reference: [1])

3.4.6 Locking

Locking is the act of putting a lock (access restriction) on an aspect of a database which at a particular given instance is being modified. Such locks can be applied on a row level, or on other levels such as an entire table. This helps maintain the integrity of the data by ensuring that only one user at a time can modify the data. Databases can also be locked for other reasons, like access restrictions for given levels of user.

Databases are also locked for routine database maintenance, which prevents changes being made during the maintenance.

3.4.7 Architecture

Depending on the intended use, there are a number of database architectures in use. Many databases use a combination of strategies. On-line Transaction Processing systems (OLTP) often use a row-oriented datastore architecture, while data-warehouse and other retrieval-focused applications like Google's BigTable, or bibliographic database(library catalogue) systems may use a column-oriented datastore architecture.
Document-Oriented, XML, Knowledgebases, as well as frame databases and rdf-stores (aka Triple-Stores), may also use a combination of these architectures in their implementation.
Finally it should be noted that not all database have or need a database 'schema' (so called schema-less databases).

3.5 Applications of databases

Databases are used in many applications, spanning virtually the entire range of computer software. Databases are the preferred method of storage for large multiuser applications, where coordination between many users is needed. Even individual users find them convenient, and many electronic mail programs and personal organizers are based on standard database technology. Software database drivers are available for most database platforms so that application software can use a common Application Programming Interface to retrieve the information stored in a database. Two commonly used database APIs are JDBC and ODBC.

For example suppliers database contains the data relating to suppliers such as;

- supplier name
- supplier code
- supplier address

3.6 Database as Cultural Form

Although originally a computer technology, the database, according to media theorist Lev Manovich, is becoming a new cultural form in its own right and a genre of new media. A cultural form is one of many ways that people represent the world—art and literature, for example. As contemporary culture is gradually computerized, Manovich argues, traditional cultural forms are being replaced with new ones that derive from the computer. He calls this transcoding. The database is the computer age's key form of cultural expression, as narrative was to the modern age via cinema. In this analysis, he is using "database" metaphorically.

CHAPTER 4

STRUCTURED QUERY LANGUAGE (SQL)

4.1 SQL

SQL (Structured Query Language) is a database computer language designed for the retrieval and management of data in relational database management systems (RDBMS), database schema creation and modification, and database object access control management.[4]

SQL is a standard interactive and programming language for querying and modifying data and managing databases. Although SQL is both an ANSI and an ISO standard, many database products support SQL with proprietary extensions to the standard language. The core of SQL is formed by a command language that allows the retrieval, insertion, updating, and deletion of data, and performing management and administrative functions. SQL also includes a Call Level Interface (SQL/CLI) for accessing and managing data and databases remotely.

The first version of SQL was developed at IBM by Donald D. Chamberlin and Raymond F. Boyce in the early 1970s. This version, initially called **SEQUEL**, was designed to manipulate and retrieve data stored in IBM's original relational database product, System R. The SQL language was later formally standardized by the American National Standards Institute (ANSI) in 1986. Subsequent versions of the SQL standard have been released as International Organization for Standardization (ISO) standards.

Originally designed as a declarative query and data manipulation language, variations of SQL have been created by SQL database management system (DBMS) vendors that add procedural constructs, control-of-flow statements, user-defined data types, and various other language extensions. With the release of the SQL:1999 standard, many such extensions were formally adopted as part of the SQL language via the SQL Persistent Stored Modules (SQL/PSM) portion of the standard.

4.2 History of SQL

During the 1970s, a group at **IBM's** San Jose research center developed the System R relational database management system, based on the model introduced by Edgar F. Codd in his influential paper, **A Relational Model of Data for Large Shared Data Banks**. Donald D. Chamberlin and Raymond F. Boyce of IBM subsequently created the **Structured English Query Language** (SEQUEL) to manipulate and manage data stored in System R. The acronym SEQUEL was later changed to SQL because "SEQUEL" was a trademark of the UK-based Hawker Siddeley aircraft company.

The first non-commercial non-SQL RDBMS, Ingres, was developed in 1974 at the U.C. Berkeley. Ingres implemented a query language known as QUEL, which was later supplanted in the marketplace by SQL.

In the late 1970s, Relational Software, Inc. (now Oracle Corporation) saw the potential of the concepts described by Codd, Chamberlin, and Boyce and developed their own SQL-based RDBMS with aspirations of selling it to the U.S. Navy, CIA, and other government agencies. In the summer of 1979, Relational Software, Inc. introduced the first commercially available implementation of SQL, Oracle V2 (Version2) for VAX computers. Oracle V2 beat IBM's release of the System/38 RDBMS to market by a few weeks.

After testing SQL at customer test sites to determine the usefulness and practicality of the system, IBM began developing commercial products based on their System R prototype including System/38, SQL/DS, and DB2, which were commercially available in 1979, 1981, and 1983, respectively.

4.2.1 Standardization

SQL was adopted as a standard by ANSI in 1986 and ISO in 1987. In the original SQL standard, ANSI declared that the official pronunciation for SQL is "es queue el". However, many English-speaking database professionals still use the nonstandard pronunciation (like the word "sequel"). Until 1996, the National Institute of Standards and Technology (NIST) data management standards program was tasked with certifying SQL DBMS compliance with the SQL standard. In 1996, however, the NIST data management standards program was dissolved, and vendors are now relied upon to self-certify their products for compliance.

The SQL standard is not freely available. SQL:2003 and SQL:2006 may be purchased from ISO or ANSI. A late draft of SQL:2003 is freely available as a zip archive, however, from Whitemarsh Information Systems Corporation. The zip archive contains a number of PDF files that define the parts of the SQL:2003 specification.

4.3 Scope and extensions

4.3.1 Procedural extensions

SQL is designed for a specific purpose: to query data contained in a relational database. SQL is a set-based, declarative query language, not an imperative language such as C or BASIC. However, there are extensions to Standard SQL which add procedural programming language functionality, such as control-of-flow constructs. In addition to the standard SQL/PSM extensions and proprietary SQL extensions, procedural and object-oriented programmability is available on many SQL platforms via DBMS integration with other languages. The SQL standard defines SQL/JRT extensions (SQL Routines and Types for the Java Programming Language) to support Java code in SQL databases. SQL Server 2005 uses the SQLCLR (SQL Server Common Language Runtime) to host managed .NET assemblies in the database, while prior versions of SQL Server were restricted to using unmanaged extended stored procedures which were primarily written in C. Other database platforms, like MySQL and Postgres, allow functions to be written in a wide variety of languages including Perl, Python, Tcl, and C.

4.3.2 Additional extensions

SQL:2003 also defines several additional extensions to the standard to increase SQL functionality overall. These extensions include:

The SQL/CLI, or **Call-Level Interface**, extension is defined in ISO/IEC 9075-3:2003. This extension defines common interfacing components (structures and procedures) that can be used to execute SQL statements from applications written in other programming languages. The SQL/CLI extension is defined in such a way that SQL statements and SQL/CLI procedure calls are treated as separate from the calling application's source code.

The SQL/MED, or **Management of External Data**, extension is defined by ISO/IEC 9075-9:2003. SQL/MED provides extensions to SQL that define foreign-data wrappers and datalink types to allow SQL to manage external data. External data is data that is accessible to, but not managed by, an SQL-based DBMS.

The SQL/OLB, or 'Object Language Bindings, extension is defined by ISO/IEC 9075-10:2003. SQL/OLB defines the syntax and symantics of SQLJ, which is SQL embedded in Java. The standard also describes mechanisms to ensure binary portability of SQLJ applications, and specifies various Java packages and their contained classes.

The SQL/Schemata, or **Information and Definition Schemas**, extension is defined by ISO/IEC 9075-11:2003. SQL/Schemata defines the Information Schema and Definition Schema, providing a common set of tools to make SQL databases and objects self-describing. These tools include the SQL object identifier, structure and integrity constraints, security and authorization specifications, features and packages of ISO/IEC 9075, support of features provided by SQL-based DBMS implementations, SQL-based DBMS implementation information and sizing items, and the values supported by the DBMS implementations.[10]

The SQL/JRT, or **SQL Routines and Types for the Java Programming Language**, extension is defined by ISO/IEC 9075-13:2003. SQL/JRT specifies the ability to invoke static Java methods as routines from within SQL applications. It also calls for the ability to use Java classes as SQL structured user-defined types.

The SQL/XML, or **XML-Related Specifications**, extension is defined by ISO/IEC 9075-14:2003. SQL/XML specifies SQL-based extensions for using conjunction with SQL.

The XML data type is introduced, as well as several routines, functions, and XML-to-SQL data type mappings to support manipulation and storage of XML in an SQL database.

The SQL/PSM, or **Persistent Stored Modules**, extension is defined by ISO/IEC 9075-4:2003. SQL/PSM standardizes procedural extensions for SQL, including flow of control, condition handling, statement condition signals and resignals, cursors and local variables, and assignment of expressions to variables and parameters. In addition, SQL/PSM formalizes declaration and maintenance of persistent database language routines (e.g., "stored procedures").

4.4 Language elements

This chart shows several of the SQL language elements that compose a single statement.

The SQL language is sub-divided into several language elements, including:

- Statements which may have a persistent effect on schemas and data, or which may control transactions, program flow, connections, sessions, or diagnostics.
- Queries which retrieve data based on specific criteria.
- Expressions which can produce either scalar values or tables consisting of columns and rows of data.
- Predicates which specify conditions that can be evaluated to SQL three-valued logic (3VL) Boolean truth values and which are used to limit the effects of statements and queries, or to change program flow.
- Clauses which are (in some cases optional) constituent components of statements and queries.
- Whitespace is generally ignored in SQL statements and queries, making it easier to format SQL code for readability.
-

4.4.1 Queries

The most common operation in SQL databases is the query, which is performed with the declarative SELECT keyword. SELECT retrieves data from a specified table, or multiple related tables, in a database. While often grouped with Data Manipulation Language (DML) statements, the standard SELECT query is considered separate from SQL DML, as it has no persistent effects on the data stored in a database. Note that there are some platform-specific variations of SELECT that can persist their effects in a database, such as the SELECT INTO syntax that exists in some databases.

SQL queries allow the user to specify a description of the desired result set, but it is left to the devices of the database management system (DBMS) to plan, optimize, and perform the physical operations necessary to produce that result set in as efficient a manner as possible. An SQL query includes a list of columns to be included in the final result immediately following the SELECT keyword. An asterisk ("*") can also be used as a "wildcard" indicator to specify that all available columns of a table (or multiple tables) are to be returned. SELECT is the most complex statement in SQL, with several optional keywords and clauses, including:

- The FROM clause which indicates the source table or tables from which the data is to be retrieved. The FROM clause can include optional JOIN clauses to join related tables to one another based on user-specified criteria.
- The WHERE clause includes a comparison predicate, which is used to restrict the number of rows returned by the query. The WHERE clause is applied before the GROUP BY clause. The WHERE clause eliminates all rows from the result set where the comparison predicate does not evaluate to True.
- The GROUP BY clause is used to combine, or group, rows with related values into elements of a smaller set of rows. GROUP BY is often used in conjunction with SQL aggregate functions or to eliminate duplicate rows from a result set.
- The HAVING clause includes a comparison predicate used to eliminate rows after the GROUP BY clause is applied to the result set. Because it acts on the results of the GROUP BY clause, aggregate functions can be used in the HAVING clause predicate.

- The ORDER BY clause is used to identify which columns are used to sort the resulting data, and in which order they should be sorted (options are ascending or descending). The order of rows returned by an SQL query is never guaranteed unless an ORDER BY clause is specified.

4.4.2 Data manipulation

First, there are the standard Data Manipulation Language (DML) elements. DML is the subset of the language used to add, update and delete data.

- MERGE is used to combine the data of multiple tables. It is something of a combination of the INSERT and UPDATE elements. It is defined in the SQL:2003 standard; prior to that, some databases provided similar functionality via different syntax, sometimes called an "upsert".

4.4.3 Transaction controls

Transactions, if available, can be used to wrap around the DML operations:

- START TRANSACTION (or BEGIN WORK, or BEGIN TRANSACTION, depending on SQL dialect) can be used to mark the start of a database transaction, which either completes completely or not at all.
- COMMIT causes all data changes in a transaction to be made permanent.
- ROLLBACK causes all data changes since the last COMMIT or ROLLBACK to be discarded, so that the state of the data is "rolled back" to the way it was prior to those changes being requested.

Once the COMMIT statement has been executed, the changes cannot be rolled back. In other words, its meaningless to have ROLLBACK executed after COMMIT statement and vice versa.COMMIT and ROLLBACK interact with areas such as transaction control and locking. Strictly, both terminate any open transaction and release any locks held on data. In the absence of a START TRANSACTION or similar statement, the semantics of SQL are implementation-dependent.

4.4.4 Data definition

The second group of keywords is the Data Definition Language (DDL). DDL allows the user to define new tables and associated elements. Most commercial SQL databases have proprietary extensions in their DDL, which allow control over nonstandard features of the database system. The most basic items of DDL are the CREATE, ALTER, RENAME, TRUNCATE and DROP statements:

- CREATE causes an object (a table, for example) to be created within the database.
- DROP causes an existing object within the database to be deleted, usually irretrievably.
- TRUNCATE deletes all data from a table (non-standard, but common SQL statement).
- ALTER statement permits the user to modify an existing object in various ways -- for example, adding a column to an existing table.

4.4.5 Data control

The third group of SQL keywords is the Data Control Language (DCL). DCL handles the authorization aspects of data and permits the user to control who has access to see or manipulate data within the database. Its two main keywords are:

- GRANT authorizes one or more users to perform an operation or a set of operations on an object.
- REVOKE removes or restricts the capability of a user to perform an operation or a set of operations.

4.4.6 Other

- ANSI-standard SQL supports double dash, --, as a single line comment identifier (some extensions also support curly brackets or C style /* comments */ for multi-line comments).

4.5 Criticisms of SQL

Technically, SQL is a declarative computer language for use with "SQL databases". Theorists and some practitioners note that many of the original SQL features were inspired by, but in violation of, the relational model for database management and its tuple calculus realization. Recent extensions to SQL achieved relational completeness, but have worsened the violations, as documented in The Third Manifesto.

In addition, there are also some criticisms about the practical use of SQL:

- Implementations are inconsistent and, usually, incompatible between vendors. In particular date and time syntax, string concatenation, nulls, and comparison case sensitivity often vary from vendor to vendor.
- The language makes it too easy to do a Cartesian join (joining all possible combinations), which results in "run-away" result sets when WHERE clauses are mistyped. Cartesian joins are so rarely used in practice that requiring an explicit CARTESIAN keyword may be warranted. SQL 1992 introduced the CROSS JOIN keyword that allows the user to make clear that a cartesian join is intended, but the shorthand "comma-join" with no predicate is still acceptable syntax.
- It is also possible to misconstruct a WHERE on an update or delete, thereby affecting more rows in a table than desired.
- The grammar of SQL is perhaps unnecessarily complex, borrowing a COBOL-like keyword approach, when a function-influenced syntax could result in more re-use of fewer grammar and syntax rules. This is perhaps due to IBM's early goal of making the language more English-like so that it is more approachable to those without a mathematical or programming background. (Predecessors to SQL were more mathematical.)

4.5.1 Reasons for lack of portability

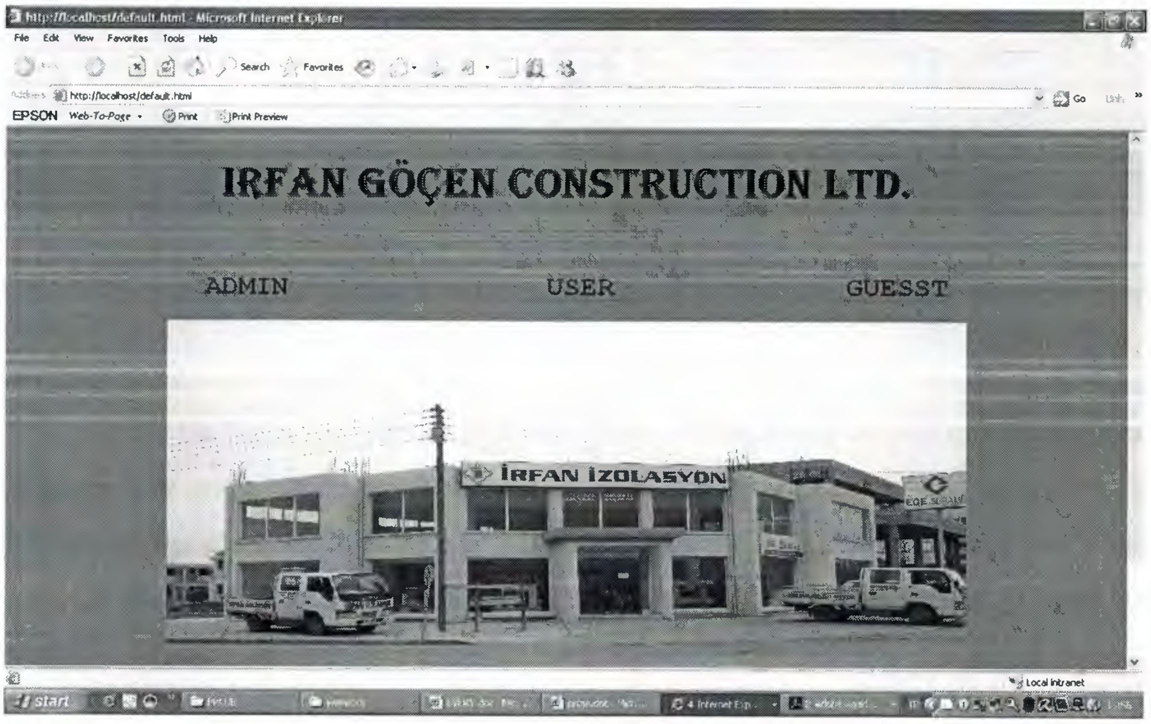
Popular implementations of SQL commonly omit support for basic features of Standard SQL, such as the DATE or TIME data types, preferring variations of their own. As a result, SQL code can rarely be ported between database systems without modifications.

There are several reasons for this lack of portability between database systems:

- The complexity and size of the SQL standard means that most databases do not implement the entire standard.
- The standard does not specify database behavior in several important areas (e.g. indexes, file storage...), leaving it up to implementations of the database to decide how to behave.
- The SQL standard precisely specifies the syntax that a conforming database system must implement. However, the standard's specification of the semantics of language constructs is less well-defined, leading to areas of ambiguity.
- Many database vendors have large existing customer bases; where the SQL standard conflicts with the prior behavior of the vendor's database, the vendor may be unwilling to break backward compatibility.

CHAPTER 5
Online Financial System For A Constuction Company

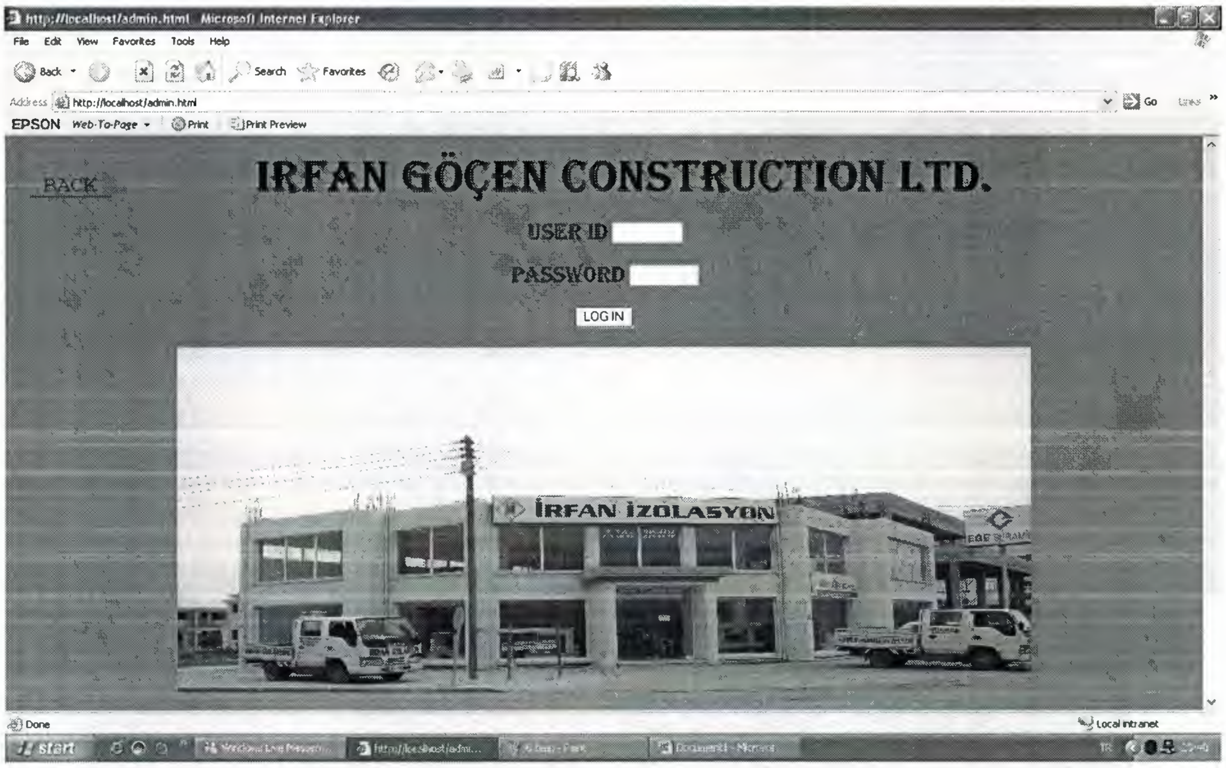
Firstly, I want to determine how I create these programs. I write HTML and ASP code in the notepad it also install all pc in automaticaly. And I install internet information services for write asp program.I put windows xp cd's in my cd rom then, go control panel, add or remove programs, add/remove windows components then windows components wilzard open, then I click Internet Information Services icons and install IIS. When I write HTML code in notepad I save my program like .html and when I write ASP code in notepad I save my program like .asp. first page for the web design it must be default.html then we open browser and we write http://localhost/default.html only I see my project in this pc like local host because every programs saved in my pc path of C:\Inetpub\wwwroot.



http://localhost/default.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
http://localhost/default.html
IRFAN GÖÇEN CONSTRUCTION LTD.
ADMIN
USER
GUESST
İRFAN İZOLASYON
Local intranet
start

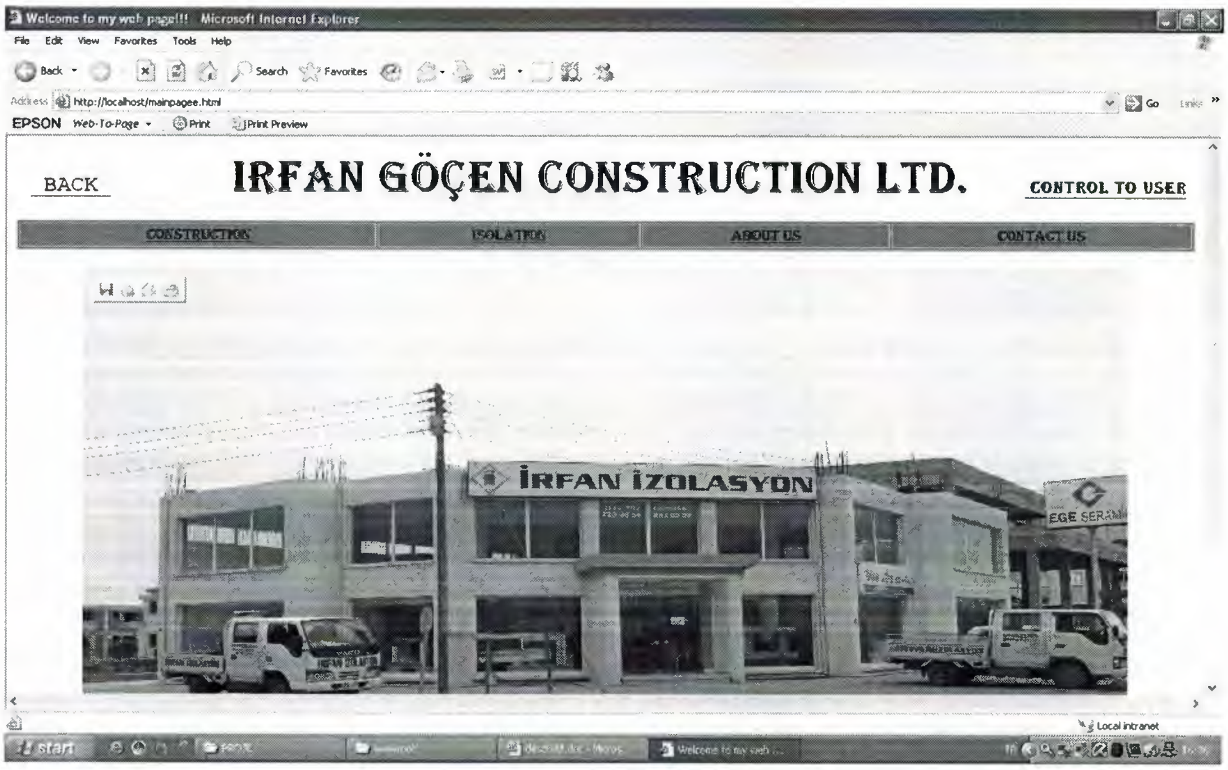
Default Page

The first page of the program is default page. It is include name of contruction and build of business.Three links for enter program. First admin, second user and last guesst links. If admin and user click you must enter program with password. But if click guesst enter program directly. I divide into 3 links becouse of admin can make search, delete, insert and update. User make search, insert, update. And guesst just search for see the product.


http://localhost/admin.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/admin.html
BACK
IRFAN GÖÇEN CONSTRUCTION LTD.
USER ID
PASSWORD
LOG IN
İRFAN İZOLASYON
Done
Local intranet
start

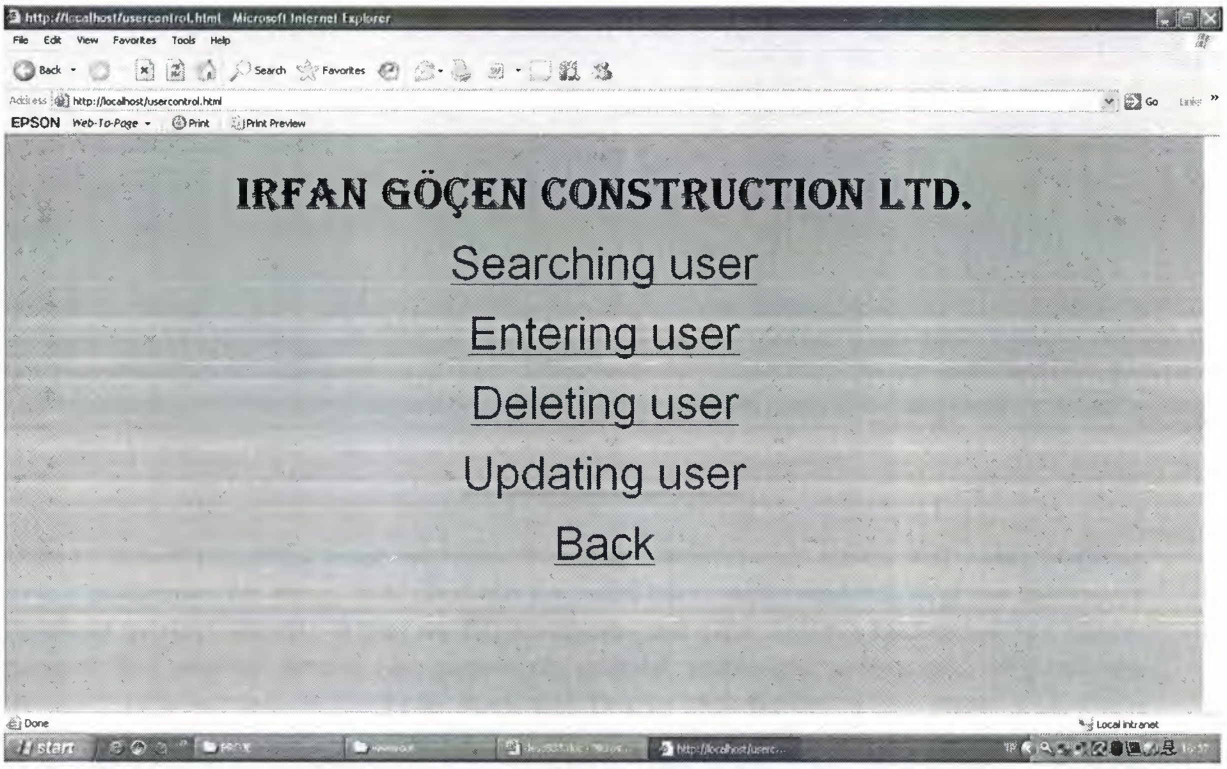
Enter The Program

This page is the login page. When the user clicks on the admin or user link, this screen appears. For enter the program, admin or user must enter user ıd and password. If details is true, can enter the mainpage. If details are invalid again this pages view. Until your dateils true. If we want to turn back we have back link. This link to turn default page.


Welcome to my web page!!! - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/mainpagee.html
Go Links
EPSON Web-To-Page Print Print Preview
BACK
IRFAN GÖÇEN CONSTRUCTION LTD.
CONTROL TO USER
CONSTRUCTION
ISOLATION
ABOUT US
CONTACT US
İRFAN İZOLASYON
EGE SERAMİK
Local intranet
start

Main Page

This page is mainpage of program. The page include title, name of construction, picture of construction and six links. First four links to details about company. First construction link, second insulation, third description about company and fourth link to contact details about company. Control to user link for admin can control staff of company and we have back link to turn default page.


http://localhost/usercontrol.html Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/usercontrol.html
Go Links
EPSON Web-To-Page Print Print Preview
Done
Local intranet
start

IRFAN GÖÇEN CONSTRUCTION LTD.

Searching user

Entering user

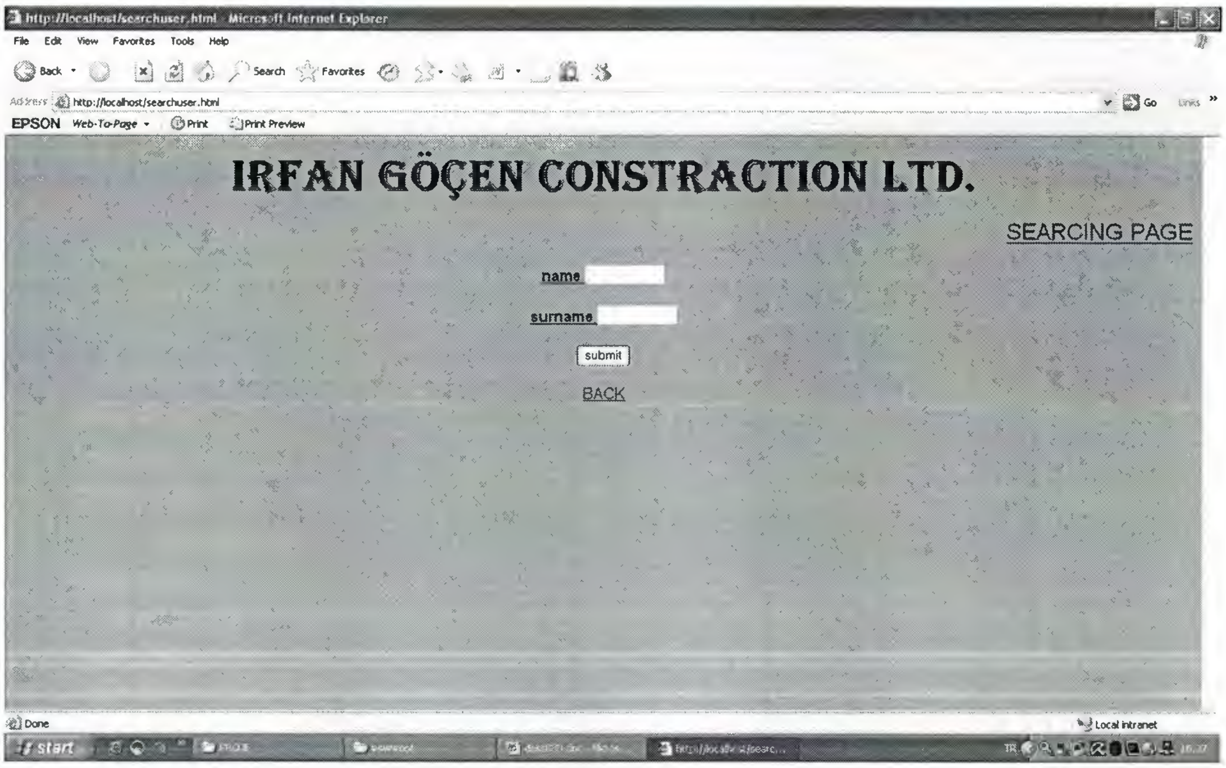
Deleting user

Updating user

Back

User Control

When we click control to user's link this page is view. Admin can see this page.And admin can make search, enter, delete and update so we have four links to make search, enter, delete and update. And back link to back before of this page.


http://localhost/searchuser.html - Microsoft Internet Explorer
http://localhost/searchuser.html

IRFAN GÖÇEN CONSTRACTION LTD.

SEARCING PAGE

name

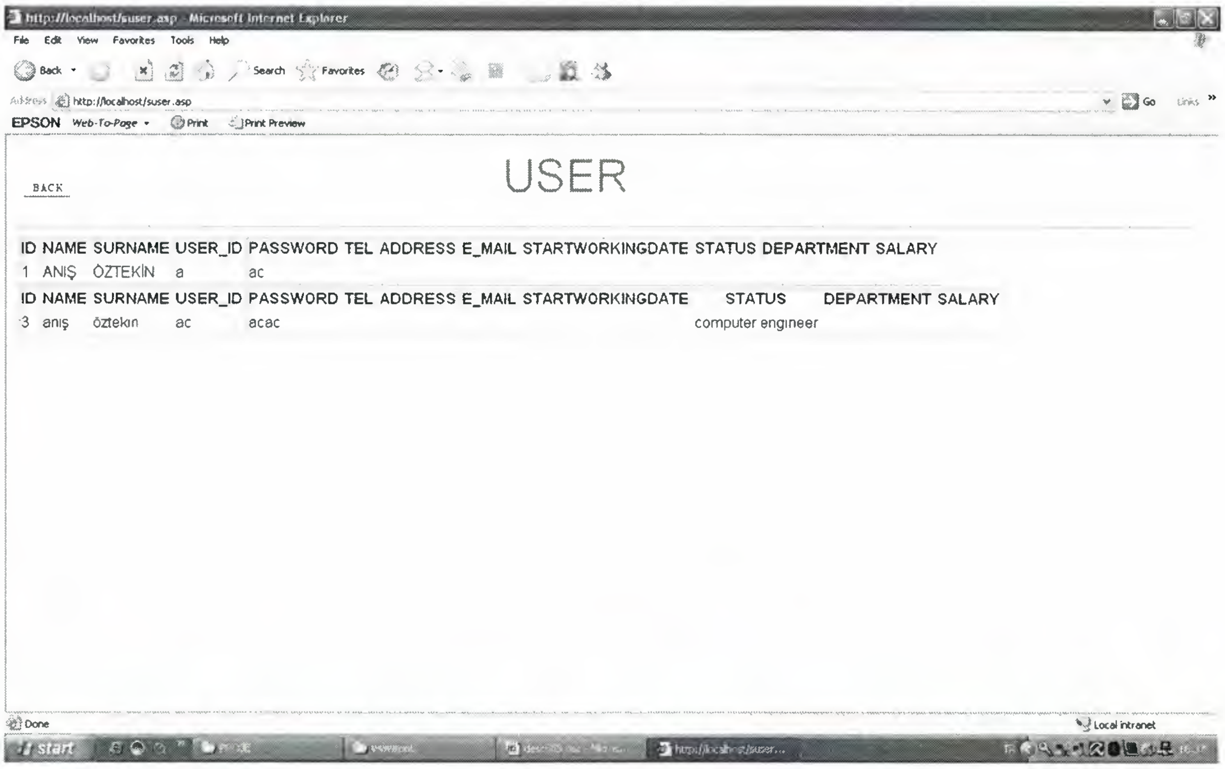
surname

submit

BACK

Searching Page

When clicked searching page we view this page. Admin can search with who working in the company. We have two textfield for enter name and surname. And we fill theese textfield to start search. When we fill textfields and click submit button we see below page to this worker details like password, user id, status and salary etc.. We can see theese details becouse of database. And back link to back before of this page.

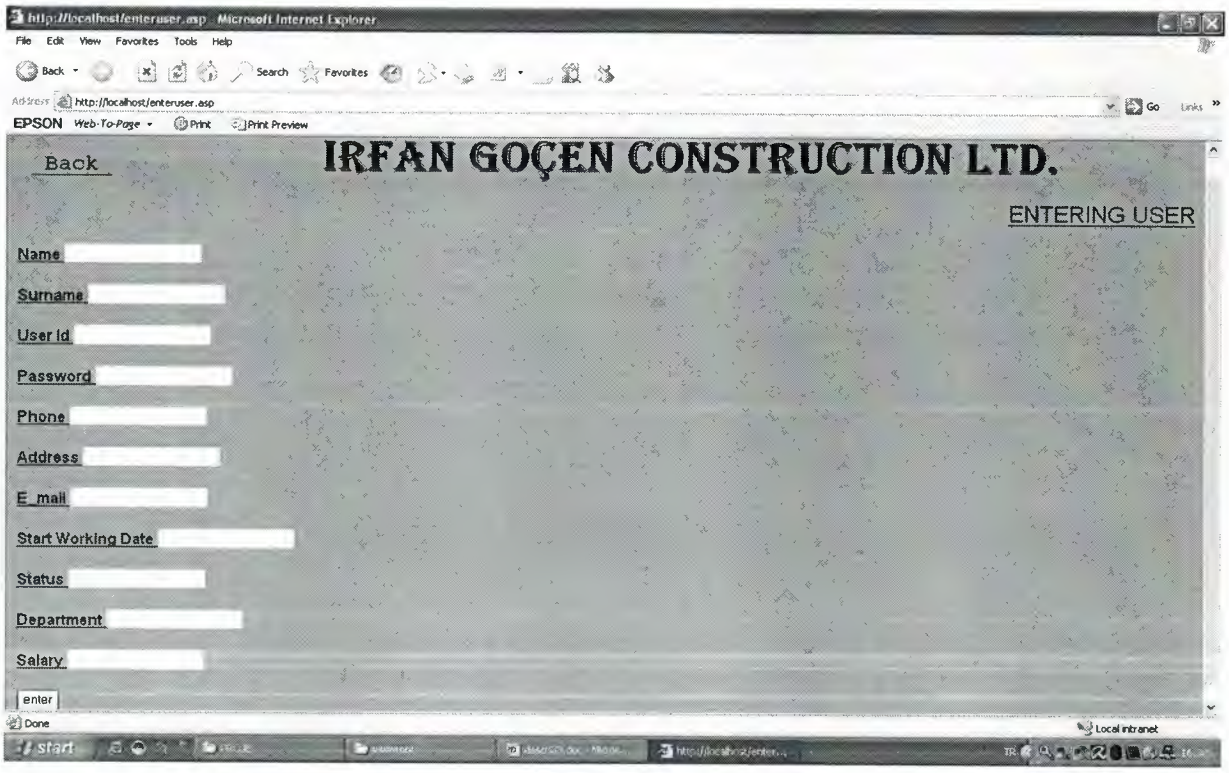

http://localhost/suser.asp - Microsoft Internet Explorer
http://localhost/suser.asp

BACK

USER

ID	NAME	SURNAME	USER_ID	PASSWORD	TEL	ADDRESS	E_MAIL	STARTWORKINGDATE	STATUS	DEPARTMENT	SALARY
1	ANIŞ	ÖZTEKİN	a	ac							

ID	NAME	SURNAME	USER_ID	PASSWORD	TEL	ADDRESS	E_MAIL	STARTWORKINGDATE	STATUS	DEPARTMENT	SALARY
3	anış	öztekın	ac	acac					computer engineer		


http://localhost/enteruser.asp - Microsoft Internet Explorer

Back

IRFAN GOÇEN CONSTRUCTION LTD.

ENTERING USER

Name

Surname

User Id

Password

Phone

Address

E_mail

Start Working Date

Status

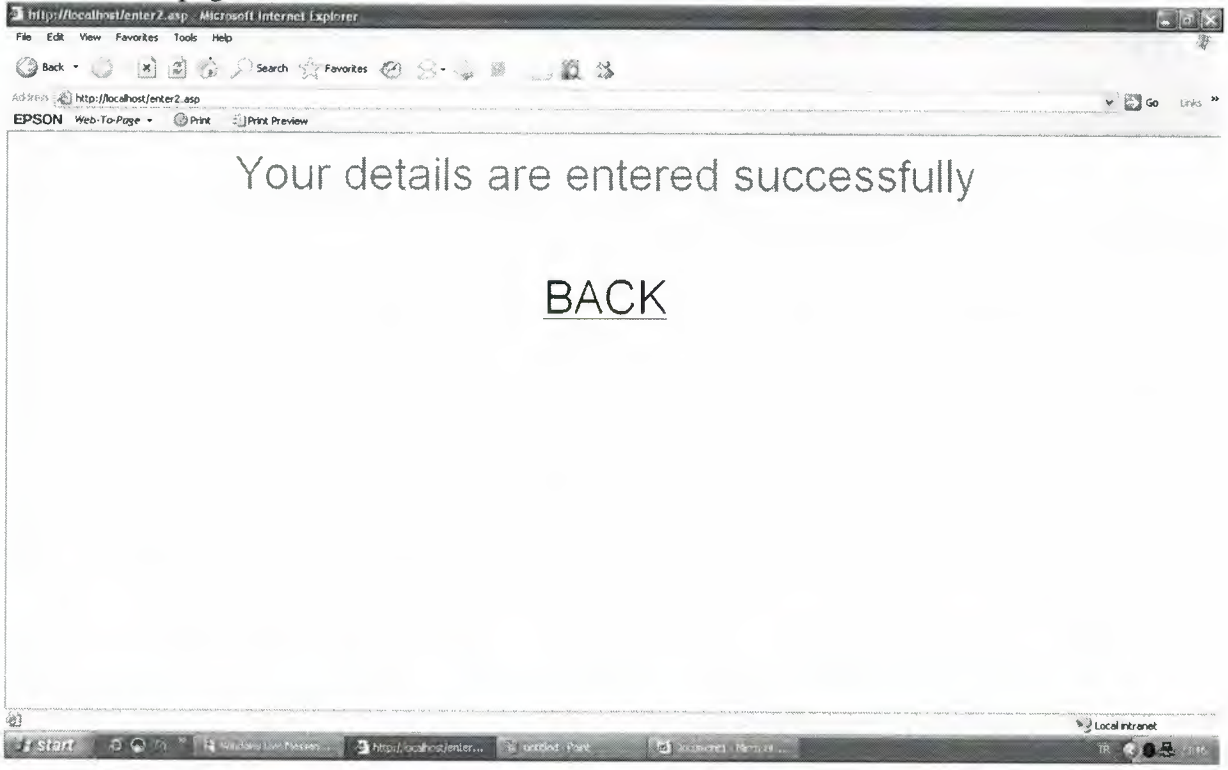
Department

Salary

enter

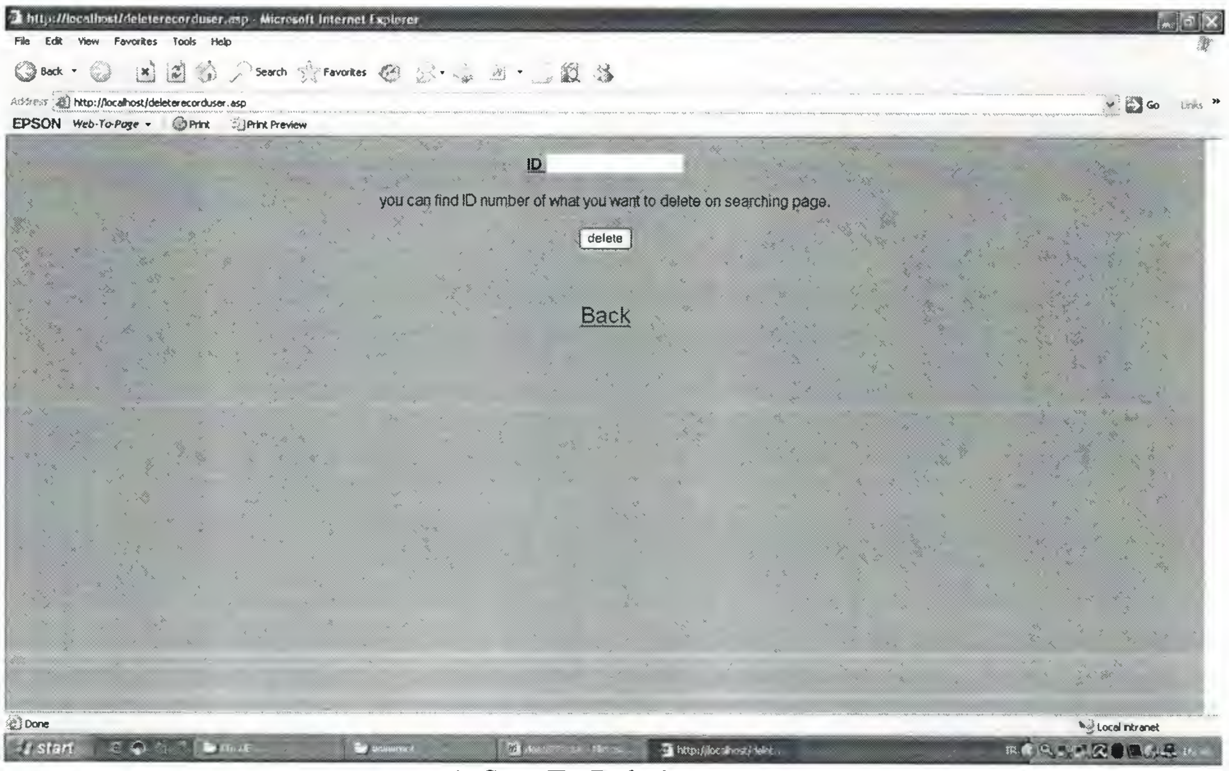
Entering Page For Workers

This page for enter the worker details in the textfield and insert database. This details for give password and user ıd for every worker to login this web page like user. We have descriptions and textfields to fill them. And we have submit button. we fill in textfields then click submit button. we insert theese details in the database. And also we have back link to turn before of this page. And if your details entered successfully you see as below page.


http://localhost/enter2.asp - Microsoft Internet Explorer

Your details are entered successfully

BACK

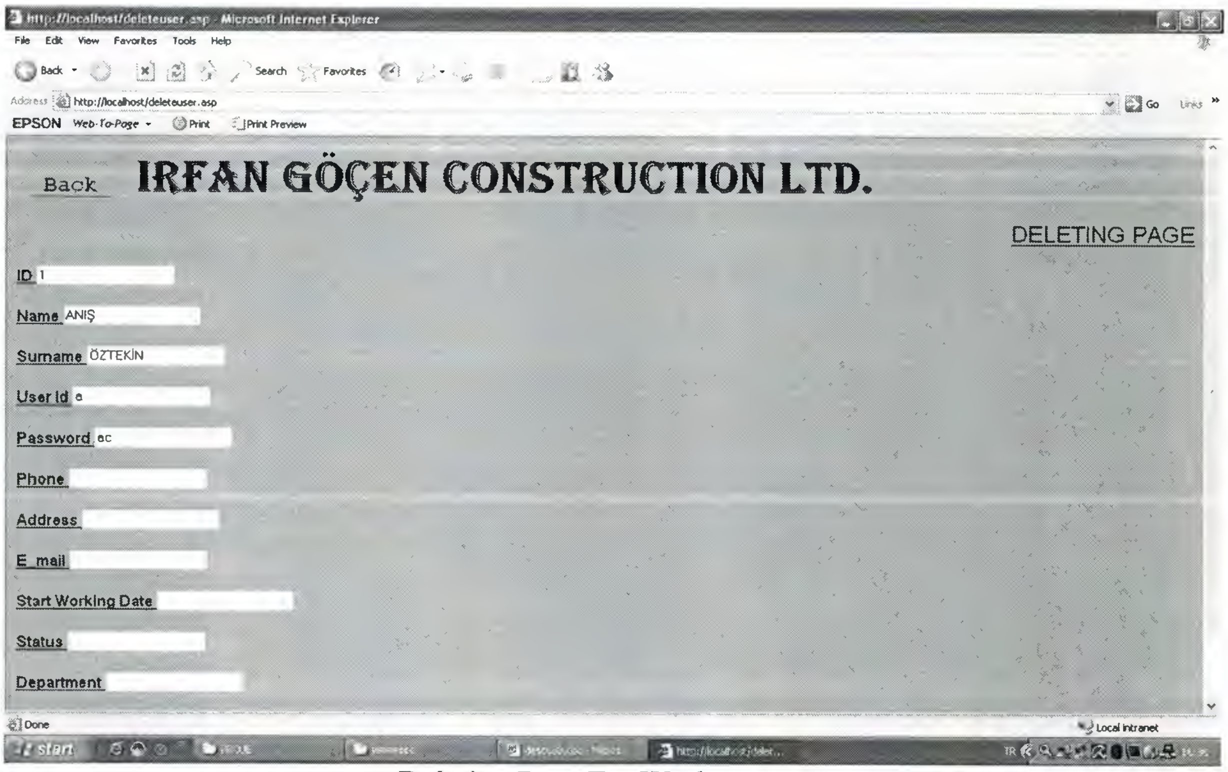

http://localhost/deleterecorduser.asp - Microsoft Internet Explorer

ID

you can find ID number of what you want to delete on searching page.

1. Step To Deleting

This page first step to deleting. Admin can delete every thing about workers. Admin write worker ıd in the texfield. Then view below page.


http://localhost/deleteuser.asp - Microsoft Internet Explorer

IRFAN GÖÇEN CONSTRUCTION LTD.

DELETING PAGE

ID 1

Name ANIŞ

Surname ÖZTEKİN

User Id a

Password ac

Phone

Address

E_mail

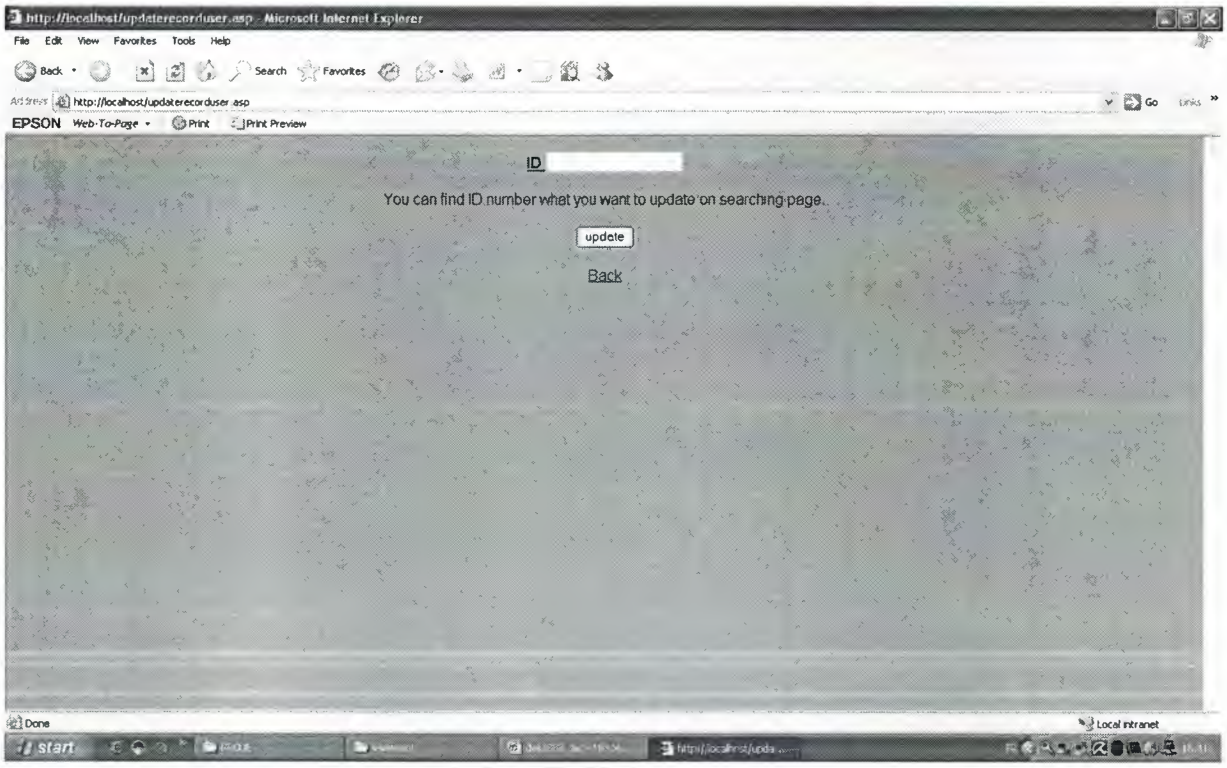
Start Working Date

Status

Department

Deleting Page For Workers

This page for delete the worker details in the database. We have descriptions and fill textfields. And we have submit button. When admin click submit button delete the worker in the database. And also we have back link to turn before of this page. And if your details deleted successfully you see page to your details deleted successfully.


http://localhost/updaterecorduser.asp - Microsoft Internet Explorer
http://localhost/updaterecorduser.asp

ID

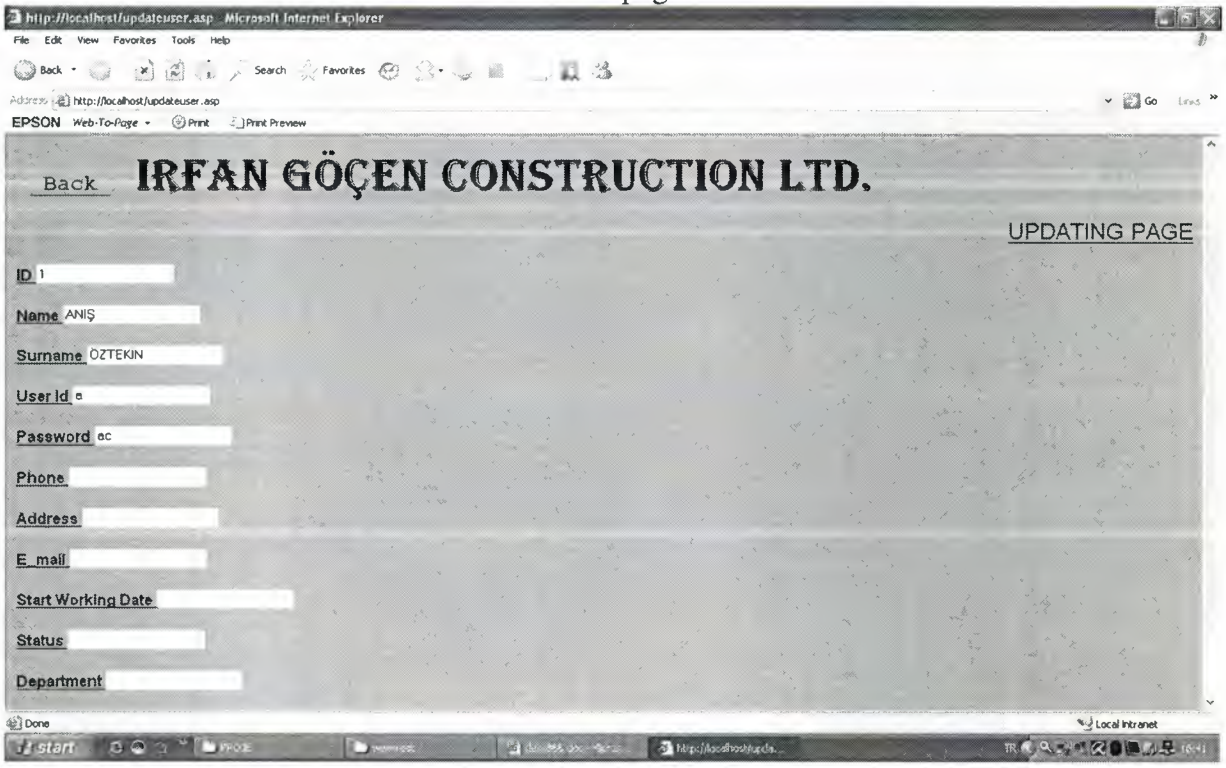
You can find ID number what you want to update on searching page.

update

Back

1. Step To Updating

This page first step to updating. Admin can update every thing about workers. Admin write house ıd in the texfield. Then view below page.


http://localhost/updateuser.asp - Microsoft Internet Explorer
http://localhost/updateuser.asp

Back IRFAN GÖÇEN CONSTRUCTION LTD.

UPDATING PAGE

ID 1

Name ANIŞ

Surname ÖZTEKIN

User Id a

Password ac

Phone

Address

E_mail

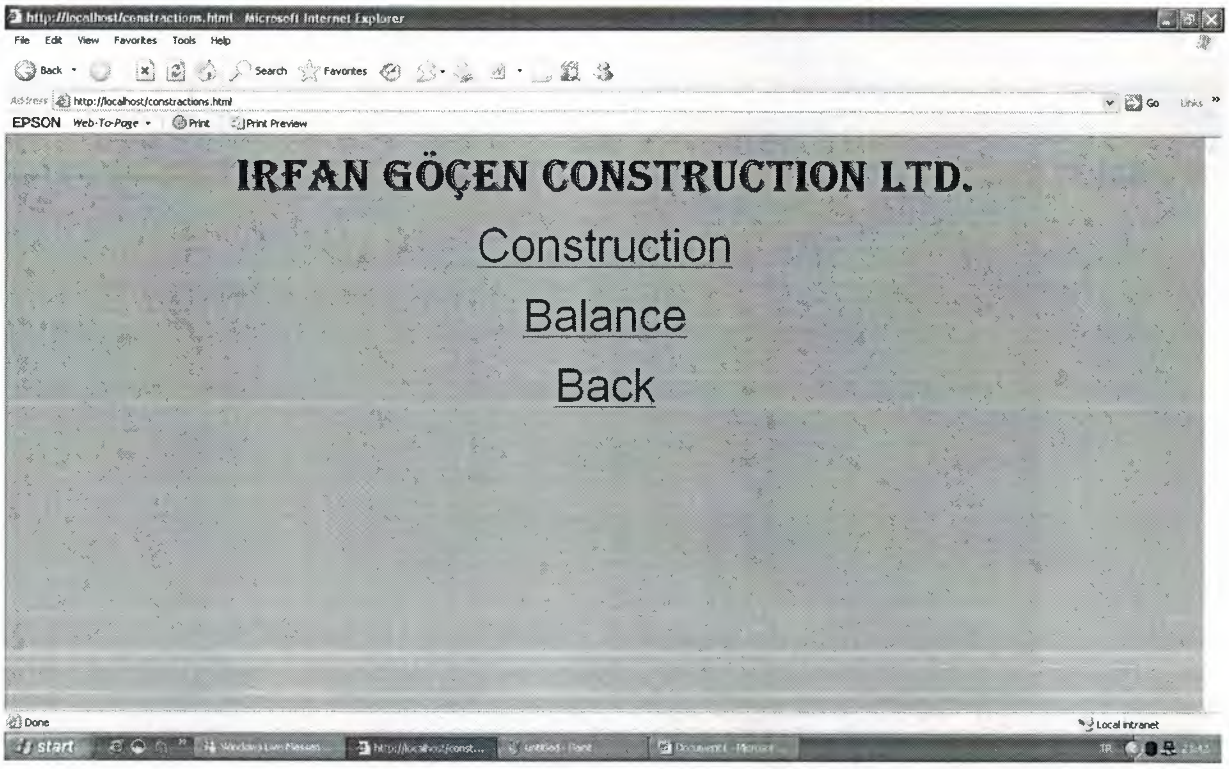
Start Working Date

Status

Department

Updating Page For Workers

This page for update the worker details in the database. We have descriptions and fill textfields. And we have submit button. When admin change in the textfield or textfields then click submit button to update the worker details in the database. And also we have back link to turn before of this page. And if your details updated successfully you see page to your details updated successfully.


IRFAN GÖÇEN CONSTRUCTION LTD.

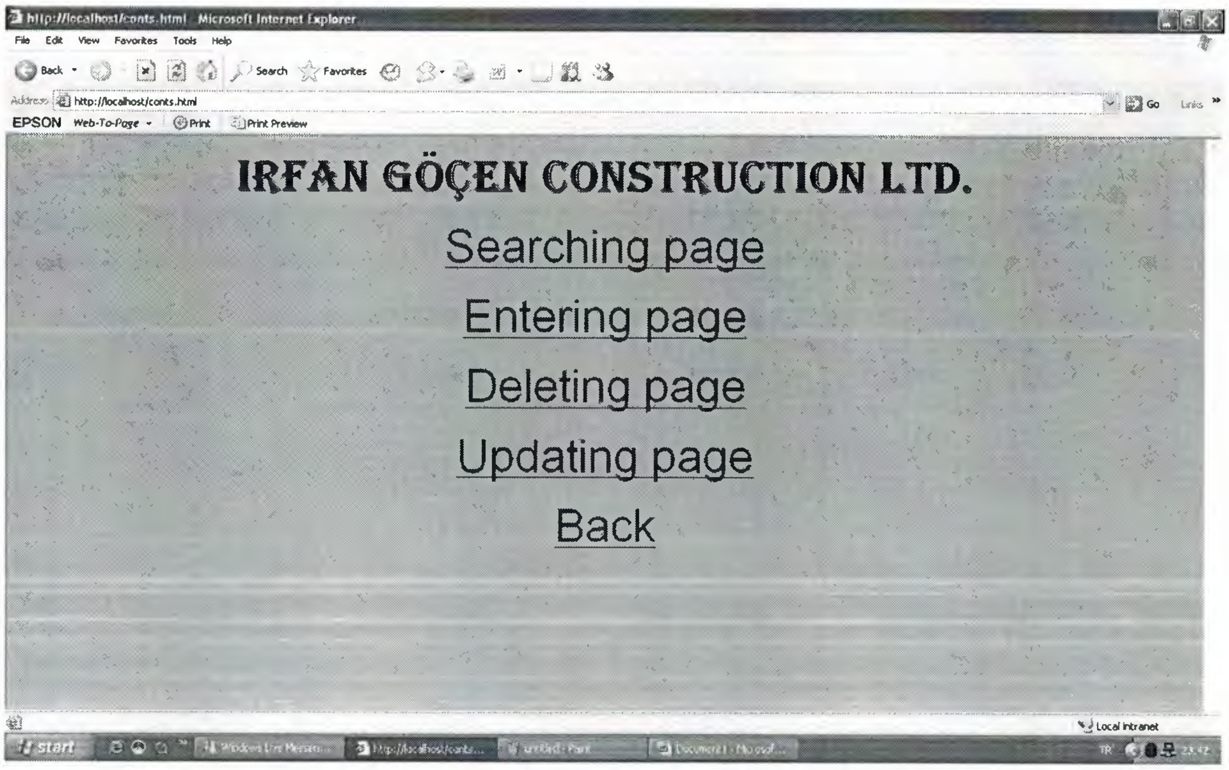
Construction

Balance

Back

Construction page

This page for details of construction. When you click construction link you view this page. And we have 3 links. And I divide construction into two links for every thing about builds of construction and balance for cost of builds. Last link to back before this page.


IRFAN GÖÇEN CONSTRUCTION LTD.

Searching page

Entering page

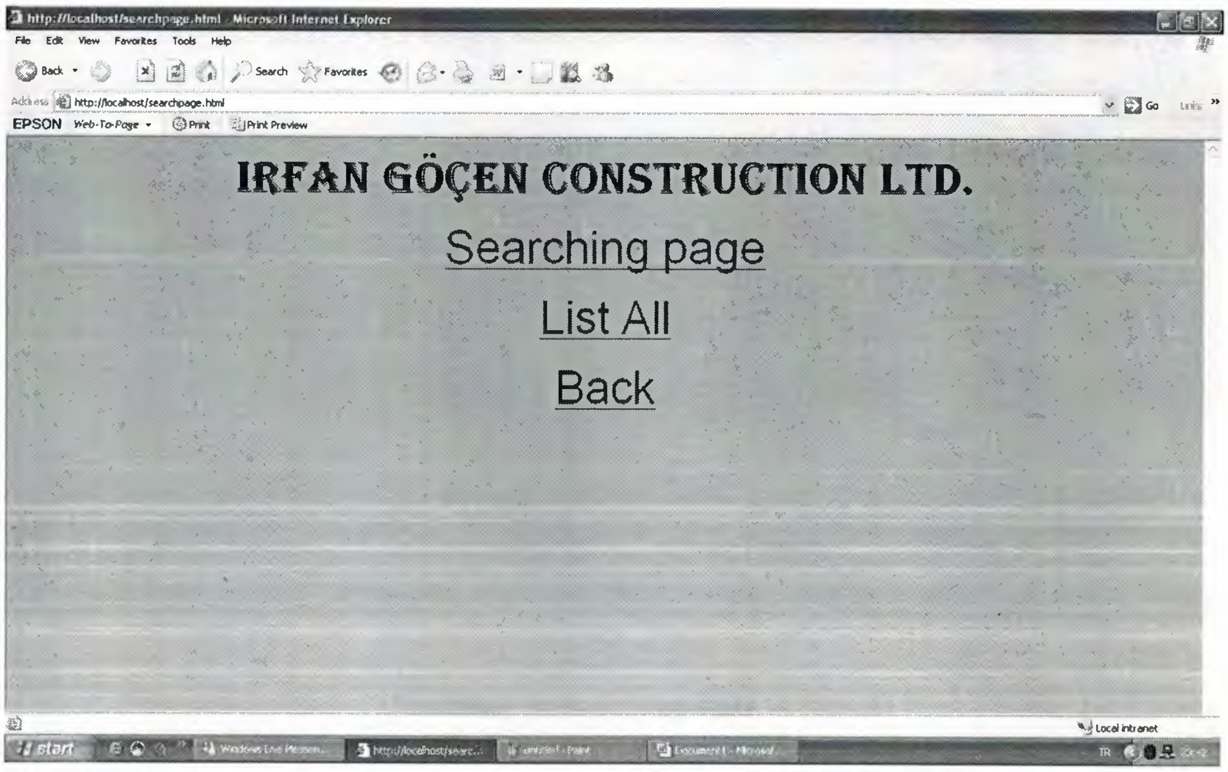
Deleting page

Updating page

Back

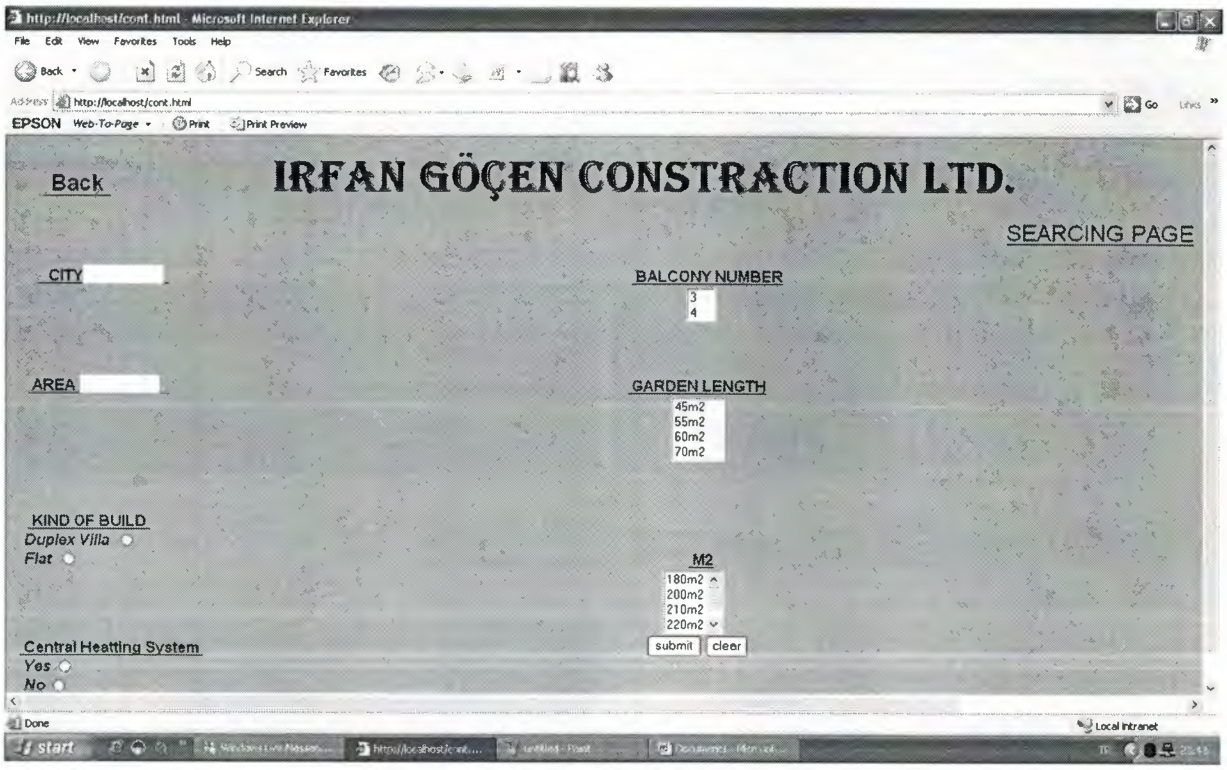
Construction to Construction

When we click construction to construction this page is view. Admin can see this page. And admin can make search, enter, delete and update so we have four links to make search, enter, delete and update. But if user enter the program user just see three links without delete, becouse user can not make delete. And back link to back before of this page.


http://localhost/searchpage.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/searchpage.html
Go Links
EPSON Web-To-Page Print Print Preview
IRFAN GÖÇEN CONSTRUCTION LTD.
Searching page
List All
Back
Local intranet
start

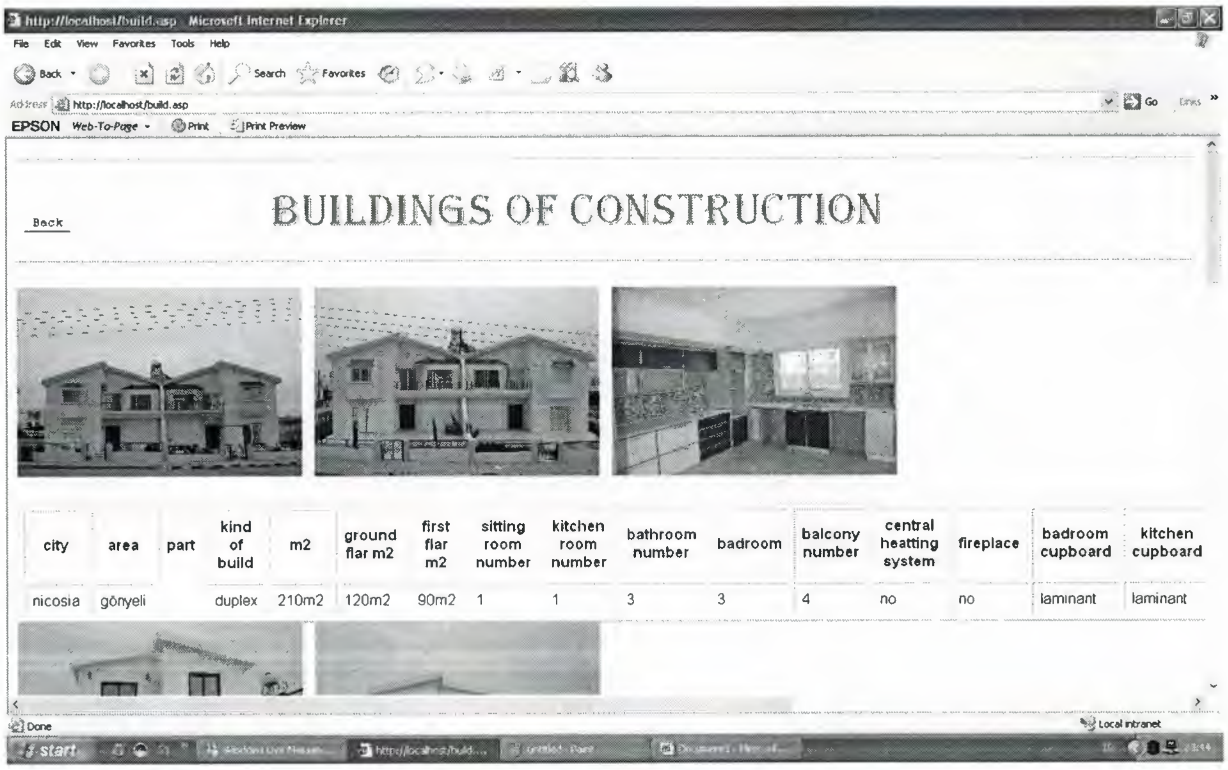
Searhing Page

When clicked searching page we view three links. First two link about build. I divide into two group for searching. And back link to back before of this page.



Searching to Searching Page

When admin or user or guess enter this page. They can search builds we have some details about builds and we can choose how builds we search. we can write city name and area in the textfiled, we choose which kind of build we want buy. If we have central heatting system we select yes section. And we choose balcony number, length of garden and m2 of house. And then we have submit button for see the build about we chooses. And if we click clear button we reset all we chooses. And back link to back before of this page

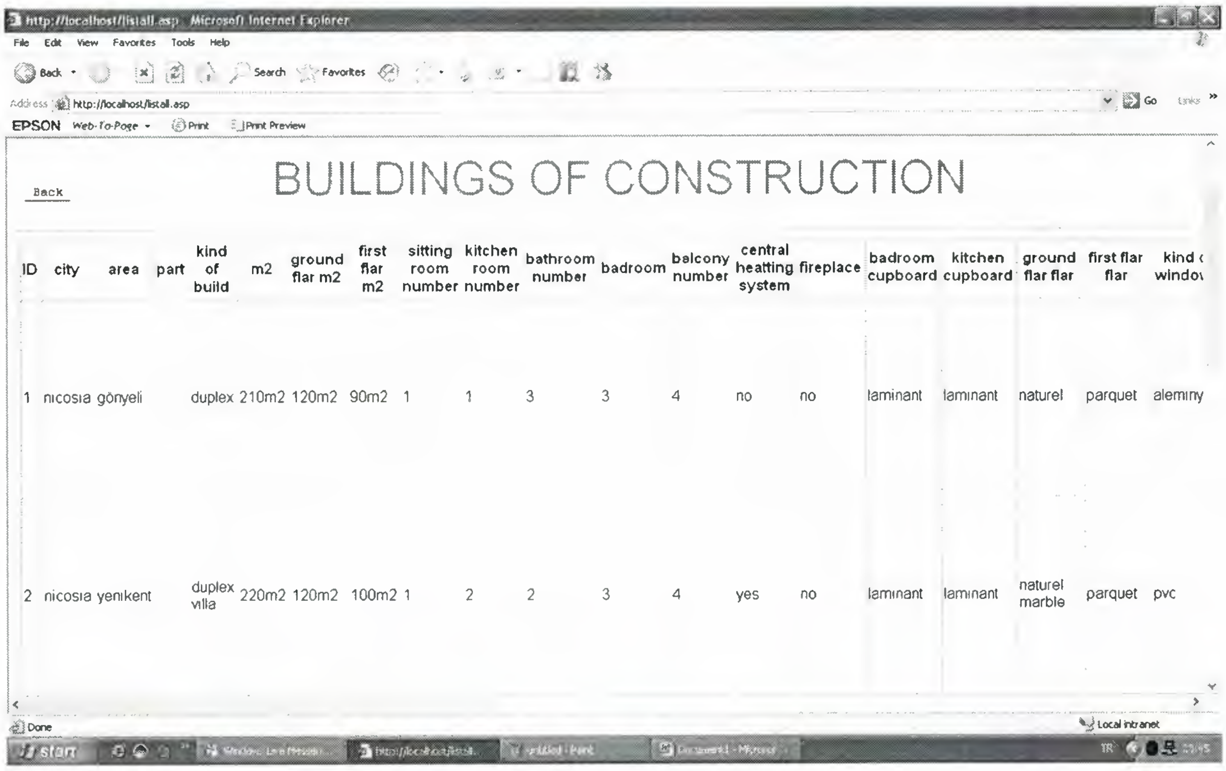

http://localhost/build.asp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address http://localhost/build.asp
Go
Links
EPSON Web-To-Page
Print
Print Preview
Back
Done
Local intranet
start

BUILDINGS OF CONSTRUCTION

city	area	part	kind of build	m2	ground flar m2	first flar m2	sitting room number	kitchen room number	bathroom number	badroom	balcony number	central heatting system	fireplace	badroom cupboard	kitchen cupboard
nicosia	gönyeli		duplex	210m2	120m2	90m2	1	1	3	3	4	no	no	laminant	laminant

Selected Kind of Build

We see house pictures and house details which kind of build we chose.

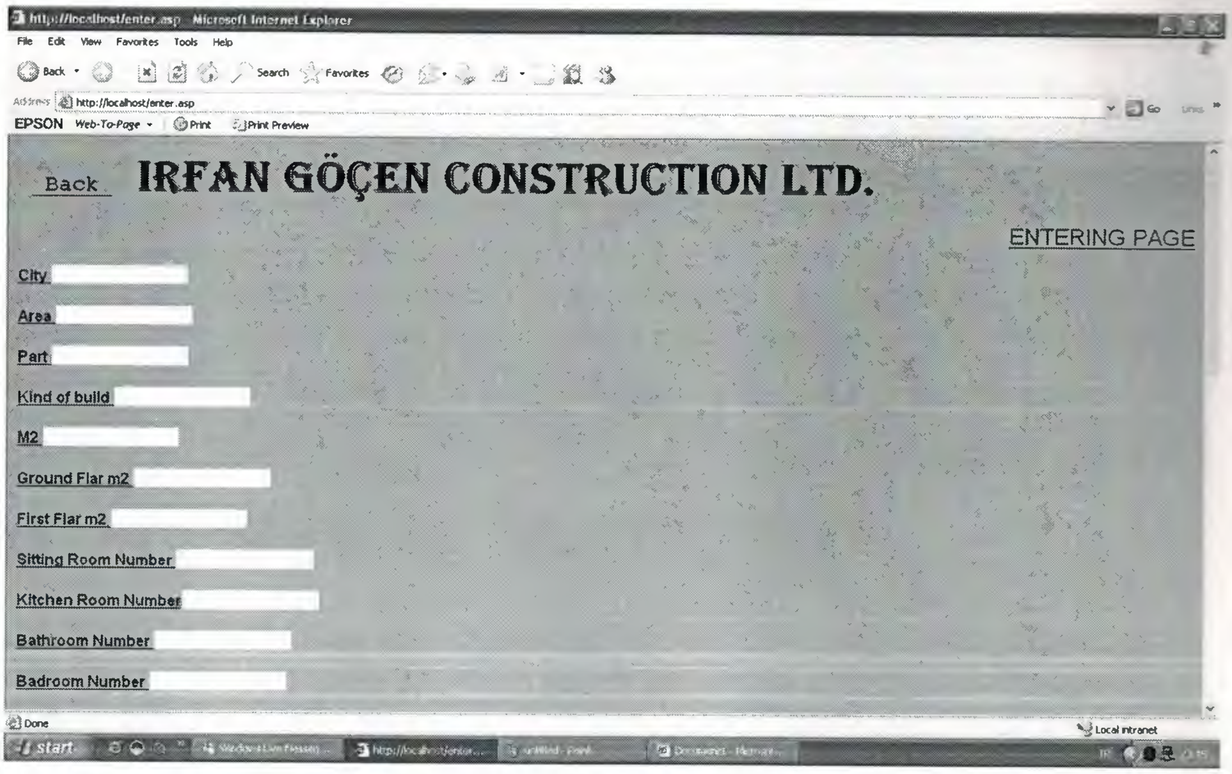

http://localhost/listall.asp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address http://localhost/listall.asp
Go
Links
EPSON Web-To-Page
Print
Print Preview
Back
Done
Local intranet
start

BUILDINGS OF CONSTRUCTION

ID	city	area	part	kind of build	m2	ground flar m2	first flar m2	sitting room number	kitchen room number	bathroom number	badroom	balcony number	central heatting system	fireplace	badroom cupboard	kitchen cupboard	ground flar flar	first flar flar	kind o windo
1	nicosia	gönyeli		duplex	210m2	120m2	90m2	1	1	3	3	4	no	no	laminant	laminant	naturel	parquet	aleminy
2	nicosia	yenikent		duplex villa	220m2	120m2	100m2	1	2	2	3	4	yes	no	laminant	laminant	naturel marble	parquet	pvc

ListAll Page

When click list all links we see evey house what company builded and house details and pictures of build.


Back

IRFAN GÖÇEN CONSTRUCTION LTD.

ENTERING PAGE

City

Area

Part

Kind of build

M2

Ground Flar m2

First Flar m2

Sitting Room Number

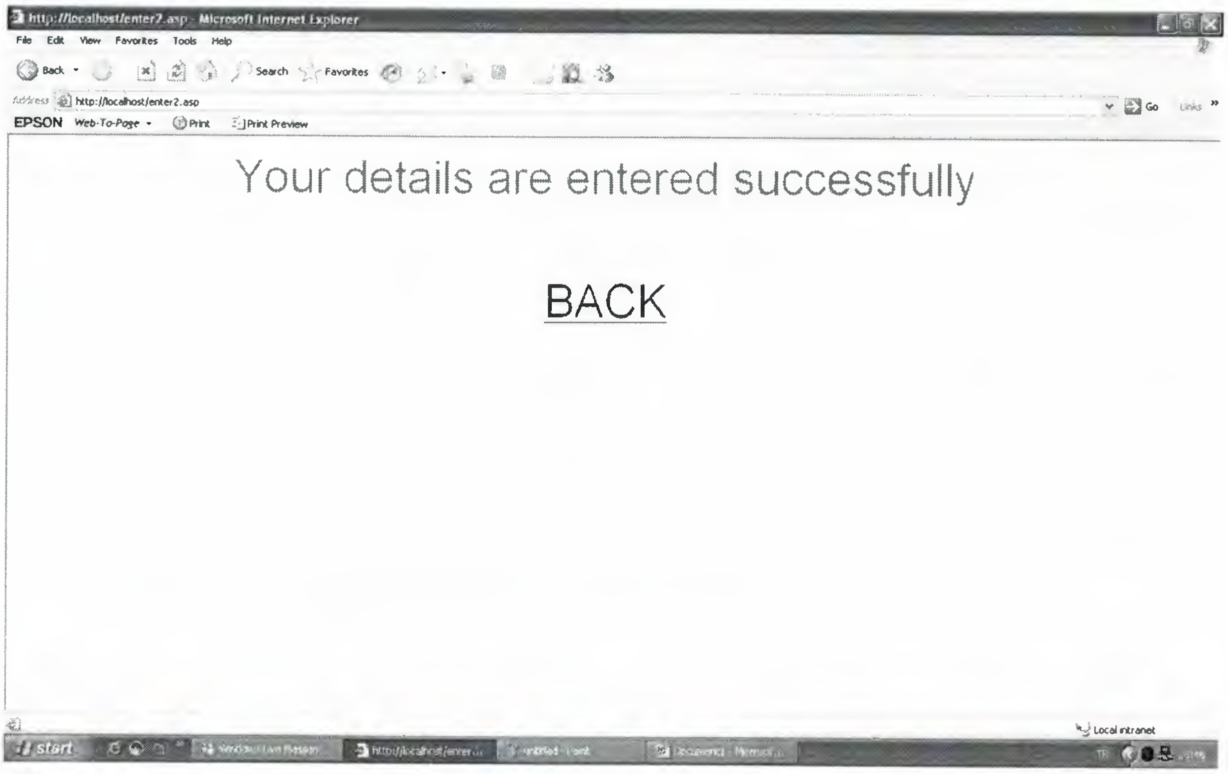
Kitchen Room Number

Bathroom Number

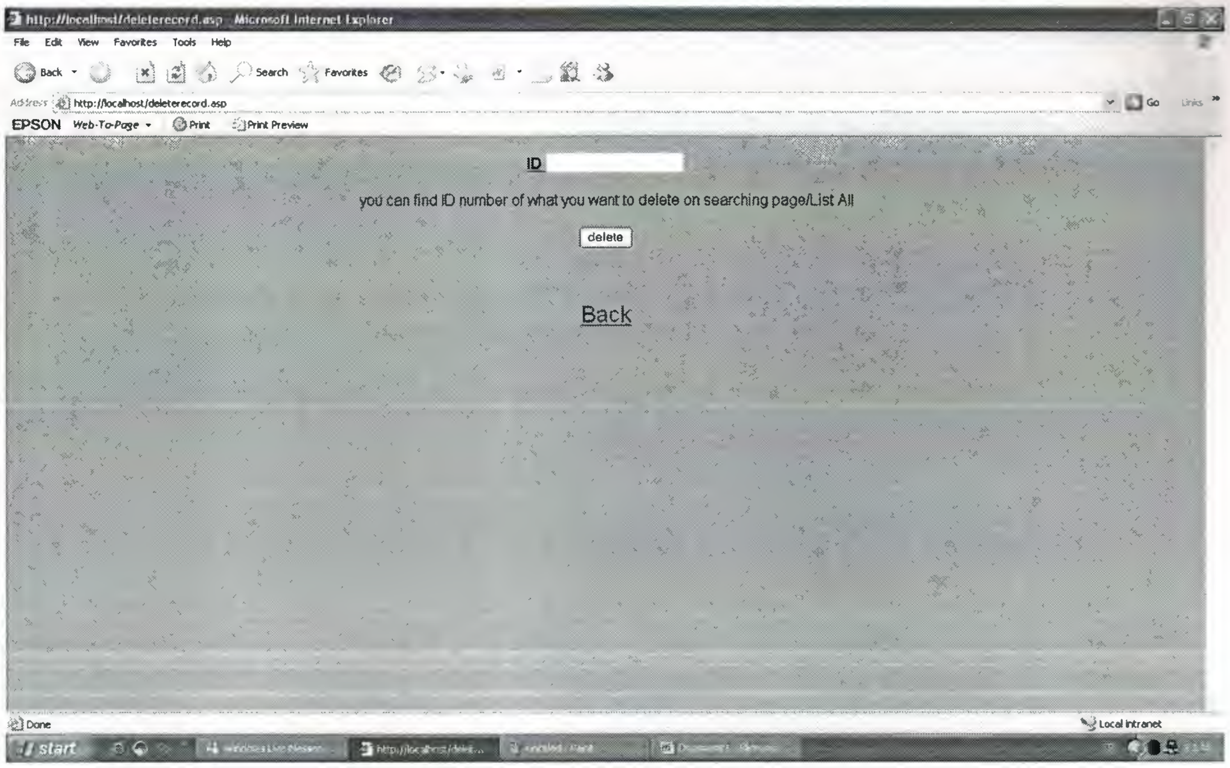
Badroom Number

Entering Page For Builds

This page for enter the house details in the textfield and insert database this details. We have descriptions and textfields to fill them. And we have submit button. we fill in textfields and we have can attach picture when click browse open file chosee and we can select build of picture then click submit button we insert theese details in the database. And also we have back link to turn before of this page. And if your details entered successfully you see as below page.


Your details are entered successfully

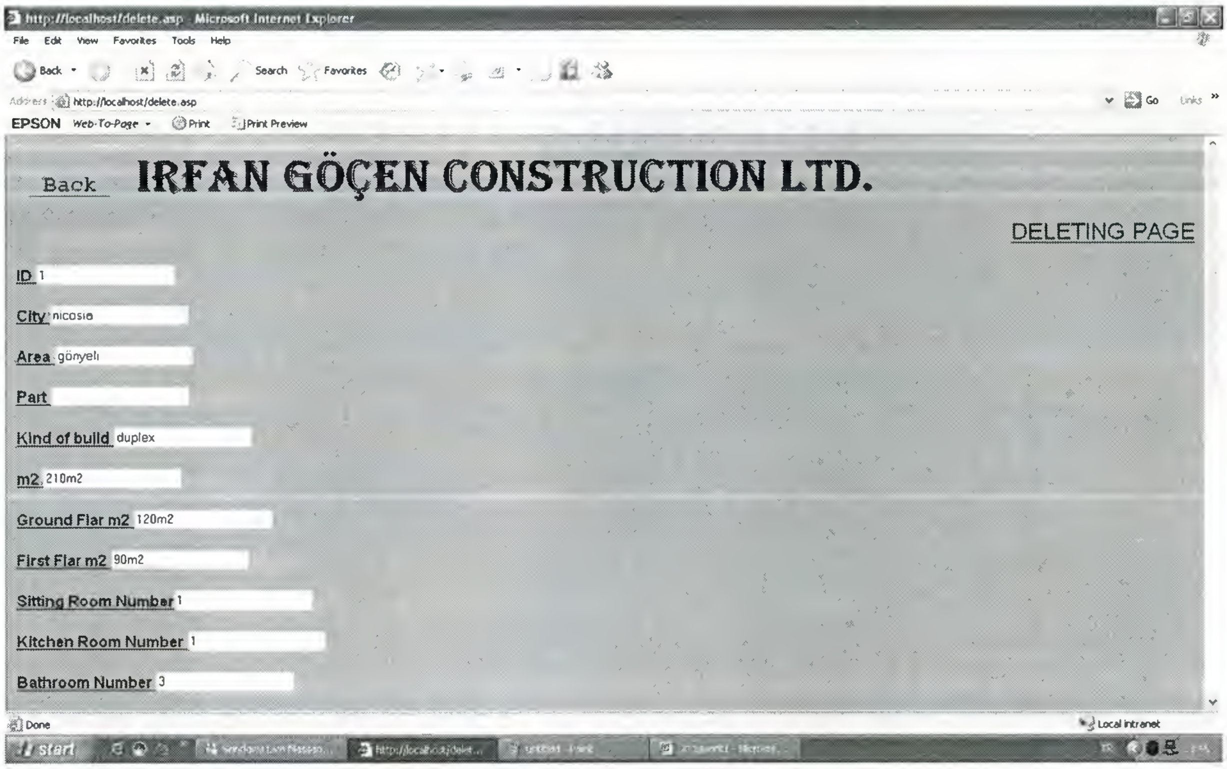
BACK


http://localhost/deleterecord.asp - Microsoft Internet Explorer
ID
delete
Back
Done
Local intranet

you can find ID number of what you want to delete on searching page/List All

1. Step To Deleting

This page first step to deleting. Admin can delete every thing about builds. Admin write house ıd in the texfield. Then view below page.



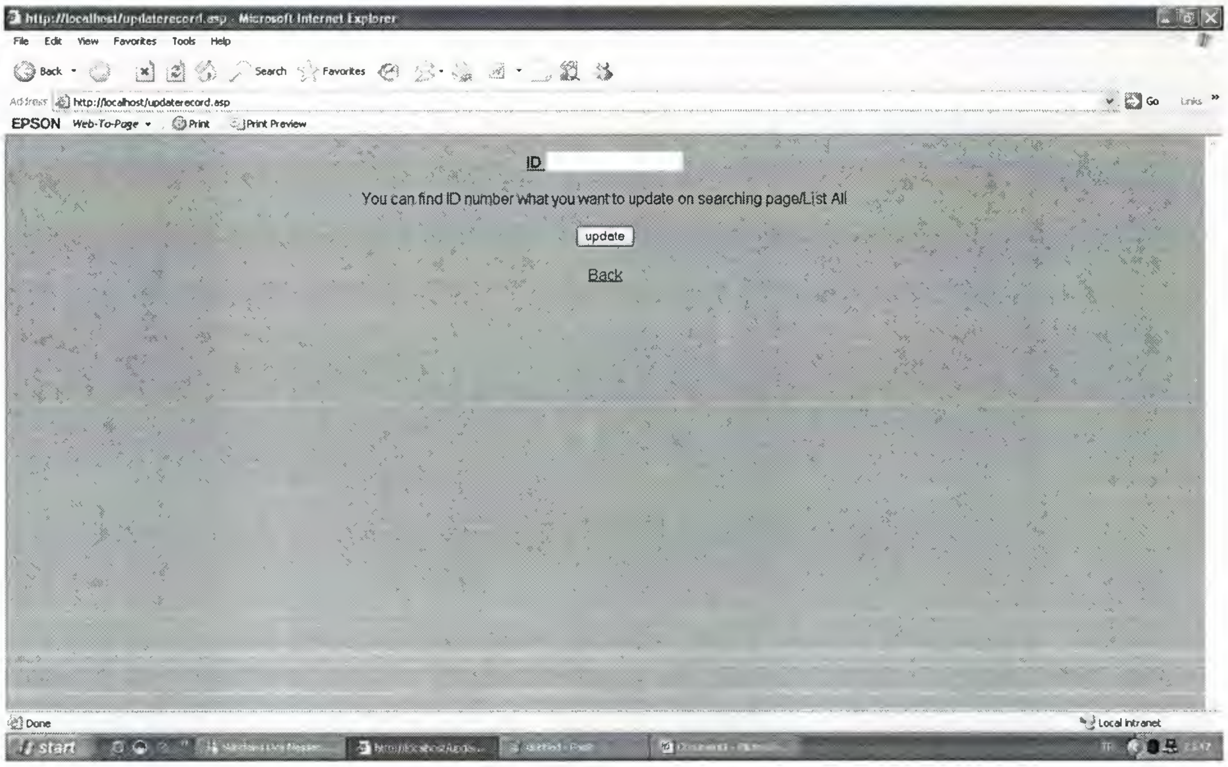
IRFAN GÖÇEN CONSTRUCTION LTD.

DELETING PAGE

ID	1
City	nicosia
Area	gönyeli
Part	
Kind of build	duplex
m2	210m2
Ground Flar m2	120m2
First Flar m2	90m2
Sitting Room Number	1
Kitchen Room Number	1
Bathroom Number	3

Deleting Page For Builds

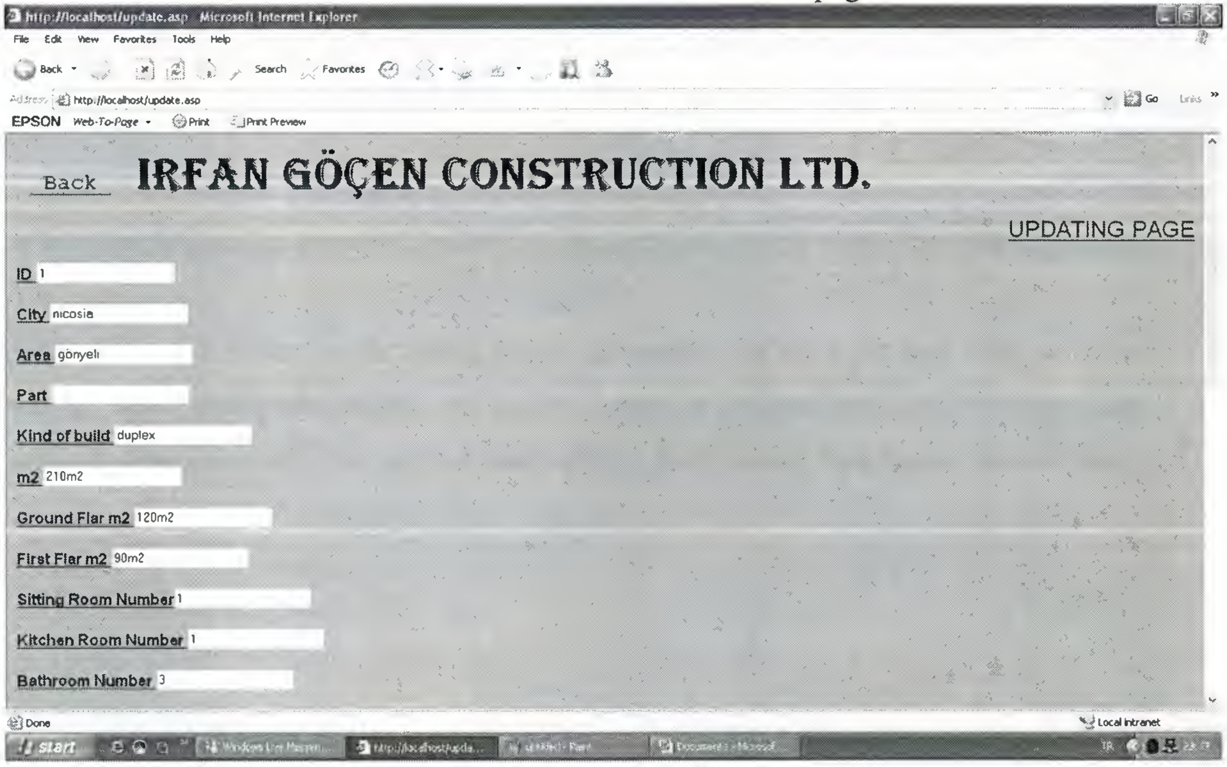
This page for delete the house details in the database. We have descriptions and fill textfields. And we have submit button. When admin click submit button delete the build in the database. And also we have back link to turn before of this page. And if your details deleted successfully you see page to your details deleted successfully.


ID

You can find ID number what you want to update on searching page/List All

1. Step To Updating

This page first step to updating. Admin or user can update every thing about builds. Admin or user write house ıd in the texfield. Then view below page.



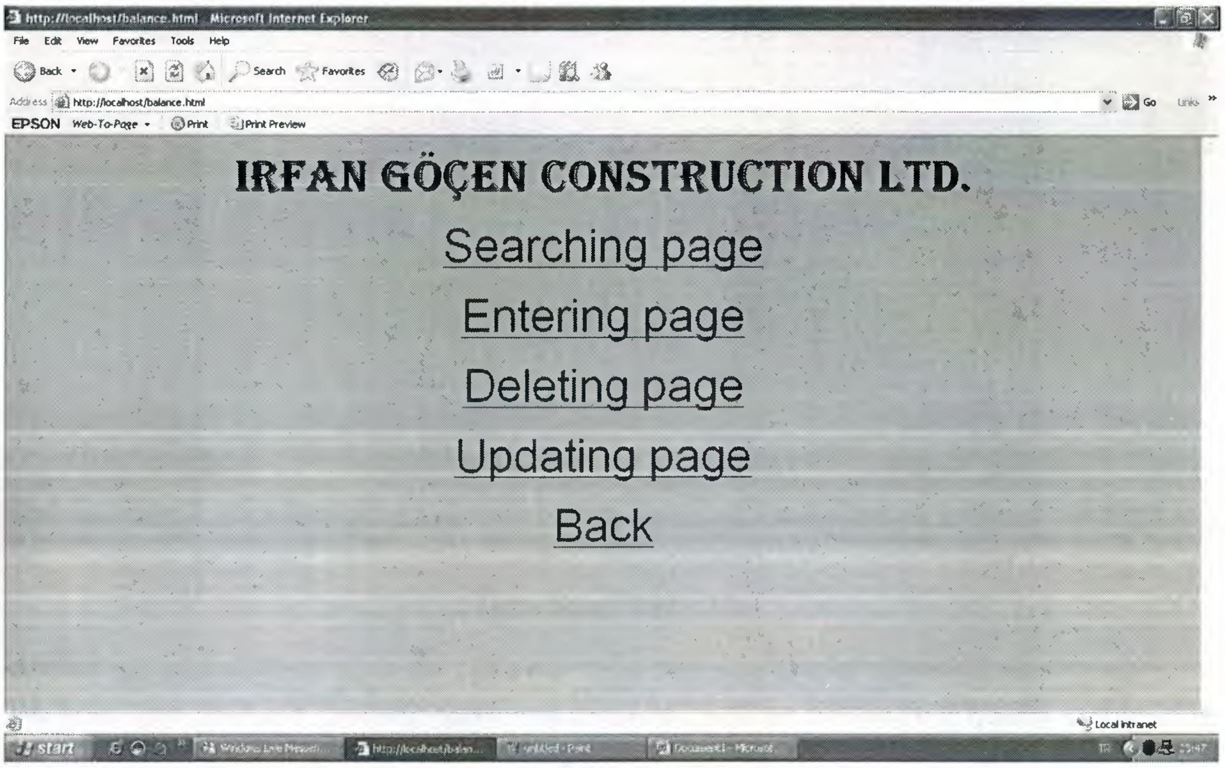
IRFAN GÖÇEN CONSTRUCTION LTD.

UPDATING PAGE

ID	1
City	nıcosia
Area	gönyelı
Part	
Kind of build	duplex
m2	210m2
Ground Flar m2	120m2
First Flar m2	90m2
Sitting Room Number	1
Kitchen Room Number	1
Bathroom Number	3

Updating Page For Builds

This page for update the house details in the database. We have descriptions and fill textfields. And we have submit button. When admin or user change in the textfield or textfields then click submit button to update the build details in the database. And also we have back link to turn before of this page. And if your details updated successfully you see page to your details updated successfully.


IRFAN GÖÇEN CONSTRUCTION LTD.

Searching page

Entering page

Deleting page

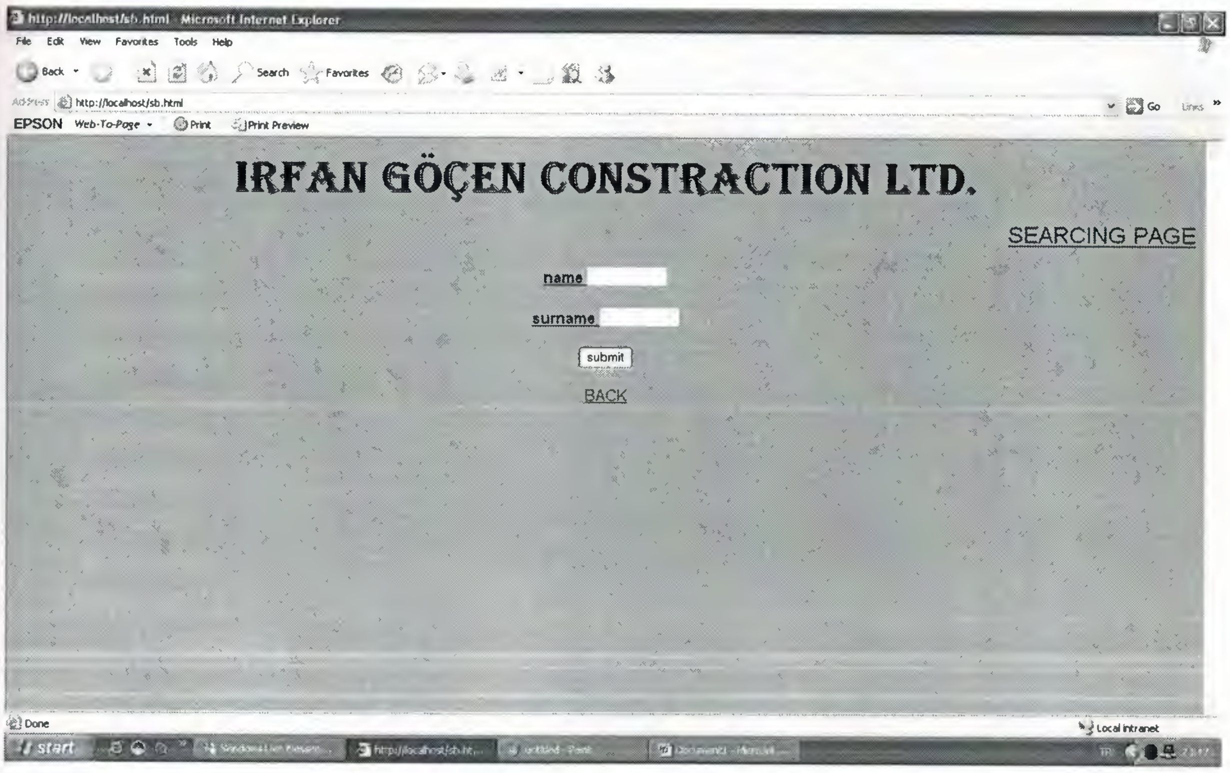
Updating page

Back

Construction to Balance

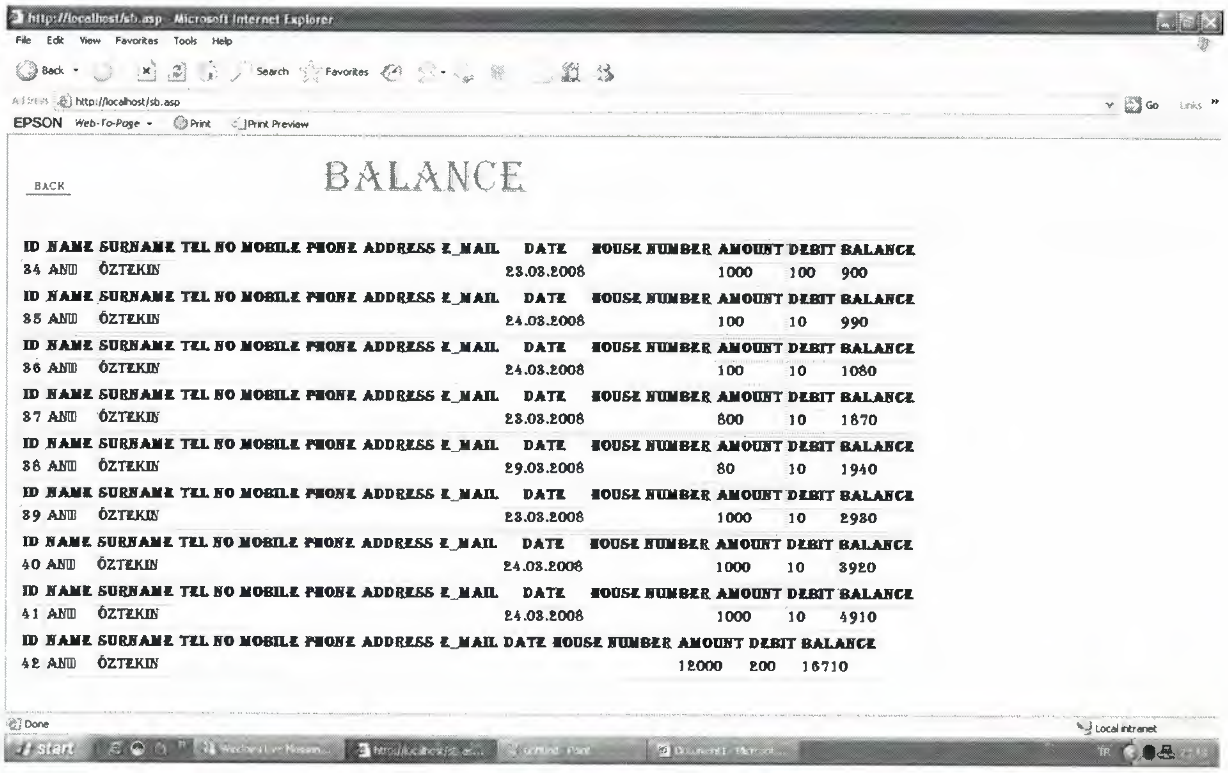
When we click on the construction link to balance this page is view. The Admin can see this page.

And admin can make search, enter, delete and update so we have four links to make search, enter, delete and update. But if user enter the program user just see three links without delete, becouse user can not make delete. Back link to back before of this page.


http://localhost/sb.html - Microsoft Internet Explorer
IRFAN GÖÇEN CONSTRACTION LTD.
SEARCING PAGE
name
surname
submit
BACK

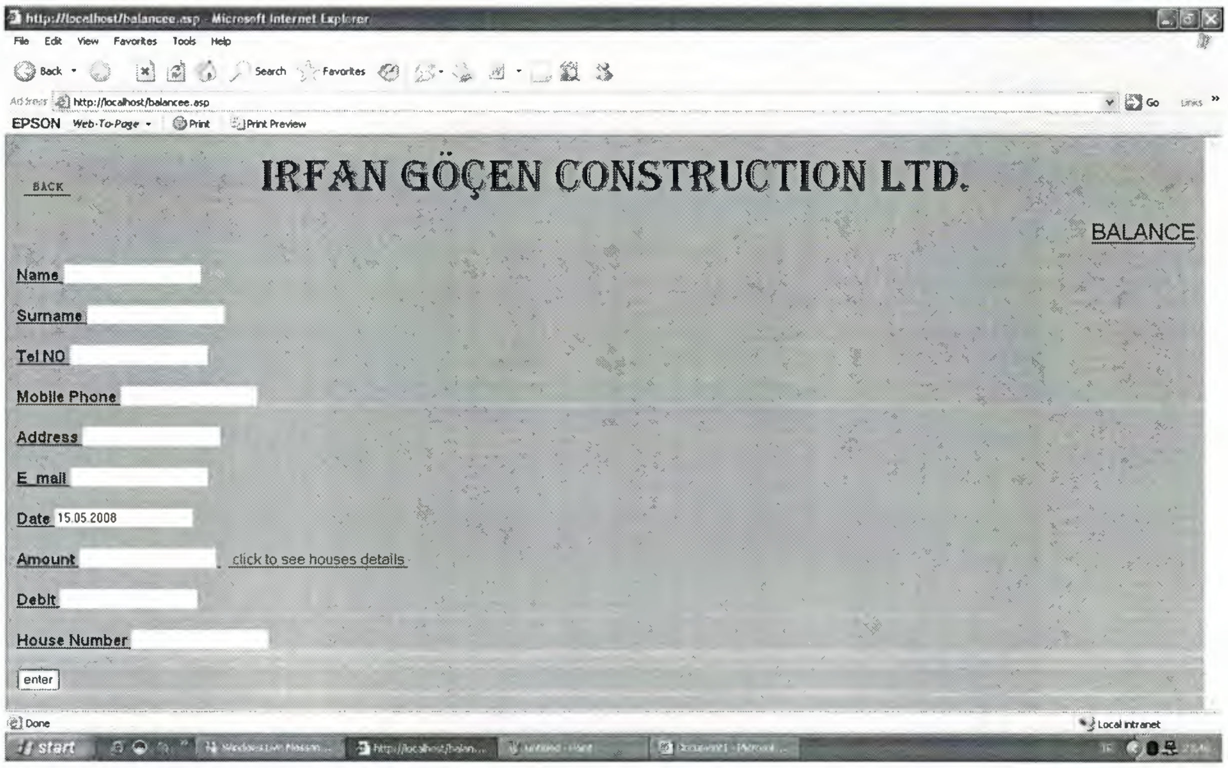
Searching Page

When clicked searching page we view this page. We can search with who buy builds. We have two textfield for enter name and surname. And we fill theese textfield to start search. When we fill textfields and click submit button we see below page to this customer which house buyed and what he/she paid and when and what the balance. We can see theese details becouse of database. And back link to back before of this page.



BALANCE

ID	NAME	SURNAME	TEL NO	MOBILE PHONE	ADDRESS	E_MAIL	DATE	HOUSE NUMBER	AMOUNT	DEBIT	BALANCE
34	ANIL	ÖZTEKIN					23.03.2008		1000	100	900
ID	NAME	SURNAME	TEL NO	MOBILE PHONE	ADDRESS	E_MAIL	DATE	HOUSE NUMBER	AMOUNT	DEBIT	BALANCE
35	ANIL	ÖZTEKIN					24.03.2008		100	10	990
ID	NAME	SURNAME	TEL NO	MOBILE PHONE	ADDRESS	E_MAIL	DATE	HOUSE NUMBER	AMOUNT	DEBIT	BALANCE
36	ANIL	ÖZTEKIN					24.03.2008		100	10	1080
ID	NAME	SURNAME	TEL NO	MOBILE PHONE	ADDRESS	E_MAIL	DATE	HOUSE NUMBER	AMOUNT	DEBIT	BALANCE
37	ANIL	ÖZTEKIN					23.03.2008		800	10	1870
ID	NAME	SURNAME	TEL NO	MOBILE PHONE	ADDRESS	E_MAIL	DATE	HOUSE NUMBER	AMOUNT	DEBIT	BALANCE
38	ANIL	ÖZTEKIN					29.03.2008		80	10	1940
ID	NAME	SURNAME	TEL NO	MOBILE PHONE	ADDRESS	E_MAIL	DATE	HOUSE NUMBER	AMOUNT	DEBIT	BALANCE
39	ANIL	ÖZTEKIN					23.03.2008		1000	10	2930
ID	NAME	SURNAME	TEL NO	MOBILE PHONE	ADDRESS	E_MAIL	DATE	HOUSE NUMBER	AMOUNT	DEBIT	BALANCE
40	ANIL	ÖZTEKIN					24.03.2008		1000	10	3920
ID	NAME	SURNAME	TEL NO	MOBILE PHONE	ADDRESS	E_MAIL	DATE	HOUSE NUMBER	AMOUNT	DEBIT	BALANCE
41	ANIL	ÖZTEKIN					24.03.2008		1000	10	4910
ID	NAME	SURNAME	TEL NO	MOBILE PHONE	ADDRESS	E_MAIL	DATE	HOUSE NUMBER	AMOUNT	DEBIT	BALANCE
42	ANIL	ÖZTEKIN							12000	200	16710


BACK

IRFAN GÖÇEN CONSTRUCTION LTD.

BALANCE

Name

Surname

Tel NO

Mobile Phone

Address

E_mail

Date 15.05.2008

Amount click to see houses details

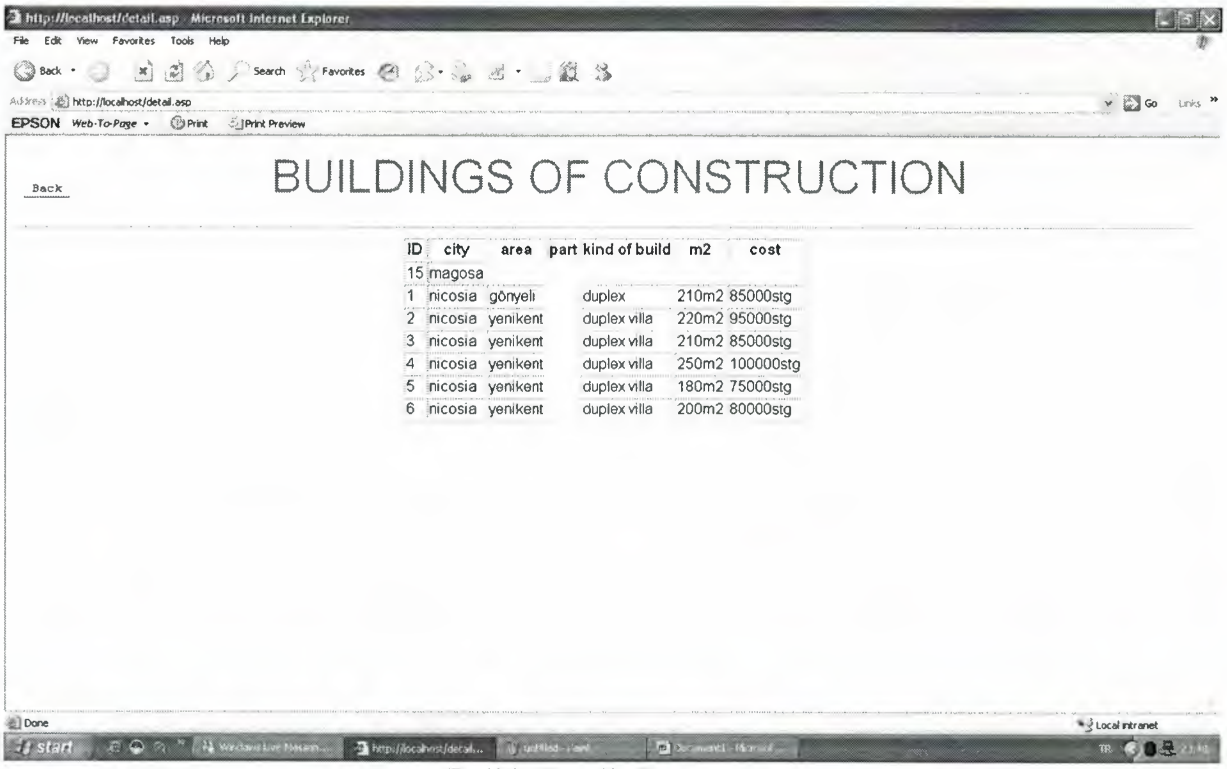
Debit

House Number

enter

Entering Page For Balance

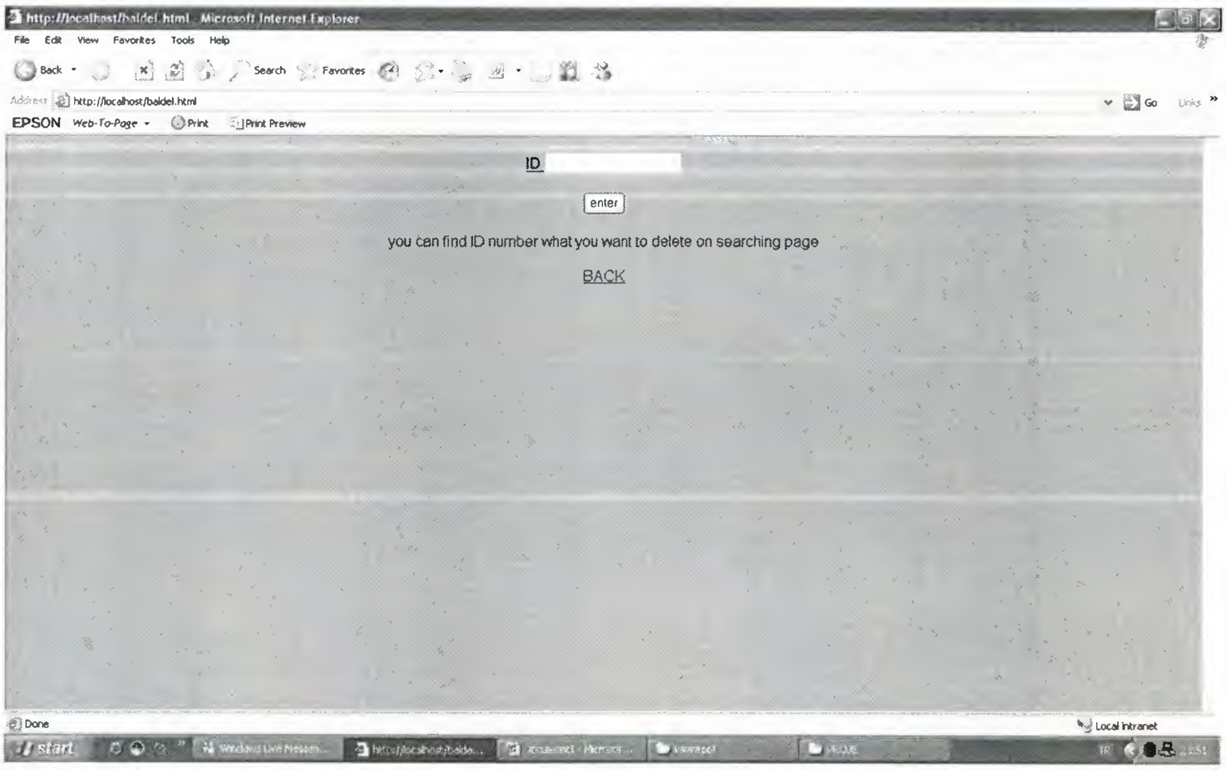
This page for enter the balance details in the textfield and insert database this details. We have descriptions and textfields to fill them. And we have submit button.we have date description and automatic writen day date in the textfield. we fill in textfields then click submit button we insert theese details in the database. And also we have back link to turn before of this page. And if your details entered successfully you see detail page to inset be successfully.


http://localhost/detail.asp - Microsoft Internet Explorer
Back
BUILDINGS OF CONSTRUCTION

ID	city	area	part	kind of build	m2	cost
15	magosa					
1	nicosia	gönyeli		duplex	210m2	85000stg
2	nicosia	yenikent		duplex villa	220m2	95000stg
3	nicosia	yenikent		duplex villa	210m2	85000stg
4	nicosia	yenikent		duplex villa	250m2	100000stg
5	nicosia	yenikent		duplex villa	180m2	75000stg
6	nicosia	yenikent		duplex villa	200m2	80000stg

Builds Details Page

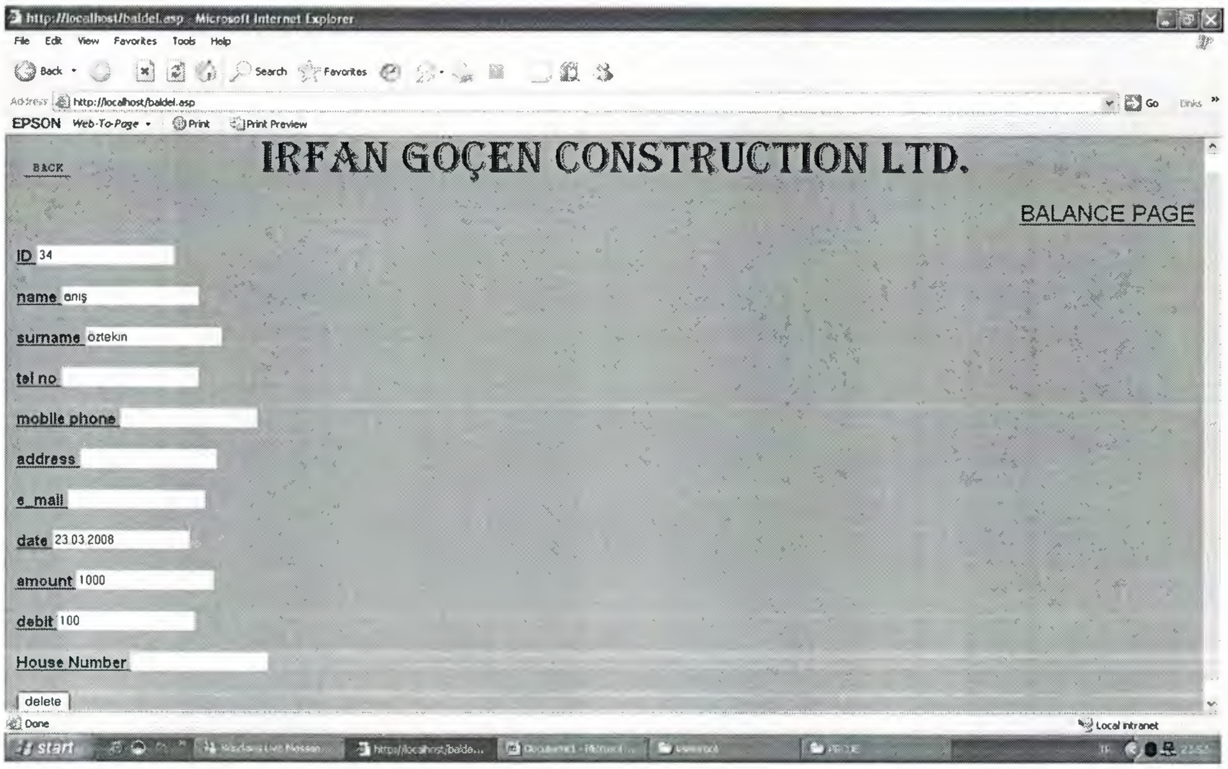
We have link in the inset page and when we click link we see this page to see builds details and we see house ıd and we can fill textfield which house buyed.



you can find ID number what you want to delete on searching page

1. Step To Deleting

This page first step to deleting. Admin can delete every thing about customers. Admin write customer ıd in the texfield. Then view below page.



http://localhost/baldel.asp - Microsoft Internet Explorer

BACK

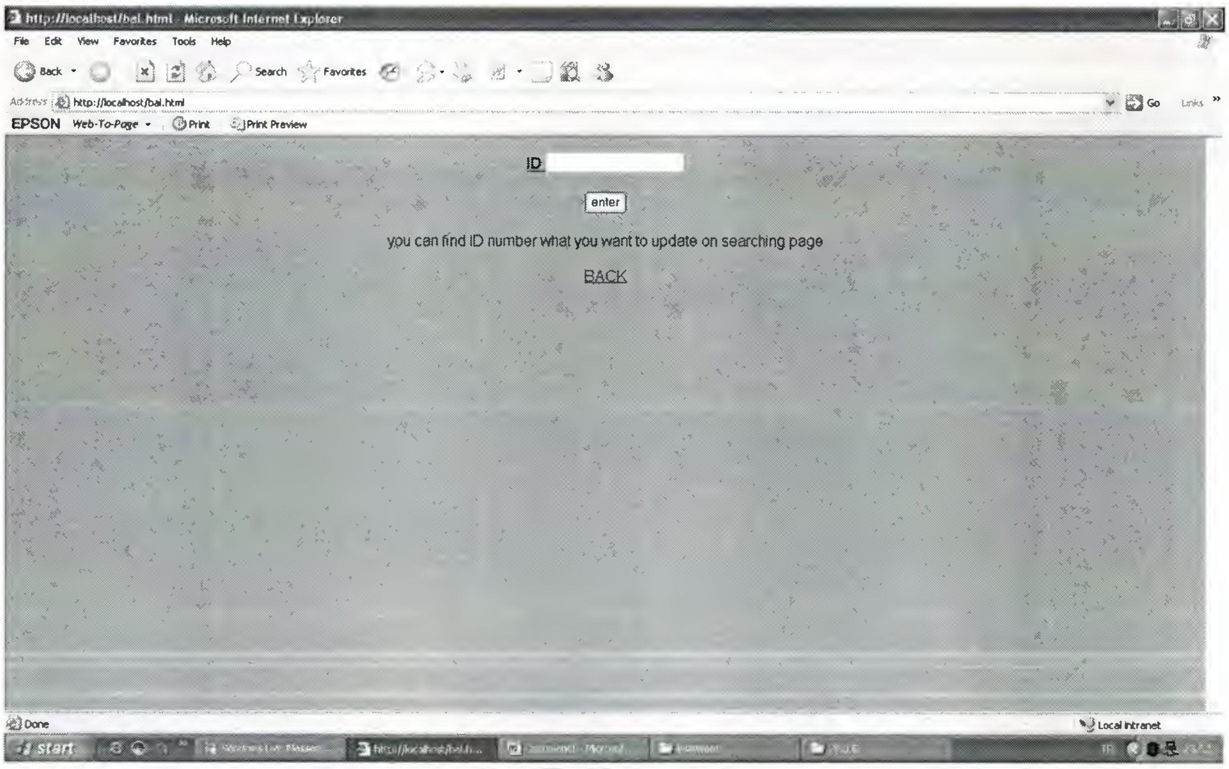
IRFAN GOÇEN CONSTRUCTION LTD.

BALANCE PAGE

ID	34
name	anış
surname	oztekin
tel no	
mobile phone	
address	
e_mail	
date	23.03.2008
amount	1000
debit	100
House Number	

Deleting Page For Balance

This page for delete the customer details in the database. We have descriptions and fill textfields. And we have submit button. When admin click submit button delete the customer in the database. And also we have back link to turn before of this page. And if your details deleted successfully you see page to your details deleted successfully.


ID

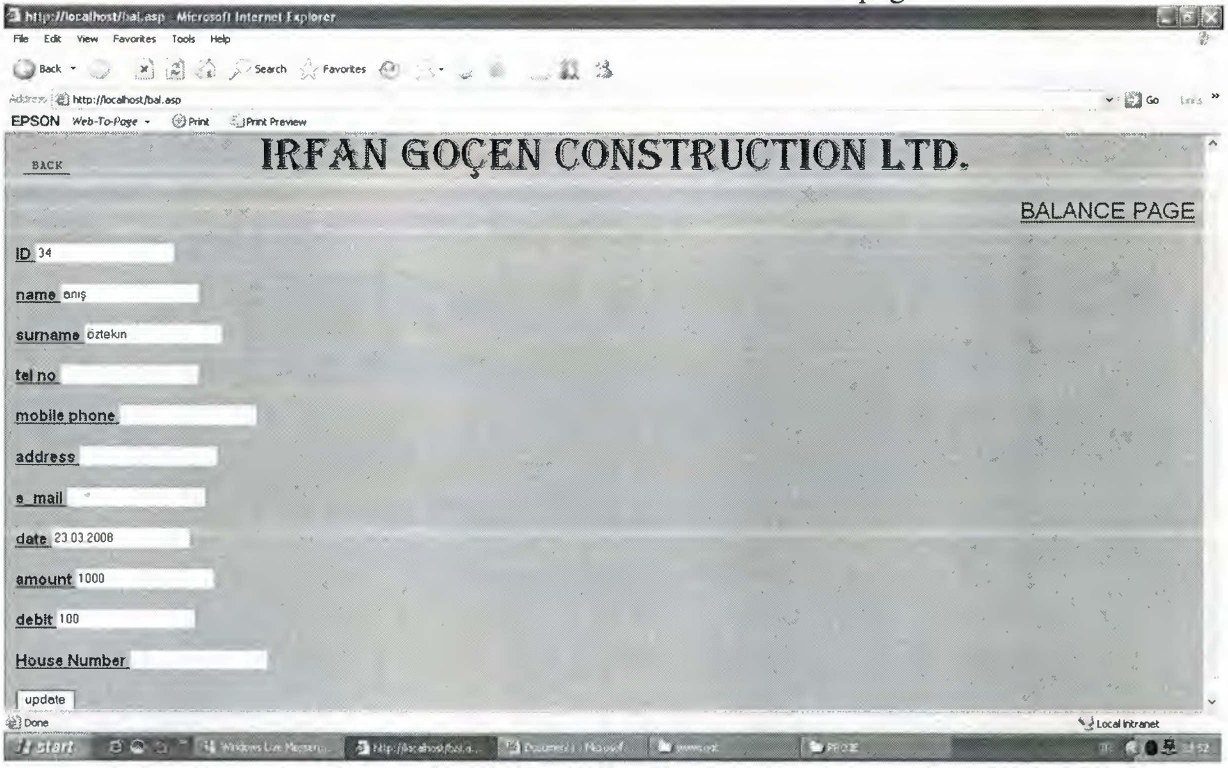
enter

you can find ID number what you want to update on searching page

BACK

1. Step To Updating

This page first step to updating. Admin or user can update every thing about customer. Admin or user write customer ıd in the texfield. Then view below page.


BACK

IRFAN GOÇEN CONSTRUCTION LTD.

BALANCE PAGE

ID 34

name anış

surname öztekın

tel no

mobile phone

address

e_mail

date 23.03.2008

amount 1000

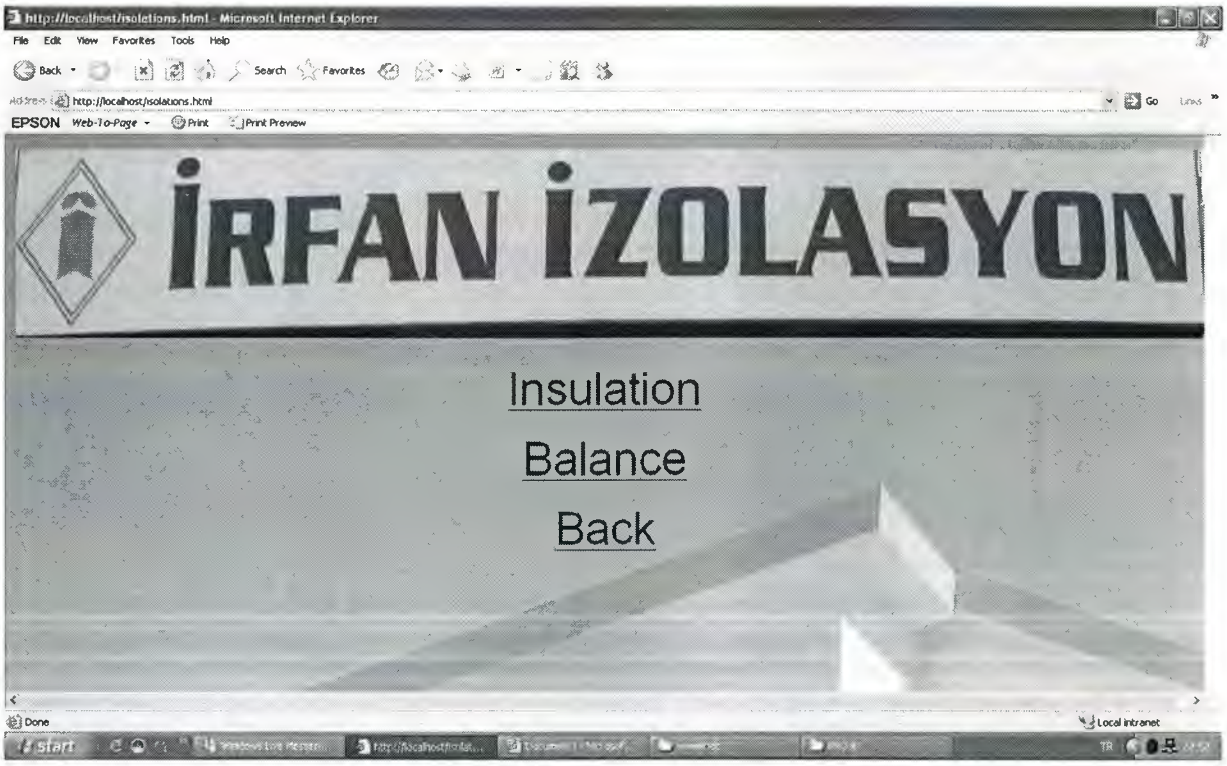
debit 100

House Number

update

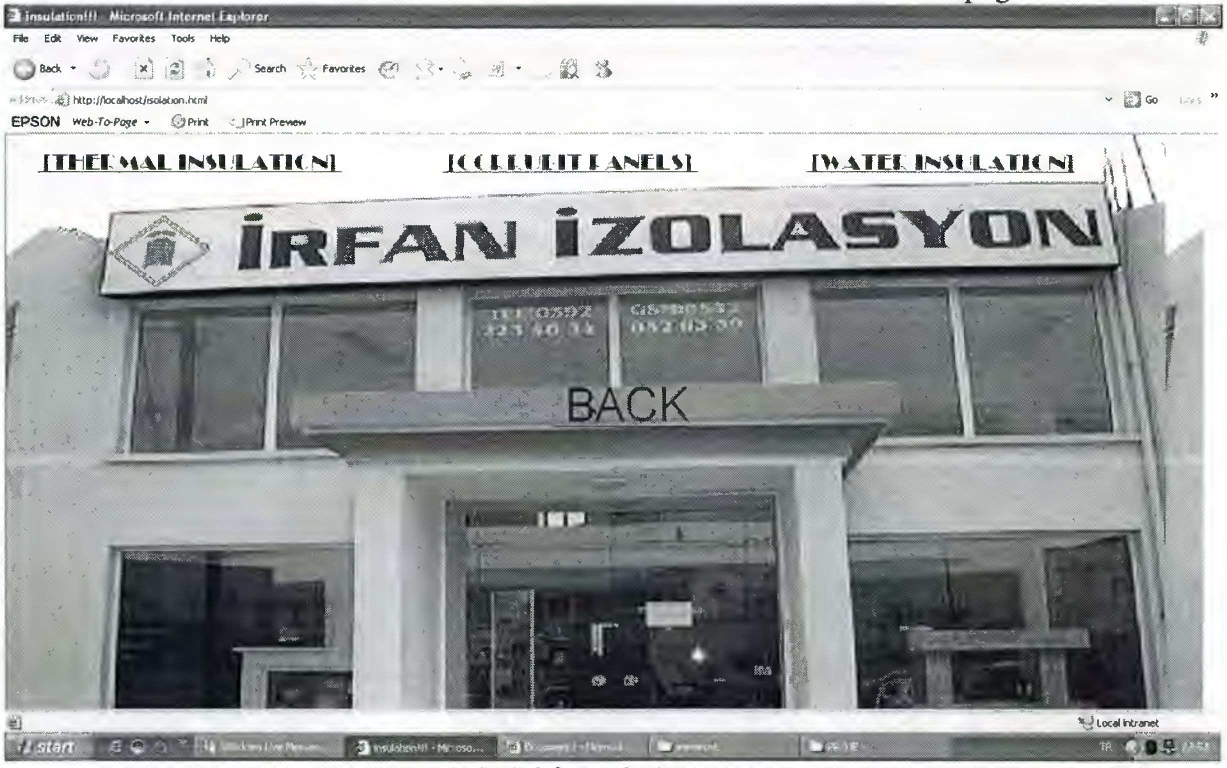
Updating Page For Balance

This page for update the customer details in the database. We have descriptions and fill textfields. And we have submit button. When admin or user change in the textfield or textfields then click submit button update the customer details in the database. And also we have back link to turn before of this page. And if your details updated successfully you see page to your details updated successfully.


http://localhost/isolations.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
http://localhost/isolations.html
İRFAN İZOLASYON
Insulation
Balance
Back
Done
Local intranet

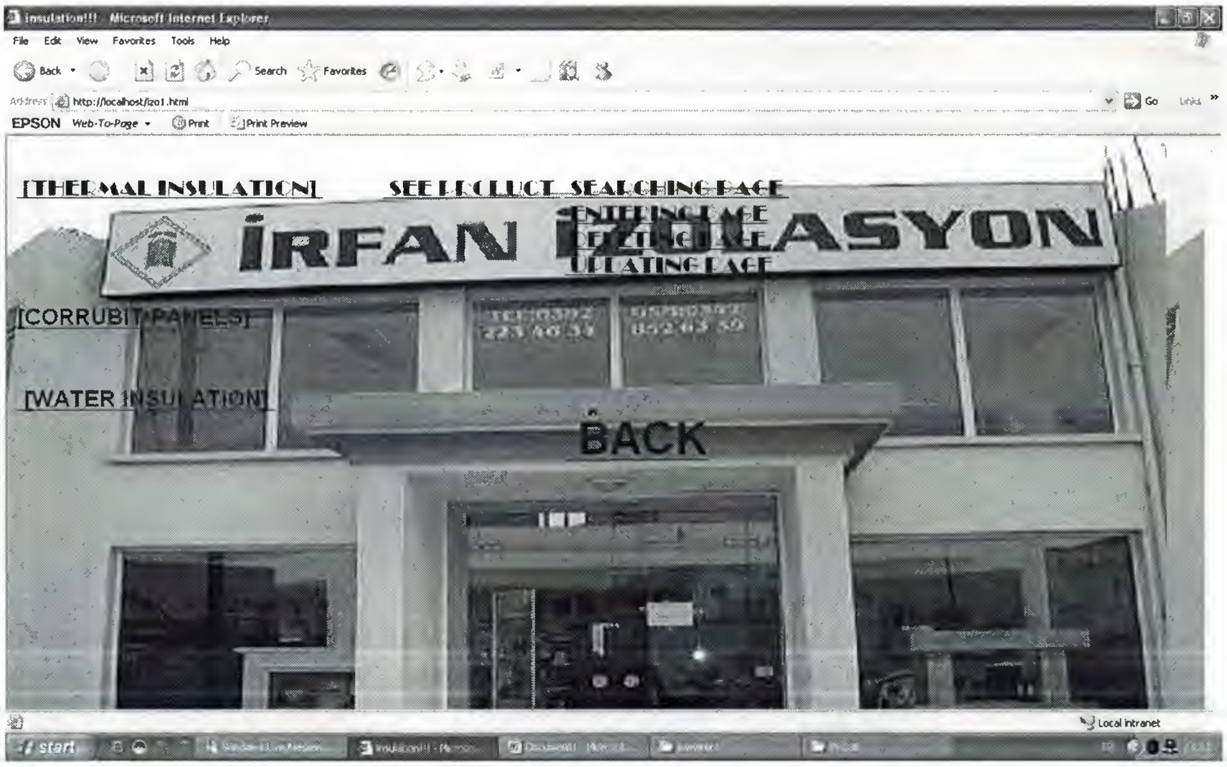
Insulation page

This page for details of insulation. When you click insulation link you view this page. And we have three links. And I divide insulation into two links for every thing about insulation and balance for cost of insulations. Last link to back before this page.


Insulation!!! - Microsoft Internet Explorer
File Edit View Favorites Tools Help
http://localhost/isolation.html
[THERMAL INSULATION]
[CORRUBIT PANELS]
[WATER INSULATION]
İRFAN İZOLASYON
BACK
Local intranet

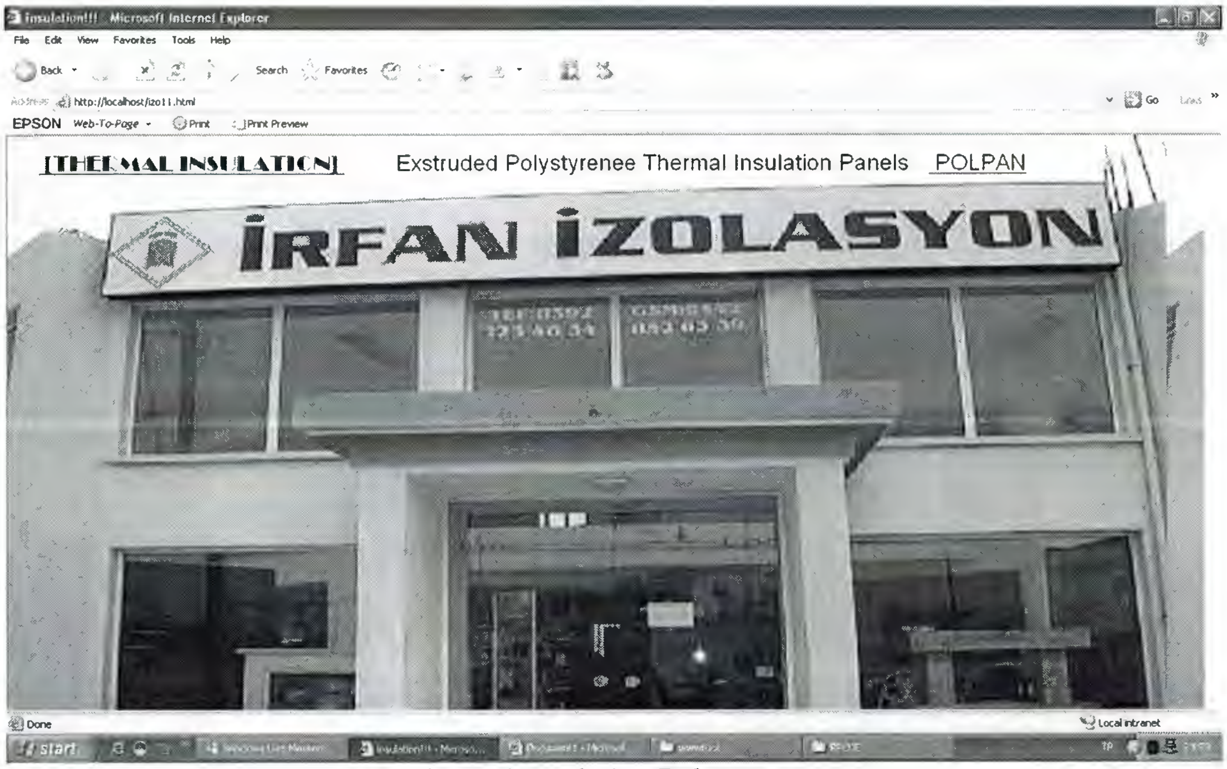
Kind Of Insulation

When clicked insulation link admin, user and guesst can see this page. We have three kind of insulation. So we have three link to thermal insulation, corrubit panels and water insulation. And also we have back link to turn before this page.



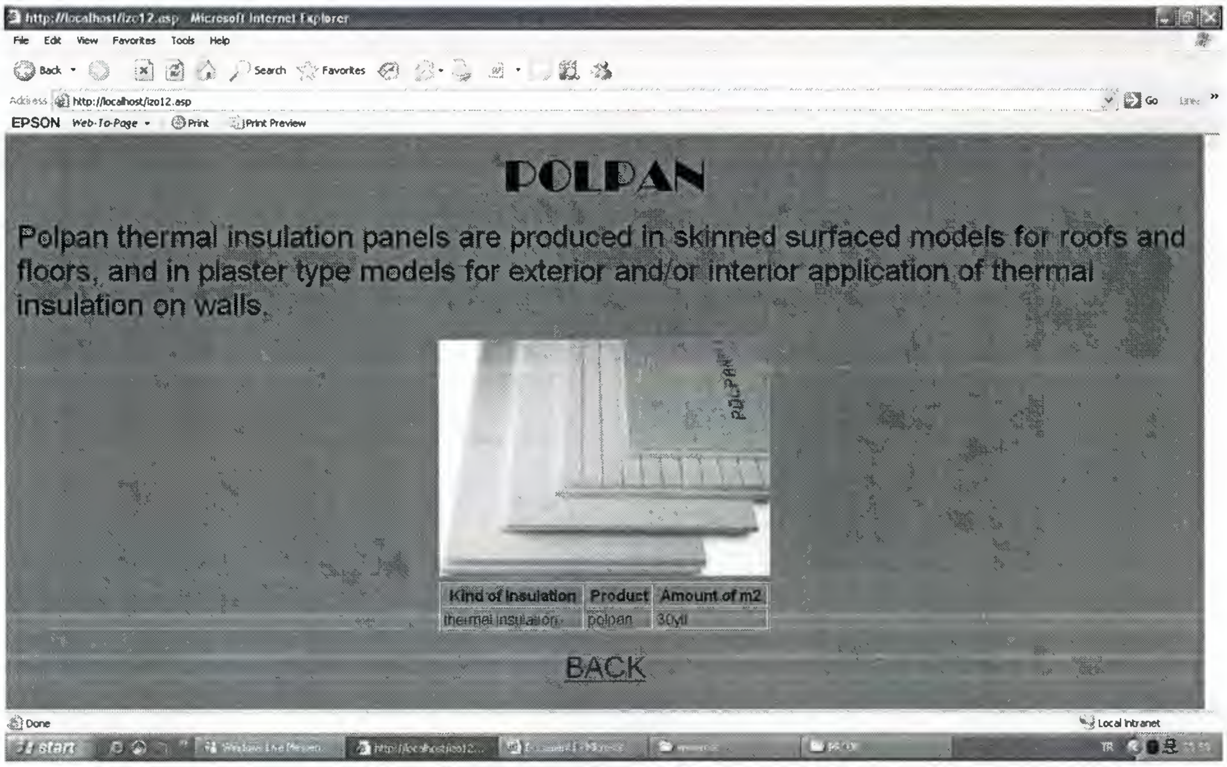
Thermal Insulation Page.

When admin, user and guesst click thermal insulation's link see this page. When admin clicked this kind of insulation, view see product, searching page, entering page, deleting page and updating page. If user click this link he/she see very links without delete becouse user can not make delete. And if guesst click this link just see see product.


insulation!!! - Microsoft Internet Explorer
http://localhost/izo11.html
[THERMAL INSULATION]
Exstruded Polystyrenee Thermal Insulation Panels
POLPAN
İRFAN İZOLASYON

Thermal Insulation Polpan

If admin, user and guesst click see product view this page. We have link to defination about polpan. Polpan is thermal insulation.


http://localhost/izo12.asp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/izo12.asp
Go
EPSON Web-To-Page Print Print Preview
Done
Local intranet
start

POLPAN

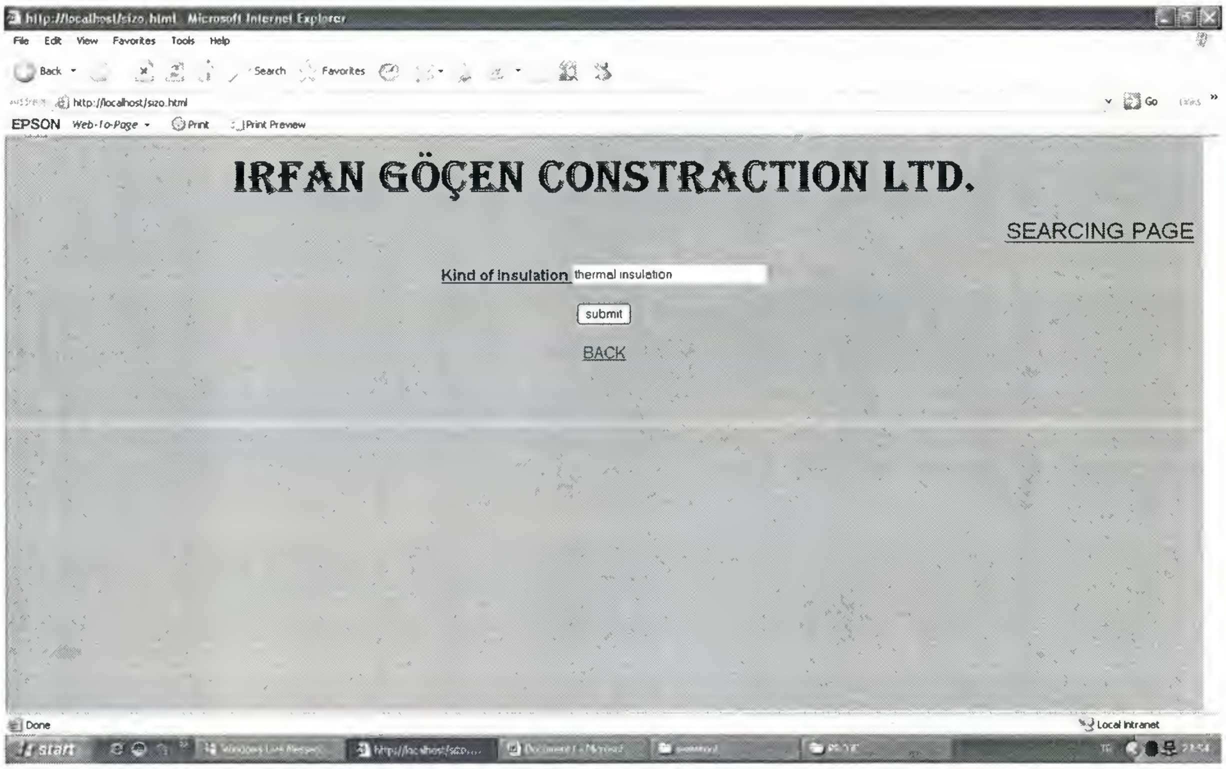
Polpan thermal insulation panels are produced in skinned surfaced models for roofs and floors, and in plaster type models for exterior and/or interior application of thermal insulation on walls.

Kind of Insulation	Product	Amount of m2
thermal insulation	polpan	30ytl

BACK

Defination Of Polpan

When admin, user or guesst click polpan's link view this page. This page give description about polpan and we have polpan picture and table about cost of insulation. We view picture and table because of database.


http://localhost/sizo.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
http://localhost/sizo.html
Go
EPSON Web-To-Page Print Print Preview
Done
Local intranet
start

IRFAN GÖÇEN CONSTRACTION LTD.

SEARCING PAGE

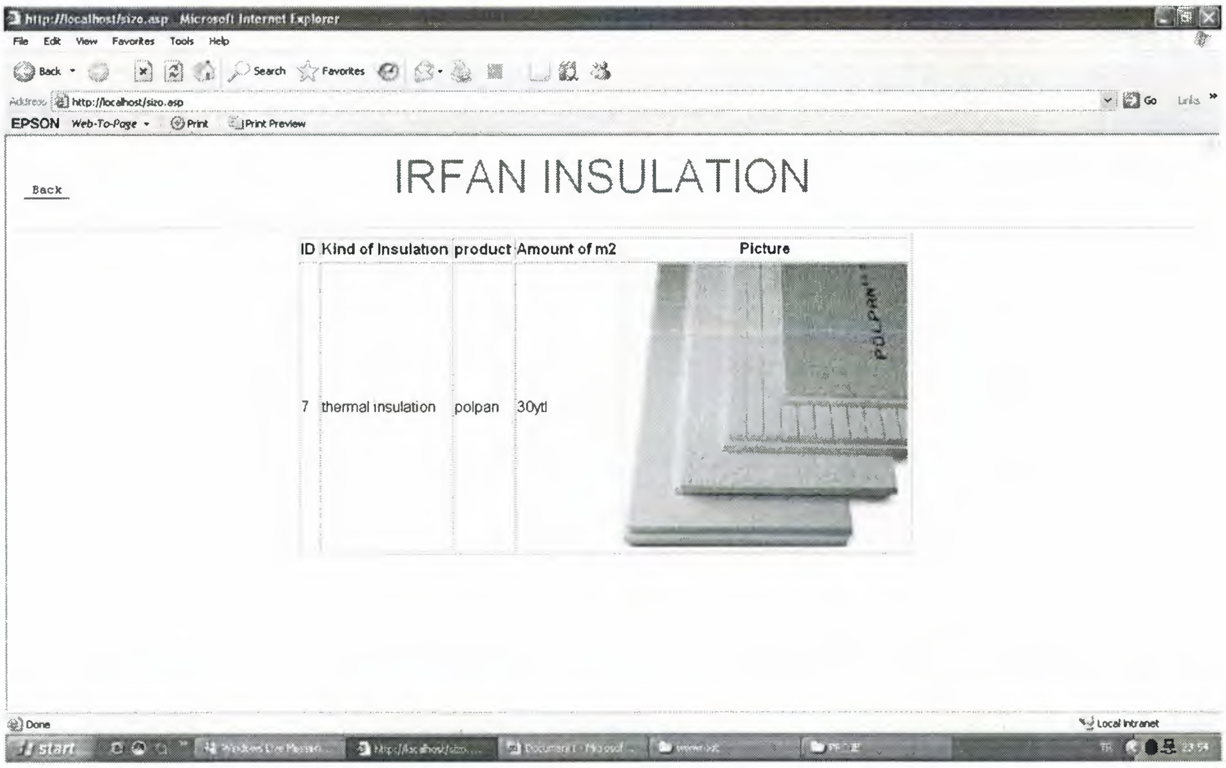
Kind of Insulation thermal insulation

submit

BACK

Searching Page To Thermal Insulation

When admin, user and guesst click this page can search thermal insulation to see every description about this kind of insulation. And see as below page.

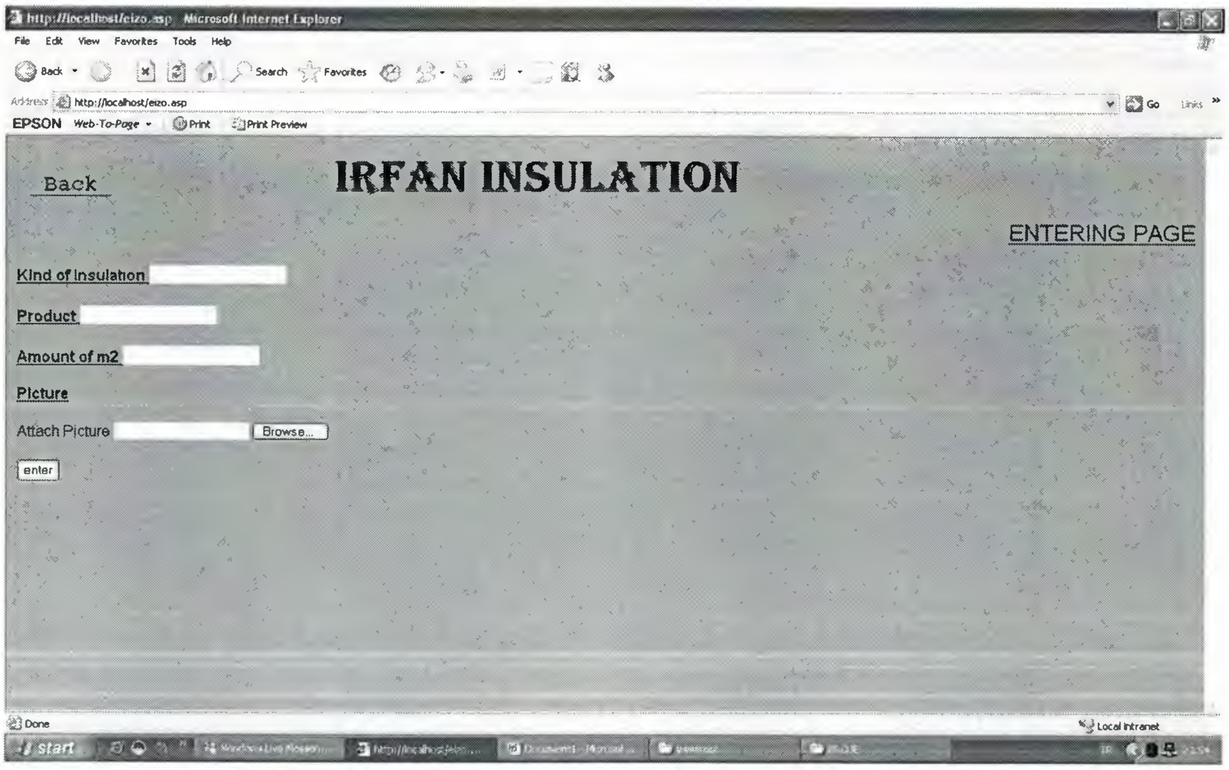

http://localhost/sizo.asp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address http://localhost/sizo.asp
Go
Links
EPSON Web-To-Page Print Print Preview
Back
Done
Local intranet
start

IRFAN INSULATION

ID	Kind of Insulation	product	Amount of m2	Picture
7	thermal ınsulation	polpan	30ytl	

Thermal Insulation Table.

This page result of search.admin, user and guesst see ıd number, kind of insulation, product, amount of m2 and picture. View because of database.


http://localhost/eizo.asp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/eizo.asp
EPSON Web-To-Page Print Print Preview

Back

IRFAN INSULATION

ENTERING PAGE

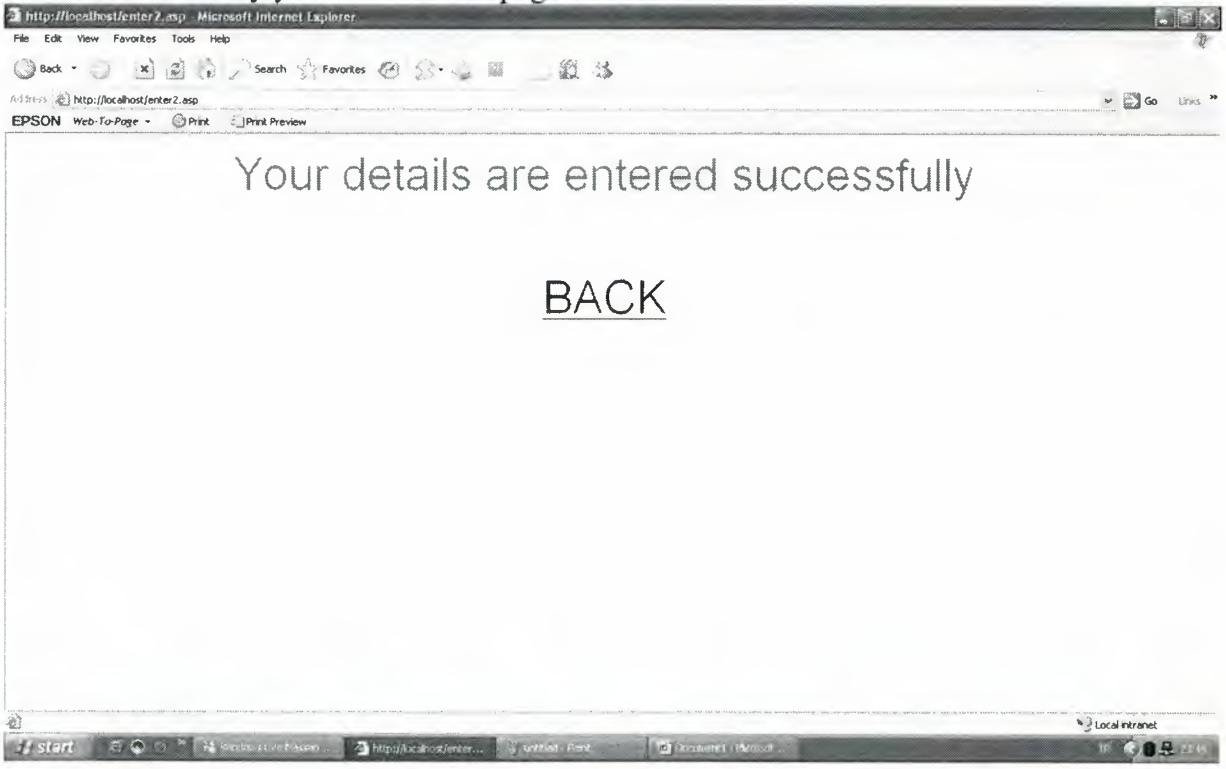
Kind of Insulation	
Product	
Amount of m2	
Picture	
Attach Picture	Browse...

enter

Done
Local intranet
start

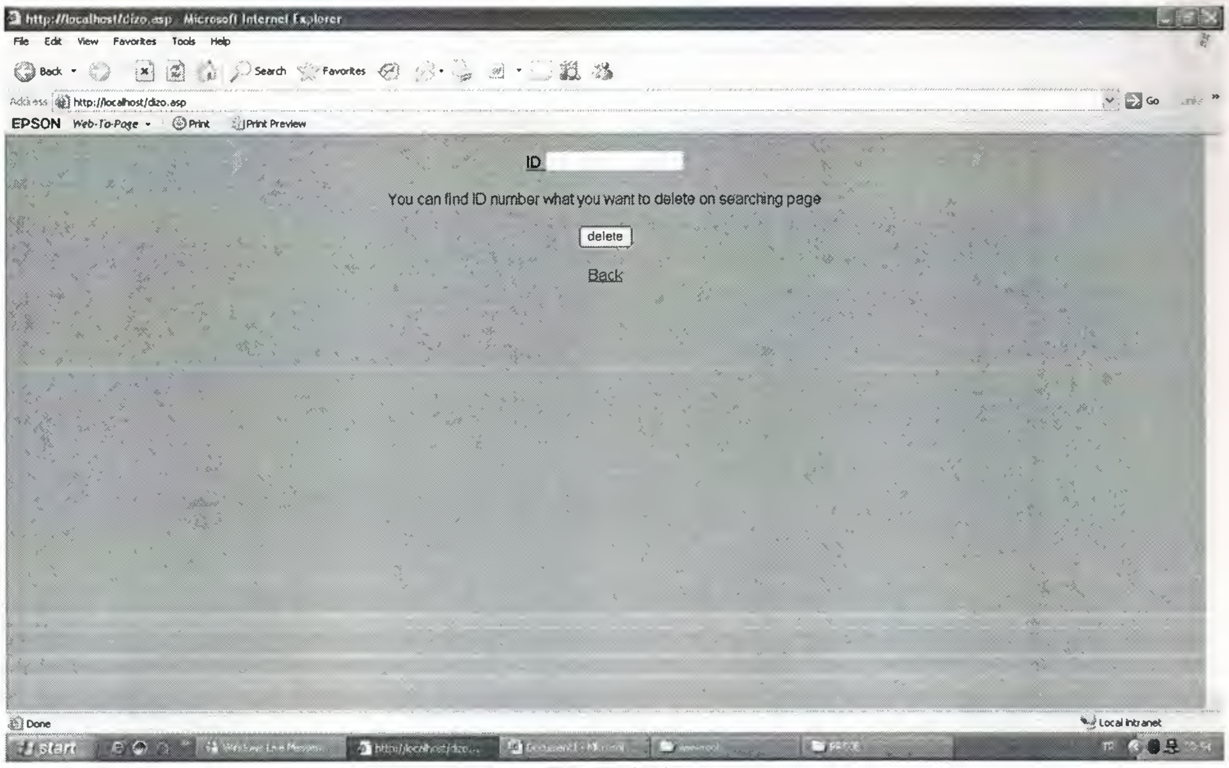
Entering Page For Kind Of Insulations

This page for enter the insulation details in the textfield and insert database this details. We have descriptions and textfields to fill them. And we have submit button. we fill in textfields and we have can attach picture when click browse open file choose and we can select insulation picture then click submit button we insert theese details in the database. And also we have back link to turn before of this page. And if your details entered successfully you see as below page.


http://localhost/enter2.asp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/enter2.asp
EPSON Web-To-Page Print Print Preview

Your details are entered successfully

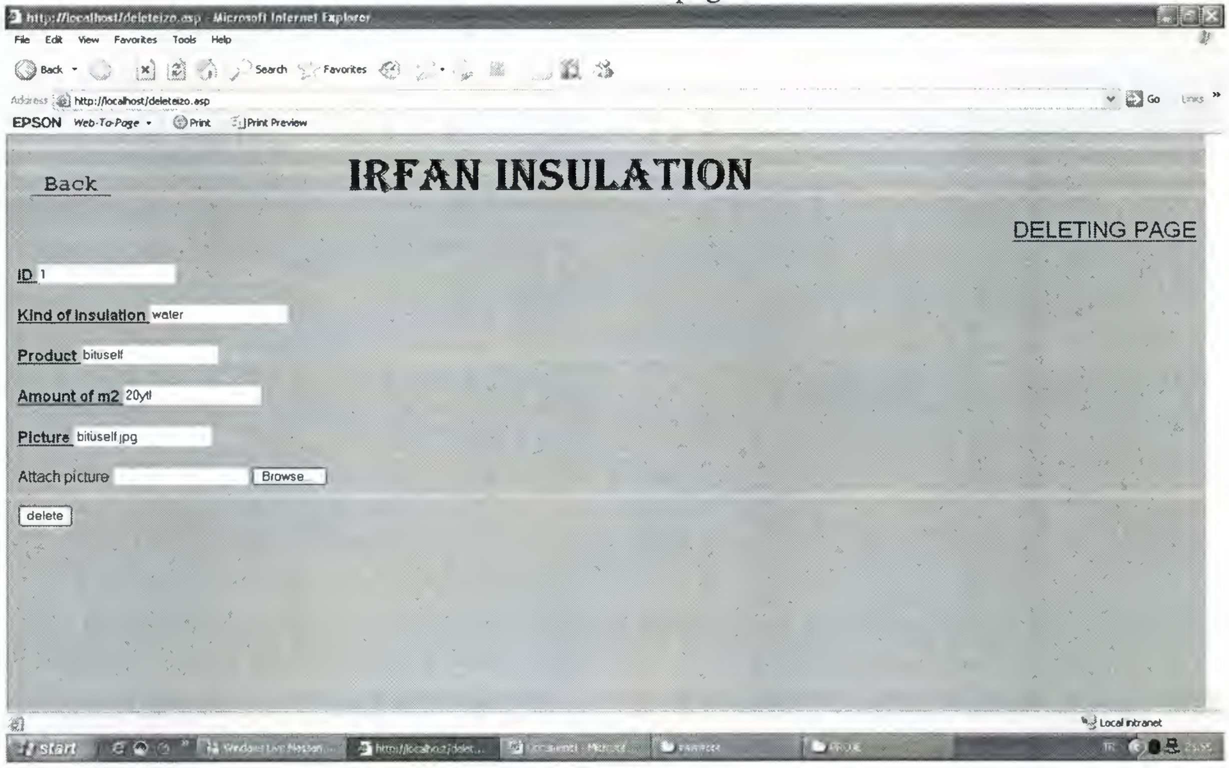
BACK


ID

You can find ID number what you want to delete on searching page

1. Step To Deleting

This page first step to deleting. Admin can delete every thing about insulations. Admin write insulation ıd in the texfield. Then view below page.


IRFAN INSULATION

DELETING PAGE

ID 1

Kind of insulation water

Product bituself

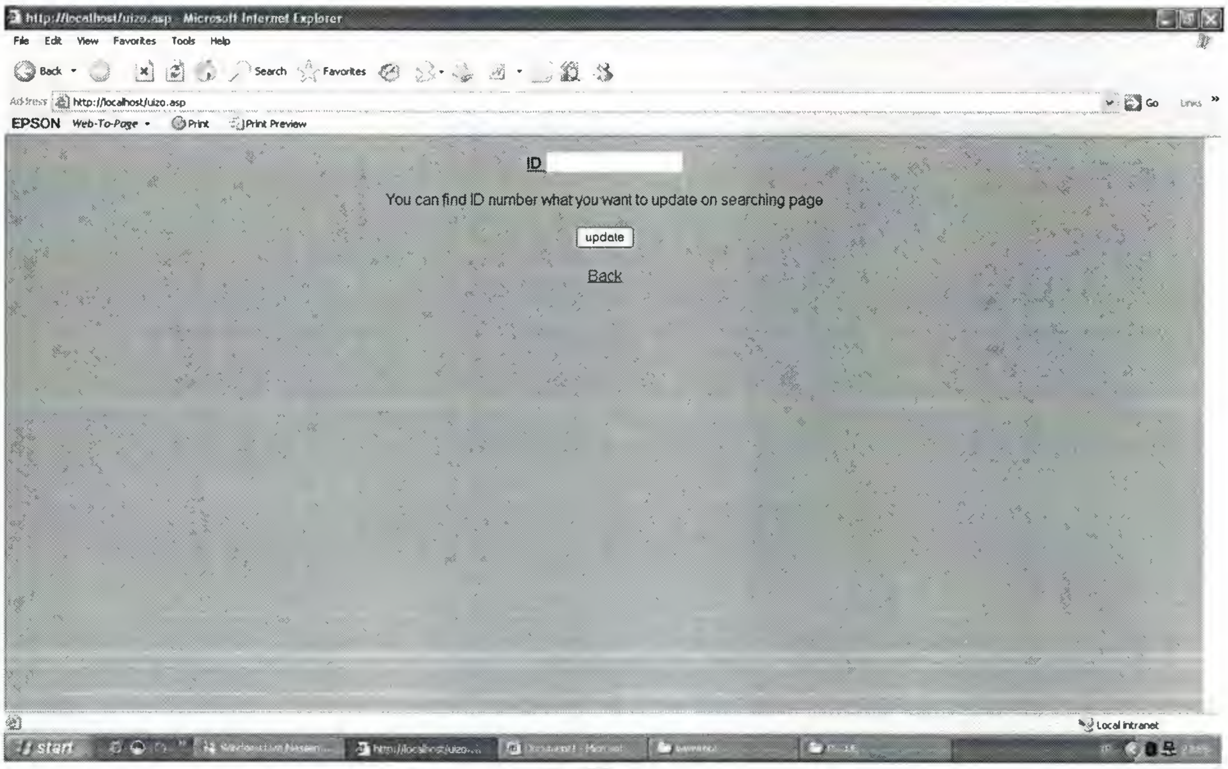
Amount of m2 20ytl

Picture bituself.jpg

Attach picture

Deleting Page For Insulations

This page for delete the insulation details in the database. We have descriptions and fill textfields. And we have submit button. When admin click submit button delete the insualtion in the database. And also we have back link to turn before of this page. And if your details deleted successfully you see page to your details deleted successfully.


ID

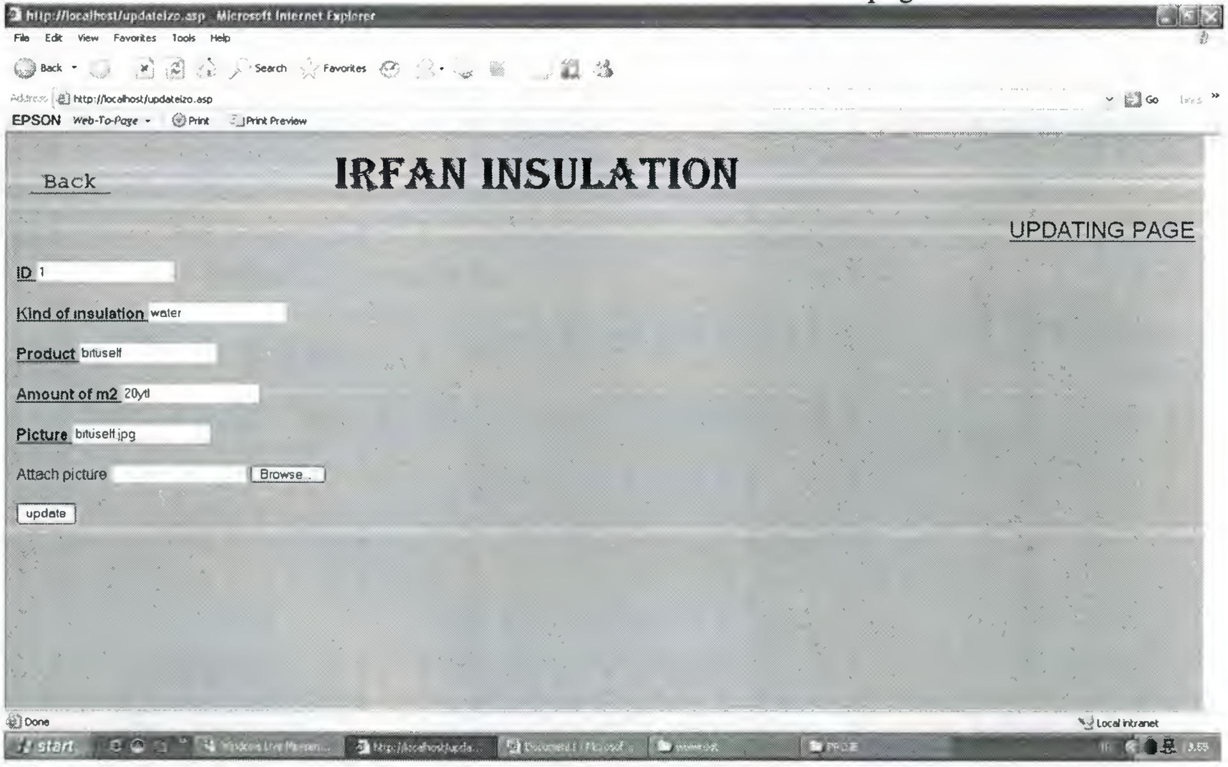
You can find ID number what you want to update on searching page

update

Back

1. Step To Updating

This page first step to updating. Admin or user can update every thing about insulations. Admin or user write insualtion ıd in the texfield. Then view below page.


Back

IRFAN INSULATION

UPDATING PAGE

ID 1

Kind of insulation water

Product brtuself

Amount of m2 20ytl

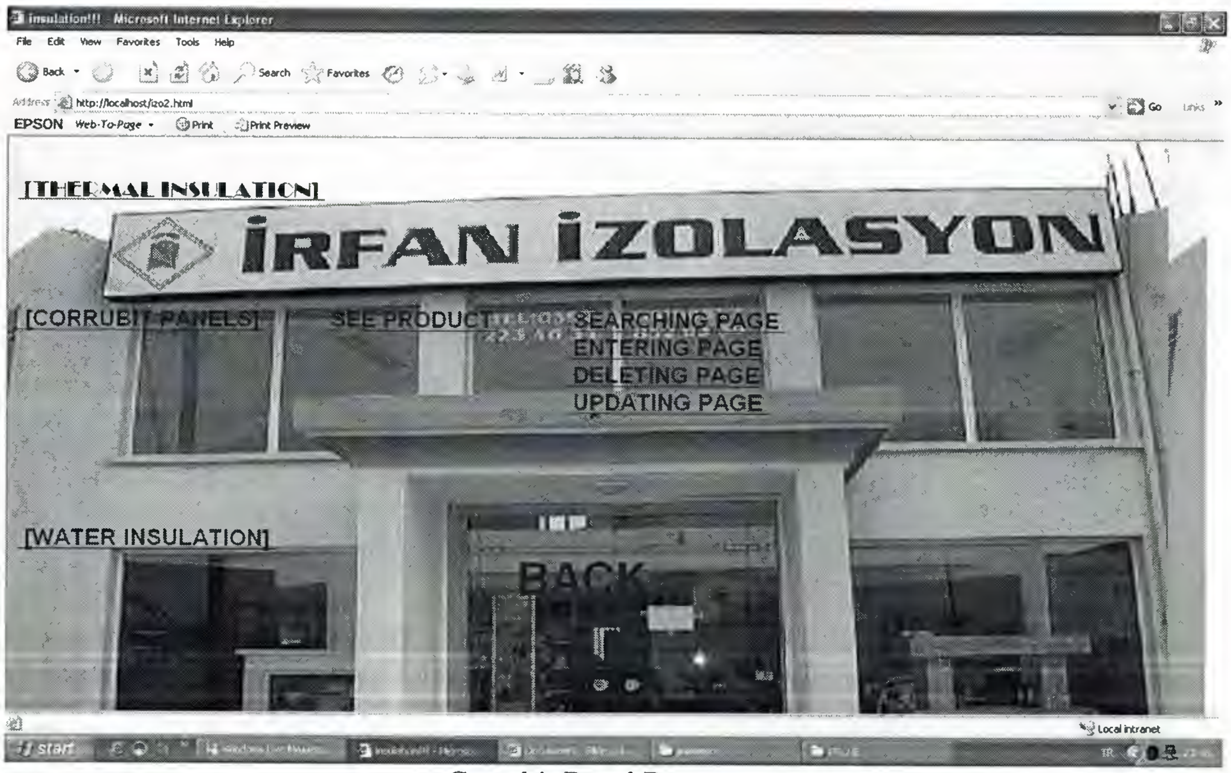
Picture brtuself.jpg

Attach picture Browse...

update

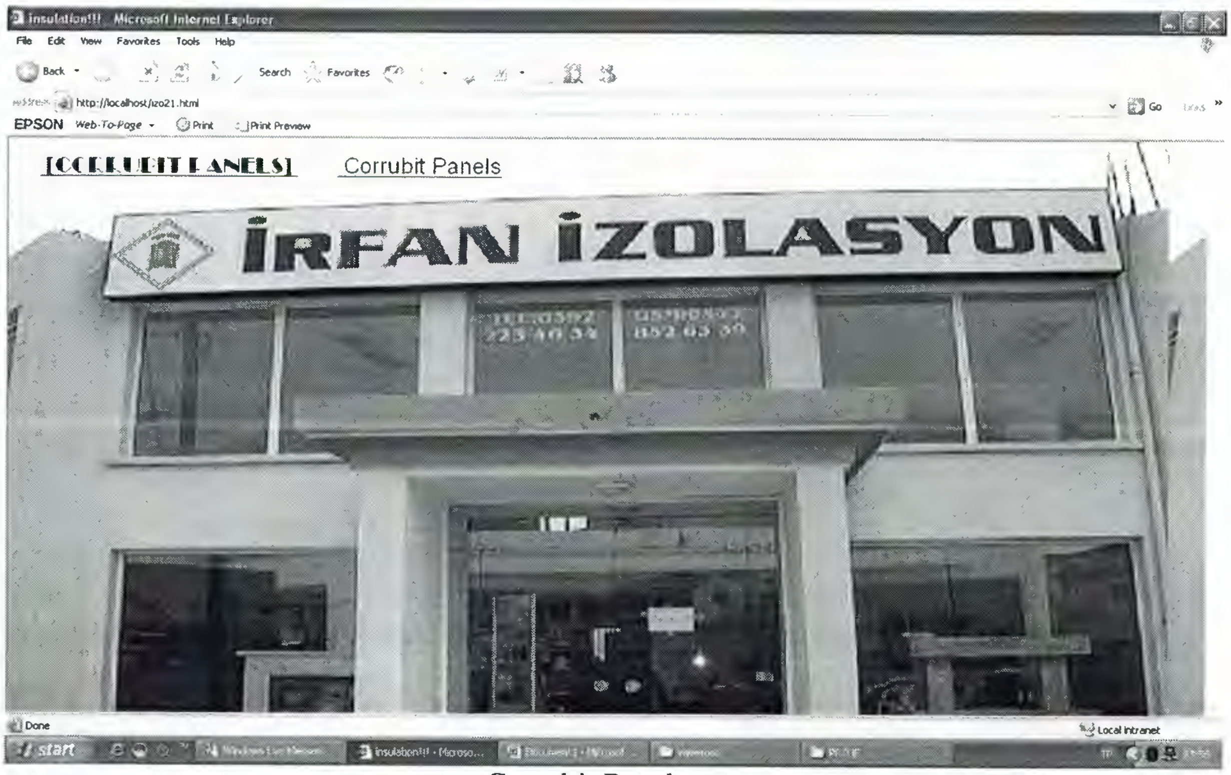
Updating Page For Insulations

This page for update the insulation details in the database. We have descriptions and fill textfields. And we have submit button. When admin or user change in the textfield or textfields then click submit button to update the insulation details in the database. And also we have back link to turn before of this page. And if your details updated successfully you see page to your details updated successfully.



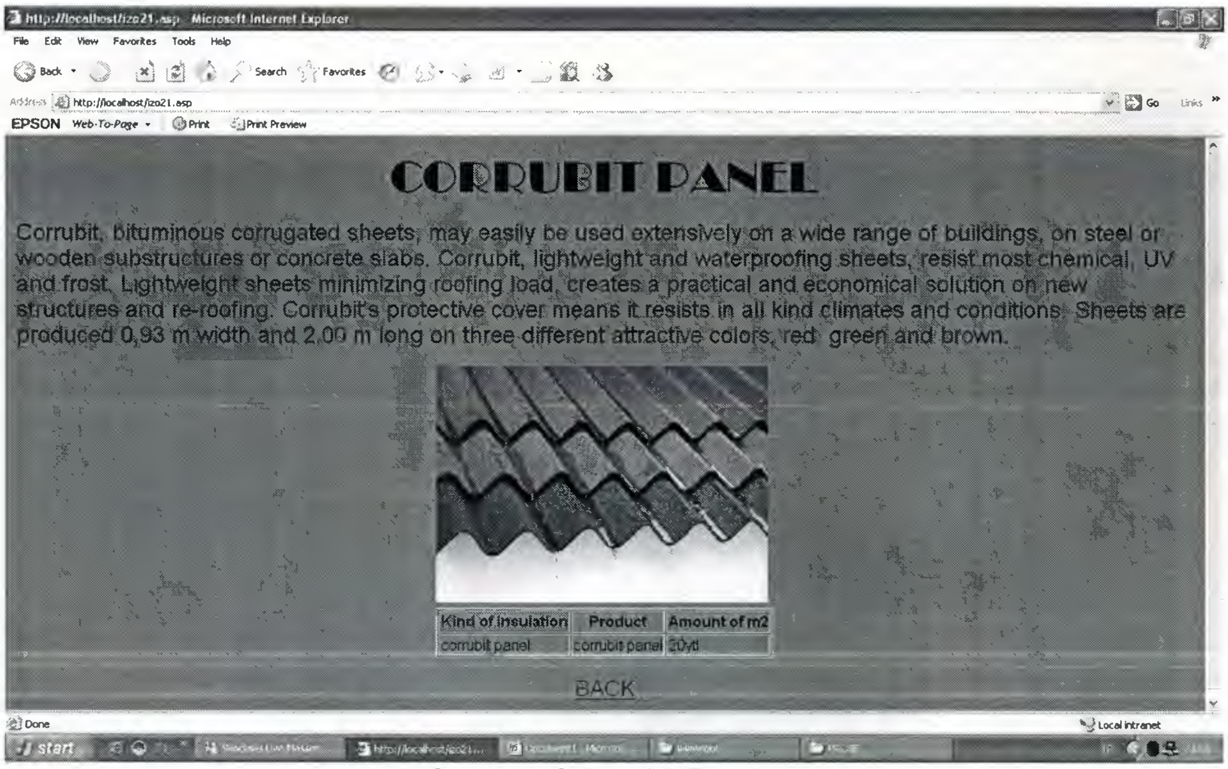
Corrubit Panel Page

When admin, user and guess click currbit panel's link see this page. When admin clicked this kind of insulation, view see product, searching page, entering page, deleting page and updating page. If user click this link he/she see very links without delete becouse user can not make delete. And if guesst click this link just see see product.


insulation!!! - Microsoft Internet Explorer
http://localhost/izo21.html
[CORRUBIT PANELS]
Corrubit Panels
İRFAN İZOLASYON

Currubit Panel

If admin, user and guess click see product view this page. We have link to defination about currubit panels.



CORRUBIT PANEL

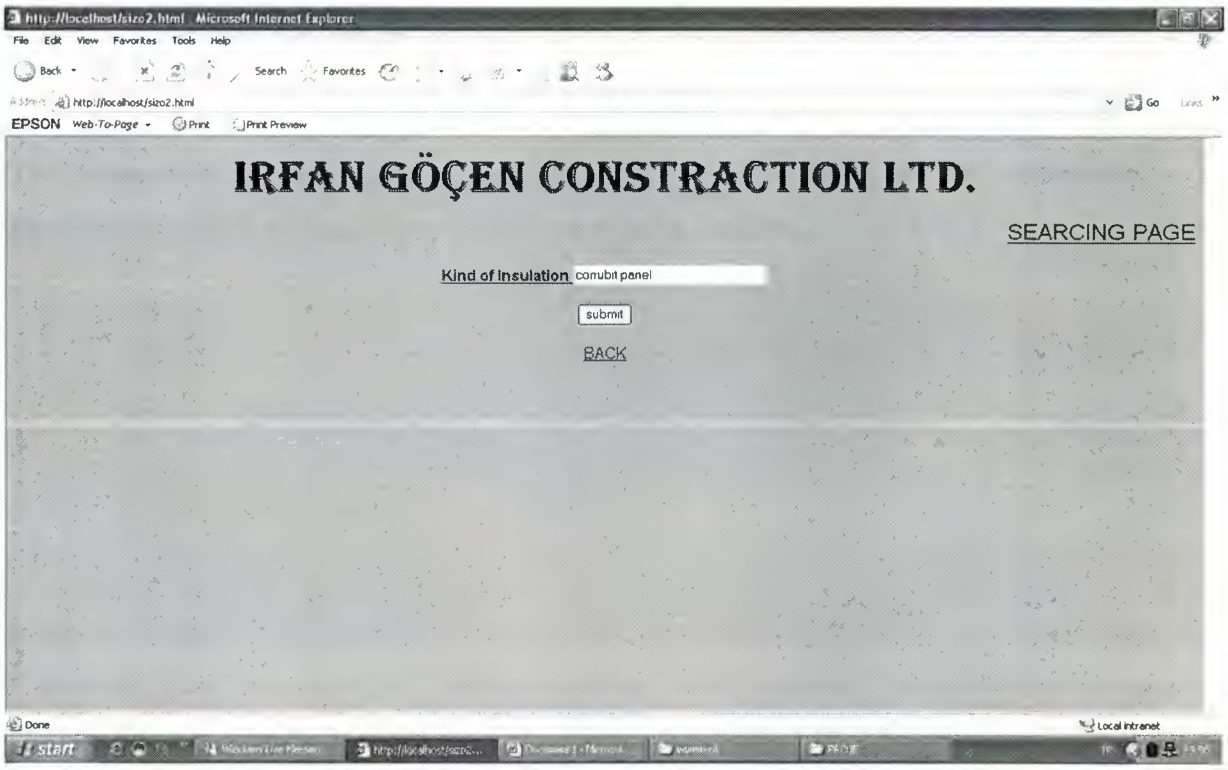
Corrubit, bituminous corrugated sheets, may easily be used extensively on a wide range of buildings, on steel or wooden substructures or concrete slabs. Corrubit, lightweight and waterproofing sheets, resist most chemical, UV and frost. Lightweight sheets minimizing roofing load, creates a practical and economical solution on new structures and re-roofing. Corrubit's protective cover means it resists in all kind climates and conditions. Sheets are produced 0,93 m width and 2,00 m long on three different attractive colors, red green and brown.

Kind of insulation	Product	Amount of m2
currubit panel	currubit panel	20ytl

BACK

Defination of Currubit Panel

When admin, user or guess click currubit panel's link view this page. This page give description about currubit panel and we have currubit picture and table about cost of insulation. We view picture and table becouse of database.



IRFAN GÖÇEN CONSTRACTION LTD.

SEARCING PAGE

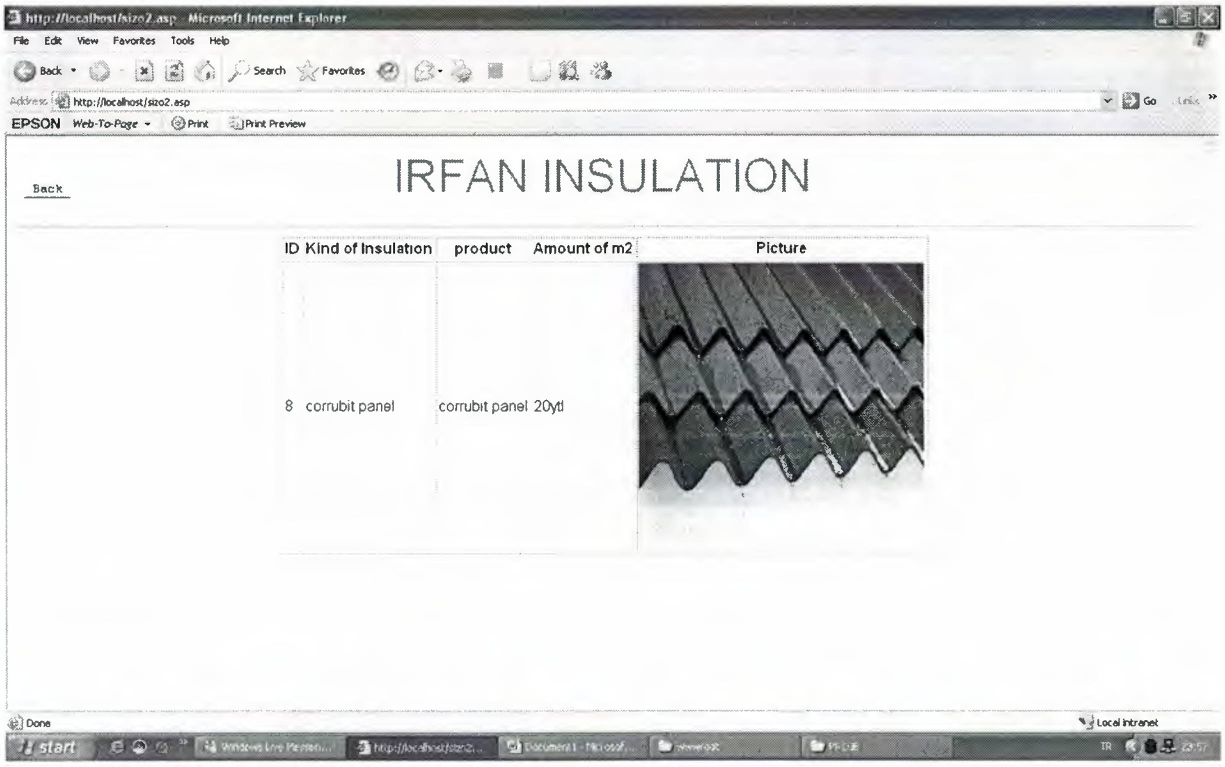
Kind of insulation: currubit panel

submit

BACK

Searching Page To Thermal Insulation.

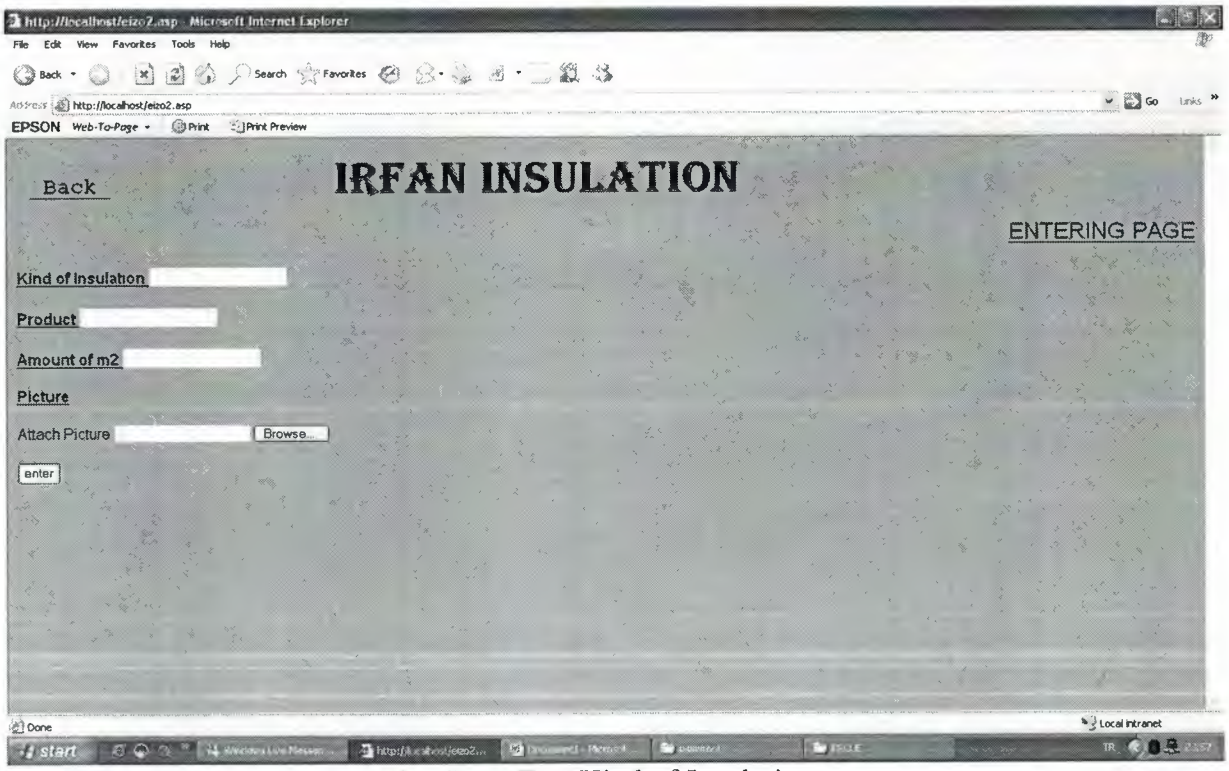
When admin, user and guesst click this page can search currubit panel to see every description about this kind of insulation. And see as below page.


http://localhost/sizo2.asp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address http://localhost/sizo2.asp
Go
EPSON Web-To-Page Print Print Preview
Back
IRFAN INSULATION
Done
Local intranet
start

ID	Kind of Insulation	product	Amount of m2	Picture
8	corrubit panel	corrubit panel	20ytl	

Thermal Insulation Table

This page result of search.admin, user and guesst see ıd number, kind of insulation, product, amount of m2 and picture. View becouse of database.


Back

IRFAN INSULATION

ENTERING PAGE

Kind of Insulation

Product

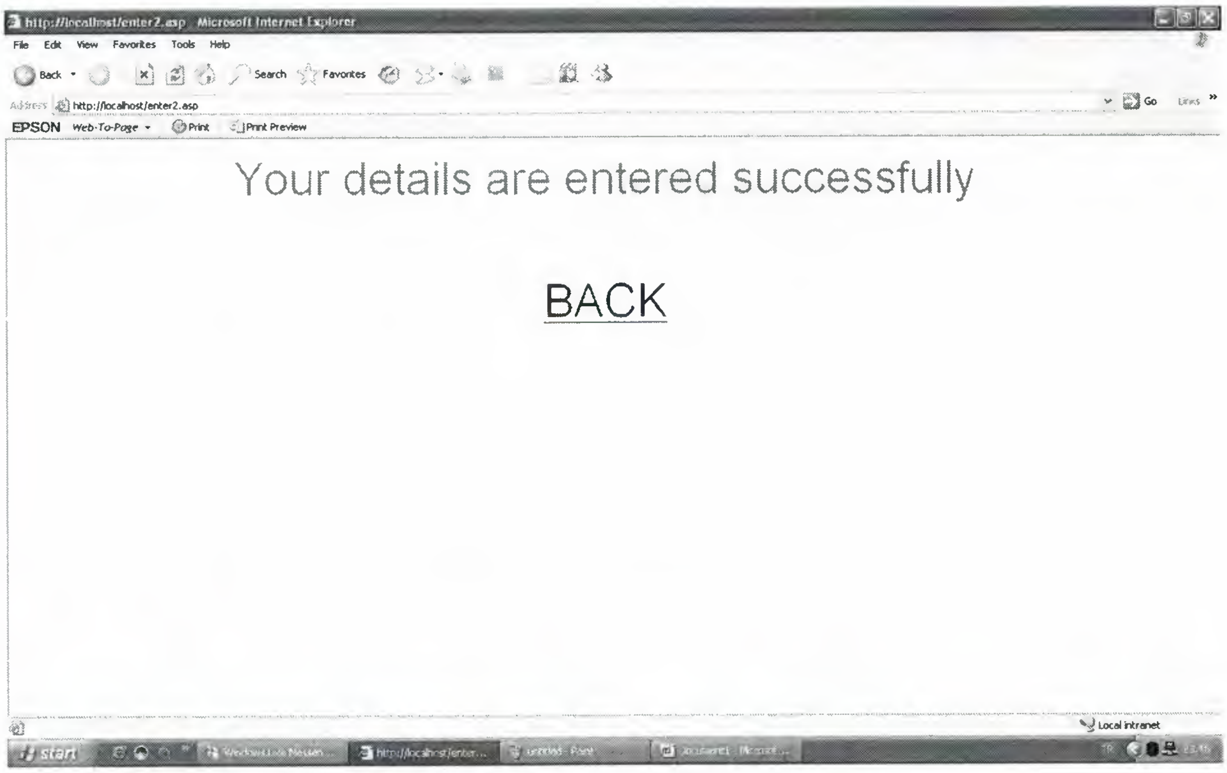
Amount of m2

Picture

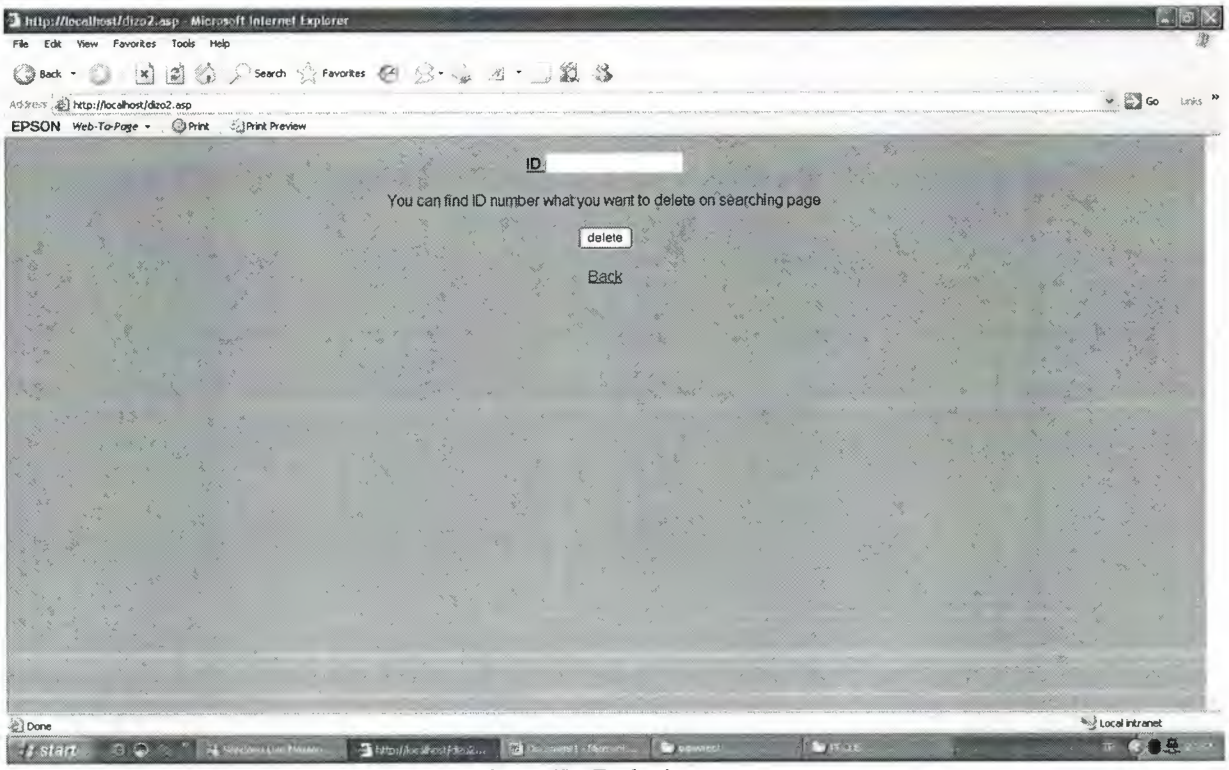
Attach Picture

Entering Page For Kind of Insulations

This page for enter the insulation details in the textfield and insert database this details. We have descriptions and textfields to fill them. And we have submit button. we fill in textfields and we have can attach picture when click browse open file choose and we can select insulation picture then click submit button we insert theese details in the database. And also we have back link to turn before of this page. And if your details entered successfully you see as below page.


Your details are entered successfully

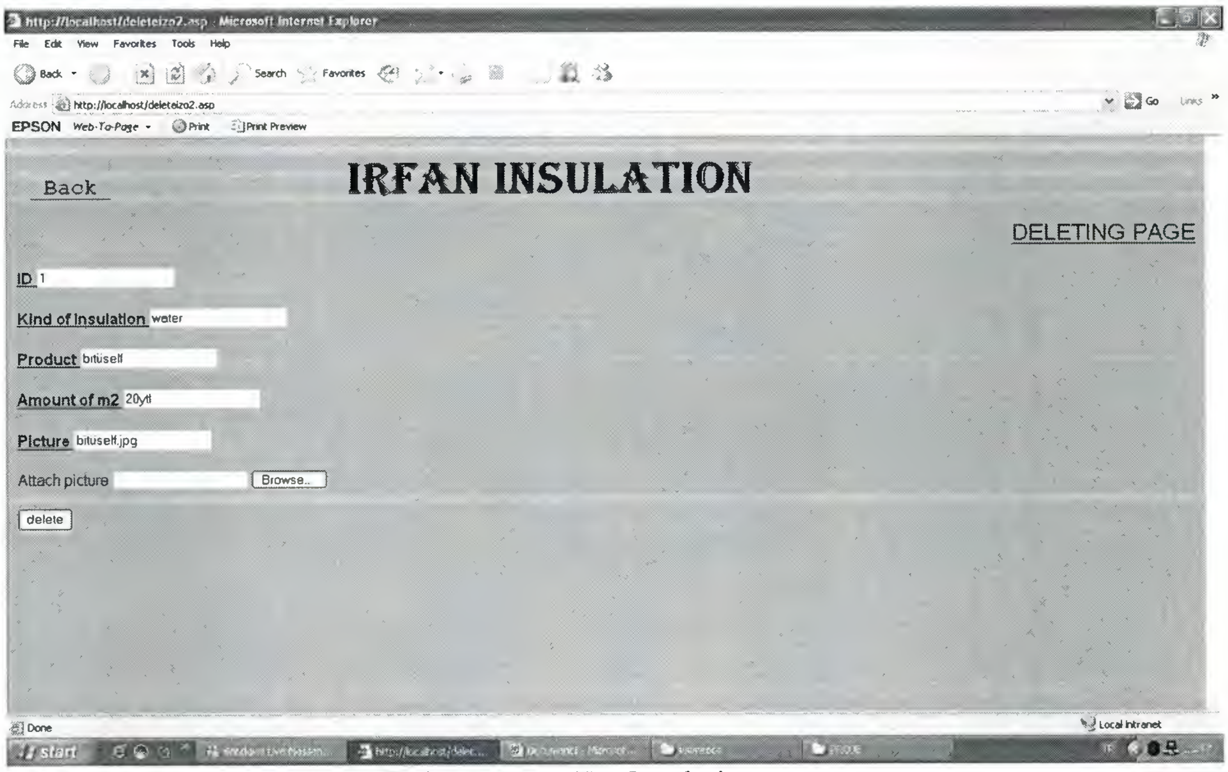
BACK


http://localhost/dizo2.asp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/dizo2.asp
ID
delete
Back
Done
Local intranet

You can find ID number what you want to delete on searching page

1. Step To Deleting

This page first step to deleting. Admin can delete every thing about insulations. Admin write insulation ıd in the texfield. Then view below page.


http://localhost/deleteizo2.asp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/deleteizo2.asp
Back
IRFAN INSULATION
DELETING PAGE
Browse..
delete
Done
Local intranet

ID 1

Kind of Insulation water

Product bitüself

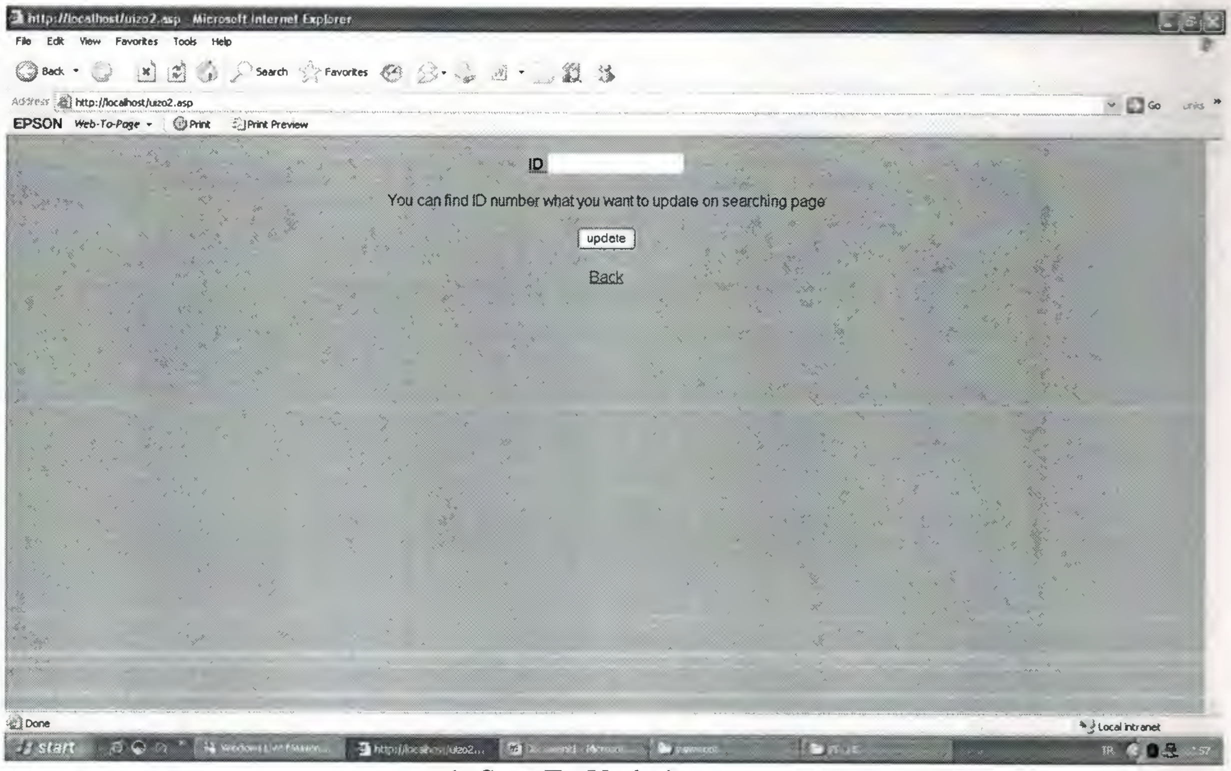
Amount of m2 20ytl

Picture bitüself.jpg

Attach picture

Deleting Page For Insulations

This page for delete the insulation details in the database. We have descriptions and fill textfields. And we have submit button. When admin click submit button delete the insualtion in the database. And also we have back link to turn before of this page. And if your details deleted successfully you see page to your details deleted successfully.


http://localhost/uizo2.asp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/uizo2.asp

ID

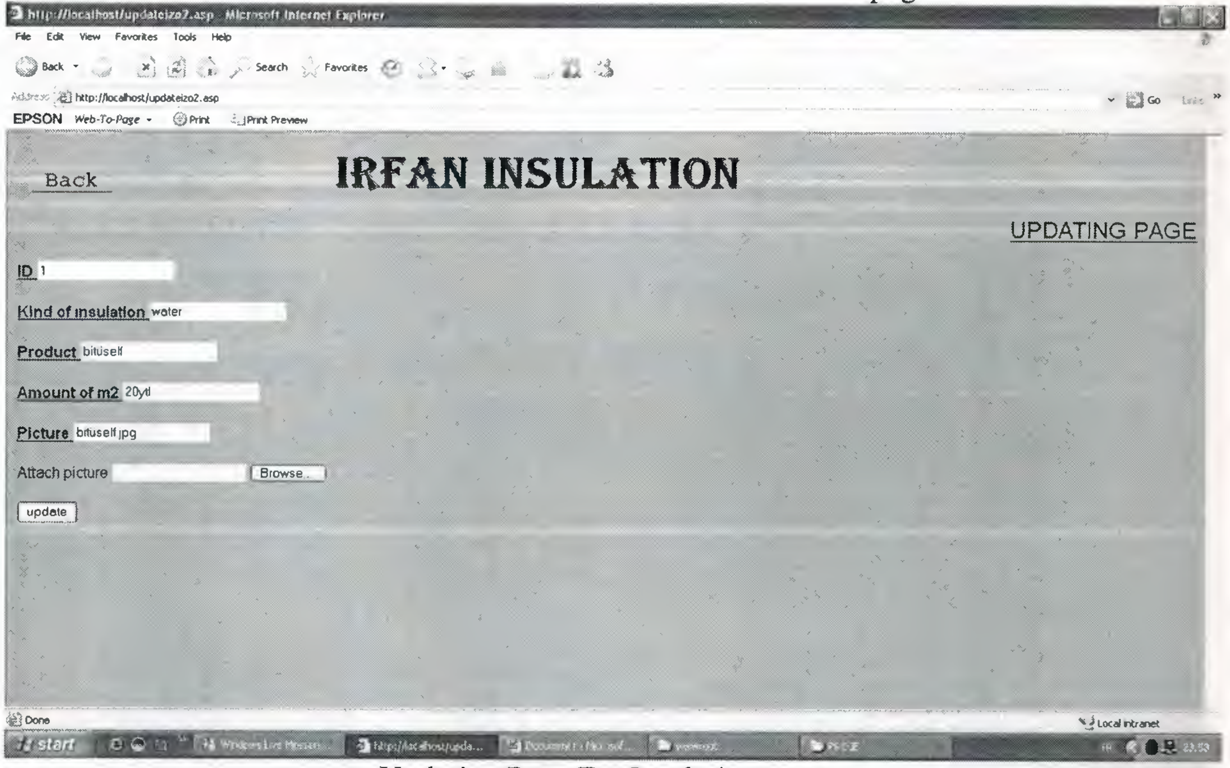
You can find ID number what you want to update on searching page

update

Back

1. Step To Updating

This page first step to updating. Admin or user can update every thing about insulations. Admin or user write insualtion ıd in the texfield. Then view below page.


http://localhost/updateizo2.asp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/updateizo2.asp

Back

IRFAN INSULATION

UPDATING PAGE

ID 1

Kind of insulation water

Product bituself

Amount of m2 20ytl

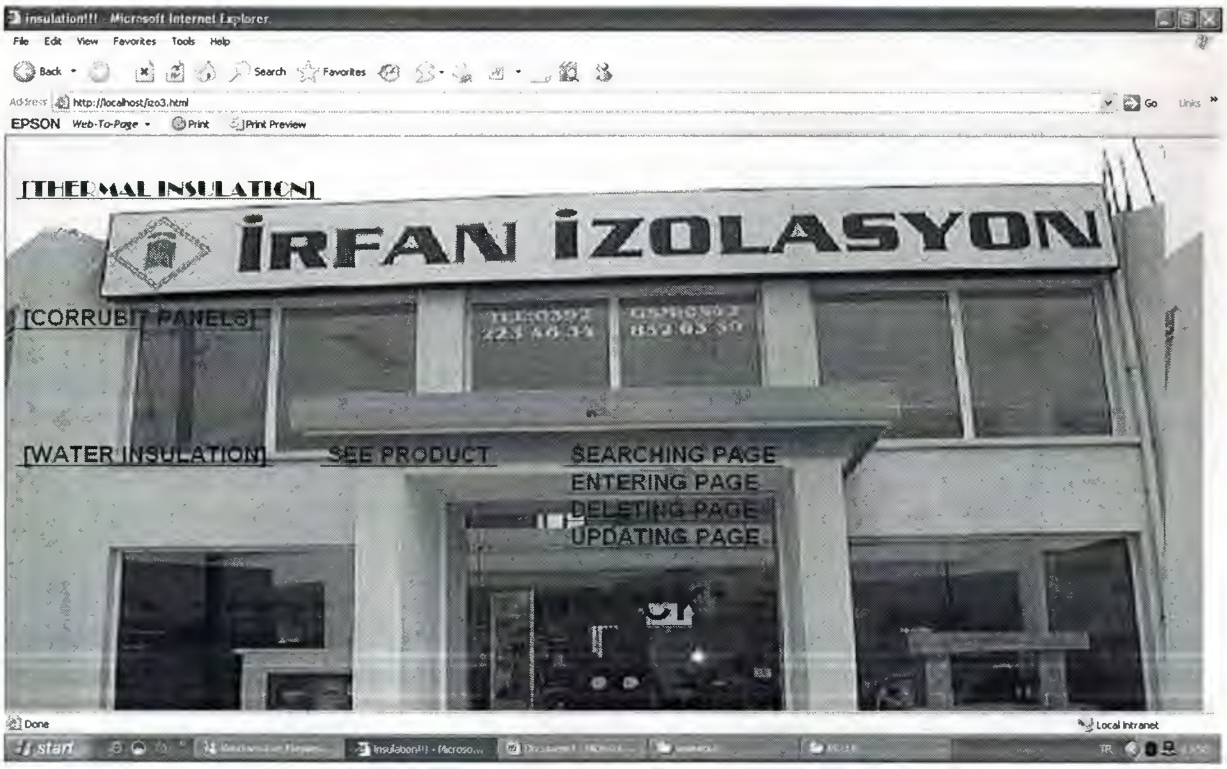
Picture bituself.jpg

Attach picture Browse...

update

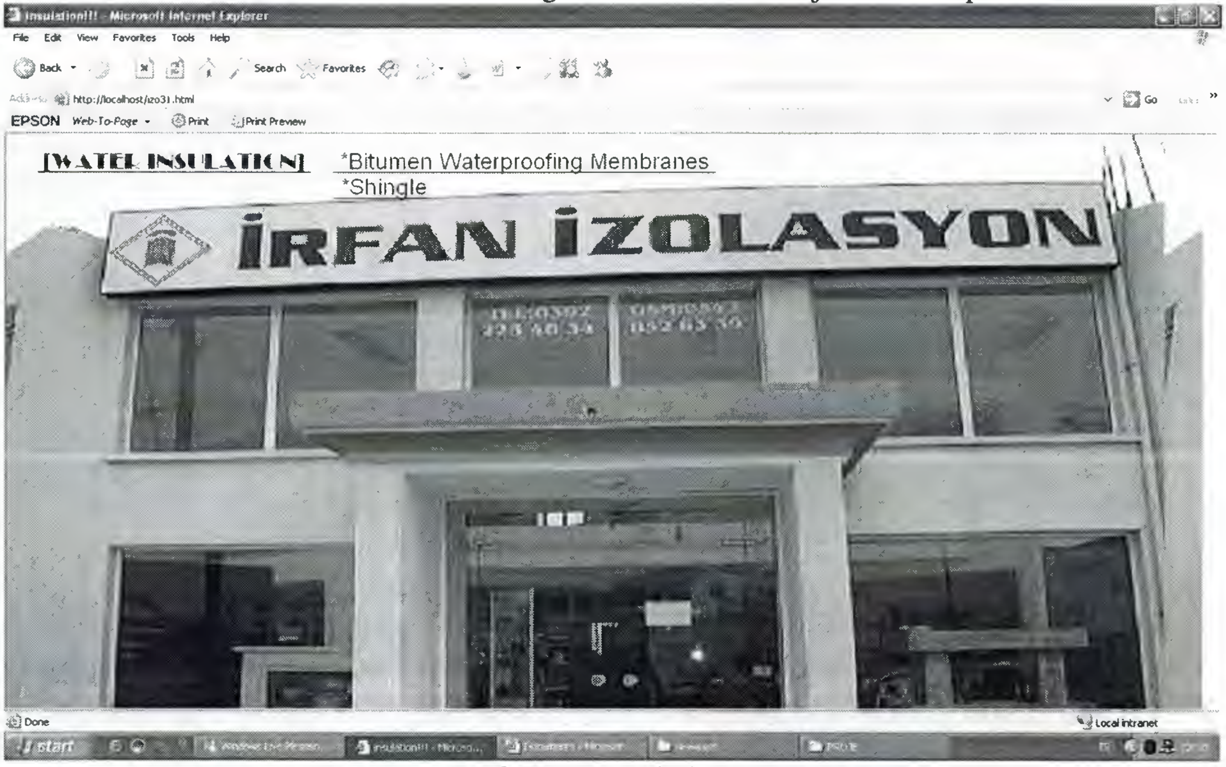
Updating Page For Insulations

This page for update the insulation details in the database. We have descriptions and fill textfields. And we have submit button. When admin or user change in the textfield or textfields then click submit button to update the insulation details in the database. And also we have back link to turn before of this page. And if your details updated successfully you see page to your details updated successfully.



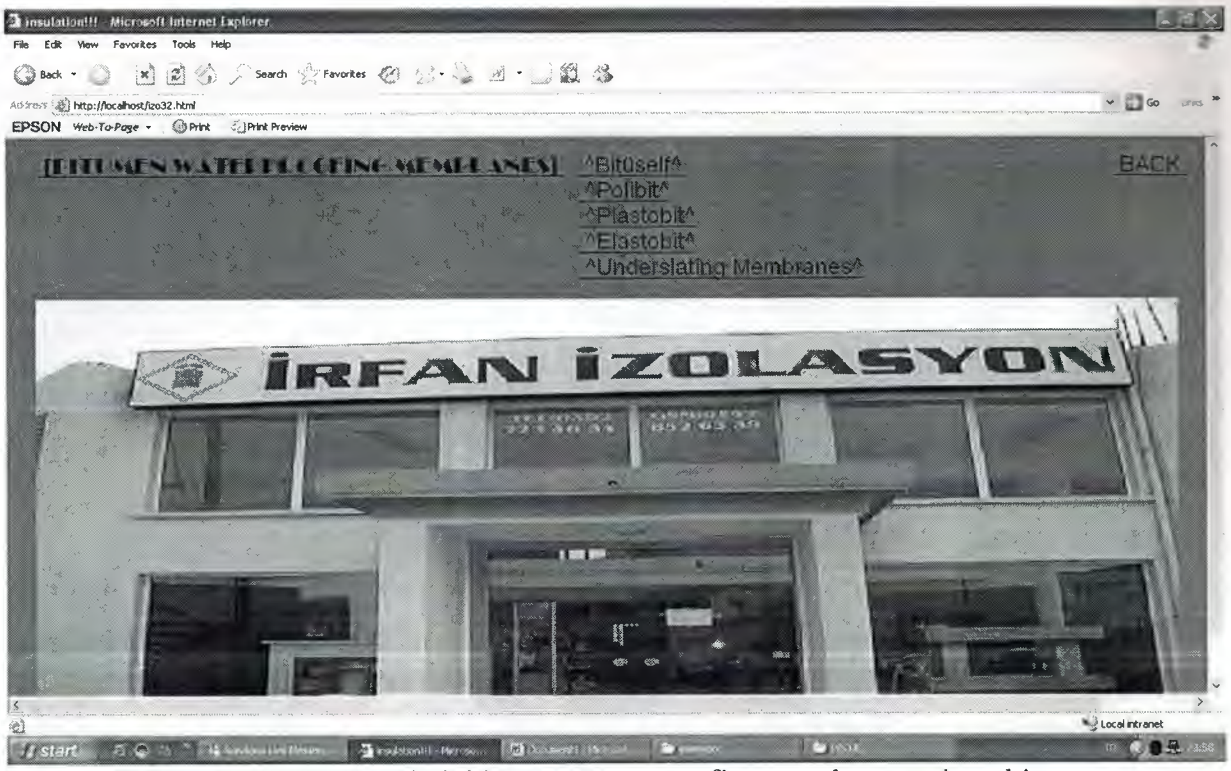
Water Insulation Page

When admin, user and guesst click water insulation’s link see this page. When admin clicked this kind of insulation, view see product, searching page, entering page, deleting page and updating page. If user click this link he/she see very links without delete becouse user can not make delete. And if guesst click this link just see see product.

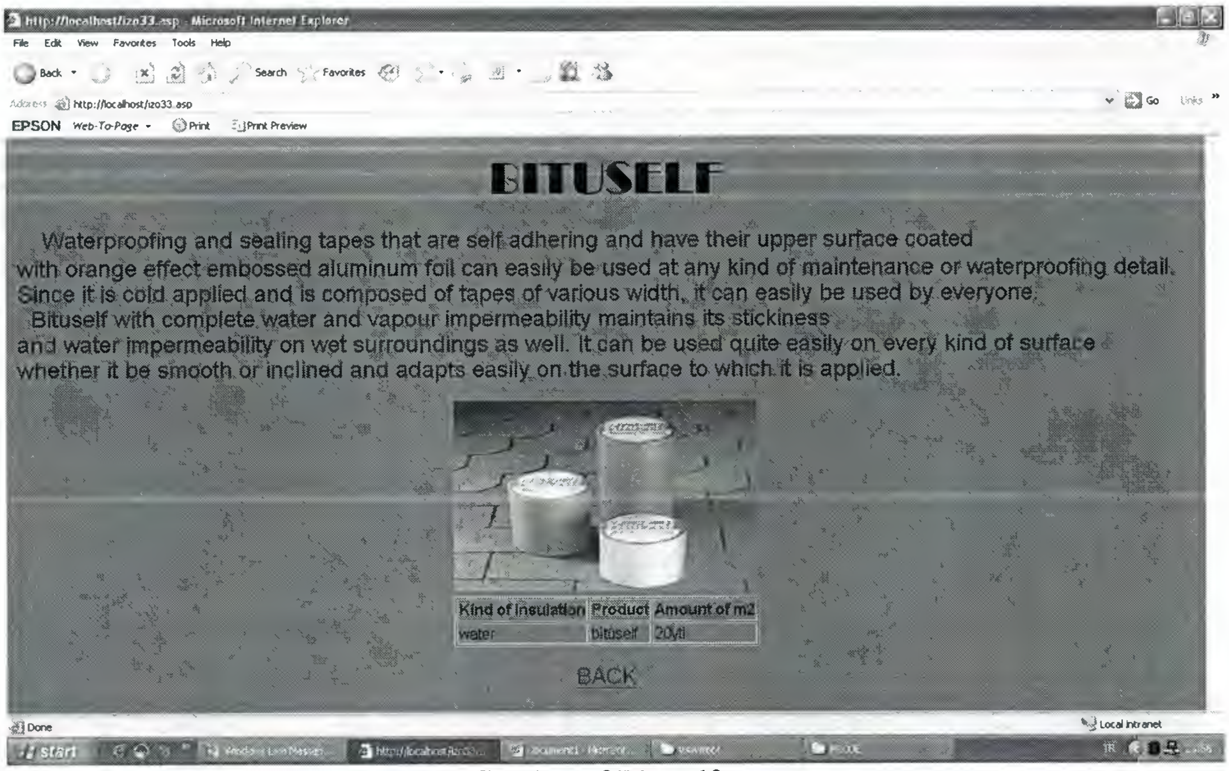

insulation!!! - Microsoft Internet Explorer
http://localhost/izo31.html
[WATER INSULATION]
*Bitumen Waterproofing Membranes
*Shingle
İRFAN İZOLASYON

Part Of Water Insulation.

If admin, user and guesst click see product view this page. We have link to defination about bitumen waterproofing membranes and shingle.


insulation!!! - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/izo32.html
Go
EPSON Web-To-Page Print Print Preview
[BITUMEN WATERPROOFING MEMBRANES]
^Bitüself^
^Polibit^
^Plastobit^
^Elastobit^
^Underslating Membranes^
BACK
İRFAN İZOLASYON
Local intranet
start

When admin, user and guesst click bitumen waterproofing membranes view this page. This kind of insulation have five part so we have five links bitüself, polibit, plastobit. Elastobit and underslating membranes. And back link to back before page.


http://localhost/izo33.asp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/izo33.asp
Go
EPSON Web-To-Page Print Print Preview
Done
Local intranet
start

BITUSELF

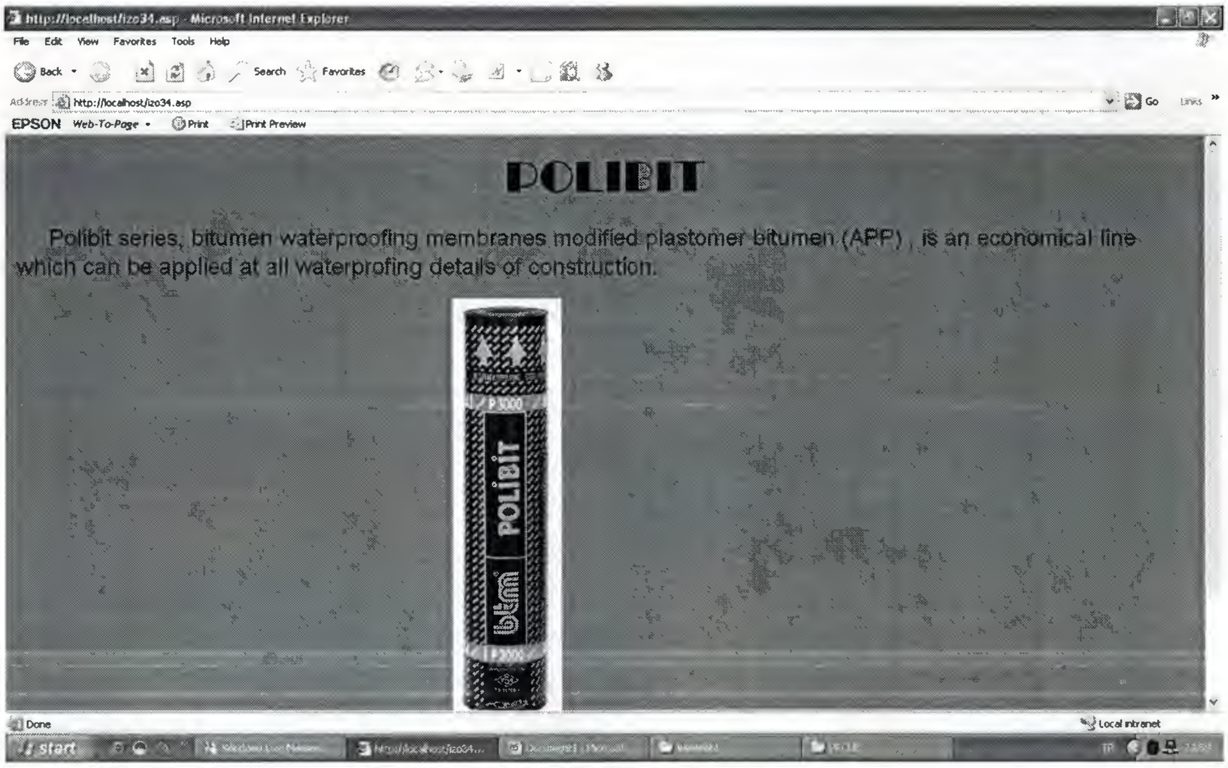
Waterproofing and sealing tapes that are self adhering and have their upper surface coated with orange effect embossed aluminum foil can easily be used at any kind of maintenance or waterproofing detail. Since it is cold applied and is composed of tapes of various width, it can easily be used by everyone.
Bituself with complete water and vapour impermeability maintains its stickiness and water impermeability on wet surroundings as well. It can be used quite easily on every kind of surface whether it be smooth or inclined and adapts easily on the surface to which it is applied.

Kind of insulation	Product	Amount of m2
water	bitüself	20ytl

BACK

Defination of Bitüself

When admin, user or guesst click polpan's link view this page. This page give description about bitüself and we have bitüself picture and table about cost of insulation. We view picture and table becouse of database.

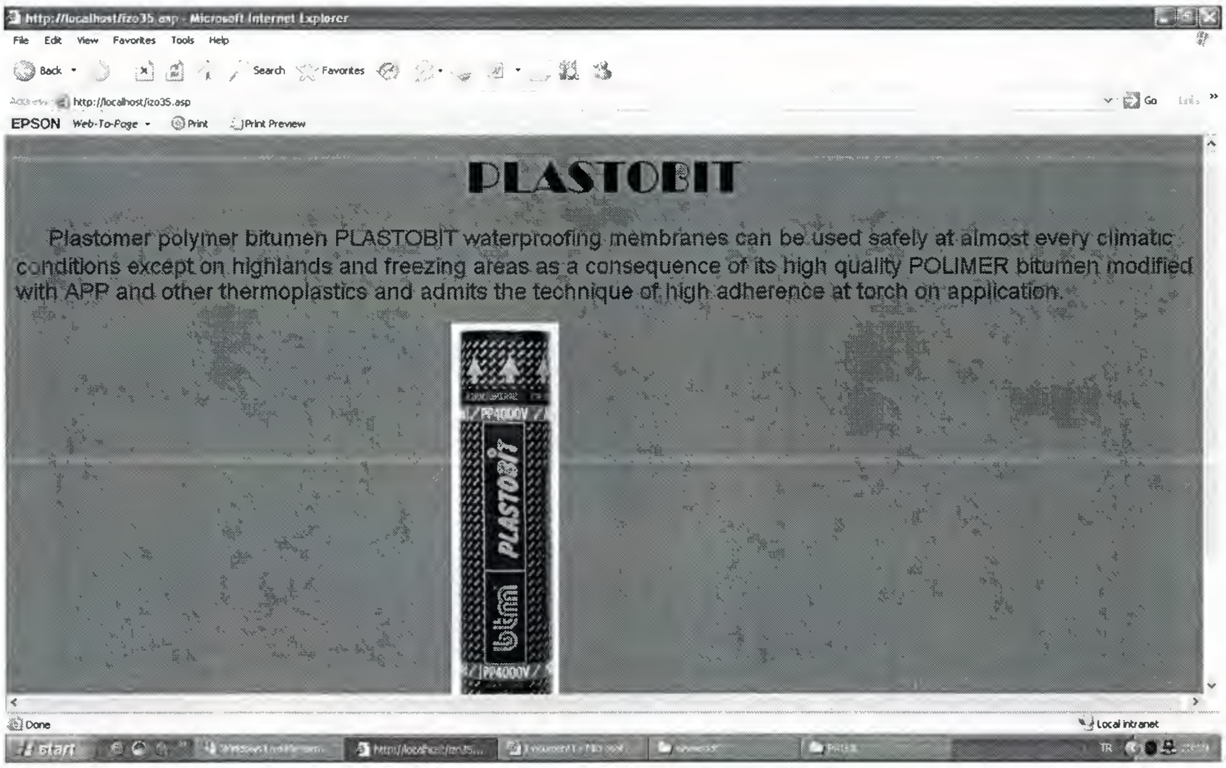

http://localhost/izo34.asp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/izo34.asp
EPSON Web-To-Page Print Print Preview
Done
Local intranet
start

POLIBIT

Polibit series, bitumen waterproofing membranes modified plastomer bitumen (APP) , is an economical line which can be applied at all waterprofing details of construction.

Defination of polibit

When admin, user or guesst click polibit's link view this page. This page give description about polibit and we have polibit picture and table about cost of insulation. We view picture and table becouse of database.

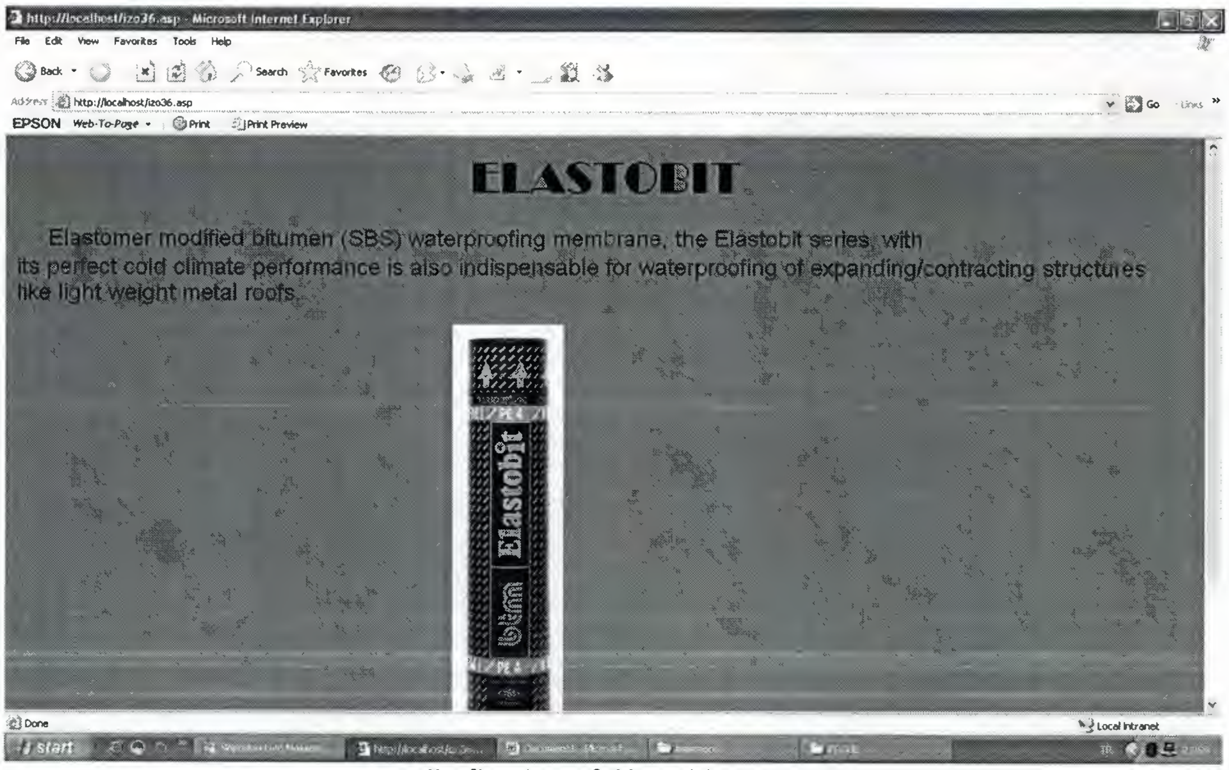

http://localhost/izo35.asp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/izo35.asp
EPSON Web-To-Page Print Print Preview
Done
Local intranet
start

PLASTOBIT

Plastomer polymer bitumen PLASTOBIT waterproofing membranes can be used safely at almost every climatic conditions except on highlands and freezing areas as a consequence of its high quality POLIMER bitumen modified with APP and other thermoplastics and admits the technique of high adherence at torch on application.

Defination of Plastobit.

When admin, user or guesst click plastobit's link view this page. This page give description about plastobit and we have plastobit picture and table about cost of insulation. We view picture and table becouse of database.

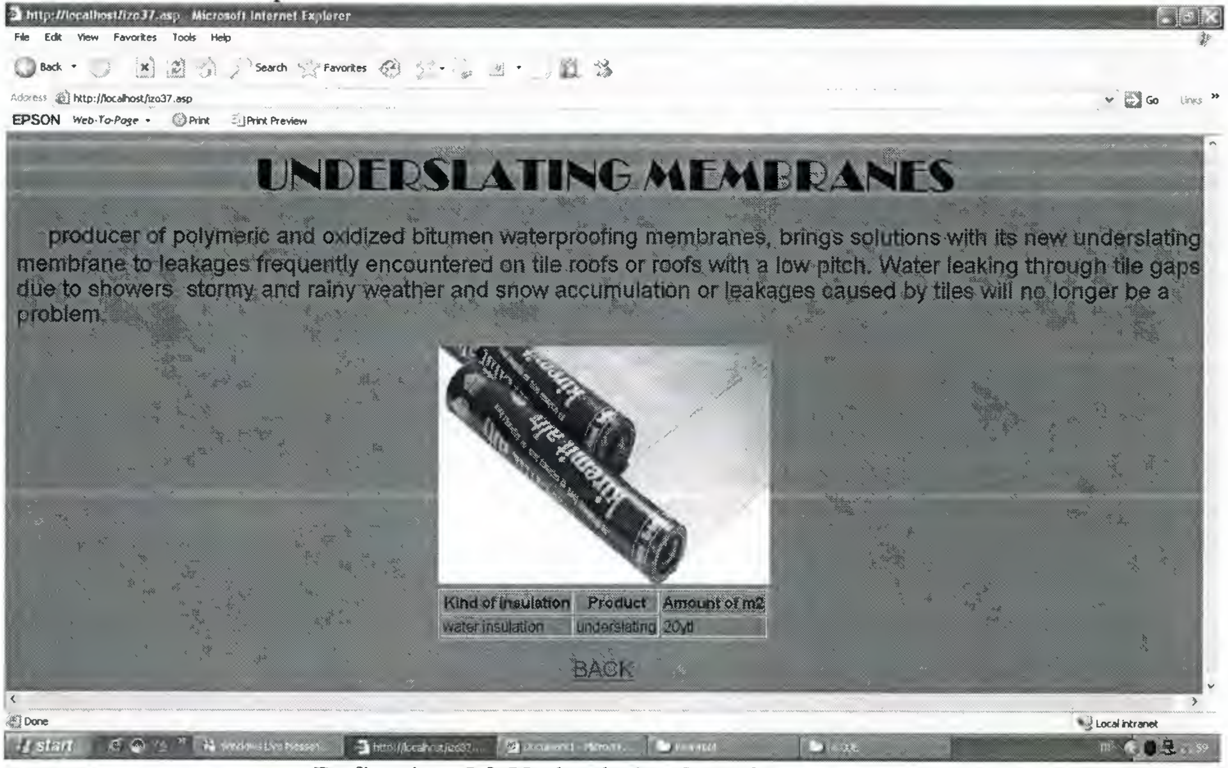

http://localhost/izo36.asp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/izo36.asp
Go Links
EPSON Web-To-Page Print Print Preview
Done
Local intranet
start

ELASTOBIT

Elastomer modified bitumen (SBS) waterproofing membrane, the Elastobit series, with its perfect cold climate performance is also indispensable for waterproofing of expanding/contracting structures like light weight metal roofs.

Defination of Elastobit

When admin, user or guess click elastobit's link view this page. This page give description about elastobit and we have elastobit picture and table about cost of insulation. We view picture and table because of database.


http://localhost/izo37.asp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/izo37.asp
Go Links
EPSON Web-To-Page Print Print Preview
Done
Local intranet
start

UNDERSLATING MEMBRANES

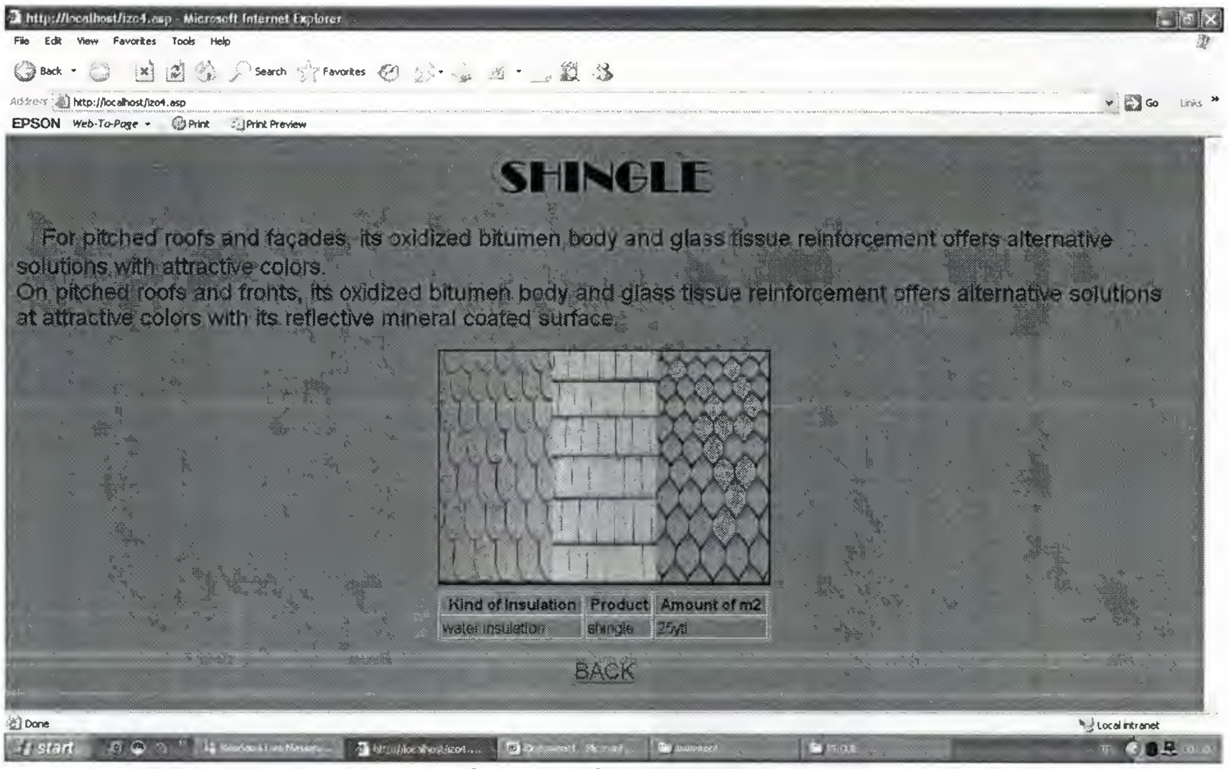
producer of polymeric and oxidized bitumen waterproofing membranes, brings solutions with its new underslating membrane to leakages frequently encountered on tile roofs or roofs with a low pitch. Water leaking through tile gaps due to showers stormy and rainy weather and snow accumulation or leakages caused by tiles will no longer be a problem.

Kind of insulation	Product	Amount of m2
water insulation	underslating	20ytl

BACK

Defination Of Underslating Membranes

When admin, user or guesst click underslating membranes's link view this page. This page give description about underslating membranes and we have underslating membranes picture and table about cost of insulation. We view picture and table becouse of database.


http://localhost/izo4.asp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/izo4.asp
Go Links
EPSON Web-To-Page Print Print Preview
Done
Local intranet
start

SHINGLE

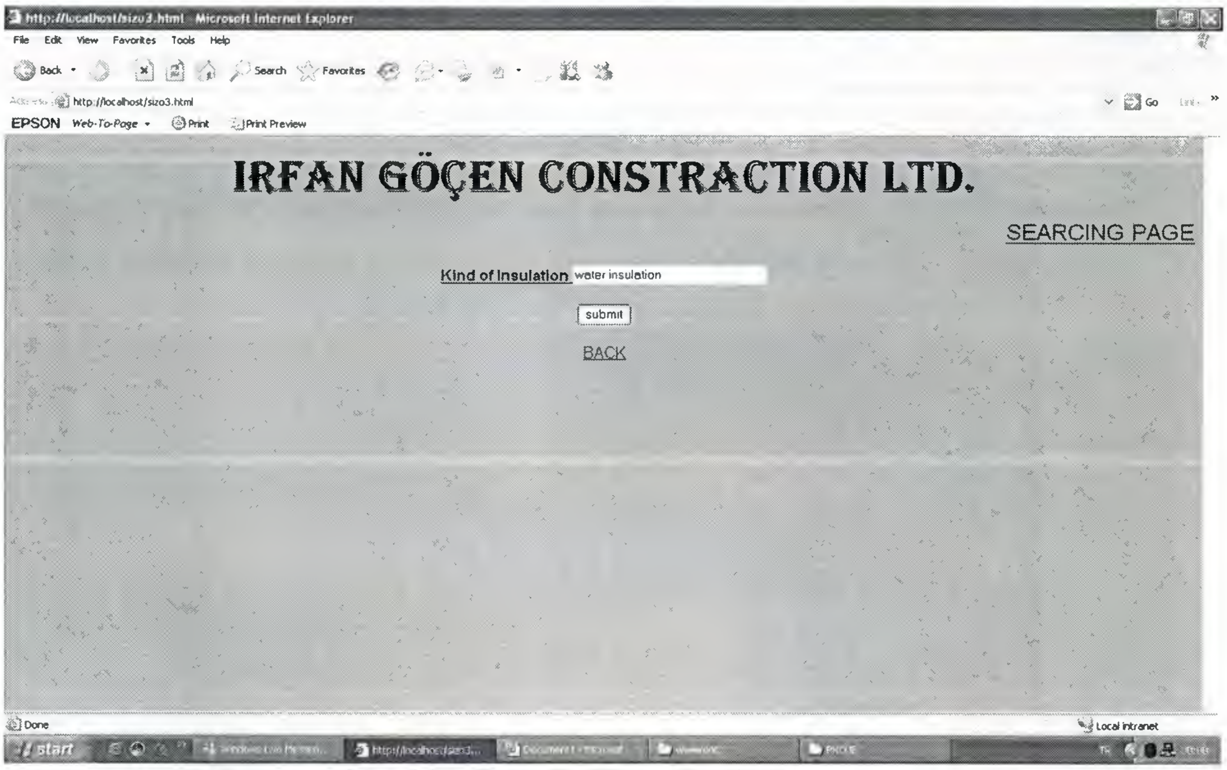
For pitched roofs and façades, its oxidized bitumen body and glass tissue reinforcement offers alternative solutions with attractive colors.
On pitched roofs and fronts, its oxidized bitumen body and glass tissue reinforcement offers alternative solutions at attractive colors with its reflective mineral coated surface.

Kind of Insulation	Product	Amount of m2
water insulation	shingle	25ytl

BACK

Defination of Shingle.

When admin, user or guesst click shingle's link view this page. This page give description about shingle and we have shingle picture and table about cost of insulation. We view picture and table becouse of database.


http://localhost/sizo3.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/sizo3.html
Go Links
EPSON Web-To-Page Print Print Preview
Done
Local intranet
start

IRFAN GÖÇEN CONSTRACTION LTD.

SEARCING PAGE

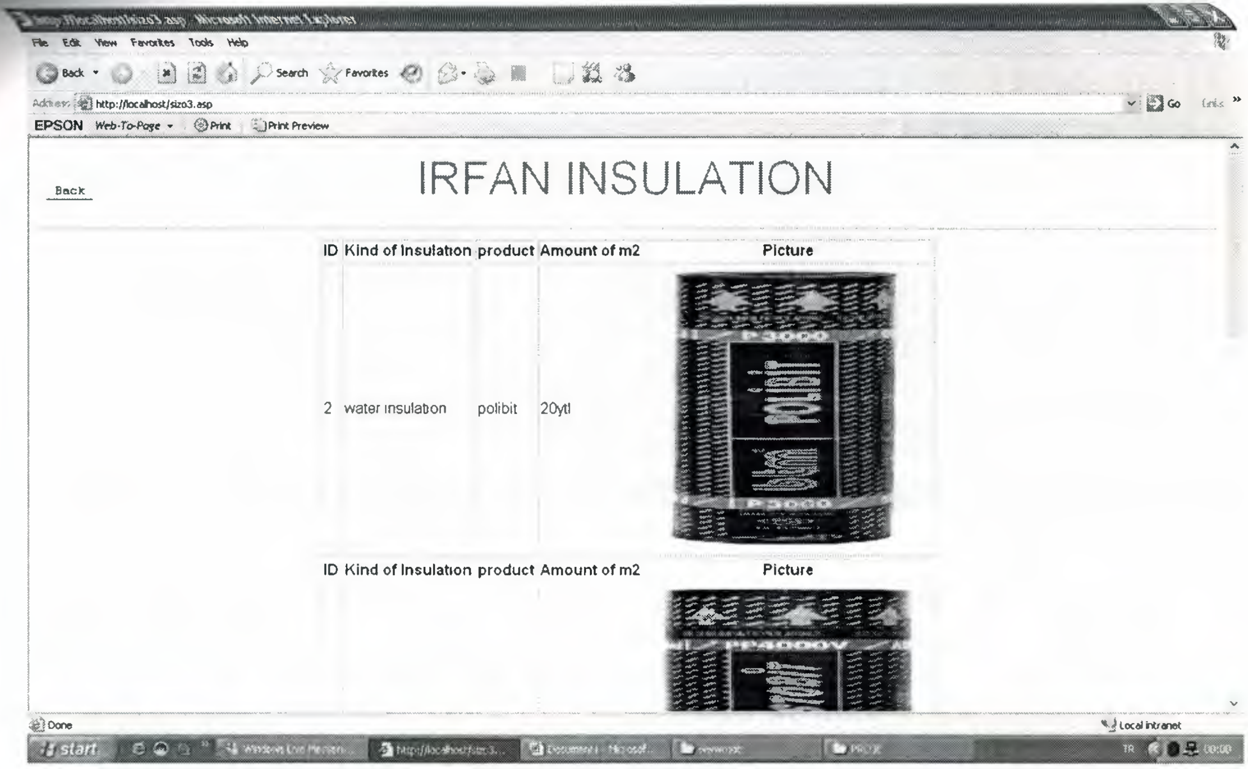
Kind of Insulation: water insulation

submit

BACK

Searching Page To Water Insulation

When admin, user and guesst click this page can search water insulation to see every description about this kind of insulation. And see as below page.



http://localhost/sizo3.asp

Back

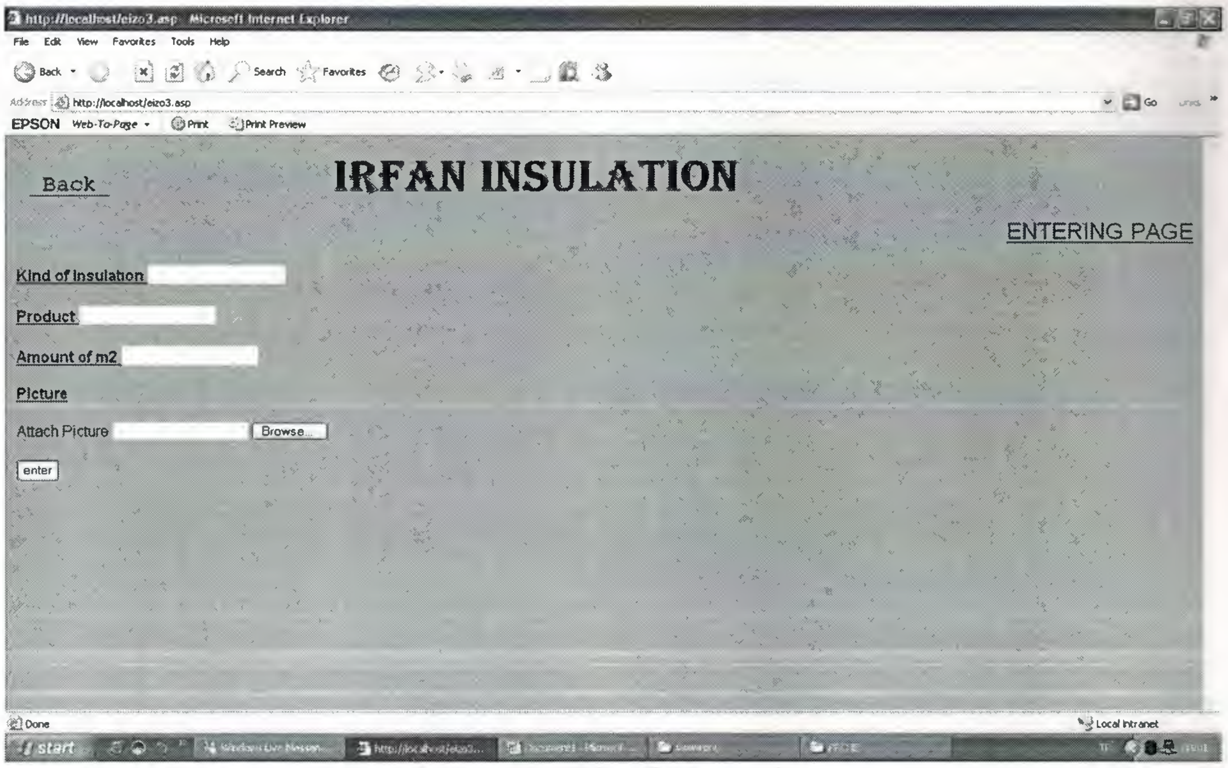
IRFAN INSULATION

ID	Kind of Insulation	product	Amount of m2	Picture
2	water insulation	polibit	20ytl	

ID	Kind of Insulation	product	Amount of m2	Picture

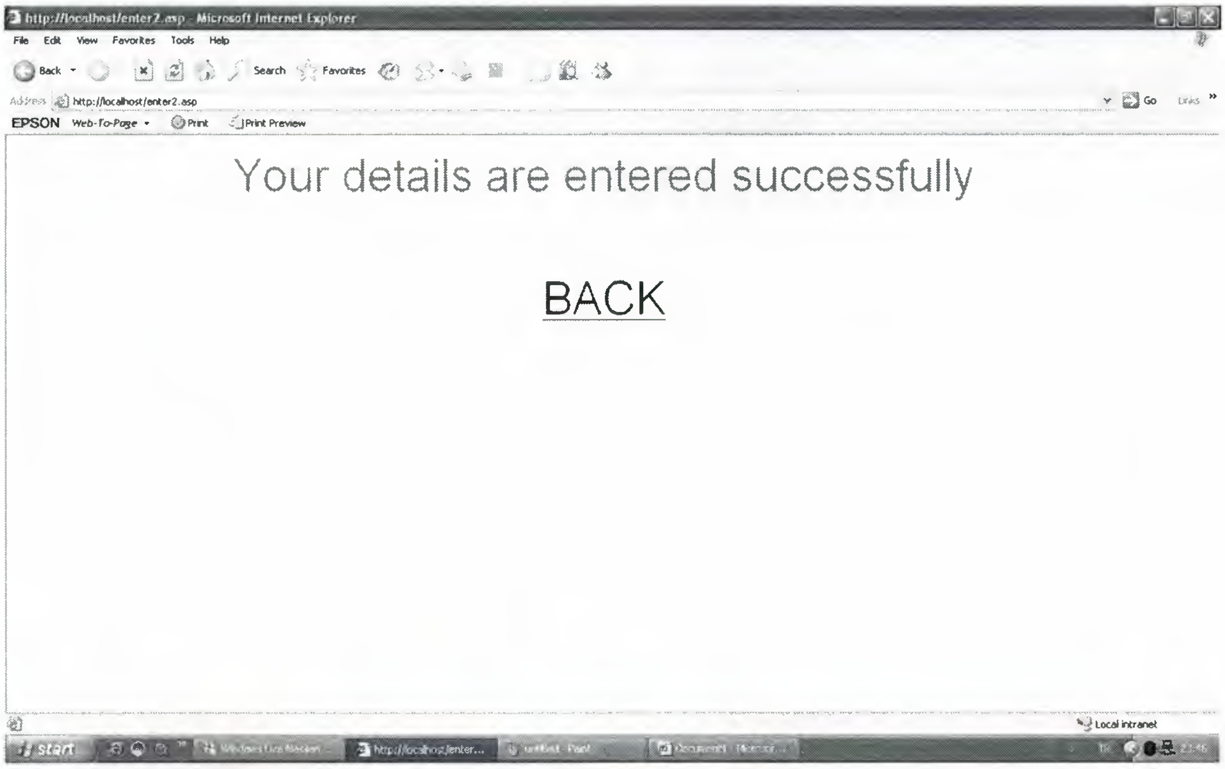
Water Insulation Table

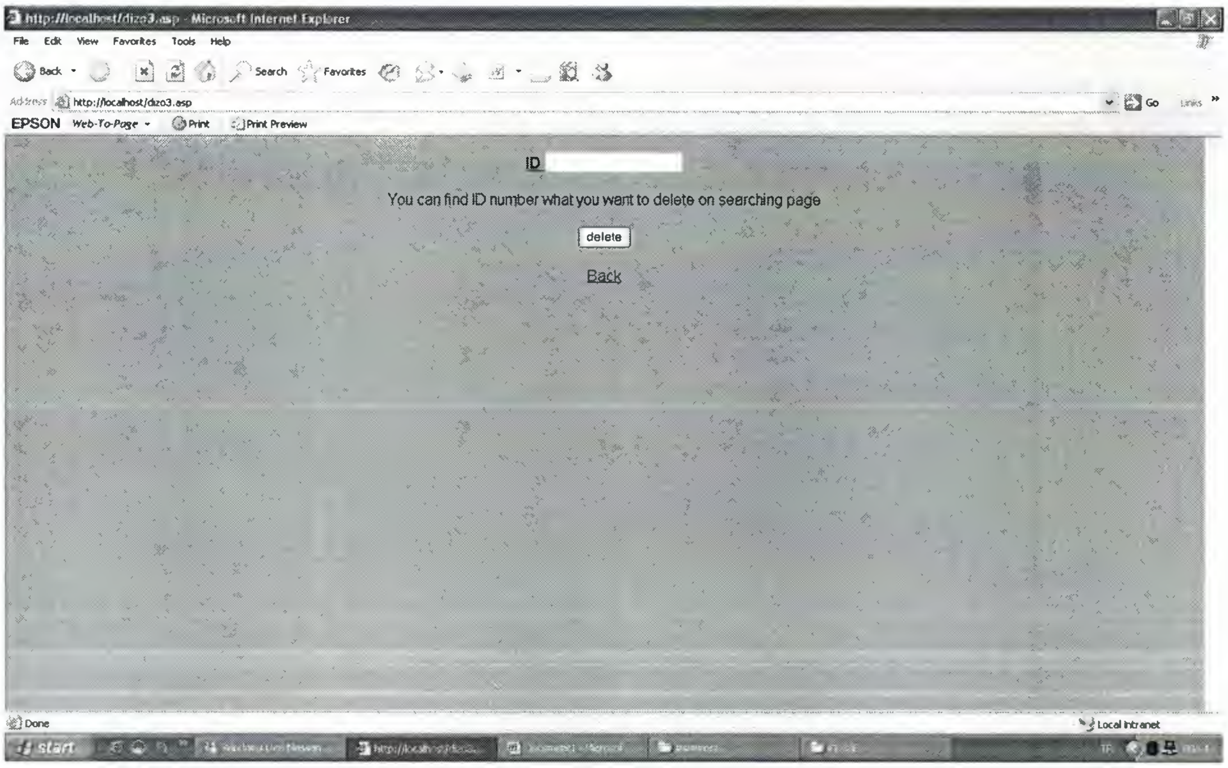
This page result of search.admin, user and guesst see ıd number, kind of insulation, product, amount of m2 and picture. View becouse of database.



Entering Page For Kind Of Insulations

This page for enter the insulation details in the textfield and insert database this details. We have descriptions and textfields to fill them. And we have submit button. we fill in textfields and we have can attach picture when click browse open file choose and we can select insulation picture then click submit button we insert theese details in the database. And also we have back link to turn before of this page. And if your details entered successfully you see as below page.


http://localhost/enter2.asp - Microsoft Internet Explorer
Your details are entered successfully
BACK


http://localhost/dizo3.asp - Microsoft Internet Explorer

ID

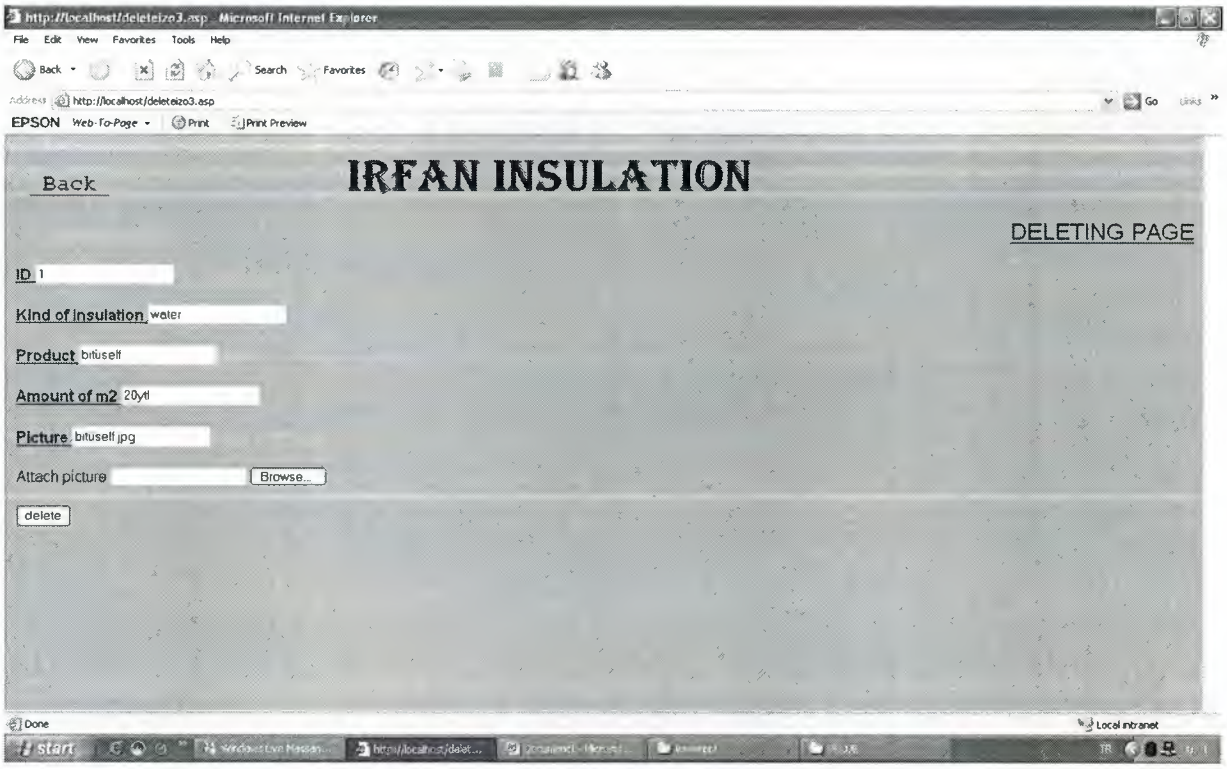
You can find ID number what you want to delete on searching page

delete

Back

1. Step To Deleting

This page first step to deleting. Admin can delete every thing about insulations. Admin write insulation ıd in the texfield. Then view below page


http://localhost/deleteizo3.asp - Microsoft Internet Explorer

Back

IRFAN INSULATION

DELETING PAGE

ID 1

Kind of Insulation water

Product bitüself

Amount of m2 20ytl

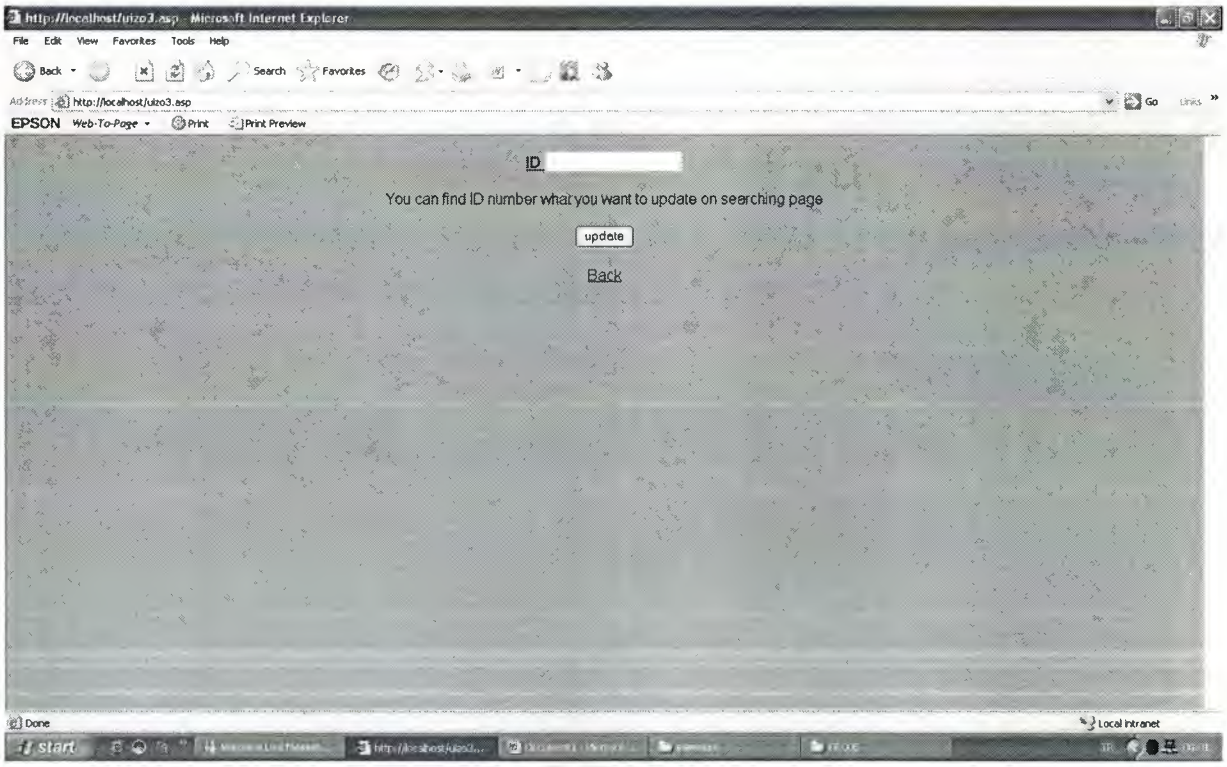
Picture bitüself.jpg

Attach picture Browse...

delete

Deleting Page For Insulations

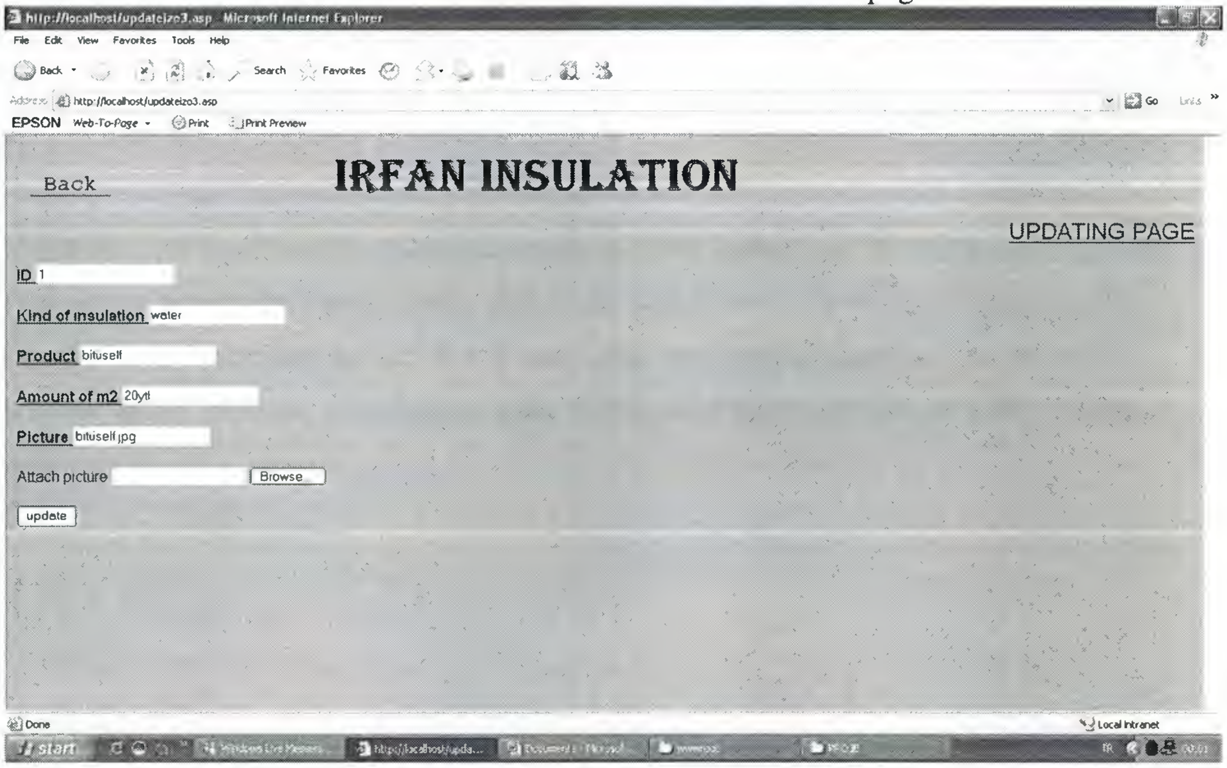
This page for delete the insulation details in the database. We have descriptions and fill textfields. And we have submit button. When admin click submit button delete the insualtion in the database. And also we have back link to turn before of this page. And if your details deleted successfully you see page to your details deleted successfully.



You can find ID number what you want to update on searching page

1. Step To Updating

This page first step to updating. Admin or user can update every thing about insulations. Admin or user write insualtion ıd in the texfield. Then view below page.


http://localhost/updateizo3.asp - Microsoft Internet Explorer
Back
IRFAN INSULATION
UPDATING PAGE
Browse
update

ID 1

Kind of insulation water

Product bituself

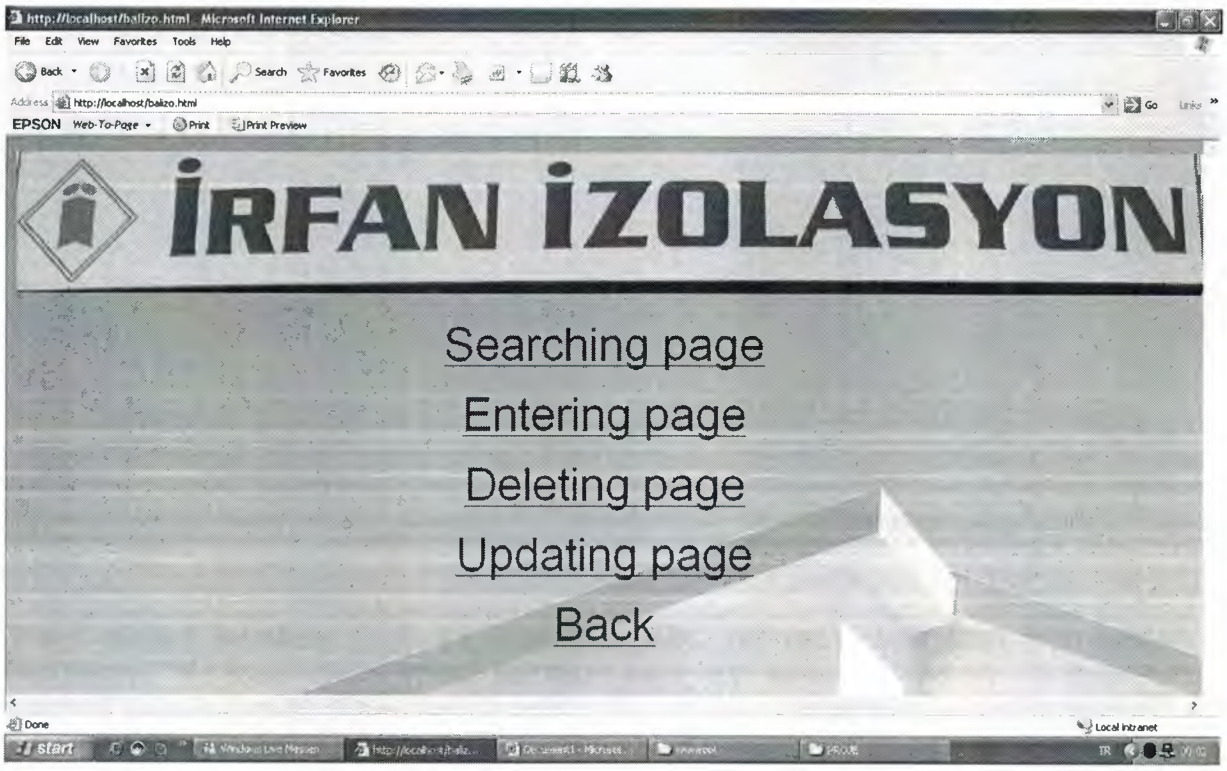
Amount of m2 20ytl

Picture bituself.jpg

Attach picture

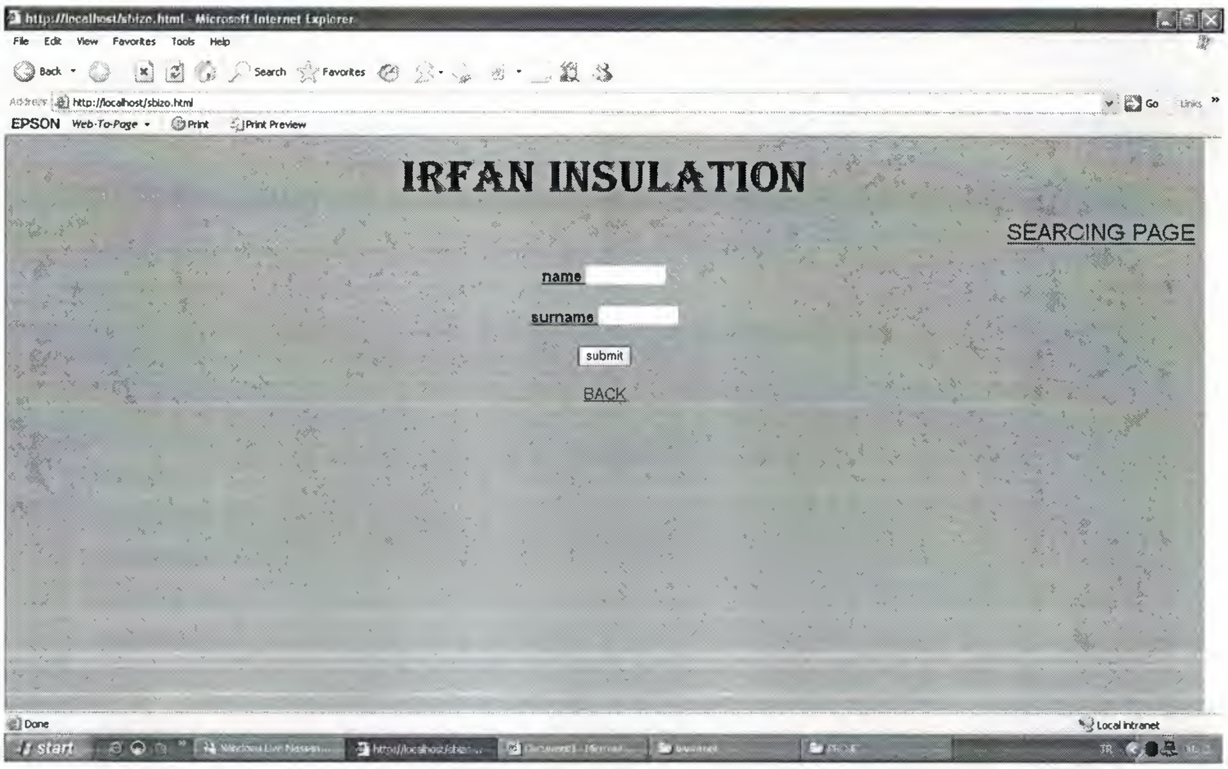
Updating Page For Insulations

This page for update the insulation details in the database. We have descriptions and fill textfields. And we have submit button. When admin or user change in the textfield or textfields then click submit button to update the insulation details in the database. And also we have back link to turn before of this page. And if your details updated successfully you see page to your details updated successfully.


http://localhost/balizo.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/balizo.html
İRFAN İZOLASYON
Searching page
Entering page
Deleting page
Updating page
Back
Done
Local intranet

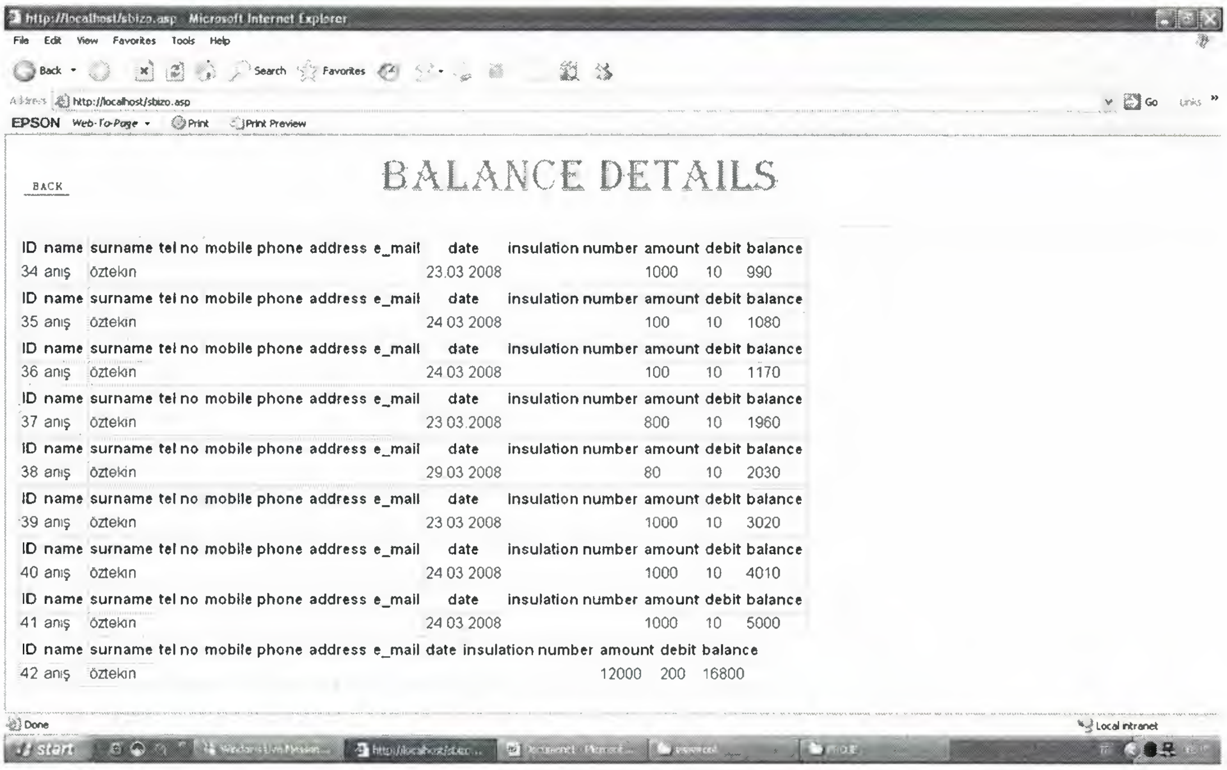
Insulationtion to Balance

When we click insulation to balance this page is view. Admin can see this page.
And admin can make search, enter, delete and update so we have four links to make search, enter, delete and update. But if user enter the program user just see three links without delete, becouse user can not make delete. And back link to back before of this page.

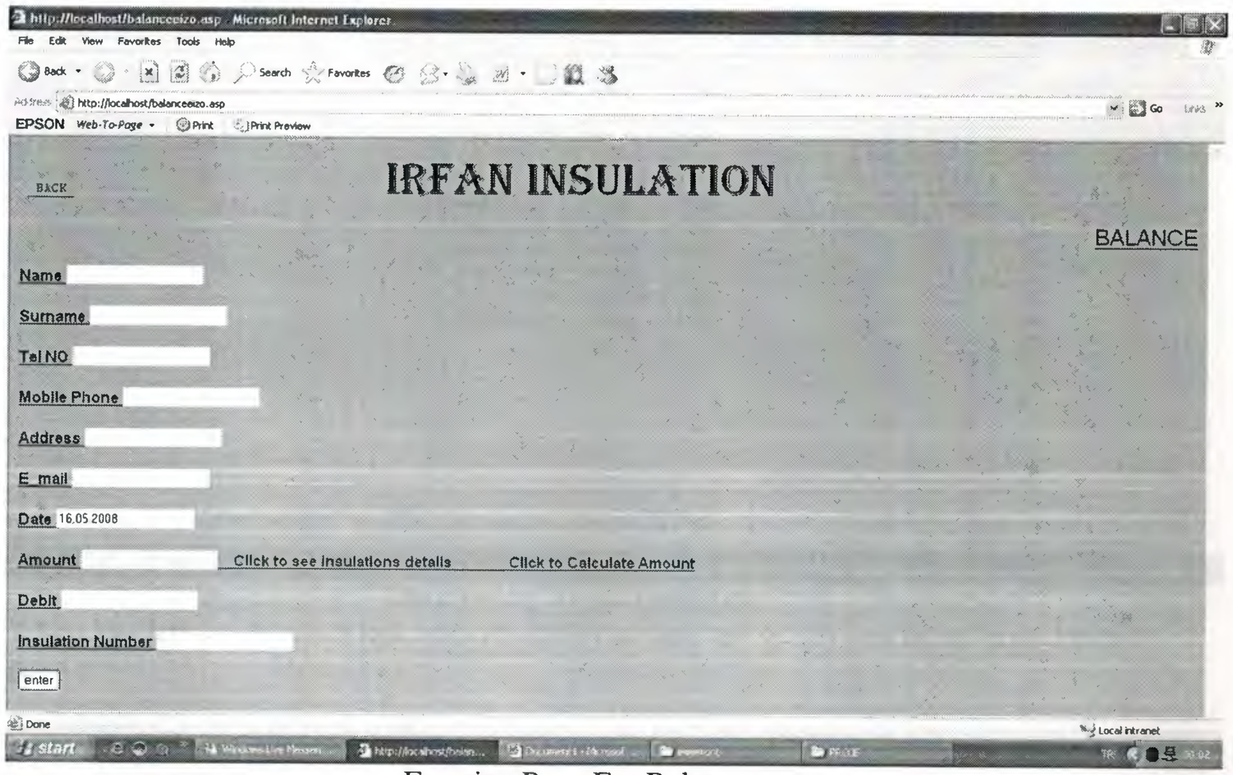


Searching Page

When clicked searching page we view this page. We can search with who buy insulations. We have two textfield for enter name and surname. And we fill theese textfield to start search. When we fill textfields and click submit button we see below page to this customer which kind of insulation buyed and what he/she paid and when and what the balance. We can see theese details becouse of database. And back link to back before of this page.



ID	name	surname	tel no	mobile phone	address	e_mail	date	insulation number	amount	debit	balance
34	anış	öztekın					23.03 2008		1000	10	990
ID	name	surname	tel no	mobile phone	address	e_mail	date	insulation number	amount	debit	balance
35	anış	öztekın					24 03 2008		100	10	1080
ID	name	surname	tel no	mobile phone	address	e_mail	date	insulation number	amount	debit	balance
36	anış	öztekın					24 03 2008		100	10	1170
ID	name	surname	tel no	mobile phone	address	e_mail	date	insulation number	amount	debit	balance
37	anış	öztekın					23 03.2008		800	10	1960
ID	name	surname	tel no	mobile phone	address	e_mail	date	insulation number	amount	debit	balance
38	anış	öztekın					29 03 2008		80	10	2030
ID	name	surname	tel no	mobile phone	address	e_mail	date	insulation number	amount	debit	balance
39	anış	öztekın					23 03 2008		1000	10	3020
ID	name	surname	tel no	mobile phone	address	e_mail	date	insulation number	amount	debit	balance
40	anış	öztekın					24 03 2008		1000	10	4010
ID	name	surname	tel no	mobile phone	address	e_mail	date	insulation number	amount	debit	balance
41	anış	öztekın					24 03 2008		1000	10	5000
ID	name	surname	tel no	mobile phone	address	e_mail	date	insulation number	amount	debit	balance
42	anış	öztekın							12000	200	16800


http://localhost/balanceeizo.asp - Microsoft Internet Explorer

http://localhost/balanceeizo.asp

BACK

IRFAN INSULATION

BALANCE

Name

Surname

Tel NO

Mobile Phone

Address

E_mail

Date 16.05.2008

Amount Click to see insulations details Click to Calculate Amount

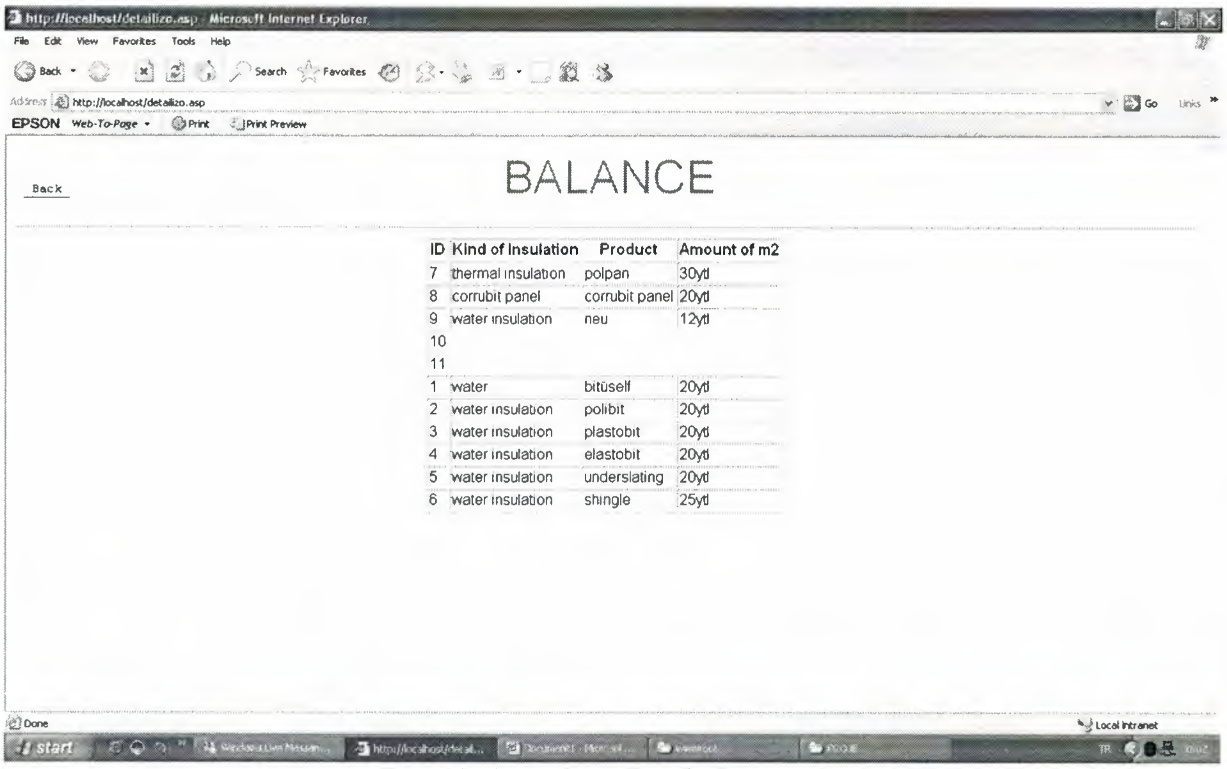
Debit

Insulation Number

enter

Entering Page For Balance

This page for enter the balance details in the textfield and insert database this details. We have descriptions and textfields to fill them. And we have submit button.we have date description and automatic written day date in the textfield. And we have 2 links next to amount for see insulation details for learn cost of insulation. The other link to calculate amount. we fill in textfields then click submit button we insert theese details in the database. And if your details entered successfully you see detail page to inset be successfully.And also we have back link to turn before of this page.

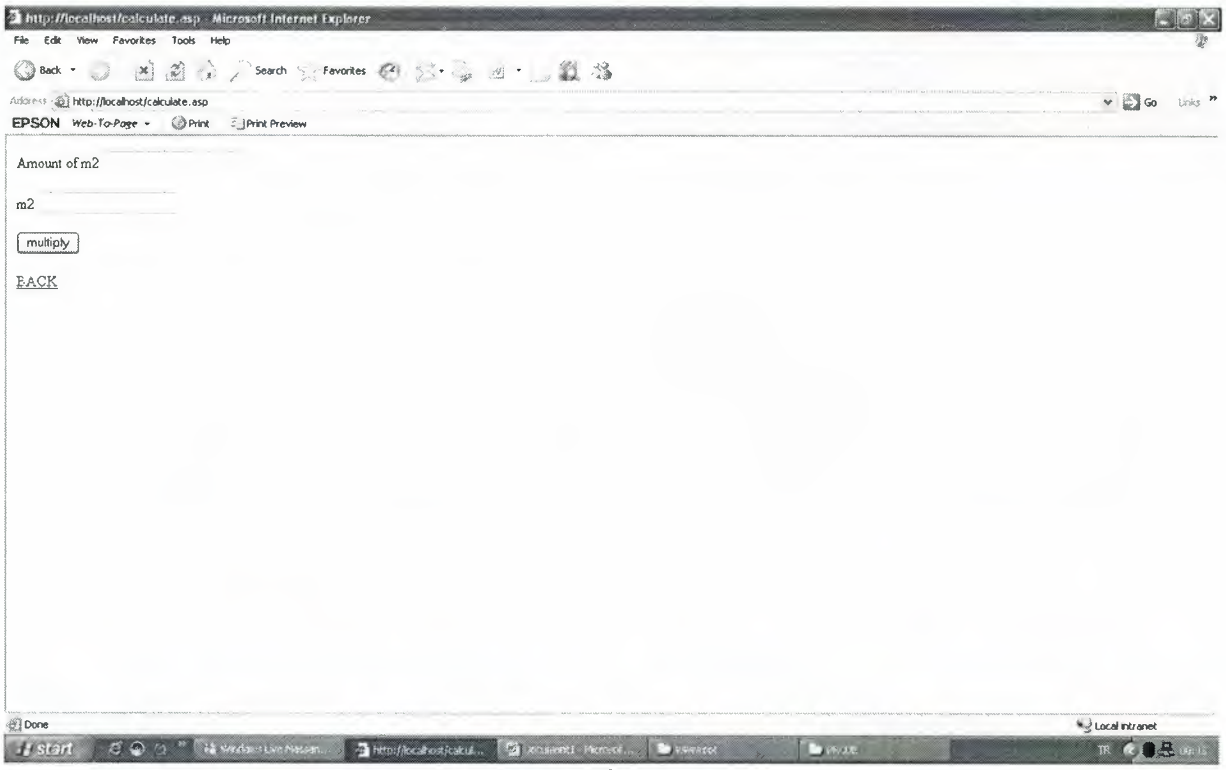

BALANCE

Back

ID	Kind of Insulation	Product	Amount of m2
7	thermal insulation	polpan	30ytl
8	corrubit panel	corrubit panel	20ytl
9	water insulation	neu	12ytl
10			
11			
1	water	bitüself	20ytl
2	water insulation	polibit	20ytl
3	water insulation	plastobit	20ytl
4	water insulation	elastobit	20ytl
5	water insulation	underslating	20ytl
6	water insulation	shingle	25ytl

Description About Insulation.

When insulations details's link clicked admin or user see which insulation how much it is.


Amount of m2

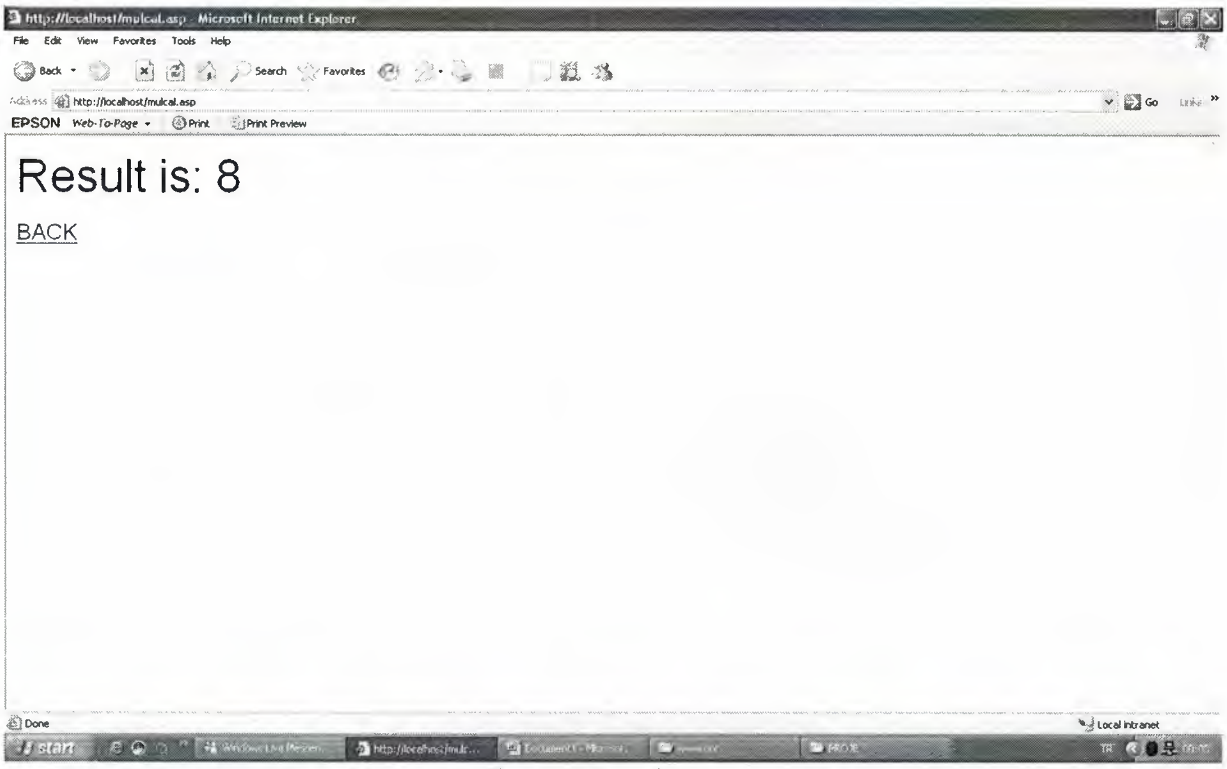
m2

multiply

BACK

Calculation Of Amount

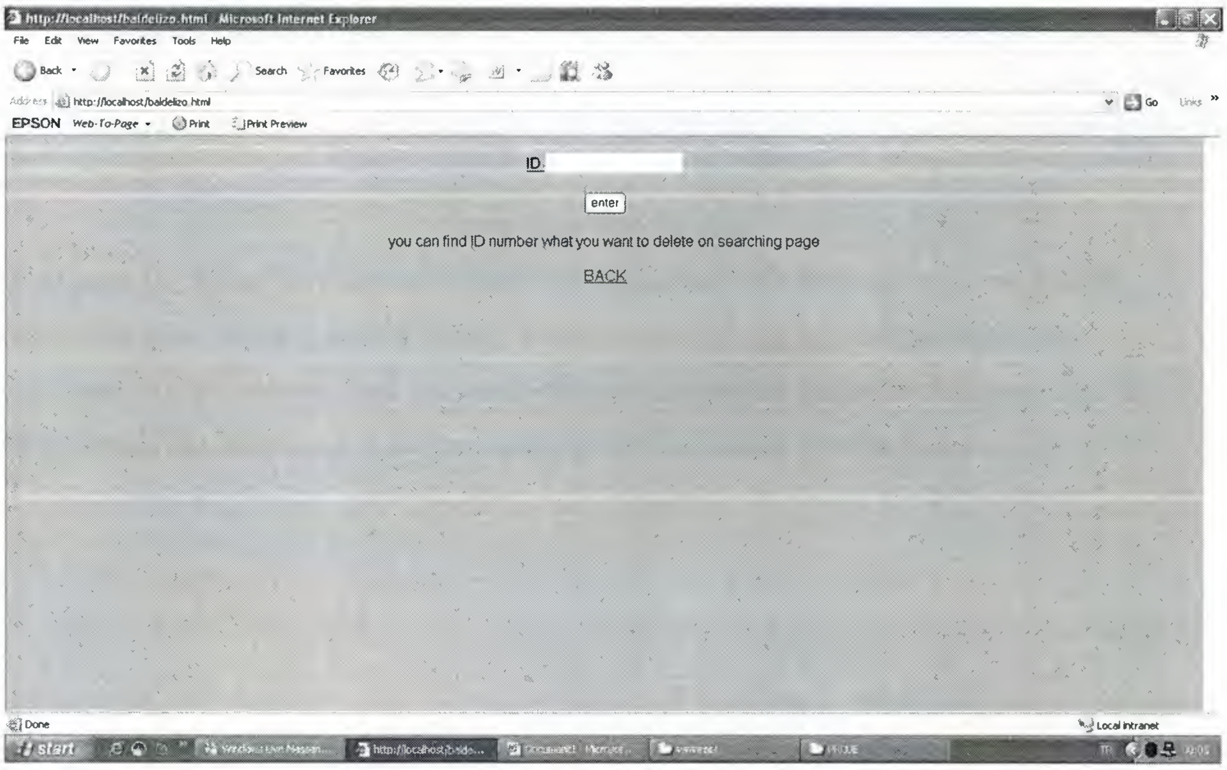
When calculate amount link clicked admin or user fill first amount of m2 and second fill how many m2 the customer buying. Then click multiply button and then see result as next page.


http://localhost/mulcal.asp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
http://localhost/mulcal.asp

Result is: 8

BACK

Showing result


http://localhost/baldelizo.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
http://localhost/baldelizo.html

ID.

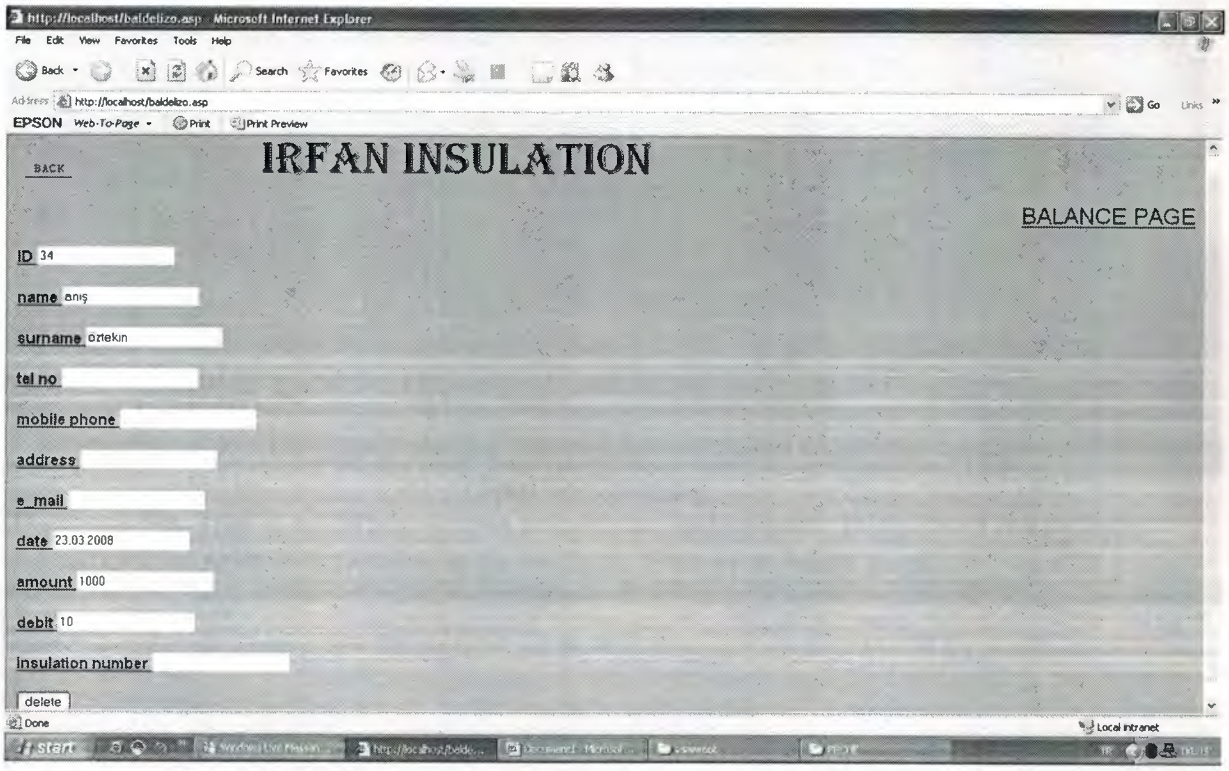
enter

you can find ID number what you want to delete on searching page

BACK

1. Step To Deleting

This page first step to deleting. Admin can delete every thing about customers. Admin write customer ıd in the texfield. Then view below page.


BACK

IRFAN INSULATION

BALANCE PAGE

ID 34

name anış

surname öztekın

tel no

mobile phone

address

e_mail

date 23.03.2008

amount 1000

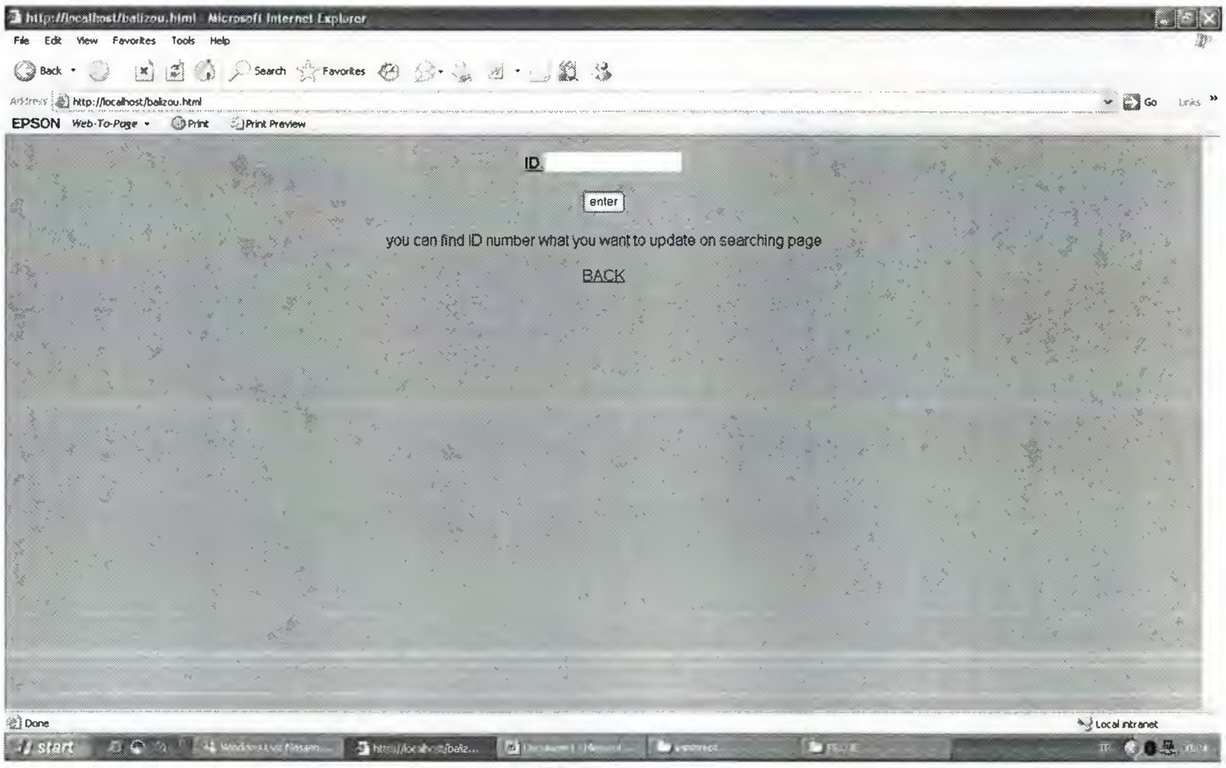
debit 10

insulation number

delete

Deleting Page For Balance

This page for delete the customer details in the database. We have descriptions and fill textfields. And we have submit button. When admin click submit button delete the customer in the database. And also we have back link to turn before of this page. And if your details deleted successfully you see page to your details deleted successfully.


ID

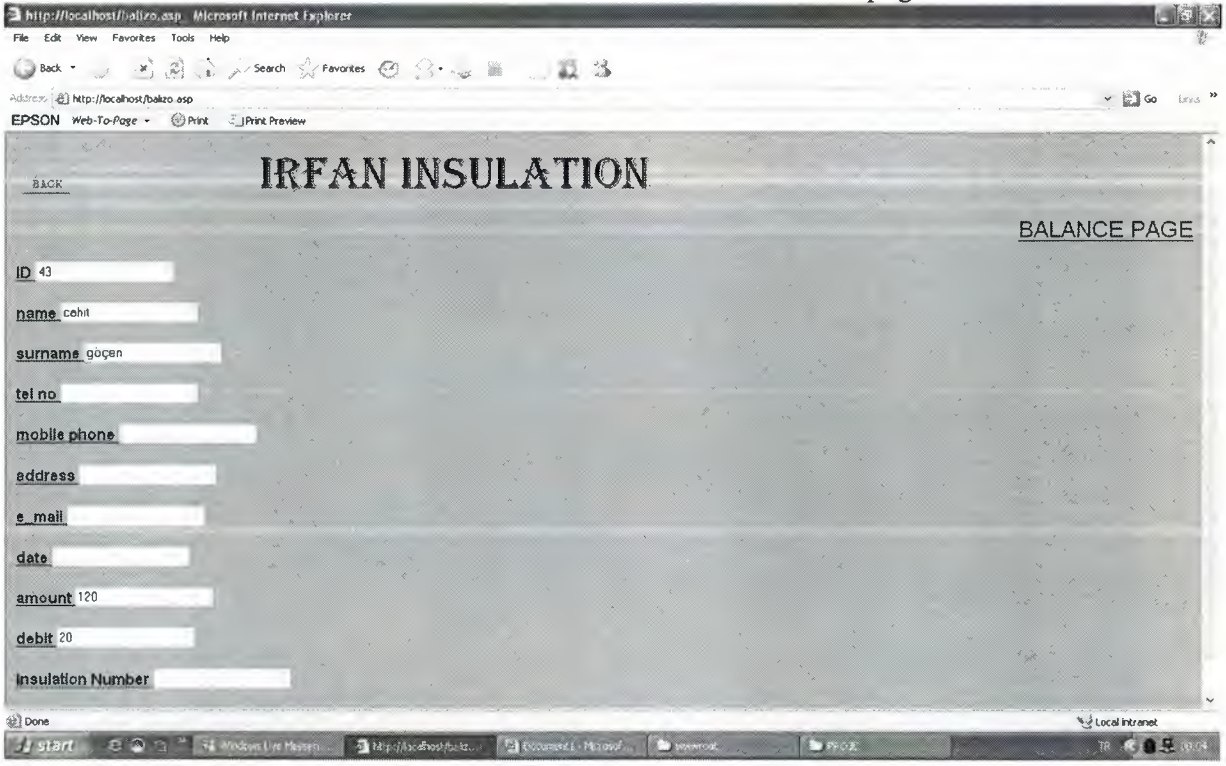
enter

you can find ID number what you want to update on searching page

BACK

1. Step To Updating

This page first step to updating. Admin or user can update every thing about insulations. Admin or user write insualtion ıd in the texfield. Then view below page.


IRFAN INSULATION

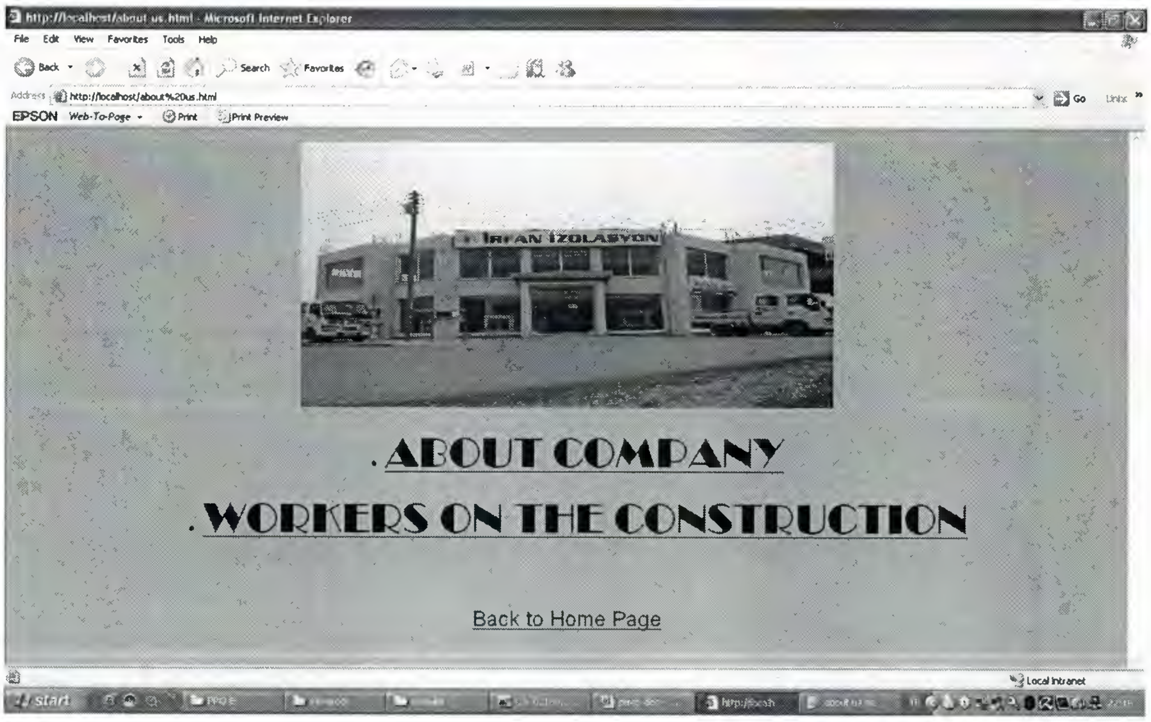
BACK

BALANCE PAGE

ID	43
name	cahit
surname	göçen
tel no	
mobile phone	
address	
e_mail	
date	
amount	120
debit	20
insulation Number	

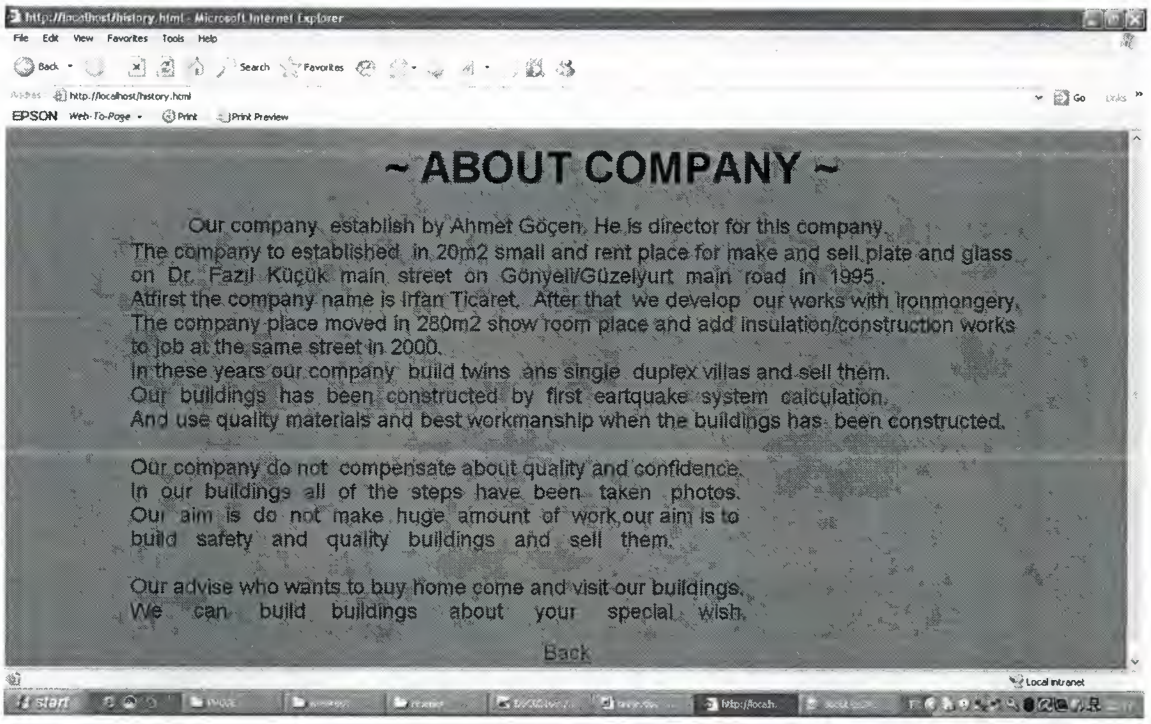
Updating Page For Balance

This page for update the customer details in the database. We have descriptions and fill textfields. And we have submit button. When admin or user change in the textfield or textfields then click submit button update the customer details in the database. And also we have back link to turn before of this page. And if your details updated successfully you see page to your details updated successfully.



About Company

This page for see details about company. We have two links. First for about company, other link about workers details. And one link to back home page.



~ ABOUT COMPANY ~

Our company establish by Ahmet Göçen. He is director for this company.
The company to established in 20m2 small and rent place for make and sell plate and glass on Dr. Fazıl Küçük main street on Gönyeli/Güzelyurt main road in 1995.
Atfirst the company name is Irfan Ticaret. After that we develop our works with ironmongery.
The company place moved in 280m2 show room place and add insulation/construction works to job at the same street in 2000.
In these years our company build twins ans single duplex villas and sell them.
Our buildings has been constructed by first eartquake system calculation.
And use quality materials and best workmanship when the buildings has been constructed.

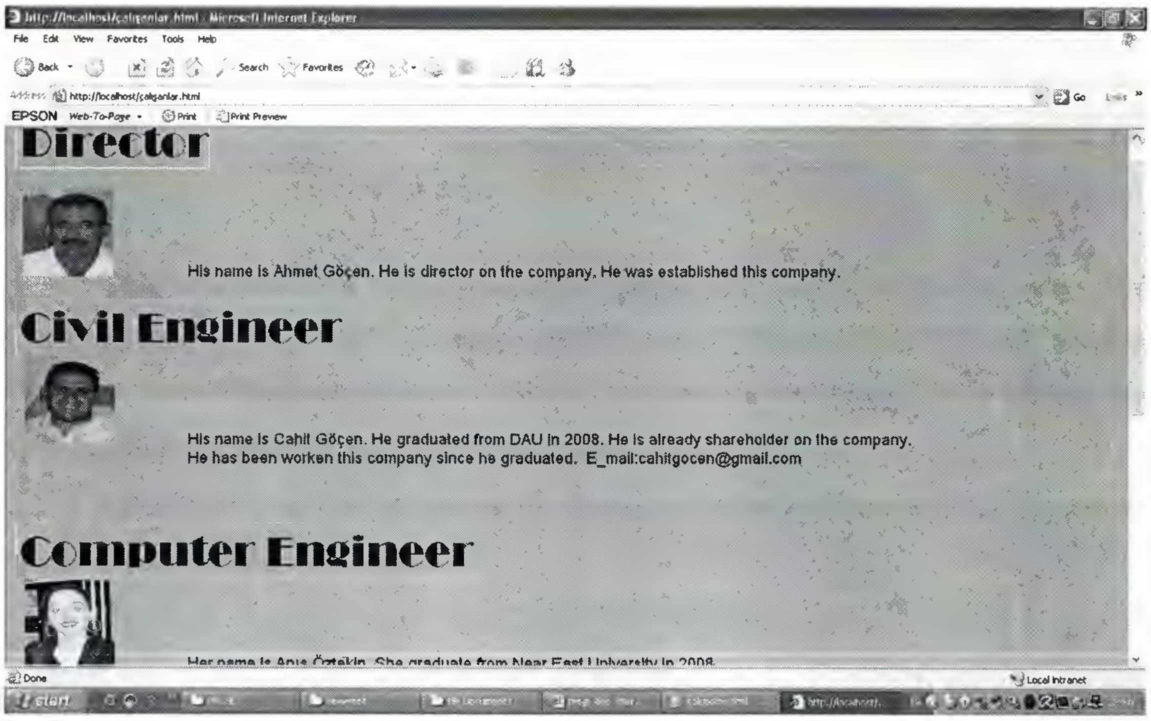
Our company do not compensate about quality and confidence.
In our buildings all of the steps have been taken photos.
Our aim is do not make huge amount of work,our aim is to build safety and quality buildings and sell them.

Our advise who wants to buy home come and visit our buildings.
We can build buildings about your special wish.

Back

Company Details

When about company link clicked this page shown. This page for history of the company, and what company do.


http://localhost/çalışanlar.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/çalışanlar.html
Go Links
EPSON Web-To-Page Print Print Preview
Done
Local intranet
start

Director

His name is Ahmet Göçen. He is director on the company. He was established this company.

Civil Engineer

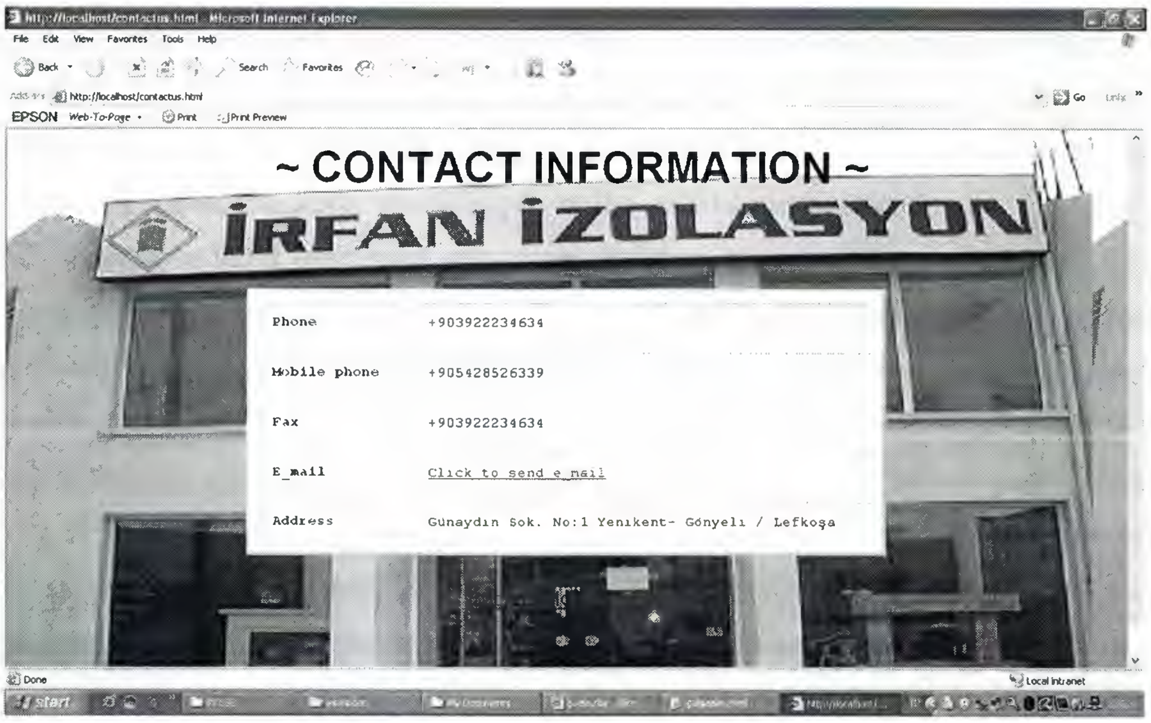
His name is Cahit Göçen. He graduated from DAU in 2008. He is already shareholder on the company.
He has been worken this company since he graduated. E_mail:cahitgocen@gmail.com

Computer Engineer

Her name is Anıs Öztekin. She graduate from Near East University in 2008.

Workers on the construction

When worker on the construction's link clicked open this page for see details about workers.


http://localhost/contactus.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/contactus.html
Go Links
EPSON Web-To-Page Print Print Preview
İRFAN İZOLASYON
Done
Local intranet
start

~ CONTACT INFORMATION ~

Phone	+903922234634
Mobile phone	+905428526339
Fax	+903922234634
E_mail	Click to send e_mail
Address	Gunaydın Sok. No:1 Yenıkent- Gönyelı / Lefkoşa

Contact to the company

When contact us's link clicked open this page for see contact details of the company. And can make contact with the company.

CONCLUSION

This program is a online financial system for a construction company. The program was implemented by using HTML and ASP and the SQL and Access Database. All of them are powerful software and help me very much. Html and asp are very useful pragramming language to create web pages.

HTML, unlike other programming languages where commands and statements are used with certain syntax rules.
HTML has tags. Tags are identifiers of lines of statements issued to translate document contetnt into words, pictures, colours and other web items by the browsers.html has text alignment on the browser, text can be aligned left, center, or right using align statement. And html has links. Hyperlinks and mail links are very important on web page. They provides connections to other web page/web sites and people. Table in the HTML mainly used to tabulate information. Very often used to partition the page for design. A form is collection of components used to accept and transmit data between the client and the web server. Get usually accept a single item and initiated by the server. Post accepts more than a sigle item to transmit and initiated by the user. Action sets the address (url or web page) that will receive the client data. Form components are text, password, radio, checkbox, scroll/dropdown list, submit, clear buttons.

ASP to build interactive web pages, scripting language like vbscript used this project. Vbscript is an offspring of the popular Visual Basic language that runs on the web server for each user through their browsers. Vbscript statements request statement and it divide two parts through form components and request from hyperlinks. The other vbscript statements is response statement. Creating interactive web pages often require that data used in transactions carried out are registered to databases. Scripts used in asp can connect to popular databases and user friendly interfaces are built for data transmission and manipulation over the net.The databases applications on the net, basically uses three steps for data communications; db connection, data manipulation and abstraction, safely closing and disconnection of db. Database connection with ASP used Microsoft Access Driver. SQL command used to manipulation of datas.

REFERENCES

[1] http://en.wikipedia.org/wiki/html

[2] http://www.w3.org/EN/html

[3] http://www.codefixer.com/tutorials/what_can_it_do.asp

[4] http://en.wikipedia.org/wiki/sql

[5] http://en.wikipedia.org/wiki/datadase

[6] David Buser, John Kauffman, "Active Server Pages 3.0". United States Of America, Press(2003)

[7] Nick Vandome, "Creating web pages in easy steps". United Kingdom, Press(2004)

[8] Ümit İlhan, Internet Programming Lecture Notes

APPENDIX A
PROGRAM CODE

A.1. default.html

```
<html>
<body bgcolor="#228B22">
<br>
<H1 align="center"><font face="algerian" size="20" > IRFAN GÖÇEN
CONSTRUCTION LTD. </H1>
<br>
<pre> <b>  <size="10">        <a href="admin.html"> ADMIN </a>
            <a href="user.html"> USER </a>                     <a
href="mainpage.html"> GUESS </a> </b> </pre>
<p align="center"> <img src  =  "C:\Documents and Settings\x\My
Documents\resimler\DSC02164.JPG"  width=900 height=500>
</body>
</html>
```

A.2. admin.html

```
<html>
<body bgcolor="#228B22">
<H1> <pre align="left"> <a href="default.html"> BACK </a>
<font face="algerian" size="20" > IRFAN GÖÇEN CONSTRUCTION LTD. </pre>
</H1>
<form method="post" action="user.asp">
<p align="center"> <font face="algerian" size="5"> USER ID   <input
type="text" name="user_id" size="8"> </p>
<p align="center"> <font face="algerian" size="5"> PASSWORD <input
type="password" name="password" size="8" maxlength="4"> </p>
<p align="center"> <input type="submit" value="LOG IN"> </p>
</form>
<p align="center"> <img src  =  "C:\Documents and Settings\x\My
Documents\resimler\DSC02164.JPG"  width=900 height=500>
</body>
</html>
```

A.3. user.asp

```
<%@Language="VBScript"%>
<%u=Request.form("user_id")
p=Request.form("password")
set db=server.createobject("ADODB.Connection")
db.open"Driver={Microsoft Access Driver
(*.mdb)};DBQ="&server.mappath("person.mdb")
SQL="Select * from personn where USER_ID='"&u&"' and PASSWORD='"&p&"'
"
set rec=db.Execute(SQL)
if rec.eof then
response.redirect "admin.html"
else
response.redirect "mainpagee.html"
end if%>
```

A.4. mainpagee.html

```
<HTML>
<Title> Welcome to my web page!!! </Title>
<Body>
<H1> <pre align="left"> <a href="default.html"> BACK </a>         <font
face="algerian" size="20" > IRFAN GÖÇEN CONSTRUCTION LTD.     <font
```

```
size="4"> <a href="usercontrol.html"> CONTROL TO USER </a> </pre>
</H1>
<Table  width="100%" Border=1 cellspacing=2 cellpadding=3
bgcolor="228B22">
<Tr> <TD> <p align="center"> <A Href = "constractions.html">
construction </A>  </TD> <TD> <p align="center"> <A Href =
"isolations.html"> isolation </A> </TD>
<TD> <p align="center"> <A Href= "about us.html"> about us </A>  </TD>
<TD> <p align="center"> <A Href = "contactus.html"> contact us </A>
</TD>
</Table>
<p align="center"> <img src  =  "C:\Documents and Settings\x\My
Documents\resimler\DSC02164.JPG"  width=1100 height=650>
</Body>
</HTML>
```

A.5. usercontrol.html

```
<html>
<body bgcolor="32CD32">
<br>
<H1 align="center"><font face="algerian" size="30" > IRFAN GÖÇEN
CONSTRUCTION LTD. </H1>
<p align="center"> <font face="arial" size="15"> <a
href="searchuser.html"> Searching user </a> </p>
<p align="center"> <font face="arial" size="15"> <a
href="enteruser.asp"> Entering user </a> </p>
<p align="center"> <font face="arial" size="15"> <a
href="deleterecorduser.asp"> Deleting user </a> </p>
<p align="center"> <font face="arial" size="15"> <a
href="updaterecorduser.asp"> Updating user </a> </p>
<p align="center"> <a href= "mainpagee.html"> Back </a> </p>
</body>
</html>
```

A.6. searchuser.html

```
<html>
<body bgcolor="32CD32">
<H1 align="center"><font face="algerian" size="30" > IRFAN GÖÇEN
CONSTRACTION LTD. </H1>
<p align="right"> <u> <font face="arial" size="5"> SEARCING PAGE </u>
</P>
<form method="post" action="suser.asp">
<p align="center"> <b> <u> <font face="arial" size="3"> name   <input
type="text" name="name" size="10" > </u> </b> </p>
<p align="center"> <b> <u> <font face="arial" size="3"> surname
<input type="text" name="surname" size="10" > </u> </b> </p>
<p align="center"> <input type="submit" value="submit"> </p>
<p align="center"> <a href="usercontrol.html"> BACK </a>
</form>
</body>
</html>
```

A.7. suser.asp

```
<%@Language="VBScript"%>
<pre> <a href="searchuser.html"> BACK </a> <font face="arial"
color="#228B22" size="20">                                      USER
</pre>
<hr>
<%n=request.form("name")
s=request.form("surname")
```

```
set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("person.mdb")
sql="select * from personnn where name='"&n&"' and surname='"&s&"'"
set rec=db.execute(sql)
while not rec.eof%>
<table border>
<tr> <th> ID </th> <th> NAME </th> <th> SURNAME </th> <th> USER_ID
</th> <th> PASSWORD </th> <th> TEL </th> <th> ADDRESS </th> <th>
E_MAIL </th> <th> STARTWORKINGDATE </th> <th> STATUS </th> <th>
DEPARTMENT </th> <th> SALARY </th>
<tr> <td> <%=rec("ID1")%> </td>
<td> <%=rec("NAME")%> </td>
<td> <%=rec("SURNAME")%> </td>
<td> <%=rec("USER_ID")%> </td>
<td> <%=rec("PASSWORD")%> </td>
<td> <%=rec("TEL")%> </td>
<td> <%=rec("ADDRESS")%> </td>
<td> <%=rec("E_MAIL")%> </td>
<td> <%=rec("STARTWORKINGDATE")%> </td>
<td> <%=rec("STATUS")%> </td>
<td> <%=rec("DEPARTMENT")%> </td>
<td> <%=rec("SALARY")%> </td>
</table>
<%rec.movenext
wend
rec.close
set rec=nothing
db.close
set db=nothing%>
```

A.8. enteruser.asp

```
<html>
<body bgcolor="32CD32">
<H1> <pre align="center"> <a href="usercontrol.html"> Back </a> <font
face="algerian" size="30" >                    IRFAN GÖÇEN
CONSTRUCTION LTD. </pre> </H1>
<p align="right"> <u> <font face="arial" size="5"> ENTERING USER </u>
</P>
<form method="post" action="enteruser2.asp">
<p> <b> <u> <font face="arial" size="3"> Name  <input type="text"
name="t1" > </u> </b> </p>
<p> <b> <u> Surname <input type="text" name="t2" >  </u> </b> </p>
<p> <b> <u> User Id <input type="text" name="t3" ></u> </b> </p>
<p> <b> <u> Password <input type="text" name="t4" > </u> </b>  </p>
<p> <b> <u> Phone <input type="text" name="t5" ></u> </b> </p>
<p> <b> <u> Address <input type="text" name="t6" ></u> </b> </p>
<p> <b> <u> E_mail <input type="text" name="t7" ></u> </b> </p>
<p> <b> <u> Start Working Date <input type="text" name="t8" ></u> </b>
</p>
<p> <b> <u> Status <input type="text" name="t19" ></u> </b> </p>
<P> <b> <u> Department <input type="text" name="t10" > </u> </b> </p>
<p> <b> <u> Salary <input type="text" name="t11" ></u> </b> </p>
<p> <input type="submit" value="enter"> </p>
</form>
</body>
</html>
```

A.9. enteruser2.asp

```
<%@Language="VBScript"%>
<%s_t1=Request.form("t1")
s_t2=Request.form("t2")
s_t3=Request.form("t3")
s_t4=Request.form("t4")
s_t5=Request.form("t5")
s_t6=Request.form("t6")
s_t7=Request.form("t7")
s_t8=Request.form("t8")
s_t9=Request.form("t9")
s_t10=Request.form("t10")
s_t10=Request.form("t11")
set db=Server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("person.mdb")
sql="Insert into personnn(NAME, SURNAME, USER_ID, PASSWORD, TEL,
ADDRESS, E_MAIL, STARTWORKINGDATE, STATUS,  DEPARTMENT, SALARY)
values('"&s_t1&"','"&s_t2&"', '"&s_t3&"', '"&s_t4&"', '"&s_t5&"',
'"&s_t6&"', '"&s_t7&"', '"&s_t8&"', '"&s_t9&"', '"&s_t10&"',
'"&s_t11&"' )"
db.execute(sql)
db.close
set db=nothing%>
<p align="center"> <font face="arial" color="#228B22" size="20"> Your
details are entered successfully </p>
<br>
<p align="center"> <a href="usercontrol.html"> BACK </a> </p>
```

A.10. deleterecorduser.asp

```
<body bgcolor="32CD32">
<form method="post" action="deleteuser.asp">
<p align="center">           <b> <u> <font face="arial" size="3" > ID
<input type="text" name="t0"> </u> </b> </p>
<p align="center"> you can find ID number of what you want to delete
on searching page. </p>
<p align="center">                        <input type="submit"
value="delete"> </p>
<br>
<br>
<p align="center"> <font face="arial" size="5"> <a
href="usercontrol.html"> Back </a> </p>
</form>
</body>
```

A.11. deleteuser.asp

```
<body bgcolor="32CD32">
<H1> <pre align="center"> <a href="deleterecorduser.asp"> Back </a>
<font face="algerian" size="30" > IRFAN GÖÇEN CONSTRUCTION LTD. </pre>
</H1>
<p align="right"> <u> <font face="arial" size="5"> DELETING PAGE </u>
</P>
<%@Language="VBScript"%>
<%del=cint(request.form("t0"))%>
<%set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("person.mdb")
SQL="select * from personnn where ID1="&del&""
set rec=db.execute(SQL)%>
<%s_t0=rec("ID1")
s_t1=rec("NAME")
```

```
s_t2=rec("SURNAME")
s_t3=rec("USER_ID")
s_t4=rec("PASSWORD")
s_t5=rec("TEL")
s_t6=rec("ADDRESS")
s_t7=rec("E_MAIL")
s_t8=rec("STARTWORKINGDATE")
s_t9=rec("STATUS")
s_t10=rec("DEPARTMENT")
s_t11=rec("SALARY")
rec.close
set rec=nothing
db.close
set db=nothing%>
<form method="post" action="deleteuser2.asp">
<p> <b> <u> <font face="arial" size="3"> ID   <input type="text"
name="t0" value=<%=s_t0%>> </u> </b> </p>
<p> <b> <u> <font face="arial" size="3"> Name   <input type="text"
name="t1" value=<%=s_t1%>> </u> </b> </p>
<p> <b> <u> Surname <input type="text" name="t2" value=<%=s_t2%>>
</u> </b> </p>
<p> <b> <u> User Id <input type="text" name="t3" value=<%=s_t3%>></u>
</b> </p>
<p> <b> <u> Password <input type="text" name="t4"  value=<%=s_t4%>>
</u> </b>  </p>
<p> <b> <u> Phone <input type="text" name="t5" value=<%=s_t5%>></u>
</b> </p>
<p> <b> <u> Address <input type="text" name="t6" value=<%=s_t6%>></u>
</b> </p>
<p> <b> <u> E_mail <input type="text" name="t7" value=<%=s_t7%>></u>
</b> </p>
<P> <b> <u> Start Working Date <input type="text" name="t8"
value=<%=s_t8%>> </u> </b> </p>
<p> <b> <u> Status <input type="text" name="t9" value=<%=s_t9%>></u>
</b> </p>
<p> <b> <u> Department <input type="text" name="t10"
value=<%=s_t10%>></u> </b> </p>
<p> <b> <u> Salary <input type="text" name="t11" value=<%=s_t11%>></u>
</b> </p>
<p> <input type="submit" value="delete"> </p>
</form>
</body>
```

A.12. deleteuser2.asp

```
<%@Language="VBScript"%>
<%dim s_t0
s_t0=CLng(Request.form("t0"))
s_t1=Request.form("t1")
s_t2=Request.form("t2")
s_t3=Request.form("t3")
s_t4=Request.form("t4")
s_t5=Request.form("t5")
s_t6=Request.form("t6")
s_t7=Request.form("t7")
s_t8=Request.form("t8")
s_t9=Request.form("t9")
s_t10=Request.form("t10")
s_t11=Request.form("t11")
set db=Server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("person.mdb")
```

```
sql= "delete from personnn where ID1="&s_t0&""
db.execute(sql)
db.close
set db=nothing%>
<p align="center"> <font face="arial" color="#228B22" size="20"> Your
details are deleted successfully </p>
<br>
<p align="center"> <a href="usercontrol.html"> Back </a> </p>
```

A.13. updaterecorduser.asp

```
<body bgcolor="32CD32">
<form method="post" action="updateuser.asp">
<p align="center"> <b> <u> <font face="arial" size="3" > ID   <input
type="text" name="t0"> </u> </b> </p>
<p align="center"> You can find ID number what you want to update on
searching page. </p>
<p align="center"> <input type="submit" value="update"> </p>
<p align="center"> <a href="usercontrol.html"> Back </a> </p>
</form>
</body>
```

A.14. updateuser.asp

```
<body bgcolor="32CD32">
<H1> <pre align="center"> <a href="updaterecorduser.asp"> Back </a>
<font face="algerian" size="30" > IRFAN GÖÇEN CONSTRUCTION LTD.</pre>
</H1>
<p align="right"> <u> <font face="arial" size="5"> UPDATING PAGE </u>
</P>
<%@Language="VBScript"%>
<%del=cint(request.form("t0"))%>
<%set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("person.mdb")
SQL="select * from personnn where ID1="&del&""
set rec=db.execute(SQL)%>
<%s_t0=rec("ID1")
s_t1=rec("NAME")
s_t2=rec("SURNAME")
s_t3=rec("USER_ID")
s_t4=rec("PASSWORD")
s_t5=rec("TEL")
s_t6=rec("ADDRESS")
s_t7=rec("E_MAIL")
s_t8=rec("STARTWORKINGDATE")
s_t9=rec("STATUS")
s_t10=rec("DEPARTMENT")
s_t11=rec("SALARY")
rec.close
set rec=nothing
db.close
set db=nothing%>
<form method="post" action="updateuser2.asp">
<p> <b> <u> <font face="arial" size="3"> ID   <input type="text"
name="t0" value=<%=s_t0%>> </u> </b> </p>
<p> <b> <u> <font face="arial" size="3"> Name   <input type="text"
name="t1" value=<%=s_t1%>> </u> </b> </p>
<p> <b> <u> Surname <input type="text" name="t2" value=<%=s_t2%>>
</u> </b> </p>
<p> <b> <u> User Id <input type="text" name="t3" value=<%=s_t3%>></u>
</b> </p>
```

```
<p> <b> <u> Password <input type="text" name="t4"  value=<%=s_t4%>>
</u> </b>  </p>
<p> <b> <u> Phone <input type="text" name="t5" value=<%=s_t5%>></u>
</b> </p>
<p> <b> <u> Address <input type="text" name="t6" value=<%=s_t6%>></u>
</b> </p>
<p> <b> <u> E_mail <input type="text" name="t7" value=<%=s_t7%>></u>
</b> </p>
<P> <b> <u> Start Working Date <input type="text" name="t8"
value=<%=s_t8%>> </u> </b> </p>
<p> <b> <u> Status <input type="text" name="t9" value=<%=s_t9%>></u>
</b> </p>
<p> <b> <u> Department <input type="text" name="t10"
value=<%=s_t10%>></u> </b> </p>
<p> <b> <u> Salary <input type="text" name="t11" value=<%=s_t11%>></u>
</b> </p>
<p> <input type="submit" value="update"> </p>
</form>
</body>
```

A.15. updateuser2.asp

```
<%@Language="VBScript"%>
<%dim s_t0
s_t0=CLng(Request.form("t0"))
s_t1=Request.form("t1")
s_t2=Request.form("t2")
s_t3=Request.form("t3")
s_t4=Request.form("t4")
s_t5=Request.form("t5")
s_t6=Request.form("t6")
s_t7=Request.form("t7")
s_t8=Request.form("t8")
s_t9=Request.form("t9")
s_t10=Request.form("t10")
s_t11=Request.form("t11")
set db=Server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("person.mdb")
sql= "UPDATE personnn set NAME='"&s_t1&"', SURNAME='"&s_t2&"',
USER_ID='"&s_t3&"', PASSWORD='"&s_t4&"', TEL='"&s_t5&"',
ADDRESS='"&s_t6&"', E_MAIL='"&s_t7&"', STARTWORKINGDATE='"&s_t8&"',
STATUS='"&s_t9&"', DEPARTMENT='"&s_t10&"', SALARY='"&s_t11&"' where
ID1="&s_t0&""
 db.execute(sql)
db.close
set db=nothing%>
<p align="center"> <font face="arial" color="#228B22" size="20"> Your
details are updated successfully </p>
<br>
<p align="center"> <a href="usercontrol.html"> Back </a> </p>
```

A.16. construction.html

```
<html>
<body bgcolor="32CD32">
<H1 align="center"><font face="algerian" size="30" > IRFAN GÖÇEN
CONSTRUCTION LTD. </H1>
<p align="center"> <font face="arial" size="15"> <a href="conts.html">
Construction </a> </p>
```

```
<p align="center"> <font face="arial" size="15"> <a
href="balance.html"> Balance </a> </p>
<p align="center"> <a href= "mainpagee.html"> Back </a> </p>
</body>
</html>
```

A.17. conts.html

```
<html>
<body bgcolor="32CD32">
<H1 align="center"><font face="algerian" size="30" > IRFAN GÖÇEN
CONSTRUCTION LTD. </H1>
<p align="center"> <font face="arial" size="15"> <a
href="searchpage.html"> Searching page </a> </p>
<p align="center"> <font face="arial" size="15"> <a href="enter.asp">
Entering page </a> </p>
<p align="center"> <font face="arial" size="15"> <a
href="deleterecord.asp"> Deleting page </a> </p>
<p align="center"> <font face="arial" size="15"> <a
href="updaterecord.asp"> Updating page </a> </p>
<p align="center"> <a href= "constractions.html"> Back </a> </p>
</body>
</html>
```

A.18. searchpage.html

```
<html>
<body bgcolor="32CD32">
<H1 align="center"><font face="algerian" size="30" > IRFAN GÖÇEN
CONSTRUCTION LTD. </H1>
<p align="center"> <font face="arial" size="15"> <a href="cont.html">
Searching page </a> </p>
<p align="center"> <font face="arial" size="15"> <a
href="listall.asp"> List All </a> </p>
<p align="center"> <a href= "conts.html"> Back </a> </p>
</body>
</html>
```

A.19. cont.html

```
<html>
<body bgcolor="32CD32">
 <H1> <pre align="left"> <font face="arial" size="5" >  <a
href="searchpage.html"> Back </a>              <font face="algerian"
size="30" > IRFAN GÖÇEN CONSTRACTION LTD.</pre> </H1>
<p align="right"> <u> <font face="arial" size="5"> SEARCING PAGE </u>
</P>
<form method="post" action="build.asp">
<pre align="left"> <b> <u> <font face="arial" size="3">CITY<input
type="text" name="city" size="10"> </u> </b>
                   <b> <u> BALCONY NUMBER </u> </b>

      <select name="mylist" size="2">

      <option value="c2"> 3

      <option value="c3"> 4

      </select>
```

```
  <b> <u> AREA <input type="text" name="area" size="10">  </u> </b>
                                          <b> <u> GARDEN LENGTH </u>
</b>

<select name="mylists" size="4">

<option value="c4"> 45m2

<option value="c5"> 55m2

<option value="c6"> 60m2

<option value="c7"> 70m2

</select>



  <b> <u> KIND OF BUILD </u> </b>

<b> <I> Duplex Villa  <input type="radio" name="R" value="dv"> </I>
</b>
<b> <I> Flat  <input type="radio" name="R" value="dv2"> </I> </b>
                                                     <b> <u> M2 </u>
</b>

<select name="mylistss" size="4">


<option value="c8"> 180m2

<option value="c9"> 200m2

<option value="c10"> 210m2

<option value="c11"> 220m2

<option value="c12"> 250m2

</select>
<b> <u> Central Heatting System </u> </b>
            <input type="submit" value="submit"> <input type="reset"
value="clear">

<b> <I> Yes <input type="radio" name="R1"  value="yes"> </I> </b>

<b> <I> No <input type="radio" name="R1"  value="no"> </I> </b>

 </pre>
</form>
</body>
</html>
```

A.20. build.asp

```
<%@Language="VBScript"%>
<hr> <pre align="center"> <a href="cont.html"> Back </a>
      <font face="algerian" color="#228B22" size="20"> BUILDINGS OF
CONSTRUCTION </pre> </hr>
<hr>
<%TC=Request.form("city")
```

```
if TC="nicosia" then
qc="nicosia"
end if
  if TC="kayrenia" then
qc="kayrenia"
end if
 if TC="famagusto" then
qc="famagusto"
end if
if TC="güzelyurt" then
qc="güzelyurt"
end if
TR=Request.form("area")
if TR="gönyeli" then
qsub="gönyeli "
end if
if TR="yenikent" then
qsub="yenikent"
end if
if TR="alsancak" then
qsub="alsancak"
end if

TD=Request.form("R")
if TD="dv" then
qd="duplex villa"
end if
if TD="dv2" then
qd="flat"
end if
TH=Request.form("R1")
if TH="yes" then
q="yes"
end if
if TH="no" then
q="no"
end if
TS=Request.form("mylist")
if TS="c2" then
qs="3"
end if
if TS="c3" then
qs="4"
end if
TT=Request.form("mylists")
if TT="c4" then
qrs="45m2"
end if
  if TT="c5" then
qrs="55m2"
end if
 if TT="c6" then
qrs="60m2"
end if
if TT="c7" then
qrs="70m2"
end if
TZ=Request.form("mylistss")
if TZ="c8" then
qz="180m2"
end if
```

```
  if TZ="c9" then
qz="200m2"
end if
 if TZ="c10" then
qz="210m2"
end if
if TZ="c11" then
qz="220m2"
end if
if TZ="c12" then
qz="250m2"
end if
set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from const where city='"&qc&"' or area='"&qsub&"' or
kind_of_build='"&qd&"' or central_heatting_system='"&q&"' or
balcony_number='"&qs&"' or  garden_length='"&qrs&"' or   m2='"&qz&"'"
set rec=db.execute(SQL)%>
<%while not rec.eof%>
<p align="center"> <font face="arial">
<table border=2 cellspacing=5 cellpadding=7>
 <tr> <th> city </th> <th> area </th> <th> part </th> <th> kind of
build </th> <th> m2 </th> <th> ground floor m2 </th> <th> first floor
m2 </th> <th> sitting room number </th> <th> kitchen room number </th>
<th> bathroom number </th> <th> badroom </th> <th> balcony number
</th> <th> central heatting system </th> <th> fireplace </th> <th>
badroom cupboard </th> <th> kitchen cupboard </th> <th> ground floor
base </th> <th> first floor base </th> <th> kind of windows </th> <th>
kind of door </th> <th> garden length </th> <th> swimming pool </th>
<th> cost </th>
<tr> <td> <%=rec("city")%> </td>
<td> <%=rec("area")%> </td>
<td> <%=rec("part")%> </td>
<td> <%=rec("kind_of_build")%> </td>
<td> <%=rec("m2")%> </td>
<td> <%=rec("ground_floor_m2")%> </td>
<td> <%=rec("first_floor_m2")%> </td>
<td> <%=rec("sitting_room_number")%> </td>
<td> <%=rec("kitchen_room_number")%> </td>
<td> <%=rec("bathroom_number")%> </td>
<td> <%=rec("badroom_number")%> </td>
<td> <%=rec("balcony_number")%> </td>
<td> <%=rec("central_heatting_system")%> </td>
<td> <%=rec("fireplace")%> </td>
<td> <%=rec("badroom_cupboard")%> </td>
<td> <%=rec("kitchen_cupboard")%> </td>
<td> <%=rec("ground_floor_base")%> </td>
<td> <%=rec("first_floor_base")%> </td>
<td> <%=rec("kind_of_windows")%> </td>
<td> <%=rec("kind_of_doors")%> </td>
<td> <%=rec("garden_length")%> </td>
<td> <%=rec("swimming_pool")%> </td>
<td> <%=rec("cost")%> </td></p>
<img src= "<%=rec ("picture")%>" height="200" width="300">
<img src="<%=rec ("pictur")%>" height="200" width="300">
<img src= "<%=rec ("pictu")%>" height="200" width="300">
<img src="<%=rec ("pict")%>" height="200" width="300">
<img src= "<%=rec ("pic")%>" height="200" width="300">
<%rec.movenext
wend
```

```
rec.close
set rec=nothing
db.close
set db=nothing%>
</table>
```

A.21. listall.asp

```
<%@Language="vbscript"%>
<body>
<pre align="left"> <a href="searchpage.html"> Back </a>
      <font face="arial" size="15" font color="#228B22"> BUILDINGS OF
CONSTRUCTION </Pre>
<hr>
<table align="center" border>
<tr> <th> ID </th> <th> city </th> <th> area </th> <th> part </th>
<th> kind of build </th> <th> m2 </th> <th> ground floor m2 </th> <th>
first floor m2 </th> <th> sitting room number </th> <th> kitchen room
number </th> <th> bathroom number </th> <th> badroom </th> <th>
balcony number </th> <th> central heatting system </th> <th> fireplace
</th> <th> badroom cupboard </th> <th> kitchen cupboard </th> <th>
ground floor base </th> <th> first floor base </th> <th> kind of
windows </th> <th> kind of door </th> <th> garden length </th> <th>
swimming pool </th> <th> cost </th>
<%set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from const"
set rec=db.execute(SQL)
while not rec.eof%>
<tr> <td> <%=rec("ID")%> </td>
<td> <%=rec("city")%> </td>
<td> <%=rec("area")%> </td>
<td> <%=rec("part")%> </td>
<td> <%=rec("kind_of_build")%> </td>
<td> <%=rec("m2")%> </td>
<td> <%=rec("ground_floor_m2")%> </td>
<td> <%=rec("first_floor_m2")%> </td>
<td> <%=rec("sitting_room_number")%> </td>
<td> <%=rec("kitchen_room_number")%> </td>
<td> <%=rec("bathroom_number")%> </td>
<td> <%=rec("badroom_number")%> </td>
<td> <%=rec("balcony_number")%> </td>
<td> <%=rec("central_heatting_system")%> </td>
<td> <%=rec("fireplace")%> </td>
<td> <%=rec("badroom_cupboard")%> </td>
<td> <%=rec("kitchen_cupboard")%> </td>
<td> <%=rec("ground_floor_base")%> </td>
<td> <%=rec("first_floor_base")%> </td>
<td> <%=rec("kind_of_windows")%> </td>
<td> <%=rec("kind_of_doors")%> </td>
<td> <%=rec("garden_length")%> </td>
<td> <%=rec("swimming_pool")%> </td>
<td> <%=rec("cost")%> </td> </p>
<td> <img src= "<%=rec ("picture")%>" height="200" width="300"> </td>
<td> <img src="<%=rec ("pictur")%>" height="200" width="300"> </td>
<td> <img src= "<%=rec ("pictu")%>" height="200" width="300"> </td>
```

```
<td> <img src="<%=rec ("pict")%>" height="200" width="300"> </td>
<td> <img src= "<%=rec ("pic")%>" height="200" width="300"> </td>
<%rec.movenext
wend
rec.close
set rec=nothing
db.close
set db=nothing%>
</table>
</body>
```

A.22. enter.asp

```
<html>
<body bgcolor="32CD32">
<H1> <pre align="center"> <a href="conts.html"> Back </a> <font
face="algerian" size="30" > IRFAN GÖÇEN CONSTRUCTION LTD. </pre> </H1>
<p align="right"> <u> <font face="arial" size="5"> ENTERING PAGE </u>
</P>
<form method="post" action="enter2.asp">
<p> <b> <u> <font face="arial" size="3"> City   <input type="text"
name="t1" > </u> </b> </p>
<p> <b> <u> Area <input type="text" name="t2" >  </u> </b> </p>
<p> <b> <u> Part <input type="text" name="t3" ></u> </b> </p>
<p> <b> <u> Kind of build <input type="text" name="t4" > </u> </b>
</p>
<p> <b> <u> M2 <input type="text" name="t5" ></u> </b> </p>
<p> <b> <u> Ground Floor m2 <input type="text" name="t6" ></u> </b>
</p>
<p> <b> <u> First Floor m2 <input type="text" name="t7" ></u> </b>
</p>
<p> <b> <u> Sitting Room Number <input type="text" name="t8" ></u>
</b> </p>
<P> <b> <u> Kitchen Room Number<input type="text" name="t9" > </u>
</b> </p>
<p> <b> <u> Bathroom Number <input type="text" name="t10" ></u> </b>
</p>
<p> <b> <u> Badroom Number <input type="text" name="t11" ></u> </b>
</p>
<p> <b> <u> Balcony Number <input type="text" name="t12" ></u> </b>
</p>
<p> <b> <u> Central Heatting System <input type="text" name="t13"
></u> </b> </p>
<p> <b> <u> Fireplace <input type="text" name="t14" ></u> </b> </p>
<p> <b> <u> Kind of badroom cupboard <input type="text" name="t15"
></u> </b> </p>
<p> <b> <u> Kind of kitchen cupboard <input type="text" name="t16"
></u> </b> </p>
<p> <b> <u> Ground Floor Base <input type="text" name="t17" ></u> </b>
</p>
<p> <b> <u> First Floor Base <input type="text" name="t18" ></u> </b>
</p>
<p> <b> <u> Kind of windows <input type="text" name="t19" ></u> </b>
</p>
<p> <b> <u> Kind of doors <input type="text" name="t20" ></u> </b>
</p>
<p> <b> <u> Garden Length <input type="text" name="t21" ></u> </b>
</p>
```

```
<p> <b> <u> Swimming Pool <input type="text" name="t22" ></u> </b>
</p>
<p> <b> <u> Picture </u> </b> </p>
<p> Attach Picture <input type="file" name="t23"
accept="appliction/pdf" /> </p>
<p> <b> <u> Picture </u> </b> </p>
<p>Attach Picture <input type="file" name="t24"
accept="appliction/pdf" /> </p>
<p> <b> <u> Picture </u> </b> </p>
<p> Attach Picture <input type="file" name="t25"
accept="appliction/pdf" /> </p>
<p> <b> <u> Picture </u> </b> </p>
<p>Attach Picture <input type="file" name="t26"
accept="appliction/pdf" /> </p>
<p> <b> <u> Picture </u> </b> </p>
<p>Attach Picture <input type="file" name="t27"
accept="appliction/pdf" /> </p>
<p> <input type="submit" value="enter"> </p>
</form>
</body>
</html>
```

A.23. enter2.asp

```
<%@Language="VBScript"%>
<%s_t1=Request.form("t1")
s_t2=Request.form("t2")
s_t3=Request.form("t3")
s_t4=Request.form("t4")
s_t5=Request.form("t5")
s_t6=Request.form("t6")
s_t7=Request.form("t7")
s_t8=Request.form("t8")
s_t9=Request.form("t9")
s_t10=Request.form("t10")
s_t11=Request.form("t11")
s_t12=Request.form("t12")
s_t13=Request.form("t13")
s_t14=Request.form("t14")
s_t15=Request.form("t15")
s_t16=Request.form("t16")
s_t17=Request.form("t17")
s_t18=Request.form("t18")
s_t19=Request.form("t19")
s_t20=Request.form("t20")
s_t21=Request.form("t21")
s_t22=Request.form("t22")
s_t23=Request.form("t23")
s_t24=Request.form("t24")
s_t25=Request.form("t25")
s_t26=Request.form("t26")
s_t27=Request.form("t27")
set db=Server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
sql="Insert into const(city, area, part, kind_of_build, m2,
ground_floor_m2, first_floor_m2, sitting_room_number,
kitchen_room_number, bathroom_number, badroom_number, balcony_number,
central_heatting_system, fireplace, badroom_cupboard,
kitchen_cupboard, ground_floor_base, first_floor_base,
kind_of_windows, kind_of_doors, garden_length, swimming_pool, picture,
pictur, pictu, pict, pic) values('"&s_t1&"','"&s_t2&"', '"&s_t3&"',
'"&s_t4&"', '"&s_t5&"', '"&s_t6&"', '"&s_t7&"', '"&s_t8&"',
'"&s_t9&"', '"&s_t10&"',
'"&s_t11&"','"&s_t12&"','"&s_t13&"','"&s_t14&"','"&s_t15&"','"&s_t16&"
','"&s_t17&"','"&s_t18&"','"&s_t19&"','"&s_t20&"','"&s_t21&"','"&s_t22
&"','"&s_t23&"','"&s_t24&"','"&s_t25&"','"&s_t26&"','"&s_t27&"')"
db.execute(sql)
db.close
set db=nothing%>
<p align="center"> <font face="arial" color="#228B22" size="20"> Your
details are entered successfully </p>
<br>
<p align="center"> <a href="conts.html"> BACK </a> </p>
```

A.24. deleterecord.asp

```
<body bgcolor="32CD32">
<form method="post" action="delete.asp">
<p align="center">          <b> <u> <font face="arial" size="3" > ID
<input type="text" name="t0"> </u> </b> </p>
<p align="center"> you can find ID number of what you want to delete
on searching page/List All </p>
<p align="center">                       <input type="submit"
value="delete"> </p>
<br>
<br>
<p align="center"> <font face="arial" size="5"> <a href="conts.html">
Back </a> </p>
</form>
</body>
```

A.25. delete.asp

```
<body bgcolor="32CD32">
<H1> <pre align="center"> <a href="deleterecord.asp"> Back </a> <font
face="algerian" size="30" > IRFAN GÖÇEN CONSTRUCTION LTD. </pre> </H1>
<p align="right"> <u> <font face="arial" size="5"> DELETING PAGE </u>
</P>
<%@Language="VBScript"%>
<%del=cint(request.form("t0"))%>
<%set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from const where ID="&del&""
set rec=db.execute(SQL)%>
<%s_t0=rec("ID")
s_t1=rec("city")
s_t2=rec("area")
s_t3=rec("part")
s_t4=rec("kind_of_build")
s_t5=rec("m2")
s_t6=rec("ground_floor_m2")
```

```
s_t7=rec("first_floor_m2")
s_t8=rec("sitting_room_number")
s_t9=rec("kitchen_room_number")
s_t10=rec("bathroom_number")
s_t11=rec("badroom_number")
s_t12=rec("balcony_number")
s_t13=rec("central_heatting_system")
s_t14=rec("fireplace")
s_t15=rec("badroom_cupboard")
s_t16=rec("kitchen_cupboard")
s_t17=rec("ground_floor_base")
s_t18=rec("first_floor_base")
s_t19=rec("kind_of_windows")
s_t20=rec("kind_of_doors")
s_t21=rec("garden_length")
s_t22=rec("swimming_pool")
s_t23=rec("picture")
s_t24=rec("pictur")
s_t25=rec("pictu")
s_t26=rec("pict")
s_t27=rec("pic")
rec.close
set rec=nothing
db.close
set db=nothing%>
<form method="post" action="delete2.asp">
<p> <b> <u> <font face="arial" size="3"> ID   <input type="text"
name="t0" value=<%=s_t0%>> </u> </b> </p>
<p> <b> <u> <font face="arial" size="3"> City   <input type="text"
name="t1" value=<%=s_t1%>> </u> </b> </p>
<p> <b> <u> Area <input type="text" name="t2" value=<%=s_t2%>>  </u>
</b> </p>
<p> <b> <u> Part <input type="text" name="t3" value=<%=s_t3%>></u>
</b> </p>
<p> <b> <u> Kind of build <input type="text" name="t4"
value=<%=s_t4%>> </u> </b>  </p>
<p> <b> <u> m2 <input type="text" name="t5" value=<%=s_t5%>></u> </b>
</p>
<p> <b> <u> Ground Floor m2 <input type="text" name="t6"
value=<%=s_t6%>></u> </b> </p>
<p> <b> <u> First Floor m2 <input type="text" name="t7"
value=<%=s_t7%>></u> </b> </p>
<P> <b> <u> Sitting Room Number<input type="text" name="t8"
value=<%=s_t8%>> </u> </b> </p>
<p> <b> <u> Kitchen Room Number <input type="text" name="t9"
value=<%=s_t9%>></u> </b> </p>
<p> <b> <u> Bathroom Number <input type="text" name="t10"
value=<%=s_t10%>></u> </b> </p>
<p> <b> <u> Badroom Number <input type="text" name="t11"
value=<%=s_t11%>></u> </b> </p>
<p> <b> <u> Balcony Number <input type="text" name="t12"
value=<%=s_t12%>></u> </b> </p>
<p> <b> <u> Central Heatting System <input type="text" name="t13"
value=<%=s_t13%>></u> </b> </p>
<p> <b> <u> Fireplace <input type="text" name="t14"
value=<%=s_t14%>></u> </b> </p>
<p> <b> <u> Kind of Badroom Cupboard  <input type="text" name="t15"
value=<%=s_t15%>></u> </b> </p>
<p> <b> <u> Kind od Kitchen cupboard <input type="text" name="t16"
value=<%=s_t16%>></u> </b> </p>
```

```
<p> <b> <u> Ground Floor Base <input type="text" name="t17"
value=<%=s_t17%>></u> </b> </p>
<p> <b> <u> First Flar Flar <input type="text" name="t18"
value=<%=s_t18%>></u> </b> </p>
<p> <b> <u> Kind of Windows <input type="text" name="t19"
value=<%=s_t19%>></u> </b> </p>
<p> <b> <u> Kind of Doors <input type="text" name="t20"
value=<%=s_t20%>></u> </b> </p>
<p> <b> <u> Garden Length <input type="text" name="t21"
value=<%=s_t21%>></u> </b> </p>
<p> <b> <u> Swimming Pool <input type="text" name="t22"
value=<%=s_t22%>></u> </b> </p>
<p> <b> <u> Picture <input type="text" name="t23" value=<%=s_t23%>>
</u> </b> </p>
<p> <b> <u> Picture <input type="text" name="t24" value=<%=s_t24%>>
</u> </b> </p>
<p> <b> <u> Picture <input type="text" name="t25" value=<%=s_t25%>>
</u> </b> </p>
<p> <b> <u> Picture <input type="text" name="t26" value=<%=s_t26%>>
</u> </b> </p>
<p> <b> <u> Picture <input type="text" name="t27" value=<%=s_t27%>>
</u> </b> </p>
<p> <input type="submit" value="delete"> </p>
</form>
</body>
```

A.26. delete2.asp

```
<%@Language="VBScript"%>
<%dim s_t0
s_t0=CLng(Request.form("t0"))
s_t1=Request.form("t1")
s_t2=Request.form("t2")
s_t3=Request.form("t3")
s_t4=Request.form("t4")
s_t5=Request.form("t5")
s_t6=Request.form("t6")
s_t7=Request.form("t7")
s_t8=Request.form("t8")
s_t9=Request.form("t9")
s_t10=Request.form("t10")
s_t11=Request.form("t11")
s_t12=Request.form("t12")
s_t13=Request.form("t13")
s_t14=Request.form("t14")
s_t15=Request.form("t15")
s_t16=Request.form("t16")
s_t17=Request.form("t17")
s_t18=Request.form("t18")
s_t19=Request.form("t19")
s_t20=Request.form("t20")
s_t21=Request.form("t21")
s_t22=Request.form("t22")
s_t23=Request.form("t23")
s_t24=Request.form("t24")
s_t25=Request.form("t25")
```

```
s_t26=Request.form("t26")
s_t27=Request.form("t27")
set db=Server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
sql= "delete from const where ID="&s_t0&""
db.execute(sql)
db.close
set db=nothing%>
<p align="center"> <font face="arial" color="#228B22" size="20"> Your
details are deleted successfully </p>
<br>
<p align="center"> <a href="conts.html"> Back </a> </p>
```

A.27. updaterecord.asp

```
<body bgcolor="32CD32">
<form method="post" action="update.asp">
<p align="center"> <b> <u> <font face="arial" size="3" > ID   <input
type="text" name="t0"> </u> </b> </p>
<p align="center"> You can find ID number what you want to update on
searching page/List All </p>
<p align="center"> <input type="submit" value="update"> </p>
<p align="center"> <a href="conts.html"> Back </a> </p>
</form>
</body>
```

A.28. update.asp

```
<body bgcolor="32CD32">
<H1> <pre align="center"> <a href="updaterecord.asp"> Back </a> <font
face="algerian" size="30" > IRFAN GÖÇEN CONSTRUCTION LTD.</pre> </H1>
<p align="right"> <u> <font face="arial" size="5"> UPDATING PAGE </u>
</P>
<%@Language="VBScript"%>
<%del=cint(request.form("t0"))%>
<%set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from const where ID="&del&""
set rec=db.execute(SQL)%>
<%s_t0=rec("ID")
s_t1=rec("city")
s_t2=rec("area")
s_t3=rec("part")
s_t4=rec("kind_of_build")
s_t5=rec("m2")
s_t6=rec("ground_floor_m2")
s_t7=rec("first_floor_m2")
s_t8=rec("sitting_room_number")
s_t9=rec("kitchen_room_number")
```

```
s_t10=rec("bathroom_number")
s_t11=rec("badroom_number")
s_t12=rec("balcony_number")
s_t13=rec("central_heatting_system")
s_t14=rec("fireplace")
s_t15=rec("badroom_cupboard")
s_t16=rec("kitchen_cupboard")
s_t17=rec("ground_floor_base")
s_t18=rec("first_floor_base")
s_t19=rec("kind_of_windows")
s_t20=rec("kind_of_doors")
s_t21=rec("garden_length")
s_t22=rec("swimming_pool")
s_t23=rec("picture")
s_t24=rec("pictur")
s_t25=rec("pictu")
s_t26=rec("pict")
s_t27=rec("pic")
rec.close
set rec=nothing
db.close
set db=nothing%>
<form method="post" action="update2.asp">
<p> <b> <u> <font face="arial" size="3"> ID   <input type="text"
name="t0" value=<%=s_t0%>> </u> </b> </p>
<p> <b> <u> <font face="arial" size="3"> City   <input type="text"
name="t1" value=<%=s_t1%>> </u> </b> </p>
<p> <b> <u> Area <input type="text" name="t2" value=<%=s_t2%>>  </u>
</b> </p>
<p> <b> <u> Part <input type="text" name="t3" value=<%=s_t3%>></u>
</b> </p>
<p> <b> <u> Kind of build <input type="text" name="t4"
value=<%=s_t4%>> </u> </b>  </p>
<p> <b> <u> m2 <input type="text" name="t5" value=<%=s_t5%>></u> </b>
</p>
<p> <b> <u> Ground Floor m2 <input type="text" name="t6"
value=<%=s_t6%>></u> </b> </p>
<p> <b> <u> First Floor m2 <input type="text" name="t7"
value=<%=s_t7%>></u> </b> </p>
<P> <b> <u> Sitting Room Number<input type="text" name="t8"
value=<%=s_t8%>> </u> </b> </p>
<p> <b> <u> Kitchen Room Number <input type="text" name="t9"
value=<%=s_t9%>></u> </b> </p>
<p> <b> <u> Bathroom Number <input type="text" name="t10"
value=<%=s_t10%>></u> </b> </p>
<p> <b> <u> Badroom Number <input type="text" name="t11"
value=<%=s_t11%>></u> </b> </p>
<p> <b> <u> Balcony Number <input type="text" name="t12"
value=<%=s_t12%>></u> </b> </p>
<p> <b> <u> Central Heatting System <input type="text" name="t13"
value=<%=s_t13%>></u> </b> </p>
<p> <b> <u> Fireplace <input type="text" name="t14"
value=<%=s_t14%>></u> </b> </p>
<p> <b> <u> Kind of Badroom Cupboard  <input type="text" name="t15"
value=<%=s_t15%>></u> </b> </p>
<p> <b> <u> Kind od Kitchen cupboard <input type="text" name="t16"
value=<%=s_t16%>></u> </b> </p>
<p> <b> <u> Ground Floor Base <input type="text" name="t17"
value=<%=s_t17%>></u> </b> </p>
<p> <b> <u> First Floor Base <input type="text" name="t18"
value=<%=s_t18%>></u> </b> </p>
```

```
<p> <b> <u> Kind of Windows <input type="text" name="t19"
value=<%=s_t19%>></u> </b> </p>
<p> <b> <u> Kind of Doors <input type="text" name="t20"
value=<%=s_t20%>></u> </b> </p>
<p> <b> <u> Garden Length <input type="text" name="t21"
value=<%=s_t21%>></u> </b> </p>
<p> <b> <u> Swimming Pool <input type="text" name="t22"
value=<%=s_t22%>></u> </b> </p>
<p> <b> <u> Picture <input type="text" name="t23" value=<%=s_t23%>>
</u> </b> </p>

<p> <b> <u> Picture <input type="text" name="t24" value=<%=s_t24%>>
</u> </b> </p>

<p> <b> <u> Picture <input type="text" name="t25" value=<%=s_t25%>>
</u> </b> </p>

<p> <b> <u> Picture <input type="text" name="t26" value=<%=s_t26%>>
</u> </b> </p>

<p> <b> <u> Picture <input type="text" name="t27" value=<%=s_t27%>>
</u> </b> </p>

<p> <input type="submit" value="update"> </p>
</form>
</body>
```

A.29. update2.asp

```
<%@Language="VBScript"%>
<%dim s_t0
s_t0=CLng(Request.form("t0"))
s_t1=Request.form("t1")
s_t2=Request.form("t2")
s_t3=Request.form("t3")
s_t4=Request.form("t4")
s_t5=Request.form("t5")
s_t6=Request.form("t6")
s_t7=Request.form("t7")
s_t8=Request.form("t8")
s_t9=Request.form("t9")
s_t10=Request.form("t10")
s_t11=Request.form("t11")
s_t12=Request.form("t12")
s_t13=Request.form("t13")
s_t14=Request.form("t14")
s_t15=Request.form("t15")
s_t16=Request.form("t16")
s_t17=Request.form("t17")
s_t18=Request.form("t18")
s_t19=Request.form("t19")
s_t20=Request.form("t20")
s_t21=Request.form("t21")
s_t22=Request.form("t22")
s_t23=Request.form("t23")
s_t24=Request.form("t24")
s_t25=Request.form("t25")
s_t26=Request.form("t26")
s_t27=Request.form("t27")
```

```
set db=Server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
sql= "UPDATE const set city='"&s_t1&"', area='"&s_t2&"',
part='"&s_t3&"', kind_of_build='"&s_t4&"', m2='"&s_t5&"',
ground_floor_m2='"&s_t6&"', first_floor_m2='"&s_t7&"',
sitting_room_number='"&s_t8&"', kitchen_room_number='"&s_t9&"',
bathroom_number='"&s_t10&"', badroom_number='"&s_t11&"',
balcony_number='"&s_t12&"', central_heatting_system='"&s_t13&"',
fireplace='"&s_t14&"', badroom_cupboard='"&s_t15&"',
kitchen_cupboard='"&s_t16&"', ground_floor_base='"&s_t17&"',
first_floor_base='"&s_t18&"', kind_of_windows='"&s_t19&"',
kind_of_doors='"&s_t20&"',  garden_length='"&s_t21&"',
swimming_pool='"&s_t22&"', picture='"&s_t23&"', pictur='"&s_t24&"',
pictu='"&s_t25&"', pict='"&s_t26&"', pic='"&s_t27&"' where
ID="&s_t0&""
 db.execute(sql)
db.close
set db=nothing%>
<p align="center"> <font face="arial" color="#228B22" size="20"> Your
details are updated successfully </p>
<br>
<p align="center"> <a href="conts.html"> Back </a> </p>
```

A.30. balance.html

```
<html>
<body bgcolor="32CD32">
<H1 align="center"><font face="algerian" size="30" > IRFAN GÖÇEN
CONSTRUCTION LTD. </H1>
<p align="center"> <font face="arial" size="15"> <a href="sb.html">
Searching page </a> </p>
<p align="center"> <font face="arial" size="15"> <a
href="balancee.asp"> Entering page </a> </p>
<p align="center"> <font face="arial" size="15"> <a
href="baldel.html"> Deleting page </a> </p>
<p align="center"> <font face="arial" size="15"> <a href="bal.html">
Updating page </a> </p>
<p align="center"> <a href= "constractions.html"> Back </a> </p>
</body>
</html>
```

A.31. sb.html

```
<html>
<body bgcolor="32CD32">
<H1 align="center"><font face="algerian" size="30" > IRFAN GÖÇEN
CONSTRACTION LTD. </H1>
<p align="right"> <u> <font face="arial" size="5"> SEARCING PAGE </u>
</P>
<form method="post" action="sb.asp">
<p align="center"> <b> <u> <font face="arial" size="3"> name   <input
type="text" name="name" size="10" > </u> </b> </p>
<p align="center"> <b> <u> <font face="arial" size="3"> surname
<input type="text" name="surname" size="10" > </u> </b> </p>
<p align="center"> <input type="submit" value="submit"> </p>
<p align="center"> <a href="balance.html"> BACK </a>
</form>
</body>
</html>
```

A.32. sb.asp

```
<%@Language="VBScript"%>
<pre> <a href="sb.html"> BACK </a> <font face="algerian"
color="#228B22" size="20">                         BALANCE </pre>
<hr>
<%n=request.form("name")
s=request.form("surname")
set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
sql="select * from balan where name='"&n&"' and surname='"&s&"'"
set rec=db.execute(sql)
tot=0
while not rec.eof%>
<table border>
<tr> <th> ID </th> <th> name </th> <th> surname </th> <th> tel no
</th> <th> mobile phone </th> <th> address </th> <th> e_mail </th>
<th> date </th> <th> house number </th> <th> amount </th> <th> debit
</th>   <th> balance </th>
<tr> <td> <%=rec("ID")%> </td>
<td> <%=rec("name")%> </td>
<td> <%=rec("surname")%> </td>
<td> <%=rec("tel_no")%> </td>
<td> <%=rec("mobile_phone")%> </td>
<td> <%=rec("address")%> </td>
<td> <%=rec("e_mail")%> </td>
<td> <%=rec("u_date")%> </td>
<td> <%=rec("house_number")%> </td>
<td> <%=rec("amount")%> </td>
<td> <%=rec("debit")%> </td>
<%tot=tot+rec("amount")-rec("debit")%>
<td> <%=tot%> </td>
</table>
<%rec.movenext
wend
rec.close
set rec=nothing
db.close
set db=nothing%>
balancee.asp
<html>
<body bgcolor="32CD32">
<pre> <a href="balance.html"> BACK </a> <font face="algerian"
size="30" >                  IRFAN GÖÇEN CONSTRUCTION LTD. </pre>
<p align="right"> <u> <font face="arial" size="5"> BALANCE </u> </P>
<form method="post" action="balanceinsert.asp">
<font face="arial" size="3">
<p> <b> <u> Name  <input type="text" name="t1" > </u> </b> </p>
<p> <b> <u> Surname <input type="text" name="t2" >  </u> </b> </p>
<p> <b> <u> Tel NO <input type="text" name="t3" ></u> </b> </p>
<p> <b> <u> Mobile Phone <input type="text" name="t4" > </u> </b>
</p>
<p> <b> <u> Address <input type="text" name="t5" ></u> </b> </p>
<P> <b> <u> E_mail <input type="text" name="t6" > </u> </b> </p>
<p> <b> <u> Date <input type="text" name="t7" value=<%=date%> ></u>
</b> </p>
```

```
<pre><b><u><font face="arial" size="3">Amount <input type="text"
name="t8" > </u> </b> <a href="detail.asp"> click to see houses
details </a>  </pre>
<p> <b> <u> Debit <input type="text" name="t9" > </u> </b> </p>
<p> <b> <u> House Number <input type="text" name="t10" > </u> </b>
</p>
<p> <input type="submit" value="enter"> </p>
</form>
</body>
</html>
```

A.33. detail.asp

```
<%@Language="vbscript"%>
<body>
<pre align="left"> <a href="balancee.asp"> Back </a>              <font
face="arial" size="15" font color="#228B22"> BUILDINGS OF CONSTRUCTION
</Pre>
<hr>
<table align="center" border>
<tr> <th> ID </th> <th> city </th> <th> area </th> <th> part </th>
<th> kind of build </th> <th> m2 </th>  <th> cost </th>
<%set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from const"
set rec=db.execute(SQL)
while not rec.eof%>
<tr> <td> <%=rec("ID")%> </td>
<td> <%=rec("city")%> </td>
<td> <%=rec("area")%> </td>
<td> <%=rec("part")%> </td>
<td> <%=rec("kind_of_build")%> </td>
<td> <%=rec("m2")%> </td>
<td> <%=rec("cost")%> </td> </p>
<%rec.movenext
wend
rec.close
set rec=nothing
db.close
set db=nothing%>
</table>
</body>
```

A.34. balanceinsert.asp

```
<%@Language="VBScript"%>
<%s_t1=Request.form("t1")
s_t2=Request.form("t2")
s_t3=Request.form("t3")
s_t4=Request.form("t4")
s_t5=Request.form("t5")
s_t6=Request.form("t6")
s_t7=Request.form("t7")
s_t8=csng(Request.form("t8"))
s_t9=csng(Request.form("t9"))
s_t10=Request.form("t10")
set db=Server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
```

```
sql="insert into balan(name, surname, tel_no, mobile_phone, address,
e_mail, u_date, amount, debit, house_number) values ('"&s_t1&"',
'"&s_t2&"', '"&s_t3&"', '"&s_t4&"', '"&s_t5&"', '"&s_t6&"',
'"&s_t7&"', "&s_t8&", "&s_t9&", '"&s_t10&"')"
db.execute(sql)%>
<%db.close
set db=nothing%>
<p align="center"> <font face="arial" color="32CD32" size="40"> YOUR
DETAILS ARE ENTERED SUCCESSFULLY </p>
<p align="center"> <a href="balance.html"> BACK </a>
```

A.35. baldel.html

```
<body bgcolor="32CD32">
<form method="post" action="baldel.asp">
<p align="center"> <b> <u> <font face="arial" size="3" > ID   <input
type="text" name="t0"> </u> </b> </p>
<p align="center"> <input type="submit" value="enter"> </p>
<p align="center"> you can find ID number what you want to delete on
searching page </p>
<p align="center"> <a href="balance.html"> BACK </a> </p>
</form>
</body>
```

A.36. baldel.asp

```
<body bgcolor="32CD32">
<pre> <a href="baldel.html"> BACK </a> <font face="algerian" size="30"
>                 IRFAN GÖÇEN CONSTRUCTION LTD. </pre>
<p align="right"> <u> <font face="arial" size="5"> BALANCE PAGE </u>
</P>
<%@Language="VBScript"%>
<%del=cint(request.form("t0"))%>
<%set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from balan where ID="&del&""
set rec=db.execute(SQL)%>
<%s_t0=rec("ID")
s_t1=rec("name")
s_t2=rec("surname")
s_t3=rec("tel_no")
s_t4=rec("mobile_phone")
s_t5=rec("address")
s_t6=rec("e_mail")
s_t7=rec("u_date")
s_t8=rec("amount")
s_t9=rec("debit")
s_t10=rec("house_number")
rec.close
set rec=nothing
```

```
db.close
set db=nothing%>
<form method="post" action="balandel.asp">
<p> <b> <u> <font face="arial" size="3"> ID   <input type="text"
name="t0" value=<%=s_t0%>> </u> </b> </p>
<p> <b> <u> <font face="arial" size="3"> name   <input type="text"
name="t1" value=<%=s_t1%>> </u> </b> </p>
<p> <b> <u> surname <input type="text" name="t2" value=<%=s_t2%>>
</u> </b> </p>
<p> <b> <u> tel no <input type="text" name="t3" value=<%=s_t3%>></u>
</b> </p>
<p> <b> <u> mobile phone <input type="text" name="t4"
value=<%=s_t4%>> </u> </b>  </p>
<p> <b> <u> address <input type="text" name="t5" value=<%=s_t5%>></u>
</b> </p>
<p> <b> <u> e_mail <input type="text" name="t6" value=<%=s_t6%>></u>
</b> </p>
<P> <b> <u> date <input type="text" name="t7" value=<%=s_t7%>> </u>
</b> </p>
<p> <b> <u> amount <input type="text" name="t8" value=<%=s_t8%>></u>
</b> </p>
<p> <b> <u> debit <input type="text" name="t9" value=<%=s_t9%>></u>
</b> </p>
<p> <b> <u> House Number <input type="text" name="t10"
value=<%=s_t10%>></u> </b> </p>
<p> <input type="submit" value="delete"> </p>
</form>
</body>
```

A.37. balandel.asp

```
<%@Language="VBScript"%>
<%dim s_t0
s_t0=CLng(Request.form("t0"))
s_t1=Request.form("t1")
s_t2=Request.form("t2")
s_t3=Request.form("t3")
s_t4=Request.form("t4")
s_t5=Request.form("t5")
s_t6=Request.form("t6")
s_t7=Request.form("t7")
s_t8=Request.form("t8")
s_t9=Request.form("t9")
s_t10=Request.form("t10")
set db=Server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
sql= "delete from balan where ID="&s_t0&""
db.execute(sql)
db.close
set db=nothing%>
<p align="center"> <font face="arial" color="#228B22" size="20"> Your
details are deleted successfully </p>
<p align="center"> <a href="balance.html"> BACK </a> </p>
```

A.38. bal.html

```
<body bgcolor="32CD32">
<form method="post" action="bal.asp">
<p align="center"> <b> <u> <font face="arial" size="3" > ID   <input
type="text" name="t0"> </u> </b> </p>
<p align="center"> <input type="submit" value="enter"> </p>
<p align="center"> you can find ID number what you want to update on
searching page </p>
<p align="center"> <a href="balance.html"> BACK </a> </p>
</form>
</body>
```

A.39. bal.asp

```
<body bgcolor="32CD32">
<pre> <a href="bal.html"> BACK </a> <font face="algerian" size="30" >
            IRFAN GÖÇEN CONSTRUCTION LTD. </pre>
<p align="right"> <u> <font face="arial" size="5"> BALANCE PAGE </u>
</P>
<%@Language="VBScript"%>
<%del=cint(request.form("t0"))%>
<%set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from balan where ID="&del&""
set rec=db.execute(SQL)%>
<%s_t0=rec("ID")
s_t1=rec("name")
s_t2=rec("surname")
s_t3=rec("tel_no")
s_t4=rec("mobile_phone")
s_t5=rec("address")
s_t6=rec("e_mail")
s_t7=rec("u_date")
s_t8=rec("amount")
s_t9=rec("debit")
s_t10=rec("house_number")
rec.close
set rec=nothing
db.close
set db=nothing%>
<form method="post" action="balan.asp">
<p> <b> <u> <font face="arial" size="3"> ID   <input type="text"
name="t0" value=<%=s_t0%>> </u> </b> </p>
<p> <b> <u> <font face="arial" size="3"> name   <input type="text"
name="t1" value=<%=s_t1%>> </u> </b> </p>
<p> <b> <u> surname <input type="text" name="t2" value=<%=s_t2%>>
</u> </b> </p>
<p> <b> <u> tel no <input type="text" name="t3" value=<%=s_t3%>></u>
</b> </p>
<p> <b> <u> mobile phone <input type="text" name="t4"
value=<%=s_t4%>> </u> </b>  </p>
<p> <b> <u> address <input type="text" name="t5" value=<%=s_t5%>></u>
</b> </p>
<p> <b> <u> e_mail <input type="text" name="t6" value=<%=s_t6%>></u>
</b> </p>
<P> <b> <u> date <input type="text" name="t7" value=<%=s_t7%>> </u>
</b> </p>
```

```
<p> <b> <u> amount <input type="text" name="t8" value=<%=s_t8%>></u>
</b> </p>
<p> <b> <u> debit <input type="text" name="t9" value=<%=s_t9%>></u>
</b> </p>
<p> <b> <u> House Number <input type="text" name="t10"
value=<%=s_t10%>></u> </b> </p>
<p> <input type="submit" value="update"> </p>
</form>
</body>
```

A.40. balan.asp

```
<%@Language="VBScript"%>
<%dim s_t0
s_t0=CLng(Request.form("t0"))
s_t1=Request.form("t1")
s_t2=Request.form("t2")
s_t3=Request.form("t3")
s_t4=Request.form("t4")
s_t5=Request.form("t5")
s_t6=Request.form("t6")
s_t7=Request.form("t7")
s_t8=Request.form("t8")
s_t9=Request.form("t9")
s_t10=Request.form("t10")
set db=Server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
sql= "UPDATE balan set name='"&s_t1&"', surname='"&s_t2&"',
tel_no='"&s_t3&"', mobile_phone='"&s_t4&"', address='"&s_t5&"',
e_mail='"&s_t6&"', u_date='"&s_t7&"', amount="&s_t8&", debit="&s_t9&",
house_number='"&s_t10&"' where ID="&s_t0&"" db.execute(sql)
db.close
set db=nothing%>
<p align="center"> <font face="arial" color="#228B22" size="20"> Your
details are updated successfully </p>
<p align="center"> <a href="balance.html"> BACK </a> </p>
```

A.41. isolations.html

```
<html>
<body background="C:\Documents and Settings\x\My
Documents\aceng\VİLLALAR\DSC_0055.JPG">
<H1 align="center"> <img src="C:\Documents and Settings\x\My
Documents\resimler\DSC021599.JPG" width="1250" height="200"> </H1>
<p align="center"> <font face="arial" size="15"> <a
href="isolation.html"> Insulation </a> </p>
<p align="center"> <font face="arial" size="15"> <a
href="balizo.html"> Balance </a> </p>
<p align="center"> <a href= "mainpagee.html"> Back </a> </p>
</body>
</html>
```

A.42. isolation.html
```
<html>
<title> insulation!!! </title>
<Body background="C:\Documents and Settings\x\My
Documents\resimler\DSC021588.JPG">
<pre>  <font face="Broadway" font size="5"> <a href="izo1.html">
[THERMAL INSULATION] </a>          <a href="izo2.html"> [CORRUBIT
PANELS] </a>           <a href="izo3.html"> [WATER INSULATION] </a>

<font face="arial" size=30>            <a href="isolations.html"> BACK
</a> </pre>
```

A.43. izo1.html
```
<html>
<title> insulation!!! </title>
<Body background="C:\Documents and Settings\x\My
Documents\resimler\DSC021588.JPG">
<pre>  <font face="Broadway" font size="5">
<a href="izo1.html"> [THERMAL INSULATION] </a> <a href="izo11.html">
SEE PRODUCT </a>  <a href="sizo.html"> SEARCHING PAGE </a>
                                                     <a
href="eizo.asp"> ENTERING PAGE </a>
                                                     <a
href="dizo.asp"> DELETING PAGE </a>
                                                     <a
href="uizo.asp"> UPDATING PAGE </a>

<font face="arial" size="5"><B><a href="izo2.html">[CORRUBIT PANELS]
</a>


<a href="izo3.html"> [WATER INSULATION] </a>
                                               <font size="15">  <a
href="isolation.html"> BACK </a>
</pre>
</html>
```

A.44. izo11.html
```
<html>
<title> insulation!!! </title>
<Body background="C:\Documents and Settings\x\My
Documents\resimler\DSC021588.JPG">
<pre>  <font face="Broadway" font size="5"> <a href="izo1.html">
[THERMAL INSULATION] </a>     <font face="arial"> Exstruded
Polystyrenee Thermal Insulation Panels   <a href="izo12.asp"> POLPAN
</a> </pre>
```

A.45. izo12.asp
```
<%@Language="vbscript"%>
<body bgcolor="#228B22">
<p align="center"> <font face="Broadway" size="15"> POLPAN </p>
```

```
<p> <font face="arial" size="6"> Polpan thermal insulation panels are
produced in skinned surfaced models for roofs and floors, and in
plaster type models for exterior and/or interior application of
thermal insulation on walls. </p>
<table align="center" border>
<tr> <th> Kind of Insulation </th> <th> Product </th> <th> Amount of
m2 </th>
<%set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from izo3 where ID=7"
set rec=db.execute(SQL)
while not rec.eof%>
<tr> <td> <%=rec("kind_of_insulation")%> </td>
<td> <%=rec("product")%> </td>
<td> <%=rec("amount_of_m2")%> </td>
<img src= "<%=rec ("picture")%>" height="250" width="350">
<%rec.movenext
wend
rec.close
set rec=nothing
db.close
set db=nothing%>
</table>
<p align="center"> <a href="isolation.html"> BACK </a> </p>
</body>
```

A.46. sizo.html

```
<html>
<body bgcolor="32CD32">
<H1 align="center"><font face="algerian" size="30" > IRFAN GÖÇEN
CONSTRACTION LTD. </H1>
<p align="right"> <u> <font face="arial" size="5"> SEARCING PAGE </u>
</P>
<form method="post" action="sizo.asp">
<p align="center"> <b> <u> <font face="arial" size="3"> Kind of
Insulation   <input type="text" name="ti" size="30" value="thermal
insulation"> </u> </b> </p>
<p align="center"> <input type="submit" value="submit"> </p>
<p align="center"> <a href="izo1.html"> BACK </a>
</form>
</body>
</html>
```

A.47. sizo.asp

```
<%@Language="vbscript"%>
<body>
<pre align="left"> <a href="izo1.html"> Back </a>
      <font face="arial" size="15" font color="#228B22"> IRFAN
INSULATION </Pre>
<hr>
<%t=request.form("ti")
set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from izo3 where kind_of_insulation='"&t&"'"
set rec=db.execute(SQL)
while not rec.eof%>
<table align="center" border>
<tr> <th> ID </th> <th> Kind of Insulatıon </th> <th> product </th>
<th> Amount of m2 </th> <th> Picture </th>
```

```
<tr> <td> <%=rec("ID")%> </td>
<td> <%=rec("kind_of_insulation")%> </td>
<td> <%=rec("product")%> </td>
<td> <%=rec("amount_of_m2")%> </td>
<td> <img src= "<%=rec ("picture")%>" height="300" width="300"> </td>
<%rec.movenext
wend
rec.close
set rec=nothing
db.close
set db=nothing%>
</table>
</body>
```

A.48. eizo.asp

```
<html>
<body bgcolor="32CD32">
<H1> <pre align="center"> <a href="izo1.html"> Back </a>
      <font face="algerian" size="30" > IRFAN INSULATION </pre> </H1>
<p align="right"> <u> <font face="arial" size="5"> ENTERING PAGE </u>
</P>
<form method="post" action="enterizo2.asp">
<p> <b> <u> <font face="arial" size="3"> Kind of Insulatıon   <input
type="text" name="t1" > </u> </b> </p>
<p> <b> <u> Product <input type="text" name="t2" >  </u> </b> </p>
<p> <b> <u> Amount of m2 <input type="text" name="t3" ></u> </b> </p>
<p> <b> <u> Picture </u> </b> </p>
<p> Attach Picture <input type="file" name="t4"
accept="appliction/pdf" /> </p>
<p> <input type="submit" value="enter"> </p>
</form>
</body>
</html>
```

A.49. enterizo2.asp

```
<%@Language="VBScript"%>
<%s_t1=Request.form("t1")
s_t2=Request.form("t2")
s_t3=Request.form("t3")
s_t4=Request.form("t4")
set db=Server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
sql="Insert into izo3(kind_of_insulation, product, amount_of_m2,
picture) values('"&s_t1&"','"&s_t2&"', '"&s_t3&"', '"&s_t4&"')"
db.execute(sql)
db.close
set db=nothing%>
<p align="center"> <font face="arial" color="#228B22" size="20"> Your
details are entered successfully </p>
<br>
<p align="center"> <a href="izo1.html"> BACK </a> </p>
```

A.50. dizo.asp

```
<body bgcolor="32CD32">
<form method="post" action="deleteizo.asp">
<p align="center"> <b> <u> <font face="arial" size="3" > ID   <input
type="text" name="t0"> </u> </b> </p>
<p align="center"> You can find ID number what you want to delete on
searching page</p>
<p align="center"> <input type="submit" value="delete"> </p>
<p align="center"> <a href="izo1.html"> Back </a> </p>
</form>
</body>
```

A.51. deleteizo.asp

```
<body bgcolor="32CD32">
<H1> <pre align="center"> <a href="dizo.asp"> Back </a>
       <font face="algerian" size="30" > IRFAN INSULATION </pre> </H1>
<p align="right"> <u> <font face="arial" size="5"> DELETING PAGE </u>
</P>
<%@Language="VBScript"%>
<%del=cint(request.form("t0"))%>
<%set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from izo3 where ID="&del&""
set rec=db.execute(SQL)%>
<%s_t0=rec("ID")
s_t1=rec("kind_of_insulation")
s_t2=rec("product")
s_t3=rec("amount_of_m2")
s_t4=rec("picture")
rec.close
set rec=nothing
db.close
set db=nothing%>
<form method="post" action="delete2izo.asp">
<p> <b> <u> <font face="arial" size="3"> ID   <input type="text"
name="t0" value=<%=s_t0%>> </u> </b> </p>
<p> <b> <u> <font face="arial" size="3"> Kind of Insulation   <input
type="text" name="t1" value=<%=s_t1%>> </u> </b> </p>
<p> <b> <u> Product <input type="text" name="t2" value=<%=s_t2%>>
</u> </b> </p>
<p> <b> <u> Amount of m2 <input type="text" name="t3" value=<%=s_t3%>>
</u> </b> </p>
<p> <b> <u> Picture  <input type="text" name="t4" value=<%=s_t4%>>
</u> </b> </p>
<p> <input type="submit" value="delete"> </p>
</form>
</body>
```

A.52. delete2izo.asp

```
<%@Language="VBScript"%>
<%dim s_t0
s_t0=CLng(Request.form("t0"))
s_t1=Request.form("t1")
s_t2=Request.form("t2")
s_t3=Request.form("t3")
s_t4=Request.form("t4")
set db=Server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
sql= "delete from izo3 where ID="&s_t0&""
db.execute(sql)
db.close
set db=nothing%>
<p align="center"> <font face="arial" color="#228B22" size="20"> Your
details are deleted successfully </p>
<br>
<p align="center"> <a href="izo1.html"> Back </a> </p>
```

A.53. uizo.asp

```
<body bgcolor="32CD32">
<form method="post" action="updateizo.asp">
<p align="center"> <b> <u> <font face="arial" size="3" > ID   <input
type="text" name="t0"> </u> </b> </p>
<p align="center"> You can find ID number what you want to update on
searching page </p>
<p align="center"> <input type="submit" value="update"> </p>
<p align="center"> <a href="izo1.html"> Back </a> </p>
</form>
</body>
updateizo.asp
<body bgcolor="32CD32">
<H1> <pre align="center"> <a href="uizo.asp"> Back </a>
       <font face="algerian" size="30" > IRFAN INSULATION </pre> </H1>
<p align="right"> <u> <font face="arial" size="5"> UPDATING PAGE </u>
</P>
<%@Language="VBScript"%>
<%del=cint(request.form("t0"))%>
<%set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from izo3 where ID="&del&""
set rec=db.execute(SQL)%>
<%s_t0=rec("ID")
s_t1=rec("kind_of_insulation")
s_t2=rec("product")
s_t3=rec("amount_of_m2")
s_t4=rec("picture")
rec.close
set rec=nothing
db.close
set db=nothing%>
<form method="post" action="update2izo.asp">
<p> <b> <u> <font face="arial" size="3"> ID   <input type="text"
name="t0" value=<%=s_t0%>> </u> </b> </p>
<p> <b> <u> <font face="arial" size="3"> Kind of ınsulation   <input
type="text" name="t1" value=<%=s_t1%>> </u> </b> </p>
```

```
<p> <b> <u> Product <input type="text" name="t2" value=<%=s_t2%>>
</u> </b> </p>
<p> <b> <u> Amount of m2 <input type="text" name="t3"
value=<%=s_t3%>></u> </b> </p>
<p> <b> <u> Picture <input type="text" name="t4" value=<%=s_t4%>> </u>
</b> </p>
<p> <input type="submit" value="update"> </p>
</form>
</body>
```

A.54. update2izo.asp

```
<%@Language="VBScript"%>
<%dim s_t0
s_t0=CLng(Request.form("t0"))
s_t1=Request.form("t1")
s_t2=Request.form("t2")
s_t3=Request.form("t3")
s_t4=Request.form("t4")
set db=Server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
sql= "UPDATE izo3 set kind_of_insulation='"&s_t1&"',
product='"&s_t2&"', amount_of_m2='"&s_t3&"', picture='"&s_t4&"'  where
ID="&s_t0&""

db.execute(sql)
db.close
set db=nothing%>
<p align="center"> <font face="arial" color="#228B22" size="20"> Your
details are updated successfully </p>
<br>
<p align="center"> <a href="izo1.html"> Back </a> </p>
```

A.55. izo2.html

```
<html>
<title> insulation!!! </title>
<Body background="C:\Documents and Settings\x\My
Documents\resimler\DSC021588.JPG">
<pre>  <font face="Broadway" font size="5">
<a href="izo1.html"> [THERMAL INSULATION] </a>




<font face="arial" size="5"><B><a href="izo2.html"> [CORRUBIT PANELS]
</a>  <a href="izo21.html"> SEE PRODUCT </a>          <a
href="sizo2.html"> SEARCHING PAGE </a>
                                                     <a
href="eizo2.asp"> ENTERING PAGE </a>
                                                     <a
href="dizo2.asp"> DELETING PAGE </a>
                                                     <a
href="uizo2.asp"> UPDATING PAGE </a>
```

```
<a href="izo3.html"> [WATER INSULATION] </a>
                                            <font size="15">  <a
href="isolation.html"> BACK </a>
</pre>
</html>
```

A.56. izo21.html

```
<html>
<title> insulation!!! </title>
<Body background="C:\Documents and Settings\x\My
Documents\resimler\DSC021588.JPG">
<pre>  <font face="Broadway" font size="5"> <a href="izo2.html">
[CORRUBIT PANELS] </a>   <font face="arial"> <a href="izo21.asp">
Corrubit Panels </a> </pre>
```

A.57. izo21.asp

```
<%@Language="vbscript"%>
<body bgcolor="#228B22">
<p align="center"> <font face="Broadway" size="15"> CORRUBIT PANEL
</p>
<p> <font face="arial" size="5"> Corrubit, bituminous corrugated
sheets, may easily be used extensively on a wide range of buildings,
on steel or wooden substructures or concrete slabs. Corrubit,
lightweight and waterproofing sheets, resist most chemical, UV and
frost. Lightweight sheets minimizing roofing load, creates a practical
and economical solution on new structures and re-roofing. Corrubit's
protective cover means it resists in all kind climates and conditions.
Sheets are produced 0,93 m width and 2,00 m long on three different
attractive colors, red, green and brown. </p>
<table align="center" border>
<tr> <th> Kind of Insulation </th> <th> Product </th> <th> Amount of
m2 </th>
<%set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from izo3 where ID=8"
set rec=db.execute(SQL)
while not rec.eof%>
<tr> <td> <%=rec("kind_of_insulation")%> </td>
<td> <%=rec("product")%> </td>
<td> <%=rec("amount_of_m2")%> </td>
<img src= "<%=rec ("picture")%>" height="250" width="350">
<%rec.movenext
wend
rec.close
set rec=nothing
```

```
db.close
set db=nothing%>
</table>
<p align="center"> <a href="isolation.html"> BACK </a> </p>
</body>
```

A.58. sizo2.html

```
<html>
<body bgcolor="32CD32">
<H1 align="center"><font face="algerian" size="30" > IRFAN GÖÇEN
CONSTRACTION LTD. </H1>
<p align="right"> <u> <font face="arial" size="5"> SEARCING PAGE </u>
</P>
<form method="post" action="sizo2.asp">
<p align="center"> <b> <u> <font face="arial" size="3"> Kind of
Insulation   <input type="text" name="cp" size="30" value="corrubit
panel"> </u> </b> </p>
<p align="center"> <input type="submit" value="submit"> </p>
<p align="center"> <a href="izo2.html"> BACK </a>
</form>
</body>
</html>
```

A.59. sizo2.asp

```
<%@Language="vbscript"%>
<body>
<pre align="left"> <a href="izo2.html"> Back </a>
      <font face="arial" size="15" font color="#228B22"> IRFAN
INSULATION </Pre>
<hr>
<%c=request.form("cp")
set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from izo3 where kind_of_insulation='"&c&"'"
set rec=db.execute(SQL)
while not rec.eof%>
<table align="center" border>
<tr> <th> ID </th> <th> Kind of Insulatıon </th> <th> product </th>
<th> Amount of m2 </th> <th> Picture </th>
<tr> <td> <%=rec("ID")%> </td>
<td> <%=rec("kind_of_insulation")%> </td>
<td> <%=rec("product")%> </td>
<td> <%=rec("amount_of_m2")%> </td>
<td> <img src= "<%=rec ("picture")%>" height="300" width="300"> </td>
<%rec.movenext
wend
rec.close
set rec=nothing
db.close
set db=nothing%>
</table>
</body>
```

A.60. eizo2.asp

```
<html>
<body bgcolor="32CD32">
<H1> <pre align="center"> <a href="izo2.html"> Back </a>
      <font face="algerian" size="30" > IRFAN INSULATION </pre> </H1>
<p align="right"> <u> <font face="arial" size="5"> ENTERING PAGE </u>
</P>
<form method="post" action="enterizo22.asp">
<p> <b> <u> <font face="arial" size="3"> Kind of Insulatıon   <input
type="text" name="t1" > </u> </b> </p>
<p> <b> <u> Product <input type="text" name="t2" >  </u> </b> </p>
<p> <b> <u> Amount of m2 <input type="text" name="t3" ></u> </b> </p>
<p> <b> <u> Picture </u> </b> </p>
<p> Attach Picture <input type="file" name="t4"
accept="appliction/pdf" /> </p>
<p> <input type="submit" value="enter"> </p>
</form>
</body>
</html>
```

A.61. enterizo22.asp

```
<%@Language="VBScript"%>
<%s_t1=Request.form("t1")
s_t2=Request.form("t2")
s_t3=Request.form("t3")
s_t4=Request.form("t4")
set db=Server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
sql="Insert into izo3(kind_of_insulation, product, amount_of_m2,
picture) values('"&s_t1&"','"&s_t2&"', '"&s_t3&"', '"&s_t4&"')"
db.execute(sql)
db.close
set db=nothing%>
<p align="center"> <font face="arial" color="#228B22" size="20"> Your
details are entered successfully </p>
<br>
<p align="center"> <a href="izo2.html"> BACK </a> </p>
```

A.62. dizo2.asp

```
<body bgcolor="32CD32">
<form method="post" action="deleteizo2.asp">
<p align="center"> <b> <u> <font face="arial" size="3" > ID   <input
type="text" name="t0"> </u> </b> </p>
<p align="center"> You can find ID number what you want to delete on
searching page</p>
<p align="center"> <input type="submit" value="delete"> </p>
<p align="center"> <a href="izo2.html"> Back </a> </p>
</form>
</body>
```

A.63. deleteizo2.asp

```
<body bgcolor="32CD32">
<H1> <pre align="center"> <a href="dizo2.asp"> Back </a>
       <font face="algerian" size="30" > IRFAN INSULATION </pre> </H1>
<p align="right"> <u> <font face="arial" size="5"> DELETING PAGE </u>
</P>
<%@Language="VBScript"%>
<%del=cint(request.form("t0"))%>
<%set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from izo3 where ID="&del&""
set rec=db.execute(SQL)%>
<%s_t0=rec("ID")
s_t1=rec("kind_of_insulation")
s_t2=rec("product")
s_t3=rec("amount_of_m2")
s_t4=rec("picture")
rec.close
set rec=nothing
db.close
set db=nothing%>
<form method="post" action="delete2izo2.asp">
<p> <b> <u> <font face="arial" size="3"> ID   <input type="text"
name="t0" value=<%=s_t0%>> </u> </b> </p>
<p> <b> <u> <font face="arial" size="3"> Kind of Insulation   <input
type="text" name="t1" value=<%=s_t1%>> </u> </b> </p>
<p> <b> <u> Product <input type="text" name="t2" value=<%=s_t2%>>
</u> </b> </p>
<p> <b> <u> Amount of m2 <input type="text" name="t3" value=<%=s_t3%>>
</u> </b> </p>
<p> <b> <u> Picture  <input type="text" name="t4" value=<%=s_t4%>>
</u> </b> </p>
<p> <input type="submit" value="delete"> </p>
</form>
</body>
```

A.64. delete2izo2.asp

```
<%@Language="VBScript"%>
<%dim s_t0
s_t0=CLng(Request.form("t0"))
s_t1=Request.form("t1")
s_t2=Request.form("t2")
s_t3=Request.form("t3")
s_t4=Request.form("t4")
set db=Server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
sql= "delete from izo3 where ID="&s_t0&""
db.execute(sql)
db.close
set db=nothing%>
<p align="center"> <font face="arial" color="#228B22" size="20"> Your
details are deleted successfully </p>
<br>
<p align="center"> <a href="izo2.html"> Back </a> </p>
```

A.65. uizo2.asp

```
<body bgcolor="32CD32">
<form method="post" action="updateizo2.asp">
<p align="center"> <b> <u> <font face="arial" size="3" > ID   <input
type="text" name="t0"> </u> </b> </p>
<p align="center"> You can find ID number what you want to update on
searching page </p>
<p align="center"> <input type="submit" value="update"> </p>
<p align="center"> <a href="izo2.html"> Back </a> </p>
</form>
</body>
```

A.66. updateizo2.asp

```
<body bgcolor="32CD32">
<H1> <pre align="center"> <a href="uizo2.asp"> Back </a>
      <font face="algerian" size="30" > IRFAN INSULATION </pre> </H1>
<p align="right"> <u> <font face="arial" size="5"> UPDATING PAGE </u>
</P>
<%@Language="VBScript"%>
<%del=cint(request.form("t0"))%>
<%set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from izo3 where ID="&del&""
set rec=db.execute(SQL)%>
<%s_t0=rec("ID")
s_t1=rec("kind_of_insulation")
s_t2=rec("product")
s_t3=rec("amount_of_m2")
s_t4=rec("picture")
rec.close
set rec=nothing
db.close
set db=nothing%>
<form method="post" action="update2izo2.asp">
<p> <b> <u> <font face="arial" size="3"> ID   <input type="text"
name="t0" value=<%=s_t0%>> </u> </b> </p>
<p> <b> <u> <font face="arial" size="3"> Kind of ınsulation   <input
type="text" name="t1" value=<%=s_t1%>> </u> </b> </p>
<p> <b> <u> Product <input type="text" name="t2" value=<%=s_t2%>>
</u> </b> </p>
<p> <b> <u> Amount of m2 <input type="text" name="t3"
value=<%=s_t3%>></u> </b> </p>
<p> <b> <u> Picture <input type="text" name="t4" value=<%=s_t4%>> </u>
</b> </p>
<p> <input type="submit" value="update"> </p>
</form>
</body>
```

A.67. update2izo2.asp

```
<%@Language="VBScript"%>
<%dim s_t0
s_t0=CLng(Request.form("t0"))
s_t1=Request.form("t1")
s_t2=Request.form("t2")
s_t3=Request.form("t3")
s_t4=Request.form("t4")
set db=Server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
sql= "UPDATE izo3 set kind_of_insulation='"&s_t1&"',
product='"&s_t2&"', amount_of_m2='"&s_t3&"', picture='"&s_t4&"'  where
ID="&s_t0&""
 db.execute(sql)
db.close
set db=nothing%>
<p align="center"> <font face="arial" color="#228B22" size="20"> Your
details are updated successfully </p>
<br>
<p align="center"> <a href="izo2.html"> Back </a> </p>
```

A.68. izo3.html

```
<html>
<title> insulation!!! </title>
<Body background="C:\Documents and Settings\x\My
Documents\resimler\DSC021588.JPG">
<pre>  <font face="Broadway" font size="5">
<a href="izo1.html"> [THERMAL INSULATION] </a>




<font face="arial" size="5"><B><a href="izo2.html"> [CORRUBIT PANELS]
</a>





<a href="izo3.html"> [WATER INSULATION] </a>    <a href="izo31.html">
SEE PRODUCT </a>        <a href="sizo3.html"> SEARCHING PAGE </a>
                                                      <a
href="eizo3.asp"> ENTERING PAGE </a>
                                                      <a
href="dizo3.asp"> DELETING PAGE </a>
                                                      <a
href="uizo3.asp"> UPDATING PAGE </a>

                                                <font size="15">  <a
href="isolation.html"> BACK </a>
</pre>
</html>
```

A.69. izo31.html

```
<html>
<title> insulation!!! </title>
<Body background="C:\Documents and Settings\x\My
Documents\resimler\DSC021588.JPG">
<pre>  <font face="Broadway" font size="5"> <a href="izo3.html">
[WATER INSULATION] </a>  <font face="arial"> <a href="izo32.html">
*Bitumen Waterproofing Membranes </a>
                                <a href="izo4.asp"> *Shingle </a>
</pre>


</html>
```

A.70. izo32.html

```
<html>
<title> insulation!!! </title>
<Body bgcolor="#228B22">
<pre>  <font face="Broadway" font size="5"> <a href="izo32.html">
[BITUMEN WATERPROOFING MEMBRANES] </a>    <font face="arial"> <a
href="izo33.asp"> ^Bitüself^ </a>
      <a href="izo31.html"> BACK </a>
                                                        <a
href="izo34.asp"> ^Polibit^ </a>
                                                        <a
href="izo35.asp"> ^Plastobit^ </a>
                                                        <a
href="izo36.asp"> ^Elastobit^ </a>
                                                        <a
href="izo37.asp"> ^Underslating Membranes^ </a> </pre>


<p align="center"> <img src  = "C:\Documents and Settings\x\My
Documents\resimler\DSC021588.JPG" width="1200" height="450">
</html>
```

A.71. izo33.asp

```
<%@Language="vbscript"%>
<body bgcolor="#228B22">
<p align="center"> <font face="Broadway" size="15"> BITUSELF </p>
<pre> <font face="arial" size="5"> Waterproofing and sealing tapes
that are self adhering and have their upper surface coated
with orange effect embossed aluminum foil can easily be used at any
kind of maintenance or waterproofing detail.
Since it is cold applied and is composed of tapes of various width, it
can easily be used by everyone.
  Bituself with complete water and vapour impermeability maintains its
stickiness
and water impermeability on wet surroundings as well. It can be used
quite easily on every kind of surface
whether it be smooth or inclined and adapts easily on the surface to
which it is applied. </pre>
<table align="center" border>
<tr> <th> Kind of Insulation </th> <th> Product </th> <th> Amount of
m2 </th>
<%set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from izo3 where ID=1"
```

```
set rec=db.execute(SQL)
while not rec.eof%>
<tr> <td> <%=rec("kind_of_insulation")%> </td>
<td> <%=rec("product")%> </td>
<td> <%=rec("amount_of_m2")%> </td>
<img src= "<%=rec("picture")%>" height="200" width="320">
<%rec.movenext
wend
rec.close
set rec=nothing
db.close
set db=nothing%>
</table>
<p align="center"> <a href="izo32.html"> BACK </a> </p>
</body>
```

A.72. izo34.asp

```
<%@Language="vbscript"%>
<body bgcolor="#228B22">
<p align="center"> <font face="Broadway" size="15"> POLIBIT </p>
<pre> <font face="arial" size="5">  Polibit series, bitumen
waterproofing membranes modified plastomer bitumen (APP) , is an
economical line
which can be applied at all waterprofing details of construction.
</pre>
<table align="center" border>
<tr> <th> Kind of Insulation </th> <th> Product </th> <th> Amount of
m2 </th>
<%set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from izo3 where ID=2"
set rec=db.execute(SQL)
while not rec.eof%>
<tr> <td> <%=rec("kind_of_insulation")%> </td>
<td> <%=rec("product")%> </td>
<td> <%=rec("amount_of_m2")%> </td>
<img src= "<%=rec ("picture")%>" height="448" width="115">
<%rec.movenext
wend
rec.close
set rec=nothing
db.close
set db=nothing%>
</table>
<p align="center"> <a href="izo32.html"> BACK </a> </p>
</body>
```

A.73. izo35.asp

```
<%@Language="vbscript"%>
<body bgcolor="#228B22">
<p align="center"> <font face="Broadway" size="15"> PLASTOBIT </p>
<pre> <font face="arial" size="5">  Plastomer polymer bitumen
PLASTOBIT waterproofing membranes can be used safely at almost every
climatic
conditions except on highlands and freezing areas as a consequence of
its high quality POLIMER bitumen modified
```

```
with APP and other thermoplastics and admits the technique of high
adherence at torch on application. </pre>
<table align="center" border>
<tr> <th> Kind of Insulation </th> <th> Product </th> <th> Amount of
m2 </th>
<%set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from izo3 where ID=3"
set rec=db.execute(SQL)
while not rec.eof%>
<tr> <td> <%=rec("kind_of_insulation")%> </td>
<td> <%=rec("product")%> </td>
<td> <%=rec("amount_of_m2")%> </td>
<img src= "<%=rec ("picture")%>" height="448" width="113">
<%rec.movenext
wend
rec.close
set rec=nothing
db.close
set db=nothing%>
</table>
<p align="center"> <a href="izo32.html"> BACK </a> </p>
</body>
<%@Language="vbscript"%>
<body bgcolor="#228B22">
<p align="center"> <font face="Broadway" size="15"> ELASTOBIT </p>
<pre> <font face="arial" size="5">  Elastomer modified bitumen (SBS)
waterproofing membrane, the Elastobit series, with
its perfect cold climate performance is also indispensable for
waterproofing of expanding/contracting structures
like light weight metal roofs. </pre>
<table align="center" border>
<tr> <th> Kind of Insulation </th> <th> Product </th> <th> Amount of
m2 </th>
<%set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from izo3 where ID=4"
set rec=db.execute(SQL)
while not rec.eof%>
<tr> <td> <%=rec("kind_of_insulation")%> </td>
<td> <%=rec("product")%> </td>
<td> <%=rec("amount_of_m2")%> </td>
<img src= "<%=rec ("picture")%>" height="448" width="118">
<%rec.movenext
wend
rec.close
set rec=nothing
db.close
set db=nothing%>
</table>
<p align="center"> <a href="izo32.html"> BACK </a> </p>
</body>
```

A.74. izo37.asp

```
<%@Language="vbscript"%>
<body bgcolor="#228B22">
<p align="center"> <font face="Broadway" size="15"> UNDERSLATING
MEMBRANES </p>
<pre> <font face="arial" size="5">  producer of polymeric and oxidized
bitumen waterproofing membranes, brings solutions with its new
underslating
membrane to leakages frequently encountered on tile roofs or roofs
with a low pitch. Water leaking through tile gaps
due to showers, stormy and rainy weather and snow accumulation or
leakages caused by tiles will no longer be a
problem. </pre>
<table align="center" border>
<tr> <th> Kind of Insulation </th> <th> Product </th> <th> Amount of
m2 </th>
<%set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from izo3 where ID=5"
set rec=db.execute(SQL)
while not rec.eof%>
<tr> <td> <%=rec("kind_of_insulation")%> </td>
<td> <%=rec("product")%> </td>
<td> <%=rec("amount_of_m2")%> </td>
<img src= "<%=rec ("picture")%>" height="250" width="350">
<%rec.movenext
wend
rec.close
set rec=nothing
db.close
set db=nothing%>
</table>
<p align="center"> <a href="izo32.html"> BACK </a> </p>
</body>
```

A.75. izo4.asp

```
<%@Language="vbscript"%>
<body bgcolor="#228B22">
<p align="center"> <font face="Broadway" size="15"> SHINGLE </p>
<pre> <font face="arial" size="5"> For pitched roofs and façades, its
oxidized bitumen body and glass tissue reinforcement offers
alternative
solutions with attractive colors.
On pitched roofs and fronts, its oxidized bitumen body and glass
tissue reinforcement offers alternative solutions
at attractive colors with its reflective mineral coated surface.
</pre>
<table align="center" border>
<tr> <th> Kind of Insulation </th> <th> Product </th> <th> Amount of
m2 </th>
<%set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from izo3 where ID=6"
set rec=db.execute(SQL)
```

```
while not rec.eof%>
<tr> <td> <%=rec("kind_of_insulation")%> </td>
<td> <%=rec("product")%> </td>
<td> <%=rec("amount_of_m2")%> </td>
<img src= "<%=rec ("picture")%>" height="250" width="350">
  <%rec.movenext
wend
rec.close
set rec=nothing
db.close
set db=nothing%>
</table>
<p align="center"> <a href="isolation.html"> BACK </a> </p>
</body>
```

A.76. sizo3.html

```
<html>
<body bgcolor="32CD32">
<H1 align="center"><font face="algerian" size="30" > IRFAN GÖÇEN
CONSTRACTION LTD. </H1>
<p align="right"> <u> <font face="arial" size="5"> SEARCING PAGE </u>
</P>
<form method="post" action="sizo3.asp">
<p align="center"> <b> <u> <font face="arial" size="3"> Kind of
Insulation   <input type="text" name="wi" size="30" value="water
insulation"> </u> </b> </p>
<p align="center"> <input type="submit" value="submit"> </p>
<p align="center"> <a href="izo3.html"> BACK </a>
</form>
</body>
</html>
```

A.77. sizo3.asp

```
<%@Language="vbscript"%>
<body>
<pre align="left"> <a href="izo3.html"> Back </a>
      <font face="arial" size="15" font color="#228B22"> IRFAN
INSULATION </Pre>
<hr>
<%w=request.form("wi")
set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from izo3 where kind_of_insulation='"&w&"'"
set rec=db.execute(SQL)
while not rec.eof%>
<table align="center" border>
<tr> <th> ID </th> <th> Kind of Insulatıon </th> <th> product </th>
<th> Amount of m2 </th> <th> Picture </th>
<tr> <td> <%=rec("ID")%> </td>
<td> <%=rec("kind_of_insulation")%> </td>
<td> <%=rec("product")%> </td>
<td> <%=rec("amount_of_m2")%> </td>
<td> <img src= "<%=rec ("picture")%>" height="300" width="300"> </td>
<%rec.movenext
wend
rec.close
set rec=nothing
```

```
db.close
set db=nothing%>
</table>
</body>
```

A.78. eizo3.asp

```
<html>
<body bgcolor="32CD32">
<%@Language="VBScript"%>
<H1> <pre align="center"> <a href="izo3.html"> Back </a>
     <font face="algerian" size="30" > IRFAN INSULATION </pre> </H1>
<p align="right"> <u> <font face="arial" size="5"> ENTERING PAGE </u>
</P>
<form method="post" action="enterizo23.asp">
<p> <b> <u> <font face="arial" size="3"> Kind of Insulatıon   <input
type="text" name="t1" > </u> </b> </p>
<p> <b> <u> Product <input type="text" name="t2" >  </u> </b> </p>
<p> <b> <u> Amount of m2 <input type="text" name="t3" ></u> </b> </p>
<p> <b> <u> Picture </u> </b> </p>
<p> Attach Picture <input type="file" name="t4"
accept="appliction/pdf" /> </p>
<p> <input type="submit" value="enter"> </p>
</form>
</body>
</html>
```

A.79. enterizo23.asp

```
<%@Language="VBScript"%>
<%s_t1=Request.form("t1")
s_t2=Request.form("t2")
s_t3=Request.form("t3")
s_t4=Request.form("t4")
set db=Server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
sql="Insert into izo3(kind_of_insulation, product, amount_of_m2,
picture) values('"&s_t1&"','"&s_t2&"', '"&s_t3&"', '"&s_t4&"')"
db.execute(sql)
db.close
set db=nothing%>
<p align="center"> <font face="arial" color="#228B22" size="20"> Your
details are entered successfully </p>
<br>
<p align="center"> <a href="izo3.html"> BACK </a> </p>
```

A.80. dizo3.asp

```
<body bgcolor="32CD32">
<form method="post" action="deleteizo3.asp">
<p align="center"> <b> <u> <font face="arial" size="3" > ID   <input
type="text" name="t0"> </u> </b> </p>
<p align="center"> You can find ID number what you want to delete on
searching page</p>
```

```
<p align="center"> <input type="submit" value="delete"> </p>
<p align="center"> <a href="izo3.html"> Back </a> </p>
</form>
</body>
```

A.81. deleteizo3.asp

```
<body bgcolor="32CD32">
<H1> <pre align="center"> <a href="dizo3.asp"> Back </a>
       <font face="algerian" size="30" > IRFAN INSULATION </pre> </H1>
<p align="right"> <u> <font face="arial" size="5"> DELETING PAGE </u>
</P>
<%@Language="VBScript"%>
<%del=cint(request.form("t0"))%>
<%set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from izo3 where ID="&del&""
set rec=db.execute(SQL)%>
<%s_t0=rec("ID")
s_t1=rec("kind_of_insulation")
s_t2=rec("product")
s_t3=rec("amount_of_m2")
s_t4=rec("picture")
rec.close
set rec=nothing
db.close
set db=nothing%>
<form method="post" action="delete2izo3.asp">
<p> <b> <u> <font face="arial" size="3"> ID   <input type="text"
name="t0" value=<%=s_t0%>> </u> </b> </p>
<p> <b> <u> <font face="arial" size="3"> Kind of Insulation   <input
type="text" name="t1" value=<%=s_t1%>> </u> </b> </p>
<p> <b> <u> Product <input type="text" name="t2" value=<%=s_t2%>>
</u> </b> </p>
<p> <b> <u> Amount of m2 <input type="text" name="t3" value=<%=s_t3%>>
</u> </b> </p>
<p> <b> <u> Picture  <input type="text" name="t4" value=<%=s_t4%>>
</u> </b> </p>
<p> <input type="submit" value="delete"> </p>
</form>
</body>
```

A.82. delete2izo3.asp

```
<%@Language="VBScript"%>
<%dim s_t0
s_t0=CLng(Request.form("t0"))
s_t1=Request.form("t1")
s_t2=Request.form("t2")
s_t3=Request.form("t3")
s_t4=Request.form("t4")
set db=Server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
sql= "delete from izo3 where ID="&s_t0&""
db.execute(sql)
db.close
```

```
set db=nothing%>
<p align="center"> <font face="arial" color="#228B22" size="20"> Your
details are deleted successfully </p>
<br>
<p align="center"> <a href="izo3.html"> Back </a> </p>
```

A.83. uizo3.asp

```
<body bgcolor="32CD32">
<form method="post" action="updateizo3.asp">
<p align="center"> <b> <u> <font face="arial" size="3" > ID   <input
type="text" name="t0"> </u> </b> </p>
<p align="center"> You can find ID number what you want to update on
searching page </p>
<p align="center"> <input type="submit" value="update"> </p>
<p align="center"> <a href="izo3.html"> Back </a> </p>
</form>
</body>
```

A.84. updateizo3.asp

```
<body bgcolor="32CD32">
<H1> <pre align="center"> <a href="uizo3.asp"> Back </a>
      <font face="algerian" size="30" > IRFAN INSULATION </pre> </H1>
<p align="right"> <u> <font face="arial" size="5"> UPDATING PAGE </u>
</P>
<%@Language="VBScript"%>
<%del=cint(request.form("t0"))%>
<%set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from izo3 where ID="&del&""
set rec=db.execute(SQL)%>
<%s_t0=rec("ID")
s_t1=rec("kind_of_insulation")
s_t2=rec("product")
s_t3=rec("amount_of_m2")
s_t4=rec("picture")
rec.close
set rec=nothing
db.close
set db=nothing%>
<form method="post" action="update2izo3.asp">
<p> <b> <u> <font face="arial" size="3"> ID   <input type="text"
name="t0" value=<%=s_t0%>> </u> </b> </p>
<p> <b> <u> <font face="arial" size="3"> Kind of ınsulation   <input
type="text" name="t1" value=<%=s_t1%>> </u> </b> </p>
<p> <b> <u> Product <input type="text" name="t2" value=<%=s_t2%>>
</u> </b> </p>
<p> <b> <u> Amount of m2 <input type="text" name="t3"
value=<%=s_t3%>></u> </b> </p>
<p> <b> <u> Picture <input type="text" name="t4" value=<%=s_t4%>> </u>
</b> </p>
<p> <input type="submit" value="update"> </p>
</form>
</body>
```

A.85. update2izo3.asp

```
<%@Language="VBScript"%>
<%dim s_t0
s_t0=CLng(Request.form("t0"))
s_t1=Request.form("t1")
s_t2=Request.form("t2")
s_t3=Request.form("t3")
s_t4=Request.form("t4")
set db=Server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
sql= "UPDATE izo3 set kind_of_insulation='"&s_t1&"',
product='"&s_t2&"', amount_of_m2='"&s_t3&"', picture='"&s_t4&"'  where
ID="&s_t0&""
db.execute(sql)
db.close
set db=nothing%>
<p align="center"> <font face="arial" color="#228B22" size="20"> Your
details are updated successfully </p>
<br>
<p align="center"> <a href="izo3.html"> Back </a> </p>
```

A.86. balizo.html

```
<html>
<body background="C:\Documents and Settings\x\My
Documents\aceng\VİLLALAR\DSC_0055.JPG">
<H1 align="center"> <img src="C:\Documents and Settings\x\My
Documents\resimler\DSC021599.JPG" width="1250" height="150"> </H1>
<p align="center"> <font face="arial" size="15"> <a href="sbizo.html">
Searching page </a> </p>
<p align="center"> <font face="arial" size="15"> <a
href="balanceeizo.asp"> Entering page </a> </p>
<p align="center"> <font face="arial" size="15"> <a
href="baldelizo.html"> Deleting page </a> </p>
<p align="center"> <font face="arial" size="15"> <a
href="balizou.html"> Updating page </a> </p>
<p align="center"> <a href= "isolations.html"> Back </a> </p>
</body>
</html>
```

A.87. sbizo.html

```
<html>
<body bgcolor="32CD32">
<H1 align="center"><font face="algerian" size="30" > IRFAN INSULATION
</H1>
<p align="right"> <u> <font face="arial" size="5"> SEARCING PAGE </u>
</P>
<form method="post" action="sbizo.asp">
<p align="center"> <b> <u> <font face="arial" size="3"> name   <input
type="text" name="name" size="10" > </u> </b> </p>
```

```
<p align="center"> <b> <u> <font face="arial" size="3"> surname
<input type="text" name="surname" size="10" > </u> </b> </p>
<p align="center"> <input type="submit" value="submit"> </p>
<p align="center"> <a href="balizo.html"> BACK </a>
</form>
</body>
</html>
```

A.88. sbizo.asp

```
<%@Language="VBScript"%>
<pre> <a href="sbizo.html"> BACK </a> <font face="algerian"
color="#228B22" size="20">                               BALANCE DETAILS
</pre>
<hr>
<%n=request.form("name")
s=request.form("surname")
set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
sql="select * from balanizo where name='"&n&"' and surname='"&s&"'"
set rec=db.execute(sql)
tot=0
while not rec.eof%>
<font face="arial" size="10">
<table border>
<tr> <th> ID </th> <th> name </th> <th> surname </th> <th> tel no
</th> <th> mobile phone </th> <th> address </th> <th> e_mail </th>
<th> date </th> <th> insulation number </th> <th> amount </th> <th>
debit </th>  <th> balance </th>
<tr> <td> <%=rec("ID")%> </td>
<td> <%=rec("name")%> </td>
<td> <%=rec("surname")%> </td>
<td> <%=rec("tel_no")%> </td>
<td> <%=rec("mobile_phone")%> </td>
<td> <%=rec("address")%> </td>
<td> <%=rec("e_mail")%> </td>
<td> <%=rec("u_date")%> </td>
<td> <%=rec("insulation_number")%> </td>
<td> <%=rec("amount")%> </td>
<td> <%=rec("debit")%> </td>
<%tot=tot+rec("amount")-rec("debit")%>
<td> <%=tot%> </td>
</table>
<%rec.movenext
wend
rec.close
set rec=nothing
db.close
set db=nothing%>
```

A.89. balanceeizo.asp

```
<%@language="vbscript"%>
<body bgcolor="32CD32">
<pre> <a href="balizo.html"> BACK </a> <font face="algerian" size="30"
>                          IRFAN INSULATION </pre>
<p align="right"> <u> <font face="arial" size="5"> BALANCE </u> </P>
<form method="post" action="balanceinsertizo.asp">
<font face="arial" size="3">
<p> <b> <u> Name  <input type="text" name="t1" > </u> </b> </p>
<p> <b> <u> Surname <input type="text" name="t2" >  </u> </b> </p>
<p> <b> <u> Tel NO <input type="text" name="t3" ></u> </b> </p>
<p> <b> <u> Mobile Phone <input type="text" name="t4" > </u> </b>
</p>
<p> <b> <u> Address <input type="text" name="t5" ></u> </b> </p>
<P> <b> <u> E_mail <input type="text" name="t6" > </u> </b> </p>
<p> <b> <u> Date <input type="text" name="t7" value=<%=date%> ></u>
</b> </p>
<pre><b><u><font face="arial" size="3">Amount <input type="text"
name="t8" >    <a href="detailizo.asp"> Click to see insulations
details </a>     <a href="calculate.asp"> Click to Calculate Amount
</a> </u> </b> </pre>
 <p> <b> <u> Debit <input type="text" name="t9" > </u> </b> </p>
<p> <b> <u> Insulation Number <input type="text" name="t10" > </u>
</b> </p>
<p> <input type="submit" value="enter"> </p>
</form>
</body>
```

A.90. balanceinsertizo.asp

```
<%@Language="VBScript"%>
<%s_t1=Request.form("t1")
s_t2=Request.form("t2")
s_t3=Request.form("t3")
s_t4=Request.form("t4")
s_t5=Request.form("t5")
s_t6=Request.form("t6")
s_t7=Request.form("t7")
s_t8=Cint(Request.form("t8"))
s_t9=Cint(Request.form("t9"))
s_t10=Request.form("t10")
set db=Server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
sql="insert into balanizo(name, surname, tel_no, mobile_phone,
address, e_mail, u_date, amount, debit, insulation_number) values
('"&s_t1&"', '"&s_t2&"', '"&s_t3&"', '"&s_t4&"', '"&s_t5&"',
'"&s_t6&"', '"&s_t7&"', "&s_t8&", "&s_t9&", '"&s_t10&"')"
db.execute(sql)%>
<%db.close
set db=nothing%>
<p align="center"> <font face="arial" color="32CD32" size="40"> YOUR
DETAILS ARE ENTERED SUCCESSFULLY </p>
<p align="center"> <a href="balizo.html"> BACK </a>
```

A.91. detailizo.asp

```
<%@Language="vbscript"%>
<body>
<pre align="left"> <a href="balanceeizo.asp"> Back </a>
      <font face="arial" size="15" font color="#228B22">
      BALANCE </Pre>
<hr>
<table align="center" border>
<tr> <th> ID </th> <th> Kind of Insulation </th> <th> Product </th>
<th> Amount of m2 </th>
<%set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from izo3"
set rec=db.execute(SQL)
while not rec.eof%>
<tr> <td> <%=rec("ID")%> </td>
<td> <%=rec("kind_of_insulation")%> </td>
<td> <%=rec("product")%> </td>
<td> <%=rec("amount_of_m2")%> </td>
<%rec.movenext
wend
rec.close
set rec=nothing
db.close
set db=nothing%>
</table>
</body>
```

A.92. calculate.asp

```
<%@Language="vbscript"%>
<form method="post" action="mulcal.asp">
<p> Amount of m2 <input type="text" name="t1"> </p>
<p> m2 <input type="text" name="t2"> </p>
<p> <input type="submit" value="multiply"> </p>
</form>
<p> <a href="balanceeizo.asp"> BACK </a> </p>
```

A.93. mulcal.asp

```
<%@Language="vbscript"%>
<%A=Cint(Request.form("t1"))
B=Cint(Request.form("t2"))%>
<p> <font face="arial" size="20"> Result is: <%Response.write(A*B)%>
</p>
<font face="arial" size="5"> <a href="balanceeizo.asp"> BACK </a>
```

A.94. baldelizo.html

```
<body bgcolor="32CD32">
<form method="post" action="baldelizo.asp">
<p align="center"> <b> <u> <font face="arial" size="3" > ID   <input
type="text" name="t0"> </u> </b> </p>
<p align="center"> <input type="submit" value="enter"> </p>
<p align="center"> you can find ID number what you want to delete on
searching page </p>
<p align="center"> <a href="balizo.html"> BACK </a> </p>
</form>
</body>
```

A.95. baldelizo.asp

```
<body bgcolor="32CD32">
<pre> <a href="baldelizo.html"> BACK </a> <font face="algerian"
size="30" >               IRFAN INSULATION </pre>
<p align="right"> <u> <font face="arial" size="5"> BALANCE PAGE </u>
</P>
<%@Language="VBScript"%>
<%del=cint(request.form("t0"))%>
<%set db=server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from balanizo where ID="&del&""
set rec=db.execute(SQL)%>
<%s_t0=rec("ID")
s_t1=rec("name")
s_t2=rec("surname")
s_t3=rec("tel_no")
s_t4=rec("mobile_phone")
s_t5=rec("address")
s_t6=rec("e_mail")
s_t7=rec("u_date")
s_t8=rec("amount")
s_t9=rec("debit")
s_t10=rec("insulation_number")
rec.close
set rec=nothing
db.close
set db=nothing%>
<form method="post" action="balandelizo.asp">
<p> <b> <u> <font face="arial" size="3"> ID   <input type="text"
name="t0" value=<%=s_t0%>> </u> </b> </p>
<p> <b> <u> <font face="arial" size="3"> name   <input type="text"
name="t1" value=<%=s_t1%>> </u> </b> </p>
<p> <b> <u> surname <input type="text" name="t2" value=<%=s_t2%>>
</u> </b> </p>
<p> <b> <u> tel no <input type="text" name="t3" value=<%=s_t3%>></u>
</b> </p>
<p> <b> <u> mobile phone <input type="text" name="t4"
value=<%=s_t4%>> </u> </b>  </p>
<p> <b> <u> address <input type="text" name="t5" value=<%=s_t5%>></u>
</b> </p>
<p> <b> <u> e_mail <input type="text" name="t6" value=<%=s_t6%>></u>
</b> </p>
<P> <b> <u> date <input type="text" name="t7" value=<%=s_t7%>> </u>
</b> </p>
```

```
<p> <b> <u> amount <input type="text" name="t8" value=<%=s_t8%>></u>
</b> </p>
<p> <b> <u> debit <input type="text" name="t9" value=<%=s_t9%>></u>
</b> </p>
<p> <b> <u> insulation number <input type="text" name="t10"
value=<%=s_t10%>></u> </b> </p>
<p> <input type="submit" value="delete"> </p>
 </form>
</body>
```

A.96. balandelizo.asp

```
<%@Language="VBScript"%>
<%dim s_t0
s_t0=CLng(Request.form("t0"))
s_t1=Request.form("t1")
s_t2=Request.form("t2")
s_t3=Request.form("t3")
s_t4=Request.form("t4")
s_t5=Request.form("t5")
s_t6=Request.form("t6")
s_t7=Request.form("t7")
s_t8=Request.form("t8")
s_t9=Request.form("t9")
s_t10=Request.form("t10")
set db=Server.createobject("ADODB.Connection")
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
sql= "delete from balanizo where ID="&s_t0&""
db.execute(sql)
db.close
set db=nothing%>
<p align="center"> <font face="arial" color="#228B22" size="20"> Your
details are deleted successfully </p>
<p align="center"> <a href="balizo.html"> BACK </a> </p>
```

A.97. balizou.html

```
<body bgcolor="32CD32">
<form method="post" action="balizo.asp">
<p align="center"> <b> <u> <font face="arial" size="3" > ID   <input
type="text" name="t0"> </u> </b> </p>
<p align="center"> <input type="submit" value="enter"> </p>
<p align="center"> you can find ID number what you want to update on
searching page </p>
<p align="center"> <a href="balizo.html"> BACK </a> </p>
</form>
</body>
```

A.98. balizo.asp

```
<body bgcolor="32CD32">
<pre> <a href="balizou.html"> BACK </a> <font face="algerian"
size="30" >                  IRFAN INSULATION </pre>
<p align="right"> <u> <font face="arial" size="5"> BALANCE PAGE </u>
</P>
<%@Language="VBScript"%>
<%del=cint(request.form("t0"))%>
<%set db=server.createobject("ADODB.Connection")
```

```
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
SQL="select * from balanizo where ID="&del&""
set rec=db.execute(SQL)%>
<%s_t0=rec("ID")
s_t1=rec("name")
s_t2=rec("surname")
s_t3=rec("tel_no")
s_t4=rec("mobile_phone")
s_t5=rec("address")
s_t6=rec("e_mail")
s_t7=rec("u_date")
s_t8=rec("amount")
s_t9=rec("debit")
s_t10=rec("insulation_number")
rec.close
set rec=nothing
db.close
set db=nothing%>
<form method="post" action="balanizo.asp">
<p> <b> <u> <font face="arial" size="3"> ID  <input type="text"
name="t0" value=<%=s_t0%>> </u> </b> </p>
<p> <b> <u> <font face="arial" size="3"> name   <input type="text"
name="t1" value=<%=s_t1%>> </u> </b> </p>
<p> <b> <u> surname <input type="text" name="t2" value=<%=s_t2%>>
</u> </b> </p>
<p> <b> <u> tel no <input type="text" name="t3" value=<%=s_t3%>></u>
</b> </p>
<p> <b> <u> mobile phone <input type="text" name="t4"
value=<%=s_t4%>> </u> </b>  </p>
<p> <b> <u> address <input type="text" name="t5" value=<%=s_t5%>></u>
</b> </p>
<p> <b> <u> e_mail <input type="text" name="t6" value=<%=s_t6%>></u>
</b> </p>
<P> <b> <u> date <input type="text" name="t7" value=<%=s_t7%>> </u>
</b> </p>
<p> <b> <u> amount <input type="text" name="t8" value=<%=s_t8%>></u>
</b> </p>
<p> <b> <u> debit <input type="text" name="t9" value=<%=s_t9%>></u>
</b> </p>
<p> <b> <u> insulation Number <input type="text" name="t10"
value=<%=s_t10%>></u> </b> </p>
<p> <input type="submit" value="update"> </p>
 </form>
</body>
```

A.99. balanizo.asp

```
<%@Language="VBScript"%>
<%dim s_t0
s_t0=CLng(Request.form("t0"))
s_t1=Request.form("t1")
s_t2=Request.form("t2")
s_t3=Request.form("t3")
s_t4=Request.form("t4")
s_t5=Request.form("t5")
s_t6=Request.form("t6")
s_t7=Request.form("t7")
s_t8=Request.form("t8")
s_t9=Request.form("t9")
s_t10=Request.form("t10")
set db=Server.createobject("ADODB.Connection")
```

```
db.open "Driver={Microsoft Access Driver (*.mdb)};
DBQ="&server.mappath("cons.mdb")
sql= "UPDATE balanizo set name='"&s_t1&"', surname='"&s_t2&"',
tel_no='"&s_t3&"', mobile_phone='"&s_t4&"', address='"&s_t5&"',
e_mail='"&s_t6&"', u_date='"&s_t7&"', amount="&s_t8&", debit="&s_t9&",
insulation_number='"&s_t10&"' where ID="&s_t0&""
db.execute(sql)
db.close
set db=nothing%>
<p align="center"> <font face="arial" color="#228B22" size="20"> Your
details are updated successfully </p>
<p align="center"> <a href="balizo.html"> BACK </a> </p>
```

A.100. About us.html

```
<html>
<body bgcolor="32CD32">
<p align="center"> <img src  =  "C:\Documents and Settings\x\My
Documents\resimler\DSC02169.JPG"  width=600 height=300>
 <font face="Broadway" size="15">   <ul><li> <p align="center"> <a
href="history.html"> ABOUT COMPANY </a> </li> </p>
                       <li> <p align="center"> <a
href="çalışanlar.html"> WORKERS ON THE CONSTRUCTION </a> </p> </li>
</ul>
<br>
<p align="center"> <font face="arial" size="5" > <a
href="mainpagee.html"> Back to Home Page </a> </p>
</body>
</html>
```

A.101. History.html

```
<html>
<Body bgcolor="#228B22">
<p align="center"> <b> <font face="arial" size="40"  > <marquee
behavior="alternate"> &#126 ABOUT COMPANY &#126 </marquee></a> </b>
</p>
<pre> <font face="arial" size="5">            Our company  establish
by Ahmet Göçen. He is director for this company.
           The company to established  in 20m2 small and rent place
for make and sell plate and glass
           on  Dr.  Fazıl  Küçük  main  street  on  Gönyeli/Güzelyurt
main  road  in  1995 .
           Atfirst the company name is Irfan Ticaret.  After that  we
develop  our works with ironmongery.
           The company place moved in 280m2 show room place and add
insulation/construction works
           to job at the same street in 2000.
           In these years our company  build twins  ans single  duplex
villas and sell them.
           Our  buildings  has  been  constructed  by  first
eartquake  system  calculation.
           And use quality materials and best workmanship when the
buildings has  been constructed.

           Our company do not  compensate about quality and
confidence.
           In  our  buildings  all  of  the  steps  have  been   taken
photos.
           Our  aim  is  do  not  make  huge  amount  of  work,our aim
is to
```

```
            build   safety   and   quality   buildings   and   sell
them.

            Our advise who wants to buy home come and visit our
buildings.
            We     can     build    buildings     about     your
special    wish. </pre>
<p align="center"> <a href="about us.html"> Back </a> </p>


</body>
</html>
```

A.102. workers.html

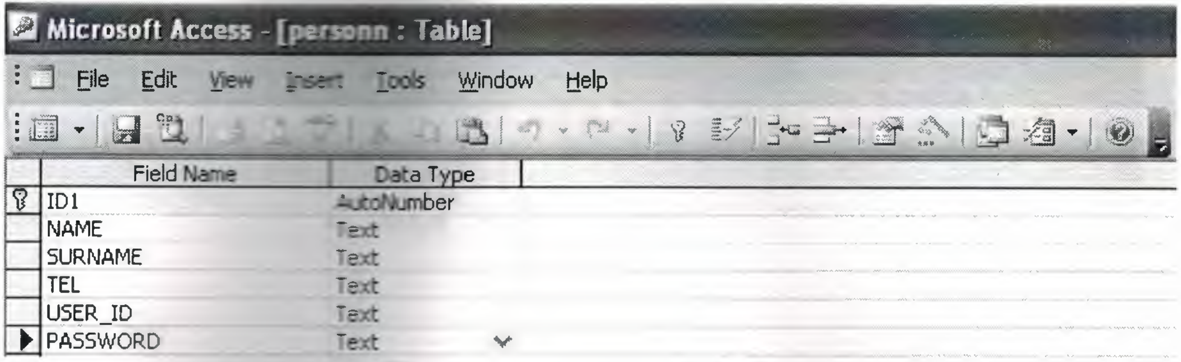
```
<html>
<body bgcolor="#32CD32">
 <table width="150" border="1" >
 <Tr> <TD> <font face="Broadway"  size="10"> Director </TD>
</table>
<pre> <img src="C:\Documents and Settings\x\My Documents\DSC00228.JPG"
width="100" height="100">  <font face="arial" size="4">    His name is
Ahmet Göçen. He is director on the company. He was established this
company.

<table width="380" border="1" >
 <Tr> <TD> <font face="Broadway"  size="10"> Civil Engineer </TD>
</table>
<pre> <img src="C:\Documents and Settings\x\My Documents\DSC00425.JPG"
width="100" height="100"> <font face="arial" size="4">           His
name is Cahit Göçen. He graduated from DAU in 2008. He is already
shareholder on the company.
                 He has been worken this company since he graduated.
E_mail:cahitgocen@gmail.com  </pre>


<table width="520" border="1" >
 <Tr> <TD> <font face="Broadway"  size="10"> Computer Engineer </TD>
</table>
<pre> <img src="C:\Documents and Settings\x\My Documents\DSC02139.JPG"
width="100" height="100"> <font face="arial" size="4">           Her
name is Anış Öztekin. She graduate from Near East University in 2008.
                 She design this web city for the company.
E_mail:anisoztekin@gmail.com


<table width="520" border="1" >
 <Tr> <TD> <font face="Broadway"  size="10"> Electrical Engineer </TD>
</table>
<pre> <img src="C:\Documents and Settings\x\My Documents\doğan.JPG">
<font face="arial" size="4">                        His name is Doğan
İZDİGİL. His graduated from Gazi University in 2001
                 He has been worken this company since 2003.
E_mail:doganizzigil@mynet.com


<table width="200" border="1" >
 <Tr> <TD> <font face="Broadway"  size="10"> Architect </TD>
</table>
<pre> <img src="C:\Documents and Settings\x\My Documents\mehmet.JPG" >
<font face="arial" size="4">                        His name is
Mehmet PEHLİVAN. His graduated from Gazi University in 1988.
```

```
                He has been worken this company since 2000.
</pre>
<p align="center"> <a href="about us.html"> Back </a> </p>
</body>
</html>
```

A.103. contactus.html

```
<HTML>
<Head> </Head>
<Body background="C:\Documents and Settings\x\My
Documents\resimler\DSC021588.JPG">
<p align="center"> <b> <font face="arial" size="40" > <marquee
behavior="alternate"> &#126 CONTACT INFORMATION &#126 </marquee></a>
</b> </p>
<br>
<pre><Table width="58%" height="50%" align="center" Border=4
cellspacing=12 cellpadding=12 bgcolor="FFFFFF"> </pre>
<Tr> <TD> <b>Phone </TD> <TD> +903922234634 </TD>
<Tr> <TD> <b>Mobile phone </TD> <TD> +905428526339 </TD>
<Tr> <TD> <b>Fax </TD> <TD> +903922234634 </TD>
<Tr> <TD> <b>E_mail </TD> <TD> <a href="mailto:anisoztekin@gmail.com">
Click to send e_mail </a> </TD>
<Tr> <TD> <b>Address </TD> <TD> Günaydın Sok. No:1 Yenikent- Gönyeli /
Lefkoşa </TD>
</Table>
<p align="center"><font face="arial" size="5" >  <a
href="mainpagee.html"> Back to Home Page </a> </p>
</Body>
</HTML>
```

APPENDIX B
Table of The Database

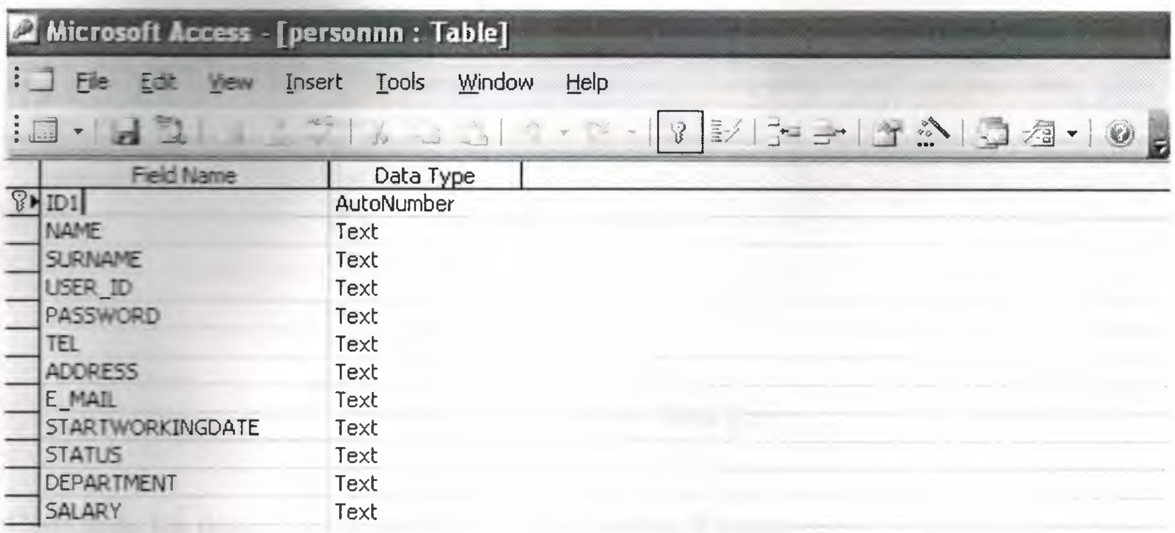


Microsoft Access - [personn : Table]
File Edit View Insert Tools Window Help

Field Name	Data Type
ID1	AutoNumber
NAME	Text
SURNAME	Text
TEL	Text
USER_ID	Text
PASSWORD	Text

Figure1. Personn Data Table

person.mdb for admin. It has include the ıd (primary key), name, surname, tel, user ıd, password datas. For admin entering the program, needed this database table. We have admin details in this table. We use this table for admin can login the web page.

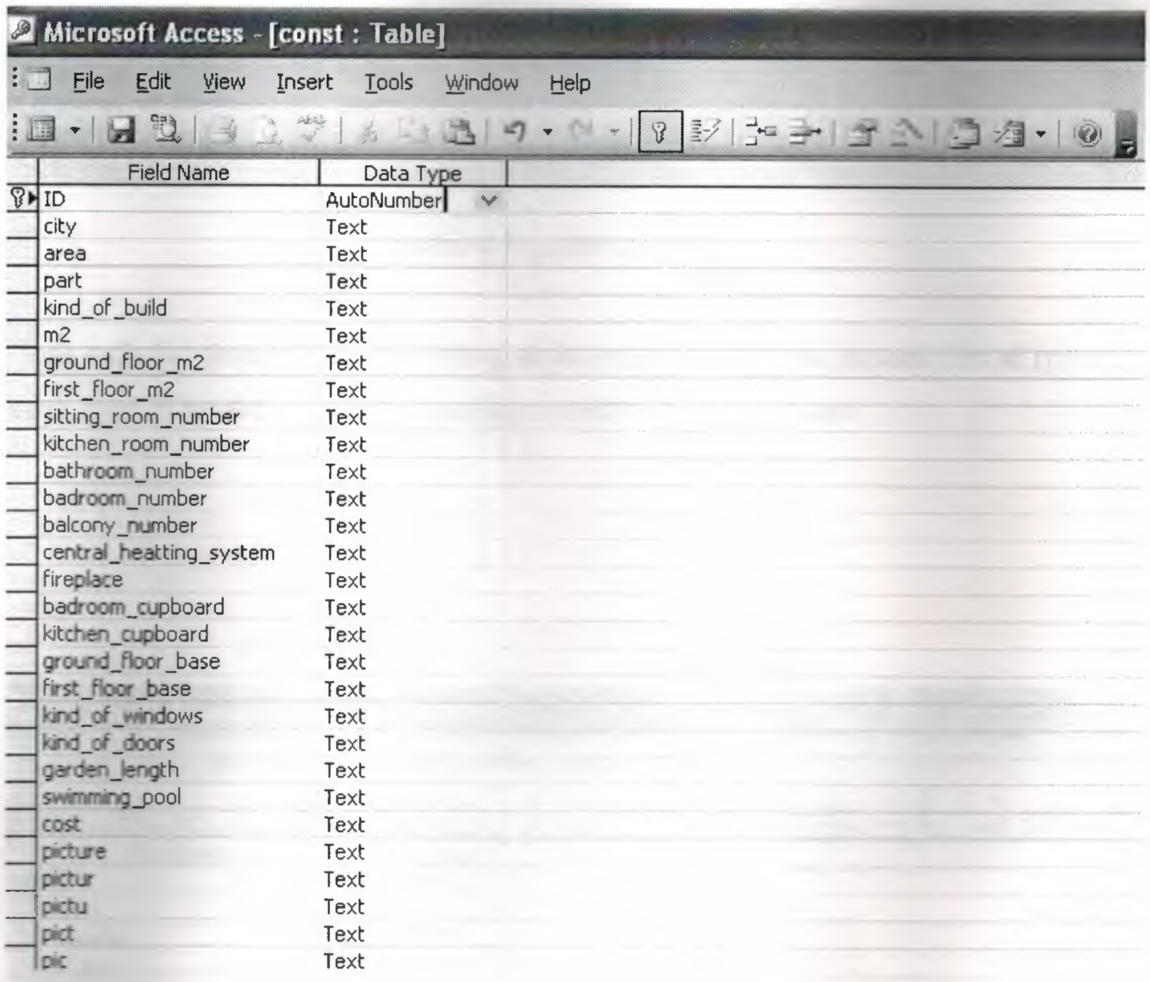


Microsoft Access - [personnn : Table]
File Edit View Insert Tools Window Help

Field Name	Data Type
ID1	AutoNumber
NAME	Text
SURNAME	Text
USER_ID	Text
PASSWORD	Text
TEL	Text
ADDRESS	Text
E_MAIL	Text
STARTWORKINGDATE	Text
STATUS	Text
DEPARTMENT	Text
SALARY	Text

Figure 2. Personnn Data Table

person.mdb for user. It has include the ıd (primary key), name, surname, user ıd, password, tel, address, e_mail, start working date, status, department, salary datas. For user entering the web page, needed this database table. We have user details in this table. We use this table for user can login the web page. Admin take personnel to worker company and admin give acknowledge the workers to enter web page like user for make search, insert and update. So in this table include user id and password for each workers get special user id and password to enter web page. And necessary details about workers to admin know workers in the company.

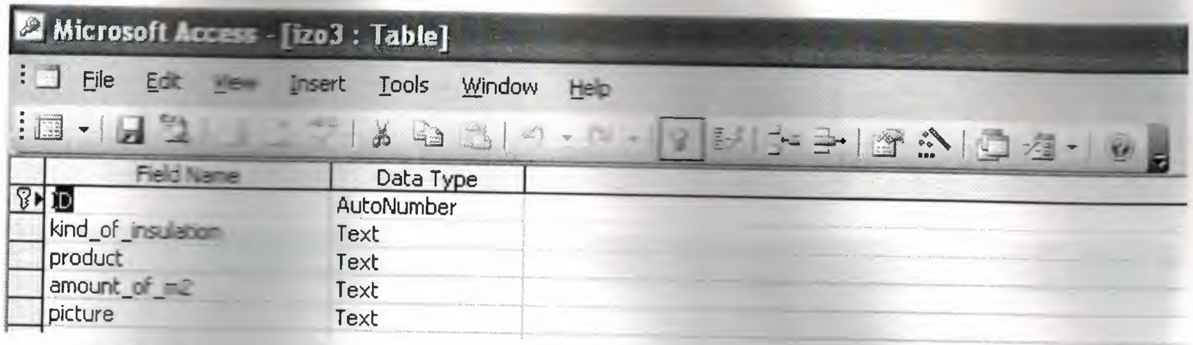


Microsoft Access - [const : Table]

Field Name	Data Type
ID	AutoNumber
city	Text
area	Text
part	Text
kind_of_build	Text
m2	Text
ground_floor_m2	Text
first_floor_m2	Text
sitting_room_number	Text
kitchen_room_number	Text
bathroom_number	Text
badroom_number	Text
balcony_number	Text
central_heatting_system	Text
fireplace	Text
badroom_cupboard	Text
kitchen_cupboard	Text
ground_floor_base	Text
first_floor_base	Text
kind_of_windows	Text
kind_of_doors	Text
garden_length	Text
swimming_pool	Text
cost	Text
picture	Text
pictur	Text
pictu	Text
pict	Text
pic	Text

Figure 3. Construction Data Table

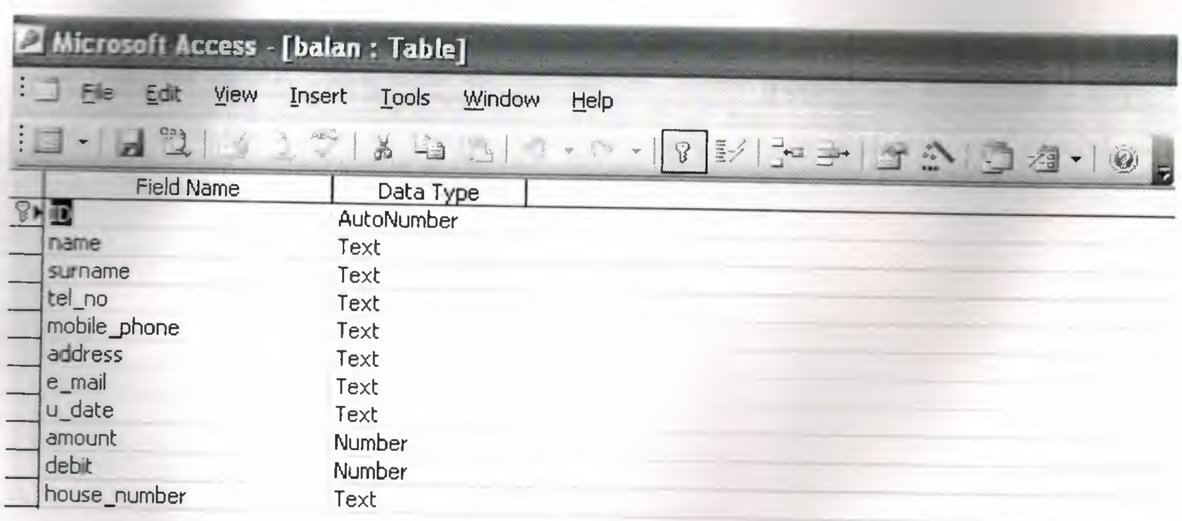
Cons.mdb for description every thing about builds. It has include the ıd (primary key) city, area, part, kind of build, m2, ground flar m2, first flar m2, sitting room number, bathroom number, badroom number, balcony number, central heatting system, fireplace, badroom cupboard, kitchen cupboard, ground flar flar, first flar flar, kind of windows, kind of doors, garden length, swimming pool, cost and 5 picture datas.


Microsoft Access - [izo3 : Table]
File Edit View Insert Tools Window Help

Field Name	Data Type
ID	AutoNumber
kind_of_insulation	Text
product	Text
amount_of_m2	Text
picture	Text

Figure 4. Insulation Data Table

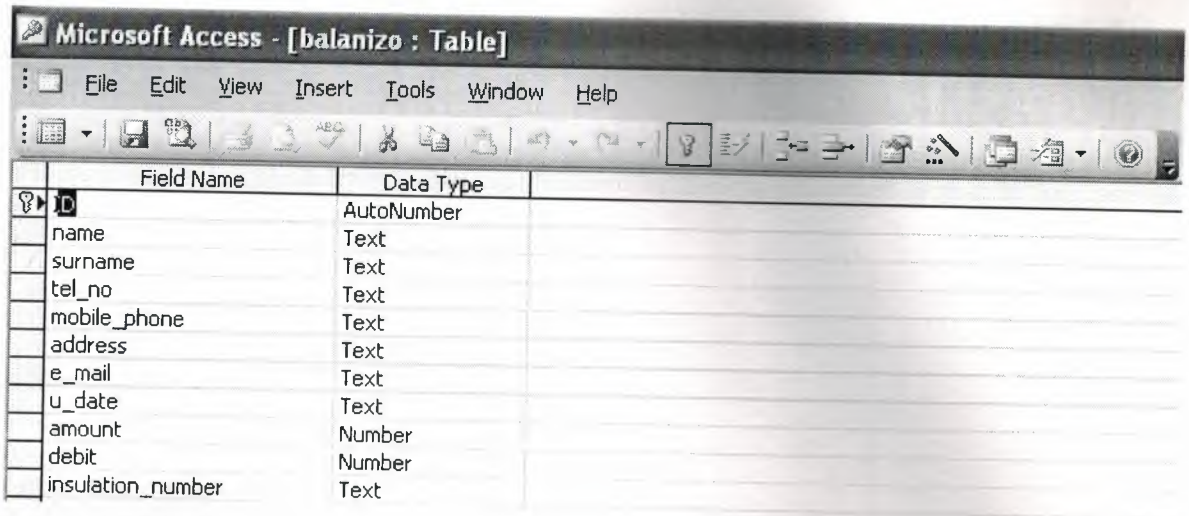
Cons.mdb for description every thing about insulations. It has include the ıd (primary key), kind of insulation, product, amount of m2 and picture datas.


Microsoft Access - [balan : Table]
File Edit View Insert Tools Window Help

Field Name	Data Type
ID	AutoNumber
name	Text
surname	Text
tel_no	Text
mobile_phone	Text
address	Text
e_mail	Text
u_date	Text
amount	Number
debit	Number
house_number	Text

Figure 5. Balance of constraction Data Table

Cons.mdb for details about who buy build. It has include the ıd (primary key), name, surname, tel no, mobile phone, address, e mail, date, amount, debit and house number datas.


Microsoft Access - [balanizo : Table]
File Edit View Insert Tools Window Help

Field Name	Data Type
ID	AutoNumber
name	Text
surname	Text
tel_no	Text
mobile_phone	Text
address	Text
e_mail	Text
u_date	Text
amount	Number
debit	Number
insulation_number	Text

Figure 6. Balance of insulation Data Table

Cons.mdb for details about who buy insulation. It has include the ıd (primary key), name, surname, tel no, mobile phone, address, e mail, date, amount, debit and insulation number datas.