Unit –3 Master page, Navigation, Validation, Login controls

3.1 Designing Master page:

A master page is the page that provides a template for other pages, master pages allow you to create a consistent look and behaviour for all the pages (or group of pages) in your web application with shared layout and functionality. The master page defines placeholders for the content, which can be overridden by content pages. The output result is a combination of the master page and the content page. The content pages contain the content you want to display. When users request the content page, ASP.NET merges the pages to produce output that combines the layout of the master page with the content of the content page.

In short:

1. The master page is a normal HTML page designed as a template for other pages.
2. The master page contains a placeholder tag <asp:ContentPlaceHolder> for individual content.
3. This master page was saved with the name "master1.master".
4. The content page is one of the individual content pages of the web.
5. The @ Page directive defines it as a standard content page.
6. This content page was saved with the name "Default.aspx".
7. When the user requests this page, ASP.NET merges the content page with the master page.
8. You can add master page from "Add New Item" option as below screen.

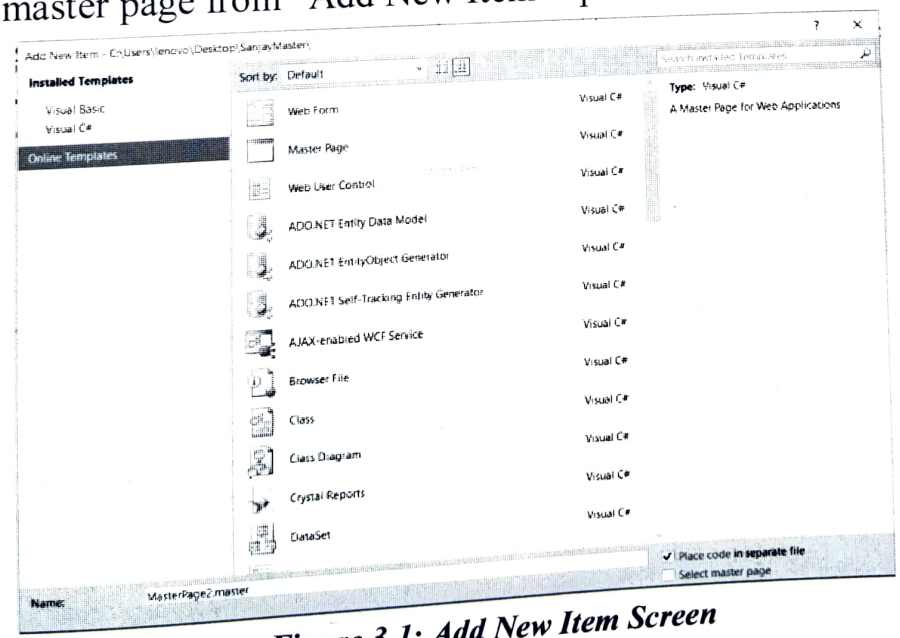


Figure 3.1: Add New Item Screen

Master Page Example:

```
<%@ Master %>
<html>
<body>
<h1>Standard Header From Masterpage</h1>
<asp:ContentPlaceHolder id="head" runat="server">
</asp:ContentPlaceHolder>
</body>
</html>
```

Content Page Example:

```
<%@ Page MasterPageFile="master1.master" %>
<asp:Content ContentPlaceHolderId="head" runat="server">
<h2>Individual Content</h2>
<p>Paragraph 1</p>
<p>Paragraph 2</p>
</asp:Content>
```

- **Demo example of the Master page:**

HTML code of MasterPage.master is,

```
<%@ Master Language="C#" AutoEventWireup="true"
   CodeFile="MasterPage.master.cs" Inherits="MasterPage" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
   Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
   transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>
        <asp:ContentPlaceHolder ID="Title" runat="server">
        </asp:ContentPlaceHolder>
    </title>
     <link href="style/StyleSheet.css" rel="stylesheet"
   type="text/css" />
       <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
```

```
    <form id="form1" runat="server">
        <div class="wrapper">
          <div class="footer">
              <center> <h2> GUJARAT UNIVERSITY,
AHMEDABAD</h2></center>
          </div>
          <div class="menu">
            <ul>
              <li> <a href="Default.aspx">Home</a></li>
              <li> <a
href="Default2.aspx">Category</a></li>
              <li> <a href="Default.aspx">Contact
Us</a></li>
              <li> <a href="Default.aspx">About
Us</a></li>
              </ul>
          </div>
          <div class="clear"></div>
          <div class="content">
              <asp:ContentPlaceHolder id="ContentBody"
runat="server">
              </asp:ContentPlaceHolder>
          </div>
          <div class="clear"></div>
          <div class="footer">
             <h2> CopyRight@sanjaySoni.com</h2>
          </div>
        </div>
    </form>
</body>
</html>
```

- **Content Page code in HTML:**

```
<%@ Page Language="C#"
   MasterPageFile="~/MasterPage.master"
   AutoEventWireup="true" CodeFile="Default.aspx.cs"
   Inherits="_Default" Title="Untitled Page" %>
```

```
<asp:Content ID="Content1" ContentPlaceHolderID="Title"
   Runat="Server">
Home
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="head"
   Runat="Server">
</asp:Content>
<asp:Content ID="Content3"
   ContentPlaceHolderID="ContentBody" Runat="Server">
     <h2> This Is Home Page</h2><br />
  </asp:Content>
```

- **CSS Code:**

```
body,div,p,ui,li
{
   padding: 0;
   margin: 0;
}
body
{
   background-color: rgb(237,237,237);
   font-family : "Arial",Sans-Serif;
   font-size: 12px;
}
.wrapper
{
   width: 1000px;
   margin: auto;
}
.content
{
   width:100%;
   background-color: rgb(254,254,254);
```

```
    margin-top:1;
    margin-bottom:1px;
    min-height:520px;
}
.menu
{
   background-color: rgb(10,110,178);
   width:100%;
   margin:0px 0px 10px;
   padding: 0px;
   height:40px;
   color:rgb(243,243,243);
}
.navigation_first_item
{
   border-left: 0px;
}
.navitem_s
{
   float: left;
   border-right: 1px solid rgb(10,85,125);
   border-left: 1px solid rgb(67,153,200);
   height:40px;
   background-color: rgb(14,79,114);
}
.menu ul {}
.menu ul li
{
   float:left;
   display:block;
   list-style:none;
   border-right: 1px solid rgb(10,85,125);
   border-left: 1px solid rgb(67,153,200);
}
```

```
.menu ul li.navigation_first_item : hover
{

}
.menu ul li a
{
   font-size: 13px;
   font-weight:bold;
   line-height:40px;
   padding: 8px 20px;
   color: rgb(255,255,255);
   text-decoration:none;
}
.menu ul li:hover
{
   background-color: rgb(14,79,114);
   border-right:1px solid rgb(14,89,130);
}
.clear
{
   clear:both;
}
.footer
{
   height:40px;
   background-color: rgb(10,110,178);
   color: rgb(255,255,255);
}
.footer h2
{
   padding: 15px;
   text-align:center;
}
```

- **Initial screen of the Master page:**

After run the web application, initial screen will display like this, and content of the default.aspx page will be displayed on the contentPlaceHolder in the master page.

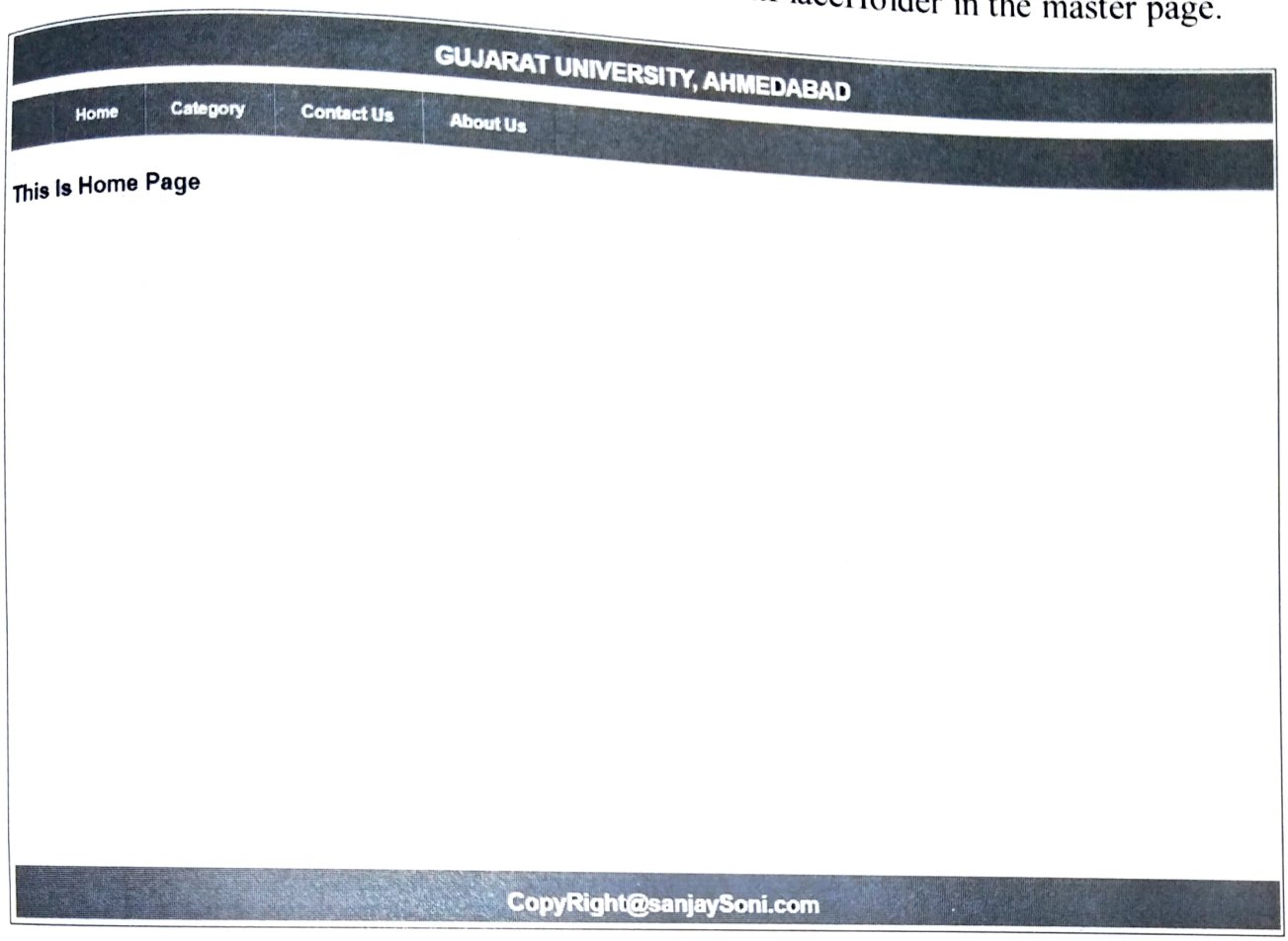


Figure 3.2: Mster Page Screen

3.2 Using Navigation Controls:

3.2.1 TreeView Control:

A TreeView is a collection of TreeNodes object. TreeView web control is useful while displaying hierarchal data in tree structure. It maintains the collection of columns. Also TreeView supports both postback-style events and simple hyperlink navigation, and rendering on different browsers. We will implement TreeView control to display the record on the browser. Each node represents by the text and value properties of TreeNodes.

The following figure 1 represents the basic format of the treeView control, as root is main directory and other parent 1 , parent 2 are sub directories, and leafs are the files/inter part of the each parent.

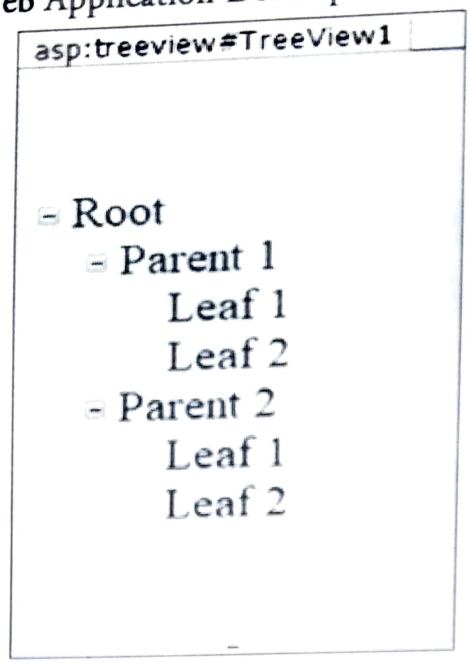


Figure 3.3: Basic format of TreeView control

We can also format of the treeview with different schemes as given below figure 2

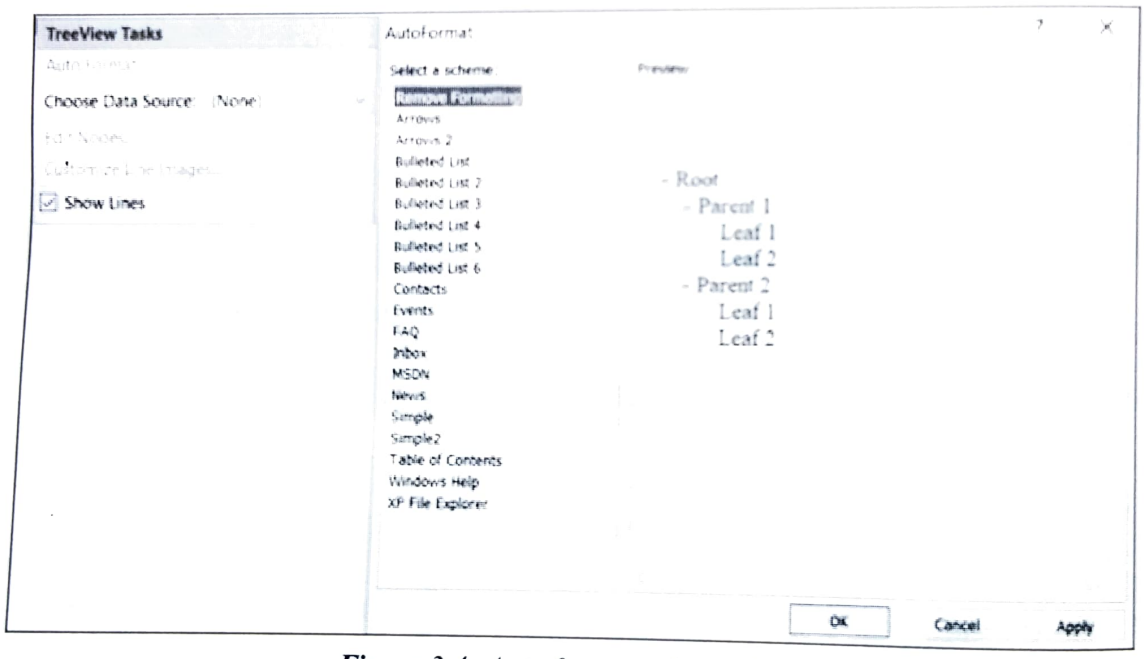


Figure 3.4: Auto format scheme dialog

We can edit the format using our required information as the based on navigation under parent or leaf nodes using the text property.

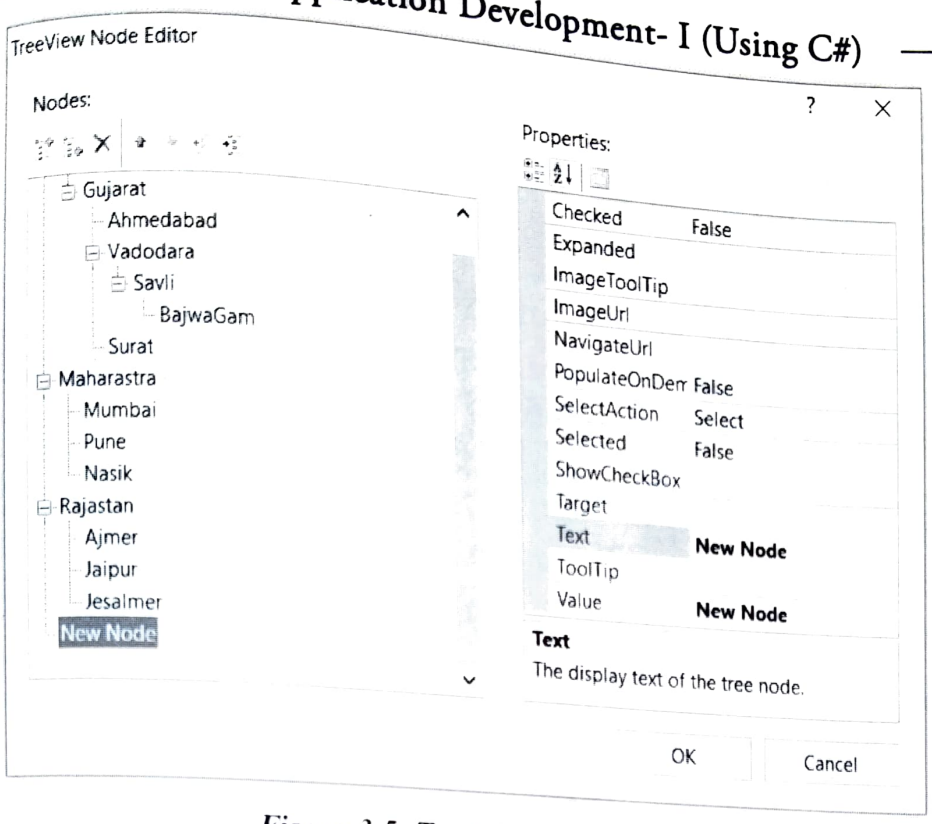


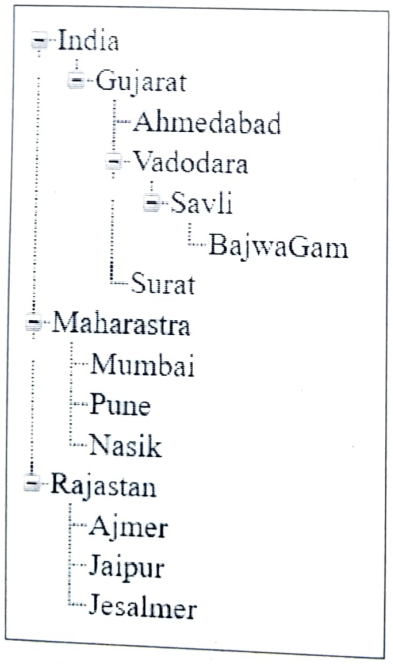
Figure 3.5: TreeView editor dialog

The following is the source code of the treeView control, in HTML body tag for displaying on the web browser

```
<asp:TreeView ID="TreeView1" runat="server" Height="244px"
ShowLines="True"
     Width="182px">
     <Nodes>
          <asp:TreeNode Text="India" Value="India">
               <asp:TreeNode Text="Gujarat" Value="Gujarat">
                    <asp:TreeNode Text="Ahmedabad"
Value="Ahmedabad"></asp:TreeNode>
                    <asp:TreeNode Text="Vadodara"
Value="Vadodara">
                         <asp:TreeNode Text="Savli"
Value="Savli">
                              <asp:TreeNode Text="BajwaGam"
Value="BajwaGam"></asp:TreeNode>
                         </asp:TreeNode>
                    </asp:TreeNode>
                    <asp:TreeNode Text="Surat"
Value="Surat"></asp:TreeNode>
```

```
            </asp:TreeNode>
          </asp:TreeNode>
          <asp:TreeNode Text="Maharastra"
Value="Maharastra">
            <asp:TreeNode Text="Mumbai"
Value="Mumbai"></asp:TreeNode>
            <asp:TreeNode Text="Pune"
Value="Pune"></asp:TreeNode>
            <asp:TreeNode Text="Nasik"
Value="Nasik"></asp:TreeNode>
          </asp:TreeNode>
          <asp:TreeNode Text="Rajastan" Value="Rajastan">
            <asp:TreeNode Text="Ajmer"
Value="Ajmer"></asp:TreeNode>
            <asp:TreeNode Text="Jaipur"
Value="Jaipur"></asp:TreeNode>
            <asp:TreeNode Text="Jesalmer"
Value="Jesalmer"></asp:TreeNode>
          </asp:TreeNode>
      </Nodes>
  </asp:TreeView>
```

After run the prescribed code, the output would be, like this, and after clicking the particular district or village, we will navigate the same web page. After linking by NavigateUrl



Properties of treeView control:

he following are some of the commonly used properties of the TreeView control

Property	Description
BackgroundImage	Gets or set the background image for the TreeView control.
BackgroundImageLayout	Gets or sets the layout of the background image for the TreeView control.
BorderStyle	Gets or sets the border style of the tree view control.
CheckBoxes	Gets or sets a value indicating whether check boxes are displayed next to the tree nodes in the tree view control.
Nodes	Gets the collection of tree nodes that are assigned to the tree view control.
RightToLeftLayout	Gets or sets a value that indicates whether the TreeView should be laid out from right-to-left.
ShowLines	Gets or sets a value indicating whether lines are drawn between tree nodes in the tree view control
Sorted	Gets or sets a value indicating whether the tree nodes in the tree view are sorted.
ForeColor	The current foreground color for this control, which is the color the control uses to draw its text
DataBindings	Gets the data bindings for the control.

Methods of the TreeView Control:

The following are some of the commonly used methods of the TreeView control

Methods	Description
ExpandAll	Expands all the nodes
CollapseAll	Collapses all the nodes including all child nodes in the tree view control.
GetNodeAt	Gets the node at the specified location
GetNodeCount	Gets the number of tree nodes
Sort	Sorts all the items in the tree view control.
ToString	Returns a string containing the name of the control

3.2.2 SiteMapPath:

The **SiteMapPath** control basically is used to access web pages from one webpage to navigate another page. It is a navigation control and displays the map of the site related to its web pages. You can click on that particular page in the Site Map to navigate to that page. We can say that the SiteMapPath control displays links for connecting to URLs of other pages. The representation of the SiteMapPath control is as follows:

Root Node->Child Node

First, add tree web forms to the application named *Home.aspx* and *AboutUs.aspx*, and *ContactUs.aspx* then,

Second, Right-click the application in the Solution Explorer window and then click the Add New Item option from the context menu. Select the Site Map Template from the Templates Pane. Note that, by default, the file has the name web.sitemap. See below,

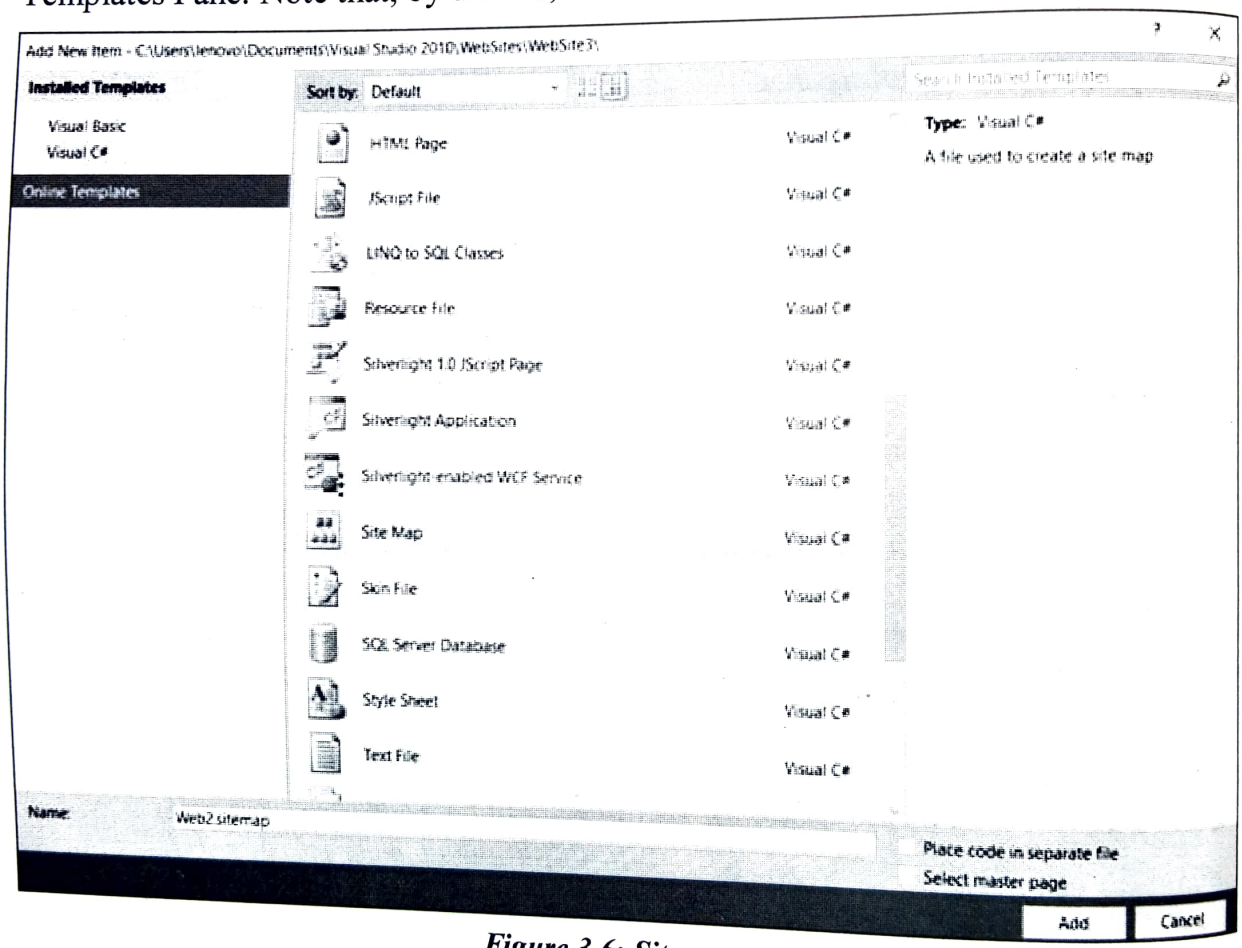


Figure 3.6: Sitemap

Edit the source code of the sitemap control, after placed with new separate file (web2.sitemap), and write the path of the above tree web forms (*Home.aspx*, *AboutUs.aspx*, and *ContactUs.aspx*) as,

Site Map file:

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap
  xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-
  1.0" >
    <siteMapNode url="home.aspx" title="Home....."
  description="">
        <siteMapNode url="aboutUs.aspx" title="About
  Us....."  description="" />
        <siteMapNode url="contactUs.aspx" title="Contact
  Us....."  description="" />
    </siteMapNode>
</siteMap>
```

Third, add the following code in all three aspx forms, as,

```
<body>
    <form id="form1" runat="server">
    <div>
            <asp:Calendar ID="Calendar1"
   runat="server"></asp:Calendar>
        <h1>Home....</h1>
        <asp:SiteMapPath ID="SiteMapPath1" runat="server">
        </asp:SiteMapPath>
    </div>
    </form>
</body>
```

Same code in aboutUs.aspx and ContactUs.aspx web form.

After inserting the above code, the design view look like below:

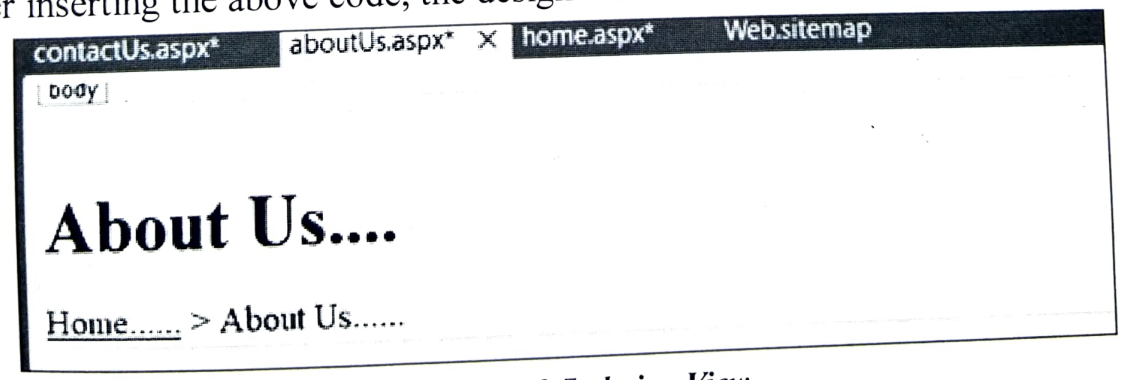


Figure 3.7: design View

Following are some important properties that are very useful.

Property	Description
PathSeparator	Gets or sets Path separator text. (By default it is >.)
NodeStyle	Sets the style of all nodes that will be displayed.
CurrentNodeStyle	Sets the style on node that represent the current page.
RootNodeStyle	Sets the style on the absolute root node.
PathSeparatorStyle	Sets the style of path separator.

3.2.3 Menu control:

The Menu control is used to create a menu of hierarchical data that can be used to navigate through the pages, it more similar to treeView control. The Menu control conceptually contains two types of items. First is StaticMenu that is always displayed on the page, second is DynamicMenu that appears when opens the parent item. Its properties like BackColor, ForeColor, BorderColor, BorderStyle, BorderWidth, Height etc. are implemented through style properties of <table, tr, td/> tag.

Following is the example of the menu control in vertical form.

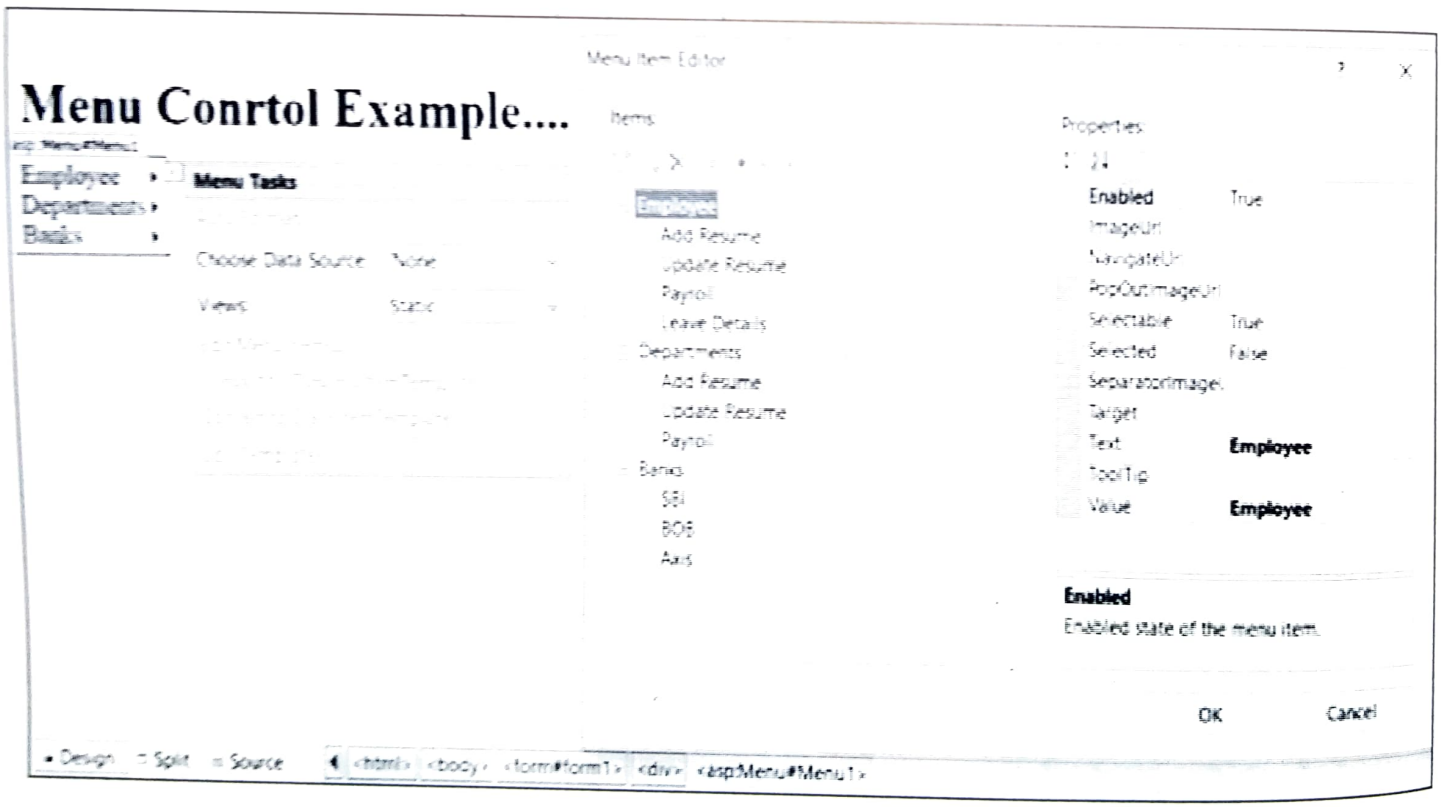


Figure 3.8: design View

After the run of the web page, the output would be look like this as vertical form.

Menu Conrtol Example....

Employee ▸
Departments ▸
Banks ▸ SBI
BOB
Axis

And if we want to display the output in horizontal form then we set the property **Orientation** to horizontal, then output will look like this,

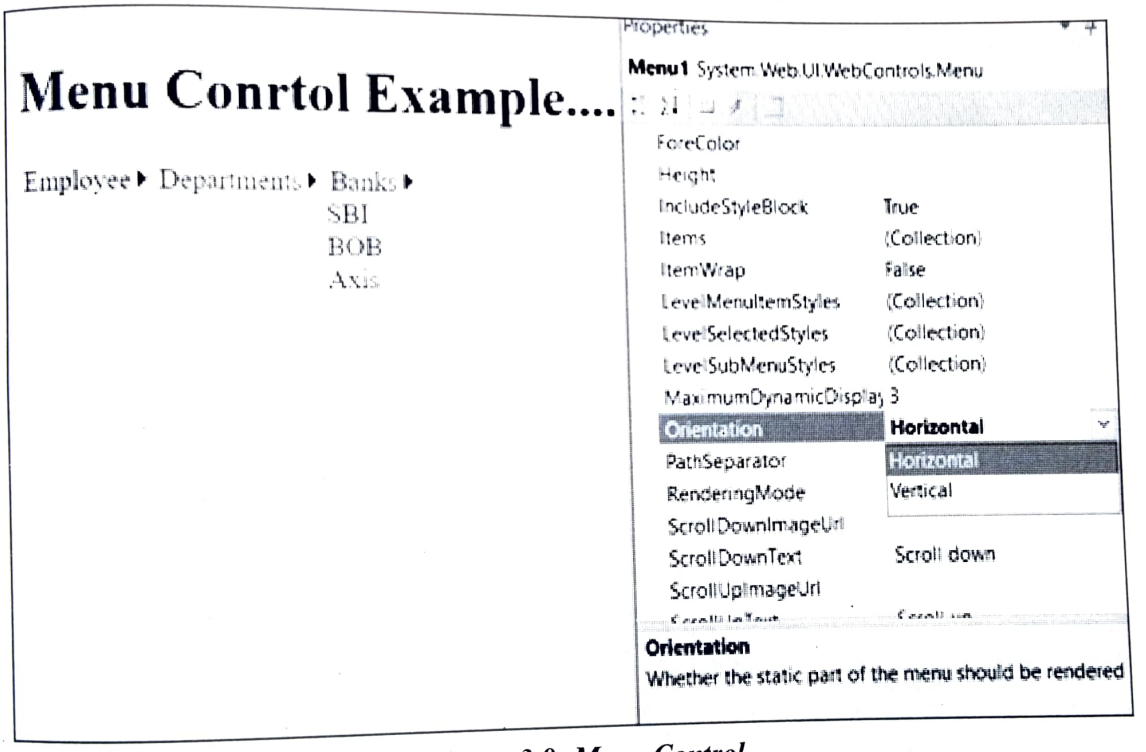


Figure 3.9: Menu Control

The form code of the menu control:

```
<Items>
 <asp:MenuItem Text="Employee" Value="Employee"
  NavigateUrl="~/employee.aspx">
 <asp:MenuItem Text="Add Resume" Value="Add Resume"
  NavigateUrl="~/addResume.aspx">></asp:MenuItem>
  <asp:MenuItem Text="Update Resume" Value="Update Delete"
  NavigateUrl="~/resume.aspx">></asp:MenuItem>
```

```
<asp:MenuItem Text="Payroll" Value="Payroll"
 NavigateUrl="~/payroll.aspx">></asp:MenuItem>
<asp:MenuItem Text="Leave Details" Value="Leave Details"
 NavigateUrl="~/leave.aspx">></asp:MenuItem>
 </asp:MenuItem>
          <asp:MenuItem Text="Departments"
Value="Departments" NavigateUrl="~/dpartment.aspx">>
              <asp:MenuItem Text="Sale" Value="Sale"
NavigateUrl="~/Sale.aspx">></asp:MenuItem>
              <asp:MenuItem Text="Perchase"
Value="Perchse"
NavigateUrl="~/Perchse.aspx">></asp:MenuItem>
              <asp:MenuItem Text="marketing"
Value="marketing"
NavigateUrl="~/marketing.aspx">></asp:MenuItem>
          </asp:MenuItem>
           <asp:MenuItem Text="Banks"
Value="Departments">
              <asp:MenuItem Text="SBI" Value="SBI"
NavigateUrl="~/SBI.aspx">></asp:MenuItem>
              <asp:MenuItem Text="BOB" Value="BOB"
NavigateUrl="~/BOB.aspx">></asp:MenuItem>
              <asp:MenuItem Text="Axis" Value="Axis"
NavigateUrl="~/Axis.aspx">></asp:MenuItem>
          </asp:MenuItem>
      </Items>
```

Basic properties of the menu control:

Property	Description
Attributes	Gets the collection of arbitrary attributes (for rendering only) that do not correspond to properties on the control.
CssClass	Gets or sets the Cascading Style Sheet (CSS) class rendered by the Web server control on the client

DataBindings	Gets a collection of MenuItemBinding objects that define the relationship between a data item and the menu item it is binding to
Items	Gets a MenuItemCollection object that contains all menu items in the Menu control.
Orientation	Gets or sets the direction in which to render the Menu control
SelectedItem	Gets the selected menu item.
Style	Gets a collection of text attributes that will be rendered as a style attribute on the outer tag of the Web server control.

3.2.4 Creating sitemap file for navigation:

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap
   xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-
   1.0" >
     <siteMapNode url="home.aspx" title="Home....."
   description="">
          <siteMapNode url="aboutUs.aspx" title="About
   Us....."  description="" />
          <siteMapNode url="contactUs.aspx" title="Contact
   Us....."  description="" />
     </siteMapNode>
 </siteMap>
```

3.3 Validation controls:

3.3.1 Required Field Validator:

Required Field Validator is used to make a control required; this validator is used to make an input control required. It will throw an error if user leaves input control empty. It is used to mandate form control required and restrict the user to provide data.

Required properties to be set of the Required Field Validator

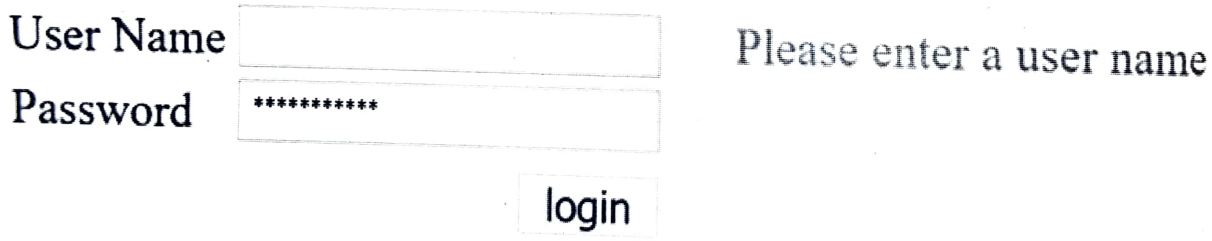
Properties	description
Access Key	It is used to set keyboard shortcut for the control.

Text	It is used to set text to be shown for the control.
ToolTip	It displays the text when mouse is over the control.
Visible	To set visibility of control on the form.
Height	It is used to set height of the control.
Width	It is used to set width of the control.
ErrorMessage	It is used to set error message that display when validation fails.
ControlToValidate	It takes ID of control to validate.

Webpage code

```
<asp:TextBox ID="username" runat="server"></asp:TextBox>
<asp: RequiredFieldValidatorID asp:
 RequiredFieldValidatorID
   ="user" runat="server" ControlToValidate="username"
ErrorMessage="Please enter a user name" ForeColor="Red">
</asp:RequiredFieldValidator>
```

If we does not enter the user name the RequiredFieldValidator control track the error message as shown below



3.3.2 Range validator:

Range validator evaluates the value of an input control to check that the value lies between specified ranges. It allows us to check whether the user input is between a specified upper and lower boundary. *This range can be numbers, alphabetic characters and dates.*

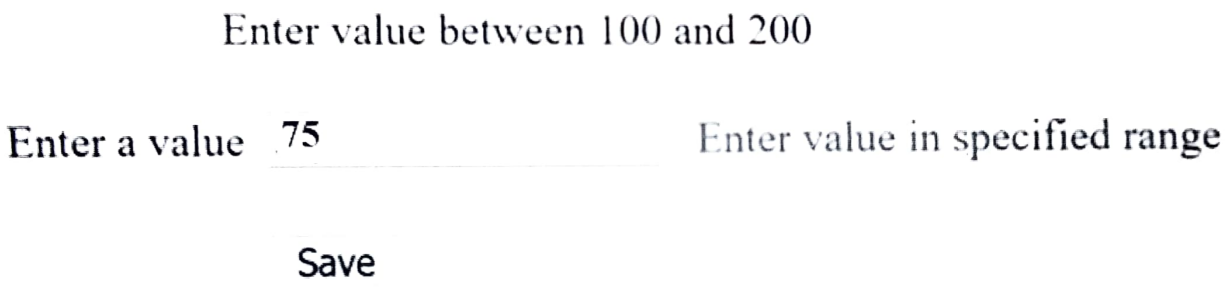
If we input the less then or greater then excluding the specified range, then the error message will throws for suggestion. The Range Validator control verifies that the input value falls within a predetermined range. It has three specific properties

Properties	description
Type	It defines the type of the data. The available values are: Currency, Date, Double, Integer, and String.
MinimumValue	It specifies the minimum value of the range.
MaximumValue	It specifies the maximum value of the range.

The syntax of the control is as given with webpage code:

```
<asp:RangeValidator ID="rvclass" runat="server"
  ControlToValidate="txtclass"
     ErrorMessage="Enter value (100 - 200)"
  MaximumValue="200"
     MinimumValue="100" Type="Integer">
</asp:RangeValidator>
```

The error message will be generated, if user enter the value less than 100 and greater than 200 as shown below



3.3.3 Regular Expression validator:

Regular Expression validator is used to validate the value of an input control against the pattern defined by a regular expression. It allows to check and validate expected sequences of characters like: ***e-mail address, telephone number, social security number*** etc. The ValidationExpression property is used to specify the regular expression, this expression is used to validate input control. BCA*/d{2}.....(BCA01)

Properties	description
ErrorMessage	It is used to set error message that display when validation fails.
ControlToValidate	It takes ID of control to validate.
ValidationExpression	It is used to set regular expression to determine validity.

ForeColor	It is used to set colour of the control text.
Text	It is used to set text to be shown for the control.
BackColor	It is used to set background colour of the control.
BorderColor	It is used to set border colour of the control.

Here, in the following example, we are validating user input by using Regular Expression validator. Source code of the example is given below.

Regular_Expression_validator _demo.aspx:

```
<asp:
   TextBox ID="username" runat="server"></asp:TextBox>
<asp:RegularExpressionValidator ID="RegularExpressionVali
   ator1" runat="server"ControlToValidate="username"
ErrorMessage="Please enter valid E-
   mail" ForeColor="Red"ValidationExpression="\w+([-
   +.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*">
</asp: RegularExpressionValidator>
```

Below error message will be generated when user does not enter proper e-mail id

3.3.3.1 Compare validator:

operator property is mandatory to set.

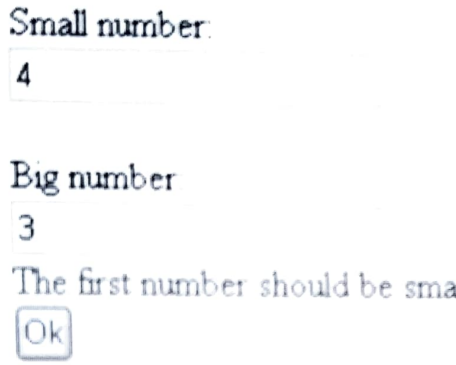
The CompareValidator control compares a value in one control with a fixed value or a value in another control. It can compare two values, for instance the values of two controls.

I will show you a small example of how it can be used.

```
Small number:<br />
<asp:TextBox runat="server" id="txtSmallNumber" /><br
   /><br />
Big number:<br />
<asp:TextBox runat="server" id="txtBigNumber" /><br />
<asp:CompareValidator runat="server" id="cmpNumbers"
controltovalidate="txtSmallNumber"
```

```
controltocompare="txtBigNumber" operator="LessThan"
type="Integer" errormessage="The first number should be
smaller than the second number!" /><br />
```

After inserting the big number in textbox 1 then the error message will be generated in red color



It has the following specific properties:

Properties	description
Type	It specifies the data type.
ControlToCompare	It specifies the value of the input control to compare with.
ValueToCompare	It specifies the constant value to compare with.
Operator	It specifies the comparison operator, the available values are: Equal, NotEqual, GreaterThan, GreaterThanEqual, LessThan, LessThanEqual, and DataTypeCheck.
ErrorMessage	It is used to display error message when validation failed

most important to set when use this validator.

Here, in the following example, we are validating user input by using CompareValidator controller. Source code of the example is given below.

3.3.3.2 Custom validator:

CustomValidator control provides the customize validation code which is created by own for required check particular validation to perform both client-side and server-side validation. For example, you can create a validation control that checks whether the value entered into a text box is 10 or more characters long. If none of the other validation controls perform the type of validation that you need, you can always use the CustomValidator control. The control allows you to validate both client side and server side, where the server side approach is probably the most powerful. You can associate your custom validation function with the CustomValidator control by handling the Server Validate event.

First example custom validator:

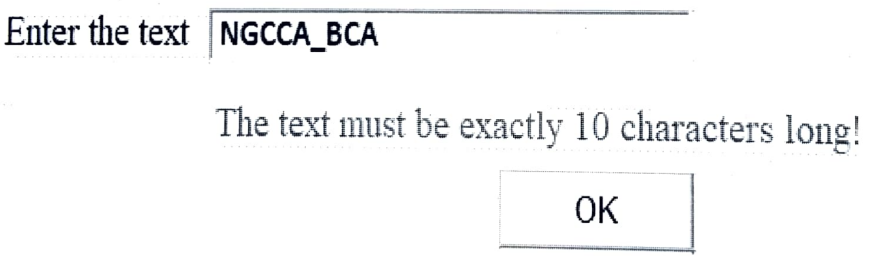
Let take simple example simply check the length of the string in the TextBox, write below method/code in **CodeBehind** which will handle the validation.

```
Protected void cusCustom_ServerValidate(object sender,
ServerValidateEventArgs e)
{
  If (e.Value.Length == 10)
      e.IsValid = true;
  else
      e.IsValid = false;
}
```

Webpage code:

```
Custom text:<br /> <asp:TextBox runat="server"
id="txtCustom" />
<asp:CustomValidator runat="server" id="cusCustom"
controltovalidate="txtCustom"
onservervalidate="cusCustom_ServerValidate"
errormessage="The text must be exactly 10 characters
long!" /> <br /><br />
```

After the run the above script of custom validator, the error message will be generated after inputting less than 10 character in the text box. Here whatever code can be define in the procedure of the *cusCustom_ServerValidate for the* validation.



Second example of custom validator:

CodeBehind

```
protected void cusCustom_ServerValidate(object source,
ServerValidateEventArgs args )
    {
        string inputData = args.Value;
        args.IsValid = false;
        if (inputData.Length < 6 || inputData.Length >
10) return;
```

```
bool upperCase = false;
foreach (char ch in inputData)
{
      if (ch >= 'A' && ch <= 'Z')
      {
            upperCase = true;
            break;
      }
}
if (!upperCase) return;
bool lowerCase = false;
foreach (char ch in inputData)
{
       if (ch >= 'a' && ch <= 'z')
       {
             lowerCase = true; break;
        }
 }
 if (!lowerCase) return;
 bool number = false;
 foreach (char ch in inputData)
{
      if (ch >= '0' && ch <= '9')
      {
            number = true; break;
      }
 }
 if (!number) return;
 args.IsValid = true;
}
```

Webpage code:

```
<asp:Label ID="Label1" runat="server"
Text="Password"></asp:Label>
 <asp:TextBox ID="txtPassword" runat="server"
Width="143px" ToolTip="Password must be between 6-12
characters and include 1 capital letter, 1 lowercase
letter, and 1 number"></asp:TextBox>  <asp:CustomValidator
ID="CustomValidator1" runat="server"
ControlToValidate="txtPassword" ErrorMessage="Password
must be between 6-10 characters with 1 letter, 1
```

```
lowercase, and 1 number"
OnServerValidate="CustomValidator1_ServerValidate"></asp:C
ustomValidator>
```

Validation suggestion\output would be,

Enter Password [BCA]

Password must be between 6-10 characters with 1 letter, 1 lowercase, and 1 number

OK

3.3.3.3 Validation summary:

Summary validator control is used to display ***list of all validation errors*** in the web form. It allows us to ***summarize the error messages*** at a **single location**. We can set Display Mode property to display error messages as a list, bullet list or single paragraph. We can set a header text for validation summary. ASP.NET Validation Summary control have many properties to design the error messages text as like fore color, back color, border color, border style, border width, theme, skin and after all CSS class.

Basic properties of the validation summary control:

Property	Description
ShowMessageBox	It displays a message box on error in up-level browsers.
ShowSummary	It is used to show summary text on the form page.
ShowValidationErrors	It is used to set whether the validation summary should be shown or not.
ForeColor	It is used to set color of the control text.
Text	It is used to set text to be shown for the control.
AccessKey	It is used to set keyboard shortcut for the control.
BackColor	It is used to set background color of the control.

Example validation summary control:

Webpage code

```
<td class="auto-style2">User Name</td>
<asp:TextBox ID="username" runat="server"></asp:TextBox>
<asp:
RequiredFieldValidator ID="user" runat="server" ControlToV
alidate="username"
ErrorMessage="Please enter a user name" ForeColor="Red">*<
/asp:RequiredFieldValidator>
<td class="auto-style2">Password</td>
<asp:TextBox ID="password" runat="server"></asp:TextBox>
<asp:
RequiredFieldValidator ID="pass" runat="server" ControlToV
alidate="password"
ErrorMessage="Please enter a password" ForeColor="Red">*</
asp:RequiredFieldValidator>
<asp:
ValidationSummary ID="ValidationSummary1" runat="server" F
oreColor="Red"/>
```

Validation suggestion\output would be,

The list of errors are displayed by the validation summary control, if user does not enter both user name and password

User Name

Password

- Please enter a user name
- Please enter a password

login

3.4 Login Controls:

3.4.1 Login:

The Login control provides a user interface which contains username and password that authenticate the user name and password and grant the access to the desired services on the basis of the authentication. There are used one methods, properties and events in this Login control, you can check manually after inserting the control on web form as given below. There are two way to create login, first using database, and second using

ASP.NET configuration wizard. Here we will check the authenticity using Authenticate event.

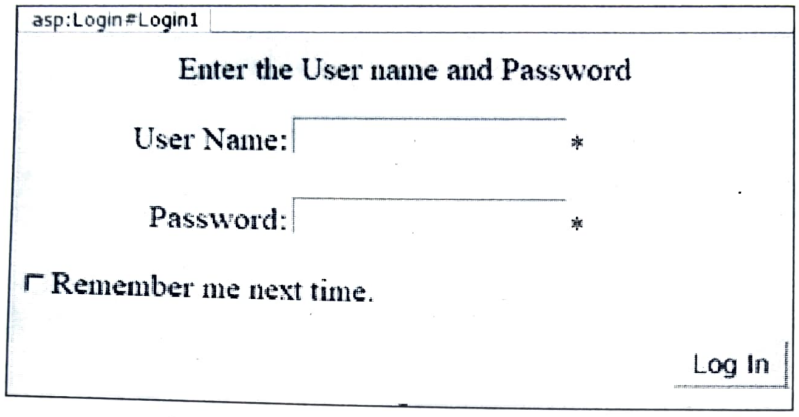


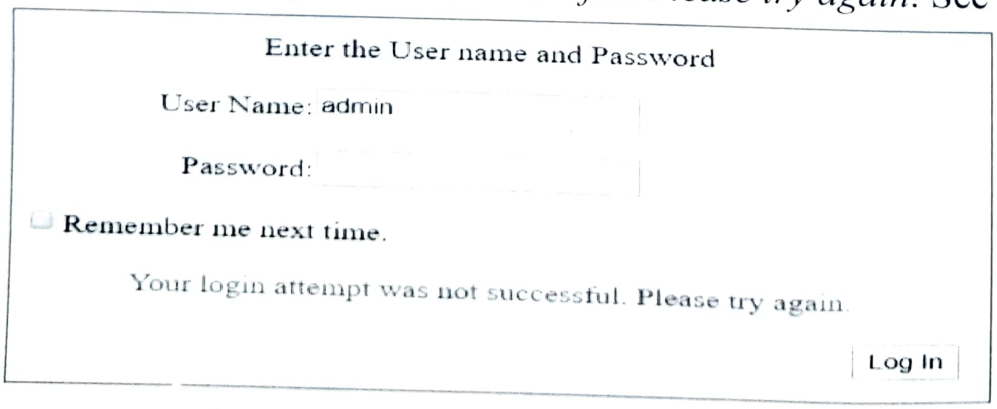
Figure 3.10: Login control Screen

You can just drop the **Login control** in your page, then in the **code behind**, catch the Login Control's ***Authenticate event***. In the Authenticate event, check the username and password that the user has entered. The username and password are properties in the login control. (LoginCtrl.UserName, LoginCtrl.Password). The code of Authenticate event is,

```
protected void Login1_Authenticate (object sender,
AuthenticateEventArgs e)
    {
    if (Login1.UserName == "admin" && Login1.Password  ==
"secret")
        {
            e.Authenticated = true;
        }
    }
```

Here, if the username and password both are correct, then we set the event *args* Authenticated property to True.

If username or password are incorrect then login control gives the embedded error message as *your login attempt was not successful. Please try again*. See like blow



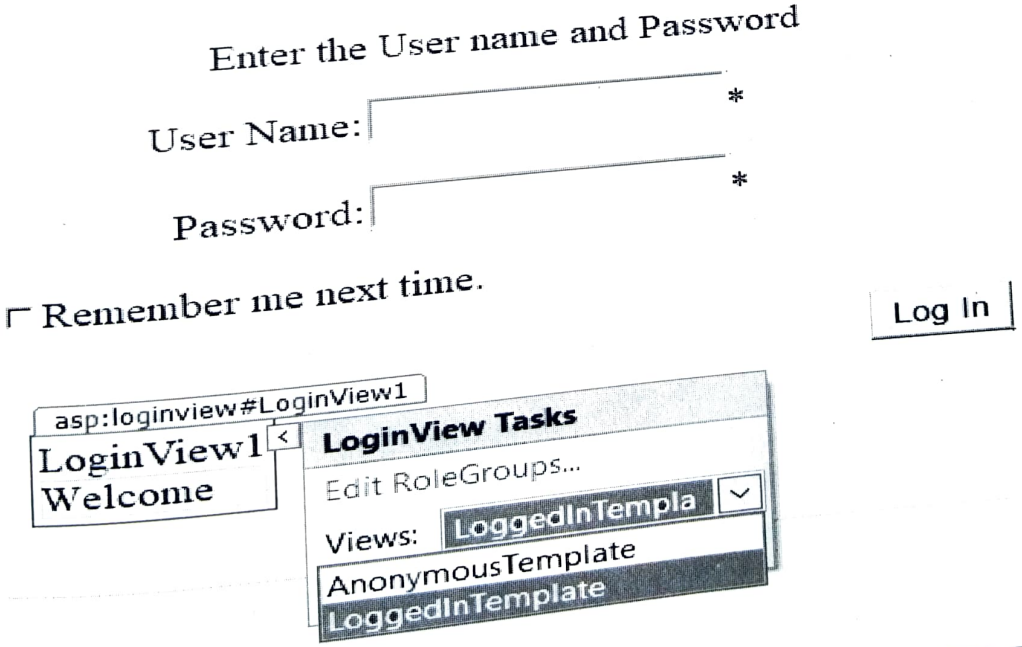
And web form code would like this, If authentication will true, then **DestinationPageUrl** property navigate control to Default2.aspx page

```
<asp:Login ID="Login1" runat="server" Height="226px"
Width="466px"
    onauthenticate="Login1_Authenticate"
DestinationPageUrl="Default2.aspx"
    TitleText="Enter the User name and Password">
</asp:Login>
```

Property	Description
CreateUserIconUrl	It retrieves the location of the image to display the link to the user.
DisplayRememberMe	It specifies or retrieves the URL for the new user page
DisplayRememberMe	It specifies the value stating whether to display the RememberMe checkbox.
FailureText	It displays the text when the login attempt fails
HelpPageText	It specifies the text of link to the login help page
Password	It retrieves the password entered by the user

3.4.2 LoginView and LoginName control:

The LoginView control is a web server control used for displaying the two different views of a web page. It helps to alter the page view for different logged in users. The current users status information is stored in the control. The control displays logged user name **using** the **LoginName control**



First take the login view control after completion of the login. Then change the views as LoggedinTemlate and write text "welcome" and take **LoginName Control** near *LoggedInTemplate* string 'welcome' like

```
<asp:LoginView ID="LoginView1" runat="server">
      <LoggedInTemplate>
          welcome
      </LoggedInTemplate>
```

LoginView1 asp:loginname#LoginName1
[UserName]

After inserting the loginName control near string 'welcome', and run the web form then displayed the logged user name, before or after the inserting the user name and password in the login control. So *LoginNmame* control is used with *loginView* control for display the name of current logged user. Like as, welcome [UserName]

3.4.3 PasswordRecovery:

Password Recovery control is used to recover or reset the password for the user. The password is sent through an email as a message at the registration time. The Membership service is used for creating and resetting the password.

The **first way** to recover the password using the ASP.NET Configuration. Using this, first we have to create user wizard, and then retrieve the password also from ASP.NET Configuration. You can open the ASP.NET Configuration wizard from the solution explorer menu as seen below image. And second way, we can create user using database connectivity and also recover the password from the database.

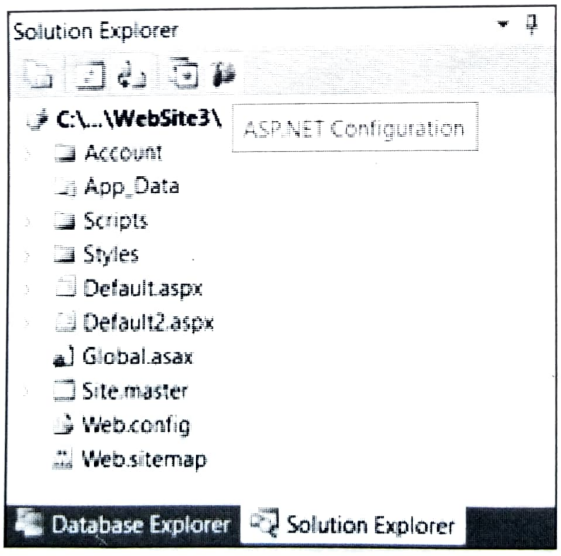


Figure 3.11: Solution Explorer

The control contains the following three views.

1. **Question:** It refers the view where the user can enter the answer to the security question.
2. **Username**: It refers to the view where the user can enter the username for the password to be recovered.
3. **Success:** It represents the view where the message is displayed to the user.

The basic properties and methods of the Password Recovery control are,

Property	Description
FailureTextStyle	It accesses the reference to the collection of properties defining the error text look
HelpPageIconUrl	It image to be displayed for the link to the password is retrieved
Answer	The answer provided by the user to confirm the password recovery through the valid user

Methods	Description
OnSendingMail	It raises the SendingMail event when the user is verified and the password is sent to the user.
OnSendMailError	It raises an error when the mail message is not sent to the user.
OnUserLookupErrror	It raises the UserLookupError when the username does not match with the one stored in the database
OnVerifyingUser	It raises the event once the username is submitted, and the membership provider verification is pending.

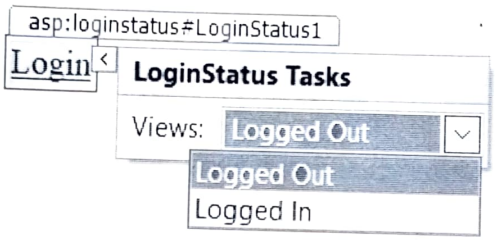
3.4.4 LoginStatus:

The Login Status control displays a login link for users who are not authenticated and a logout link for users who are authenticated. The login link takes the user to a login page. The logout link resets the current user's identity to be an anonymous user. It specifies that a particular user has logged into the web site. The login status is displayed as a text with hyperlink and provides the navigation to the login page. For example,

```
<asp:LoginStatus ID="LoginStatus1" runat="server"
LoginText="Sign In"
LogoutText="Sign Out"
LogoutPageUrl="~/Default.aspx"
LogoutAction="Redirect"
onloggedout="LoginStatus1_LoggedOut" />
```

To do this, Login Status control uses the authentication section of the web.config file. This control provides the following two views as,

1. **LoggedIn:** It will be displayed, when the user is not Logged In.
2. **Logout:** It will be displayed when the user is Logged In.



The basic properties and methods of the login status control are,

Property	Description
LoginTex	: It access the text added for the login link
LoginImageUrl	It accesses or specifies the URL of the image used for the login link
LogoutAction	It retrieves the value for determining the action when the user logs out of the web site.
LogoutText	It retrieves the text used for logout the link

Methods	Description
LogginOut	It is initiated when the user sends the logout request to the server
LoggedOut	It is initiated by the LoginStatus class when the user logout process is completed

3.4.5 CreateUserWizard:

This control used to create a new user in the membership data store. The CreateUserWizard control is provides by the CreateUserWizard class and can be customized by using template and style properties .Using this control any user can easily create an account and login to the web page. We can create user wizard using

ASP.NET configuration, and using database also. Here we have to create wizard using 'Customize Create User Step'

You can drag and drop CreateUserWizared control on the web page as shown below

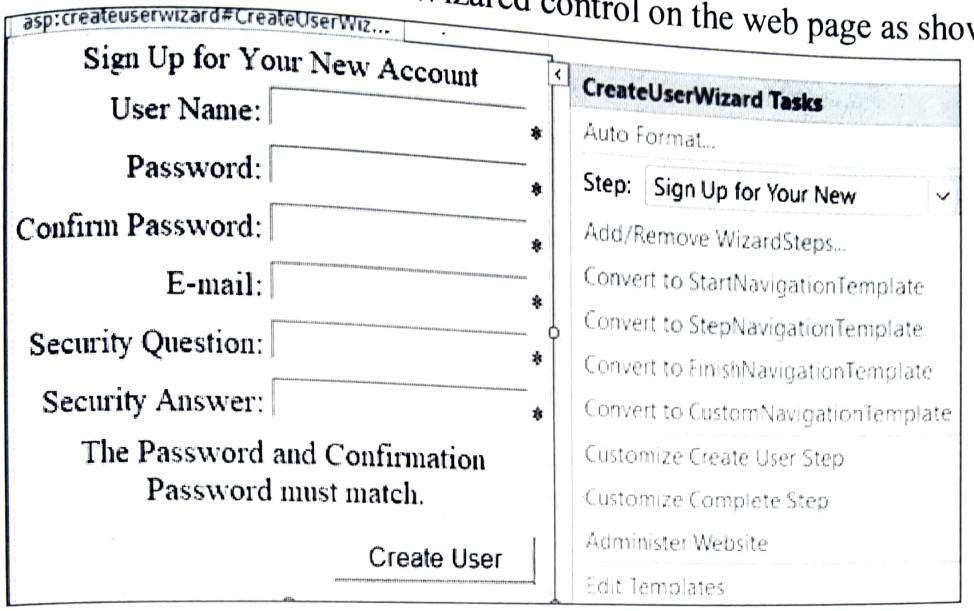


Figure 3.12: Create User Wizard Screen

We will create new account with '*Customize Create User Steps*' and create database file for storage user record. The system will check validations of required field validation and password not matched with confirm password. And store the user record into the database.

Basic properties of the CreateUserWizard:

properties	Description
LoginCreatedUser	It accesses or specifies the value indicating the new user login once the account is created.
CompleteStep	It shows the final step of the process for creating the user account.
Answer	It retrieves or specifies the answer to the password recovery confirmation question.

Email	It retrieves the email address of the user
ContinueButtonText	It accesses or specifies the collection of properties defining the look of the control

3.4.6 ChangePassword:

Using this control, user can easily change your existing/old password on the ASP.NET Website. This control prompts uses to provide the current/old password first and then set the new password, if the old password is not correct then new password can't be set. This is also helps to send the email to the respective users about the new password, if the login, registration are set by ASP.NET Configuration using security question as below.

Figure 3.13: Change Password Screen

We asked same question, when we defined in the user registration, the system check in the configuration file and give appropriate answer to the user. And if we used database to store the user registered record, then system check the password in the database.

System give the password by sending e-mail which is given in new user registration, or name of the password from the database.

Basic properties of the ChangePassword:

properties	Description
CurrentPassword	It retrieves the current password of a user
NewPassword	It retrieves the new password entered by the user
UserName	It shows the username for which the password is to be modified.
DisplayUserName	It retrieves the value indicating whether the ChangePassword control should be display the control and label
CancelDestinationPageUrl	It accesses or retrieves the URL of the page that the user is shown once it clicks the Cancel button

3.5 Creating and Managing Roles:

The most important features in building strong web applications is ensuring that users have access to just the areas or web pages they need. This is usually achieved by creating different user roles and assigning users to the roles created. These roles define what a user can and cannot do in the web application. Here we are creating custom user roles in ASP.NET core and authorizing users based on these roles. ASP.NET Core Identity is the membership or identity management system shipped with the ASP.NET Core web development stack, for building web applications. It includes membership, login, and management of user data. ASP.NET Core Identity allows you to add authentication features and customize data about the logged in user in your application.

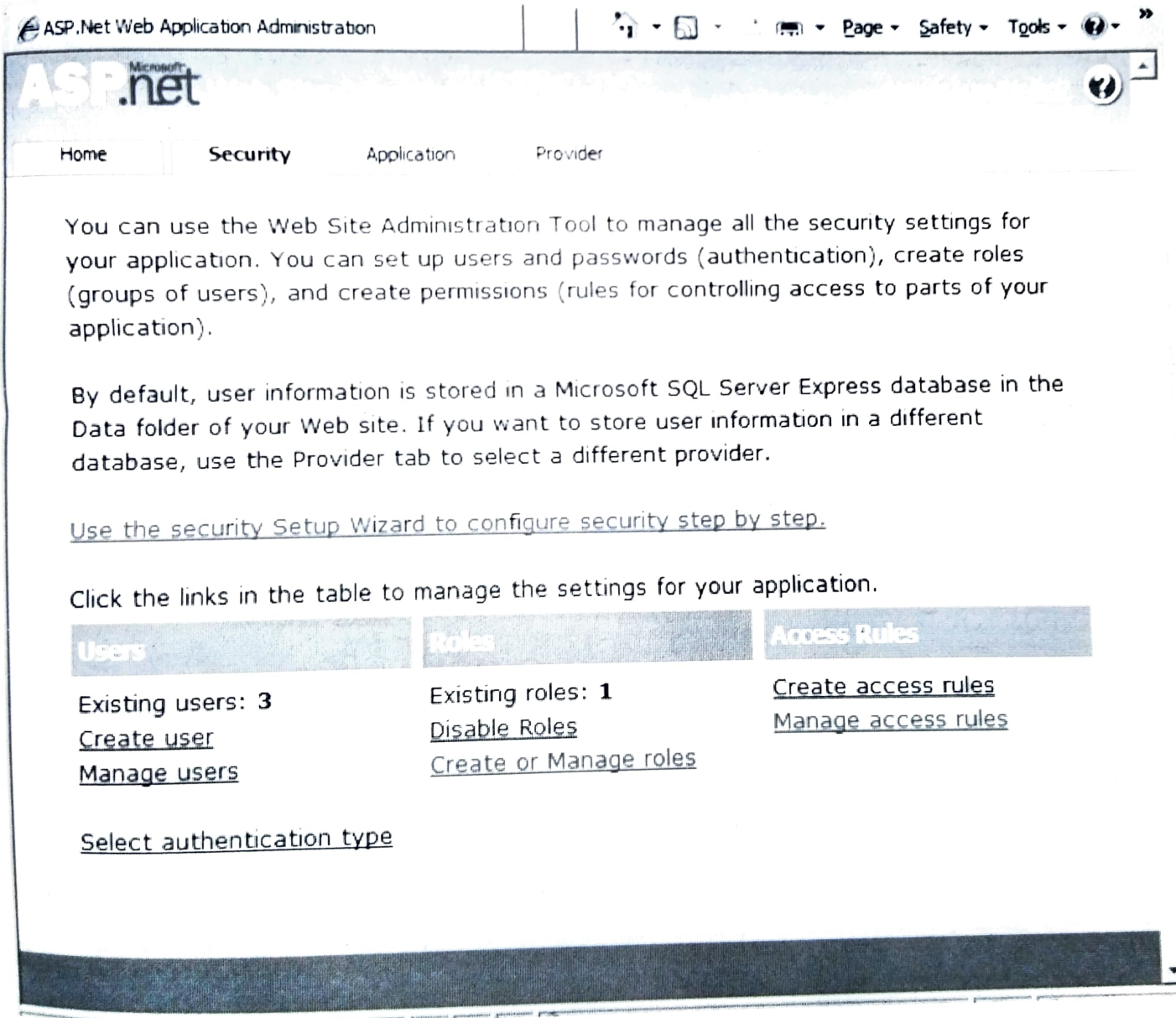


Figure 3.14: Web Site Administation Tool

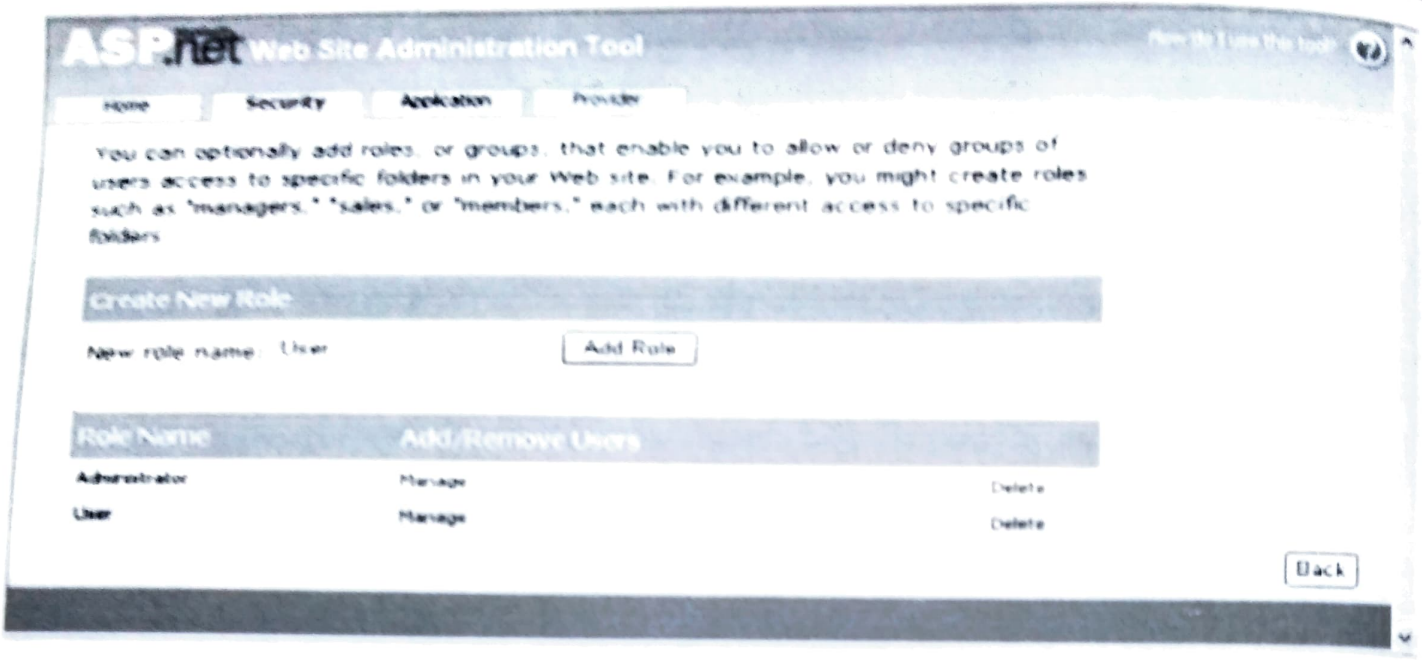


Figure 3.15: Create New Role

3.6 Creating and Managing Access Rules:

Click on "Create Access Rules" to add access rules for different users.

Before navigating to create access rules just move to your development window and add a folder and name it as "user1".

Add another folder in the same manner and name it as "user2".

Note: This is only to differentiate different users as per the user role basis; it's not a compulsion that this has to be done only in this way. You can have your own way of doing that. click on the "create access rules" link as below link

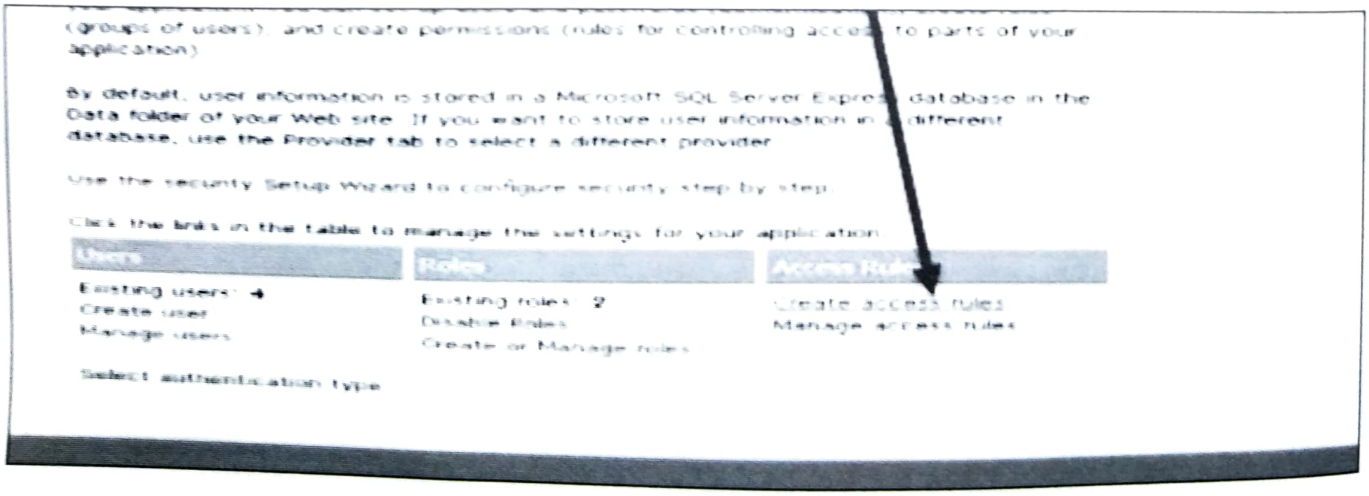


Figure 3.16: Create Access Rules

You will be redirected to a page similar to the one shown below

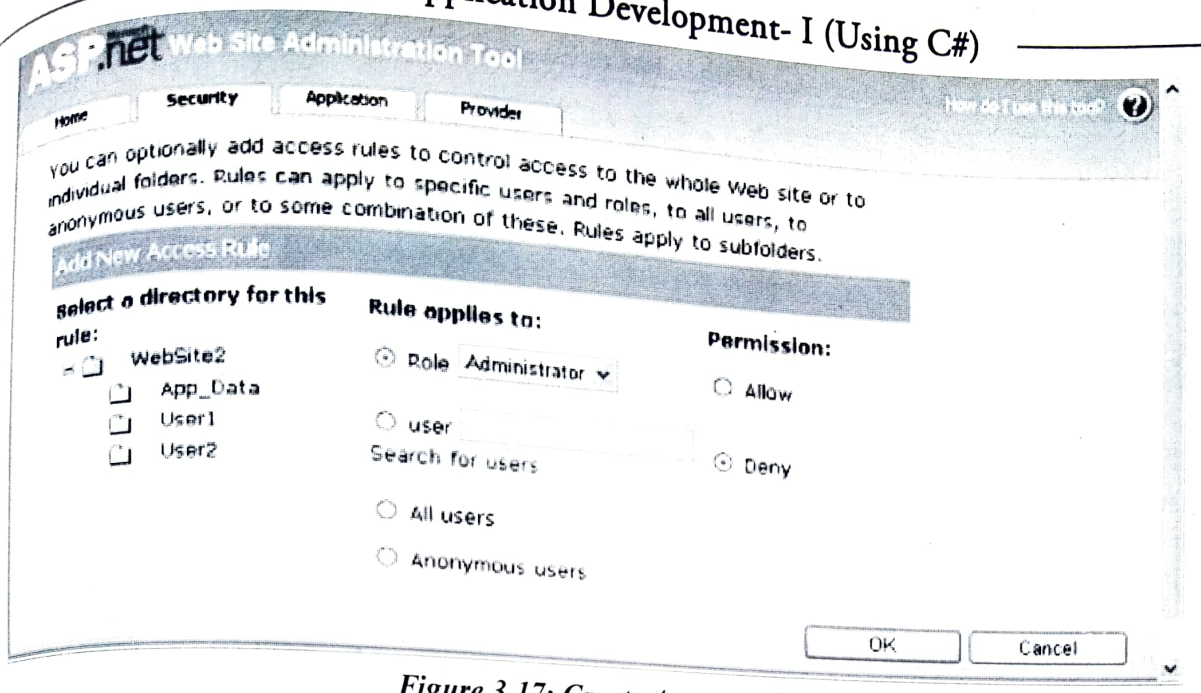


Figure 3.17: Create Access Rules

In this page as you can observe we have three different options:

1. Select a different directory for this rule
2. Rule applies to
3. Permissions.

"Select a different directory for this rule"-is the one where we you can select or deselect the directories you assign for each user or to the administrator.

For example, if you want to assign the directory "user1" only to user1 and not user2, then you can select that directory and select user1 from "Rule applies to" and allow option from "Permissions".

"Rule applies to"---is the one where you will be selecting the user /users /admin for that specific directory to access.

"Permissions"---is the one where you will be giving the actual permissions i.e. to accept or deny the user for that particular directory.

Now our job is to assign that directory to user1 and Administrator. So select "Administrator" from Rule applies to and Select "Allow" from Permissions. Now the screen should look something similar to the one above. Press ok and that role will be assigned with the specified one.

As soon as you click "OK" button, you will be automatically redirected to the "Security Home Page". Now click on "Create access rules" again to assign the same to User1 also.

Now select "user" from the role dropdown box and type "user1" in the textbox provided beside user. Again select "allow" from permissions as you have did earlier.

The screen will be similar to the one below:

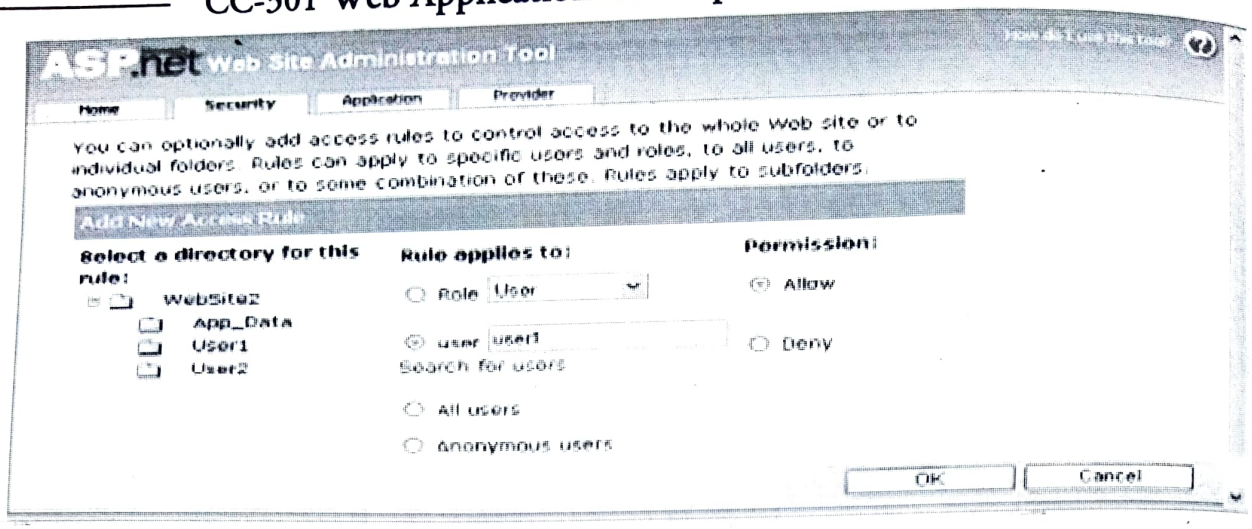


Figure 3.18: Create Access Rules

So now that you have assigned that directory to user1 and administrator. Now we need to restrict user2 to access that directory.

To achieve that you repeat the same process as you have done for "user1", but at Permissions select "Deny" instead of "Allow". And so on

3.7 Creating and Managing Profile:

1. Click the Security tab.
2. Under Access Rules, click Manage access rules.
3. Click the Add new access rule link.
4. Under Select a directory for this rule, select the Members Pages folder.
5. Under Rule applies to, select All users.
6. Under Permission, select Deny and then click OK. This rule ensures that everyone is denied access to the Members Pages folder.
7. Click the Add new access rule link.
8. Select the Members Pages folder again.
9. Under Rule applies to, select Role and select Member from the drop-down list.
10. Under Permission, select Allow, click OK, and then click done. The rule you just created allows only users who are in the Member role to access the Members Pages folder.
11. Return to the Security tab and click Manage access rules.
12. If the Members Pages folder is not selected, select it.
13. Under Users and Roles, click Member and click Move Up to move Member to the top of the list.
14. Click done to return to the Security tab.

☺ ☺ ☺ ☺ ☺

Exercises

Answer the following:

1. What is a master page, explain the concern task of master page
2. Explain the importance of the Navigation Controls
3. check and validate expected sequences of characters like e-mail address, telephone number, social security number, using Regular Expression validator
4. Explain about Custom validator with appropriate example
5. How to create the roles in ASP.NET web application administration

True/False:

1. Master page was saved with the name .master extension. T
2. Master page and Home page both are same. F
3. TreeView control is a collection of object, and displaying hierarchal data. T
4. The Menu control is similar to TreeView control as the base of navigation T
5. Required Field Validator is used for null value (-1) checking in the control F
6. This range Validator has numbers, alphabetic characters and dates. T
7. ControlToValidate property not exist in Regular Expression validator F
8. ControlToCompare property used in Compare Validator control T
9. Mostly custom validator control code written in Code Behind in the web page T
10. The LoginView control is used for view a logged user T

Answers (True/False):

1. True	2. False	3.True	4. True	5. True
6. True	7. False	8. True	9. True	10. True

MCQs (Multiple Choice Questions):

1. In master page, contents are displayed in content page using _____ control
   - **a.** Label
   - **b.** content page
   - **c.** ContentPlaceHolder
   - **d.** Rich Textbox
2. Manu control navigation is similar with below control

a. TreeView Control
b. Sitemap Control
c. Panel Control
d. Navigate Control

3. Minimum and maximum values are considered in _____
a. Required Field Validator
b. Range validator
c. Regular Expression validator
d. Compare Validator

4. The below control display the group of error _____
a. Required Field Validator
b. Range validator
c. Regular Expression validator
d. Validation summary

5. The control used for display a login link (Logged in/Logged Out) _____
a. Login Status
b. Login Name
c. Login View
d. None of these

Answer of MCQs:

1. C 2. A 3. B 4. D 5. A

☺ ☺ ☺ ☺ ☺

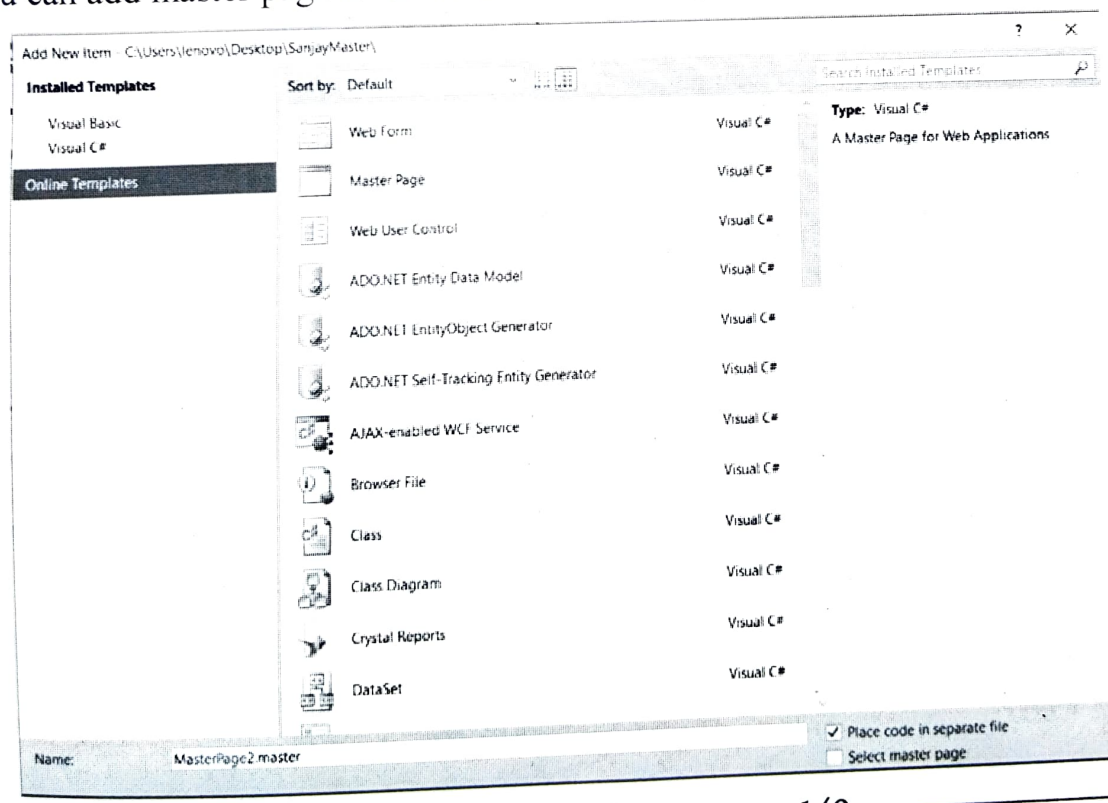
CC-304 Unit – 3 Web Application Development-I Practicals

Program : 1

Design a site for "Gujarat University". Design master page having header, sidebar, footer and content section. Put copywrite warning in the footer and university name in the header. In the sidebar put Treeview control, filled from sitemap. Create following hierarchy in the sitemap and provide links to various pages.

In Master page

- The master page is a normal HTML page designed as a template for other pages.
- The master page contains a placeholder tag <asp:ContentPlaceHolder> for individual content.
- This master page was saved with the name "master1.master".
- The content page is one of the individual content pages of the web
- The @ Page directive defines it as a standard content page.
- This content page was saved with the name "Default.aspx".
- When the user requests this page, ASP.NET merges the content page with the master page
- You can add master page from "Add New Item" option as below screen



Master Page Example:

```
<%@ Master %>
<html>
<body>
<h1>Standard Header From Masterpage</h1>
<asp:ContentPlaceHolder id="head" runat="server">
</asp:ContentPlaceHolder>
</body>
</html>
```

Content Page Example:

```
<%@ Page MasterPageFile="master1.master" %>
<asp:Content ContentPlaceHolderId="head" runat="server">
  <h2>Individual Content</h2>
  <p>Paragraph 1</p>
  <p>Paragraph 2</p>
</asp:Content>
```

Demo example of the Master page :

HTML code of MasterPage.master is,

```
<%@ Master Language="C#" AutoEventWireup="true"
CodeFile="MasterPage.master.cs" Inherits="MasterPage" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>
        <asp:ContentPlaceHolder ID="Title" runat="server">
        </asp:ContentPlaceHolder>
    </title>
     <link href="style/StyleSheet.css" rel="stylesheet"
type="text/css" />
       <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
 </head>
<body>
    <form id="form1" runat="server">
```



```
        <div class="wrapper">
          <div class="footer">
              <center> <h2> GUJARAT UNIVERSITY,
AHMEDABAD</h2></center>
            </div>
            <div class="menu">
              <ul>
               <li> <a href="Default.aspx">Home</a></li>
               <li> <a
href="Default2.aspx">Category</a></li>
               <li> <a href="Default.aspx">Contact
Us</a></li>
               <li> <a href="Default.aspx">About
Us</a></li>
               </ul>
             </div>
             <div class="clear"></div>
             <div class="content">
                 <asp:ContentPlaceHolder id="ContentBody"
runat="server">
                 </asp:ContentPlaceHolder>
             </div>
             <div class="clear"></div>
             <div class="footer">
                <h2> CopyRight@sanjaySoni.com</h2>
             </div>
          </div>
      </form>
  </body>
  </html>
```

Content Page code in HTML:

```
<%@ Page Language="C#"
   MasterPageFile="~/MasterPage.master"
   AutoEventWireup="true" CodeFile="Default.aspx.cs"
   Inherits="_Default" Title="Untitled Page" %>
```

```
<asp:Content ID="Content1" ContentPlaceHolderID="Title"
   Runat="Server">
Home
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="head"
   Runat="Server">
</asp:Content>
<asp:Content ID="Content3"
   ContentPlaceHolderID="ContentBody" Runat="Server">
     <h2> This Is Home Page</h2><br />
  </asp:Content>
```

CSS Code (Take CSS file from add new items):

```
body,div,p,ui,li
{
   padding: 0;
   margin: 0;
}
body
{
   background-color: rgb(237,237,237);
   font-family : "Arial",Sans-Serif;
   font-size: 12px;
}
.wrapper
{
   width: 1000px;
   margin: auto;
}
.content
{
   width:100%;
   background-color: rgb(254,254,254);
   border:1px solid rgb(224,224,224);
   float:left;
   margin-top:1;
```

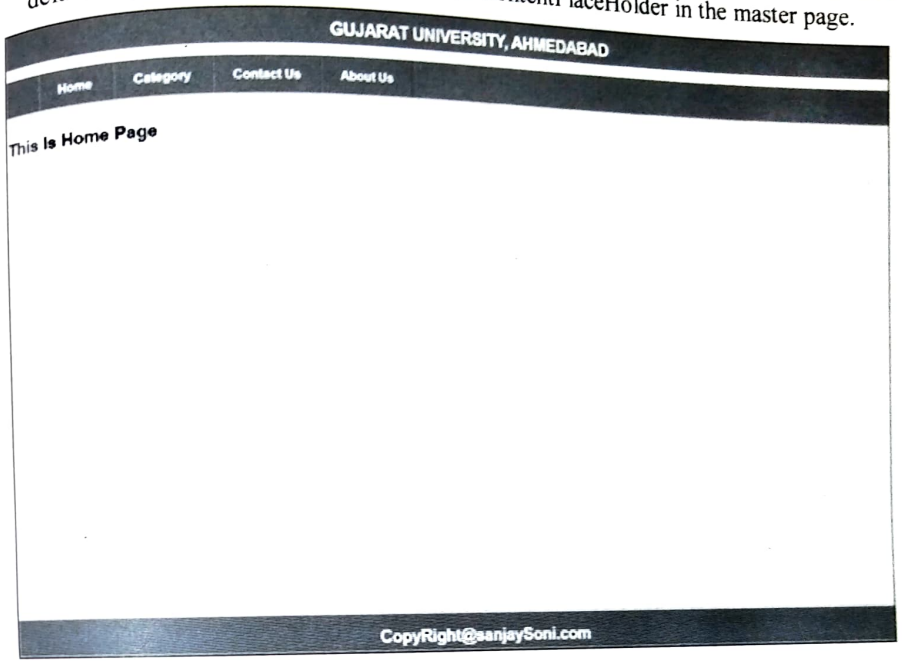


```
   margin-bottom:1px;
   min-height:520px;
}
.menu
{
   background-color: rgb(10,110,178);
   width:100%;
   margin:0px 0px 10px;
   padding: 0px;
   height:40px;
   color:rgb(243,243,243);
}
.navigation_first_item
{
   border-left: 0px;
}
.navitem_s
{
   float: left;
   border-right: 1px solid rgb(10,85,125);
   border-left: 1px solid rgb(67,153,200);
   height:40px;
   background-color: rgb(14,79,114);
}
.menu ul {}
.menu ul li
{
   float:left;
   display:block;
   list-style:none;
   border-right: 1px solid rgb(10,85,125);
   border-left: 1px solid rgb(67,153,200);
}
.menu ul li.navigation_first_item : hover
```

```
{

}
.menu ul li a
{
   font-size: 13px;
   font-weight:bold;
   line-height:40px;
   padding: 8px 20px;
   color: rgb(255,255,255);
   text-decoration:none;
}
.menu ul li:hover
{
   background-color: rgb(14,79,114);
   border-right:1px solid rgb(14,89,130);
}
.clear
{
   clear:both;
}
.footer
{
   height:40px;
   background-color: rgb(10,110,178);
   color: rgb(255,255,255);
}
.footer h2
{
   padding: 15px;
   text-align:center;
}
```

Initial screen of the Master page:

After run the web application, initial screen will display like this, and content of the default.aspx page will be displayed on the contentPlaceHolder in the master page.

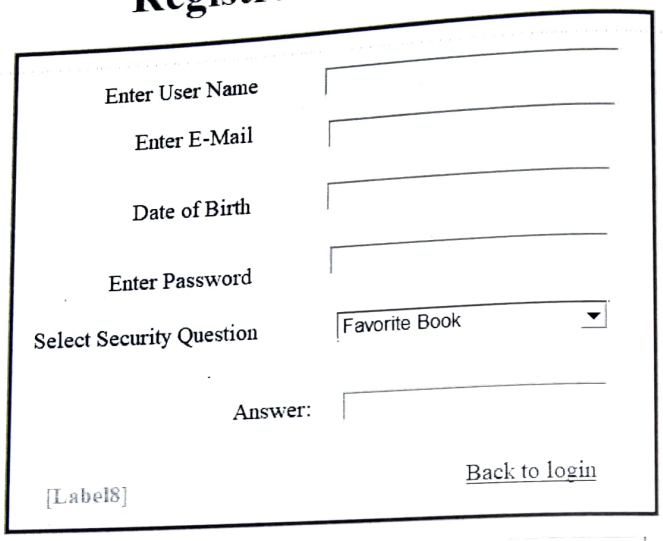


Program : 2

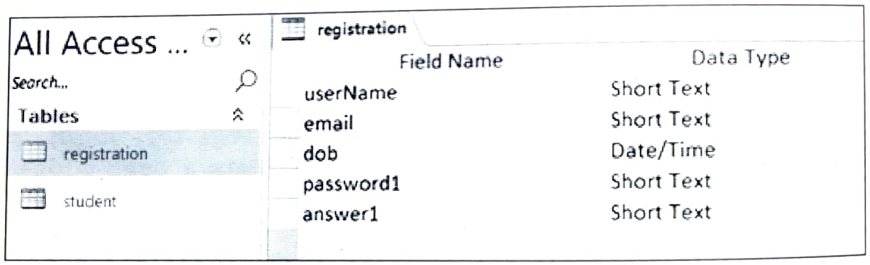
Design a web site which allows user to register, login, changer password and forgot password features. Create a page which can be opened only by authenticated users, also create a page which can be opened on by the user who belongs to 'Admin' role. On the home page display Welcome message base on the type of user. For example, for anonymous user show "Welcome Visitor", for User show "Welcome <UserName>" and for any user belongs to admin role, show "Welcome Administrator".

Design of the Registration page:

Registration System



Database Structure:



Code of the Registration page:

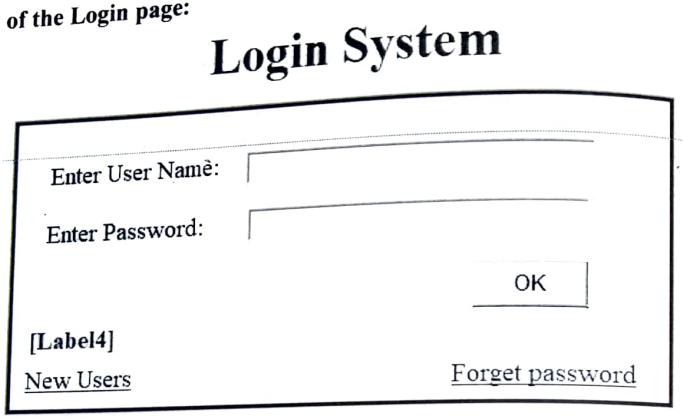
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.OleDb;
```



```
public partial class unit3_2 : System.Web.UI.Page
{
    OleDbConnection con = new
OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=E:\\asp1.accdb");
    protected void Page_Load(object sender, EventArgs e)
    {
        con.Open();
    }
    protected void Button1_Click(object sender, EventArgs
e)
    {
        try
        {

            OleDbCommand cmd = new OleDbCommand("insert
into registration values('" + TextBox1.Text + "','" +
TextBox2.Text + "',#" + TextBox3.Text + "#, '" +
TextBox4.Text + "','" + TextBox5.Text + "')", con);
            cmd.CommandType =
System.Data.CommandType.Text;
            cmd.ExecuteNonQuery();
            Label8.Text = "Registration successfully
completed...";
            cmd.Dispose();
        }

        catch (Exception ex)
        {
            Label1.Text = ex.Message;
        }
    }
    protected void LinkButton2_Click(object sender,
EventArgs e)
    {
        Response.Redirect("login.aspx");
```

```
    }
}
```

Design of the Login page:



Code of the login page:

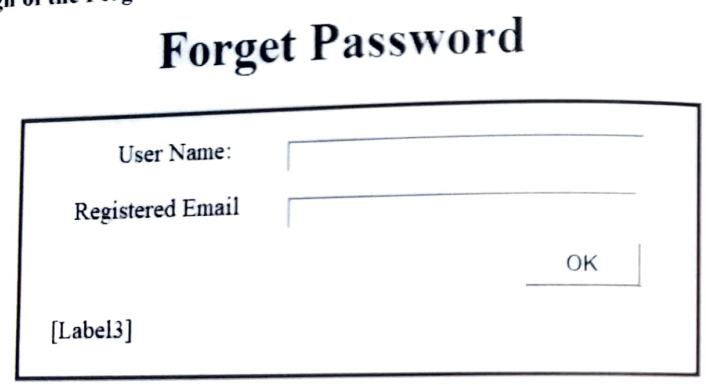
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.OleDb;

public partial class logic : System.Web.UI.Page
{
    OleDbConnection con = new
  OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
  Source=E:\\asp1.accdb");
    protected void Page_Load(object sender, EventArgs e)
    {
        con.Open();
    }
```



```
  protected void LinkButton1_Click(object sender,
EventArgs e)
  {
      Response.Redirect("unit3_2.aspx");
  }
protected void Button1_Click(object sender, EventArgs e)
  {
      try
      {
          OleDbCommand cmd = new OleDbCommand("select *
from registration where userName ='" + TextBox1.Text +
"' and password1 ='" + TextBox2.Text.Trim() + "'", con);
          OleDbDataReader red;
          cmd.CommandType =
System.Data.CommandType.Text;
          red = cmd.ExecuteReader();
          if (red.Read() == true)
          {
              Label4.Text = "Welcome " +
red["userName"].ToString();
              //Response.Redirect("Default2.aspx");
              cmd.Dispose();
          }
          else
          {
              Label4.Text = "User Name or Password
Incorrect";
              cmd.Dispose();
          }
      }

      catch (Exception ex)
      {
          Label4.Text = ex.Message;
      }
```

```
    }
    protected void LinkButton2_Click(object sender,
EventArgs e)
    {
        Response.Redirect("forgetPassword.aspx");
    }
}
```

Design of the Forget Password page:



Code of the Forget Password page:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.OleDb;
using System.Data;
using System.Net.Mail;
using System.Net;

public partial class forgetPassword : System.Web.UI.Page
{
```

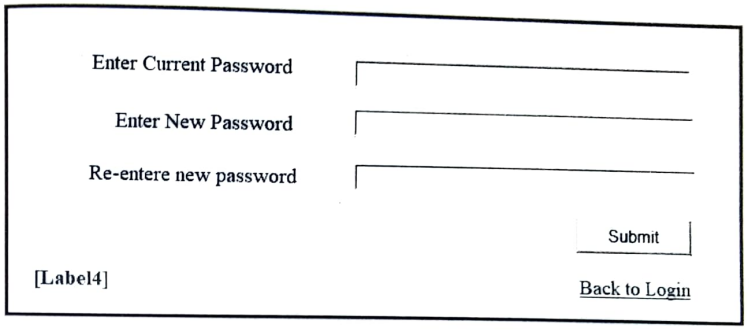


```
    OleDbConnection con = new
OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=E:\\asp1.accdb");
    protected void Page_Load(object sender, EventArgs e)
    {
        con.Open();
    }
protected void Button1_Click(object sender, EventArgs e)
    {
        try
        {
            string password1;
            OleDbCommand cmd = new OleDbCommand("select *
from registration where userName ='" + TextBox1.Text +
"' and email ='" + TextBox2.Text.Trim() + "'", con);
            OleDbDataReader red;
            cmd.CommandType =
System.Data.CommandType.Text;
            red = cmd.ExecuteReader();
            if (red.Read() == true)
            {
                password1 = red["password1"].ToString();
                passwordSend(password1, TextBox2.Text);
            }
            else
            {
                Label3.Text = "User Name or Password
Incorrect";
                cmd.Dispose();
            }
        }

        catch (Exception ex)
        {
            Label3.Text = ex.Message;
```

```
        }
    }
    private void passwordSend(string password1, string
email)
    {
        try
        {
        SmtpClient smtp = new SmtpClient();
        smtp.Host = "smtp.gmail.com";
        smtp.Port = 587;  // or 465
        smtp.Credentials = new
System.Net.NetworkCredential("sanjaysoni@gmail.com",
"your password");
        smtp.EnableSsl = true;
        MailMessage msg = new MailMessage();
        msg.Subject = "Forger password";
        msg.Body = "Dear " + TextBox1.Text + ", your
password is " + password1 + "\n\n\n thanks & regards";
        string toaddress = TextBox2.Text;
        msg.To.Add(toaddress);
        string fromaddress = "NGCCA
<sanjaysoni@gmail.com>";
        msg.From = new MailAddress(fromaddress);

        smtp.Send(msg);
        Label3.Text = "Password sent to your registered
mail";
        }
        catch (Exception ex)
        {
            Label3.Text = ex.Message;
        }
    }
}
```

Design of the Change password

Change Password



Code of the Change password page:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.OleDb;

public partial class changePassword : System.Web.UI.Page
{
    OleDbConnection con = new
OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=E:\\asp1.accdb");

    protected void Page_Load(object sender, EventArgs e)
    {
        con.Open();
    }
    protected void LinkButton1_Click(object sender,
EventArgs e)
    {
        Response.Redirect("login.aspx");
```

```
    }
protected void Button1_Click(object sender, EventArgs e)
    {
        try
        {
            OleDbCommand cmd = new OleDbCommand("select *
from registration where password1 ='" + TextBox1.Text +
"'", con);
            OleDbDataReader red;
            cmd.CommandType =
System.Data.CommandType.Text;
            red = cmd.ExecuteReader();
            if (red.Read() == true)
            {
                change();
            }
            else
            {
                Label4.Text = "Password Incorrect";
                cmd.Dispose();
            }

        }

        catch (Exception ex)
        {
            Label4.Text = ex.Message;
        }
    }
public void change()
{
    try
    {
        if (TextBox2.Text == TextBox3.Text)
        {
```



```
            OleDbCommand cmd = new
OleDbCommand("update registration set upassword='" +
TextBox2.Text + "' where userName='" + TextBox1.Text +
"'", con);
            cmd.CommandType =
System.Data.CommandType.Text;
            cmd.ExecuteNonQuery();
            Label3.Text = "User password Updated";
            cmd.Dispose();
        }
        else
        {
            Label4.Text = "Password and Rr-password
not same";
        }
    }

    catch (Exception ex)
    {
        Label1.Text = ex.Message;
    }
  }
}
```

Program : 3 and 4

3. Design a web form and perform the following validations:

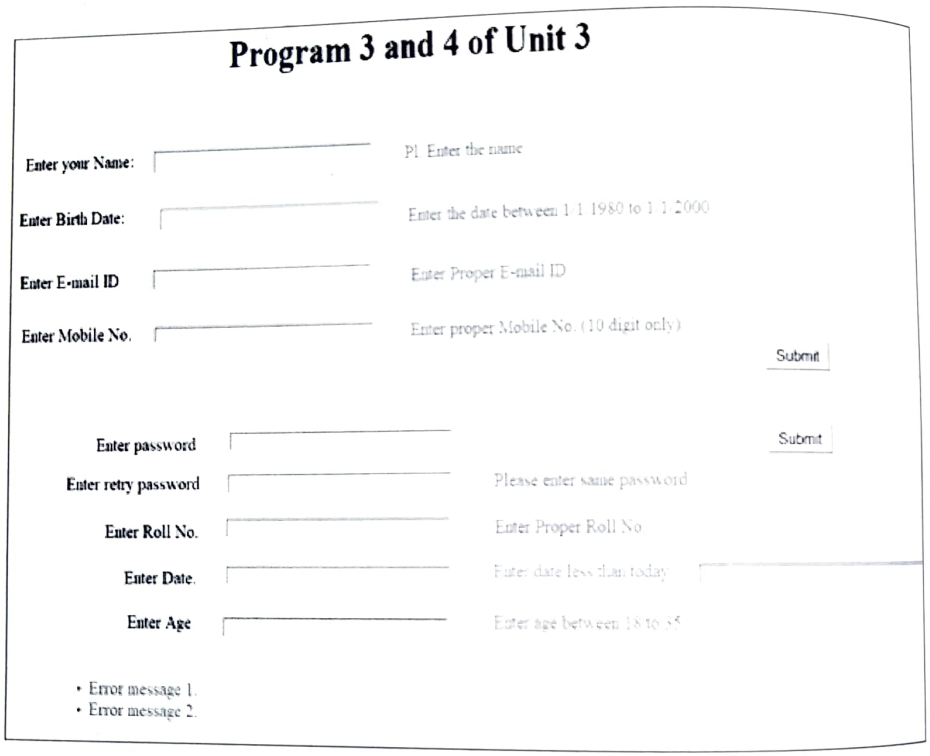
- Null value is not allowed.
- The birth date should appear between "1/1/1980" and "1/1/2000".
- Email should be valid id.
- Contact number exactly of 10 digits

4. Create one registration page and perform the following validations.

- To validate email_id
- To compare new password and retype password
- The roll no should contain first 3 characters BCA. Example BCA01, BCA02

- Restrict the user to enter only date in textbox and it must not accept date greater than current date.
- Age should be between 18 to 35
- Name field is compulsory
- Mobile number must be of 10 digits only
- Give demo of validation summary

Design of the Program 3 and 4



Set the below properties for each validation control first (This for Null value is not allowed)

BorderStyle	NotSet
BorderWidth	
ClientIDMode	Inherit
ControlToValidate	**TextBox1**
CssClass	
Display	Static
EnableClientScript	True
Enabled	True
EnableTheming	True
EnableViewState	True
ErrorMessage	**Pl. Enter the name**
Font	
ForeColor	**#FF3300**
Height	
InitialValue	

Set the below properties for the date between 1/1/1980 to 1/1/2000

EnableViewState	True
ErrorMessage	**Enter the date betwee**
Font	
ForeColor	**#FF0066**
Height	
MaximumValue	**2000/1/1**
MinimumValue	**1980/1/1**
SetFocusOnError	False
SkinID	
TabIndex	0
Text	
ToolTip	
Type	**Date**

Set the below properties for the E-mail ID:

EnableViewState	True
ErrorMessage	**Enter Proper E-mail ID**
Font	
ForeColor	**#FF0066**
Height	
SetFocusOnError	False
SkinID	
TabIndex	0
Text	
ToolTip	
ValidationExpression	**\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)***

Set the below properties for the Mobile Number 10 digit:

EnableTheming	True
EnableViewState	True
ErrorMessage	**Enter proper Mobile No. (10 digit only)**
Font	
ForeColor	**#FF0066**
Height	
SetFocusOnError	False
SkinID	
TabIndex	0
Text	
ToolTip	
ValidationExpression	**\d{10}**
ValidationGroup	

Set the below properties for entering same password:

ControlToCompare	**TextBox6**
ControlToValidate	**TextBox5**
CssClass	
CultureInvariantValues	False
Display	Static
EnableClientScript	True
Enabled	True
EnableTheming	True
EnableViewState	True
ErrorMessage	**Please enter same password**
Font	
ForeColor	**#FF0066**
Height	
Operator	Equal
SetFocusOnError	False
SkinID	
TabIndex	0
Text	
ToolTip	
Type	String
ValidationGroup	

Set the below properties for enter date less than today:

BorderStyle	NotSet
BorderWidth	
ClientIDMode	Inherit
ControlToCompare	**TextBox9**
ControlToValidate	**TextBox8**
CssClass	
CultureInvariantValues	False
Display	Static
EnableClientScript	True
Enabled	True
EnableTheming	True
EnableViewState	True
ErrorMessage	**Enter date less than today**
Font	
ForeColor	**#FF5050**
Height	
Operator	**LessThan**
SetFocusOnError	False
SkinID	

Set the below properties for enter age between 18 to 35:

ClientIDMode	Inherit
ControlToValidate	**TextBox10**
CssClass	
CultureInvariantValues	False
Display	Static
EnableClientScript	True
Enabled	True
EnableTheming	True
EnableViewState	True
ErrorMessage	**Enter age between 18 to 35**
Font	
ForeColor	**#FF5050**
Height	
MaximumValue	**35**
MinimumValue	**18**
SetFocusOnError	False

Not set any property for validation summary

Program : 6

Design a website having master page. Create sitemap file with suitable assumption and use it in TreeView and Menu control for navigation purpose. Also show demo of SiteMapPath Control.

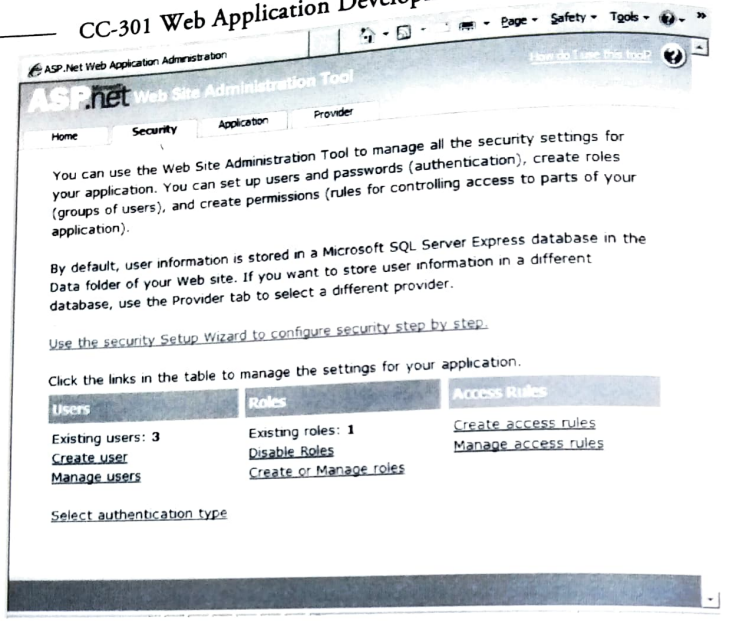
Answer:

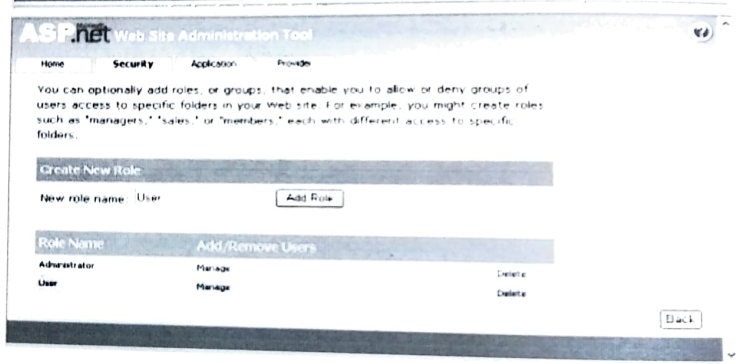
- **Same as Program 1 by adding Navigation control to master page side**

Note: Program 5, 7 and 8 are wizard program

Answer:

- **These programs are created using wizard controls as login, registration, change password, Login Name etc. by the below ASP.NET web application administration dialog box.**
- **Go to security option and set /manage the roles and authentication of the user.**
- **Please read the theory concept for more details for practical**
- **You can change/add/modify the wizard fields as requirements of program 5**





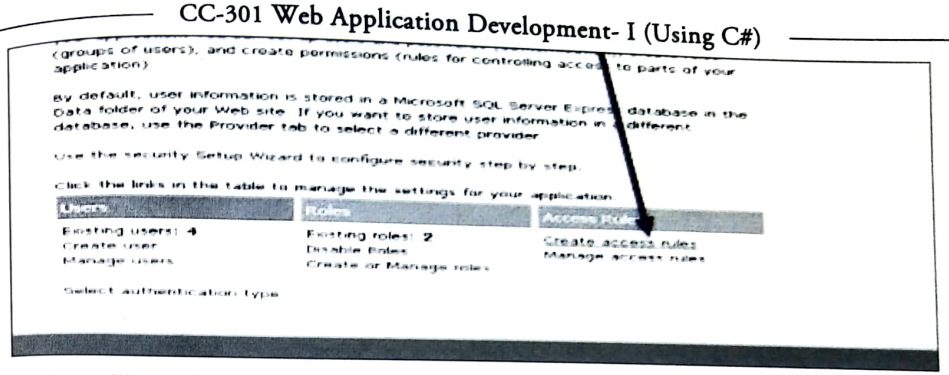
Creating and Managing Access Rules:

Click on "Create Access Rules" to add access rules for different users.

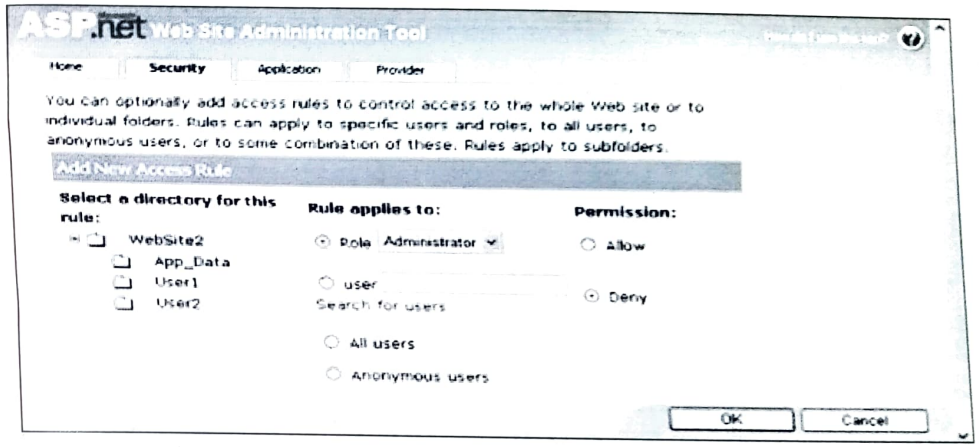
Before navigating to create access rules just move to your development window and add a folder and name it as "user1".

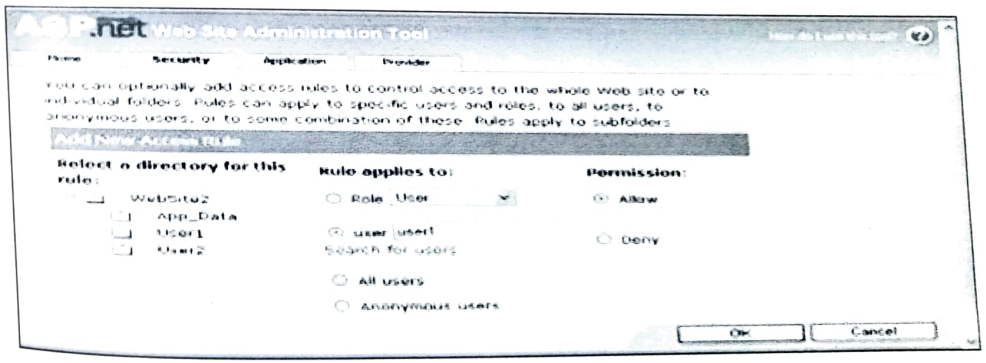
Add another folder in the same manner and name it as "user2".

***Note*:** This is only to differentiate different users as per the user role basis; it's not a compulsion that this has to be done only in this way. You can have your own way of doing that. click on the "create access rules" link as below link



You will be redirected to a page similar to the one shown below



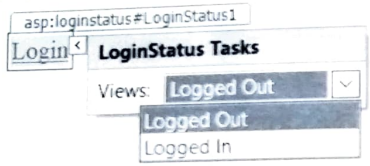


LoginStatus:

```
<asp:LoginStatus ID="LoginStatus1" runat="server"
LoginText="Sign In"
LogoutText="Sign Out"
LogoutPageUrl="~/Default.aspx"
LogoutAction="Redirect"
onloggedout="LoginStatus1_LoggedOut" />
```

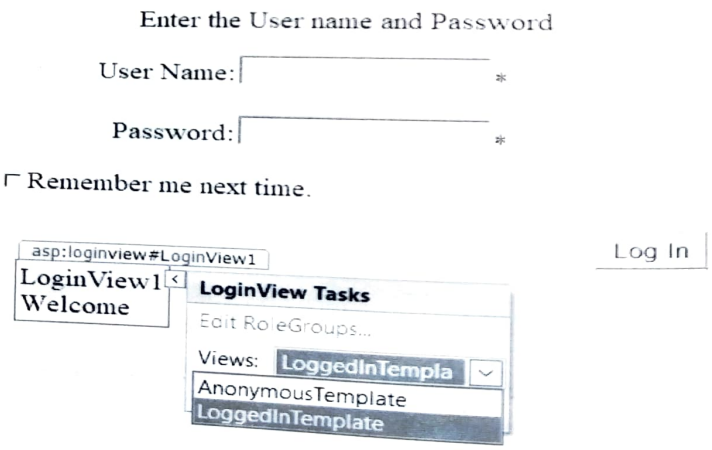
To do this, Login Status control uses the authentication section of the web.config file. This control provides the following two views as,

3. **LoggedIn:** It will be displayed, when the user is not Logged In.
4. **Logout:** It will be displayed when the user is Logged In.



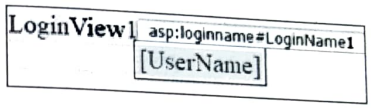
LoginView and LoginName control:

The LoginView control is a web server control used for displaying the two different views of a web page. It helps to alter the page view for different logged in users. The current user's status information is stored in the control. The control displays logged user name **using** the **LoginName control**



First take the login view control after completion of the login. Then change the views as LoggedinTemlate and write text "welcome" and take **LoginName Control** near *LoggedInTemplate* string 'welcome' like

```
<asp:LoginView ID="LoginView1" runat="server">
    <LoggedInTemplate>
        welcome
    </LoggedInTemplate>
```



After inserting the loginName control near string 'welcome', and run the web form then displayed the logged user name, before or after the inserting the user name and password in the login control. So *LoginNmame* control is used with *loginView* control for display the name of current logged user. Like as, welcome [UserName]

☺ ☺ ☺ ☺ ☺