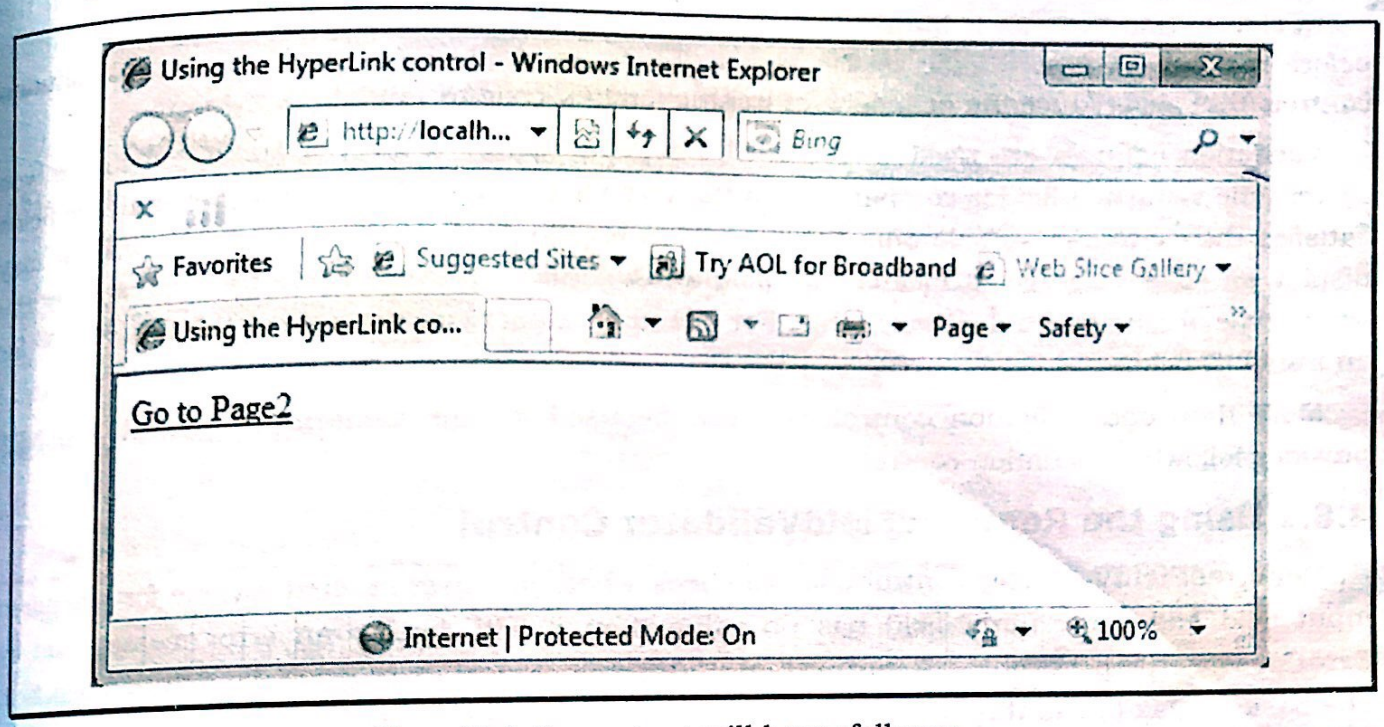
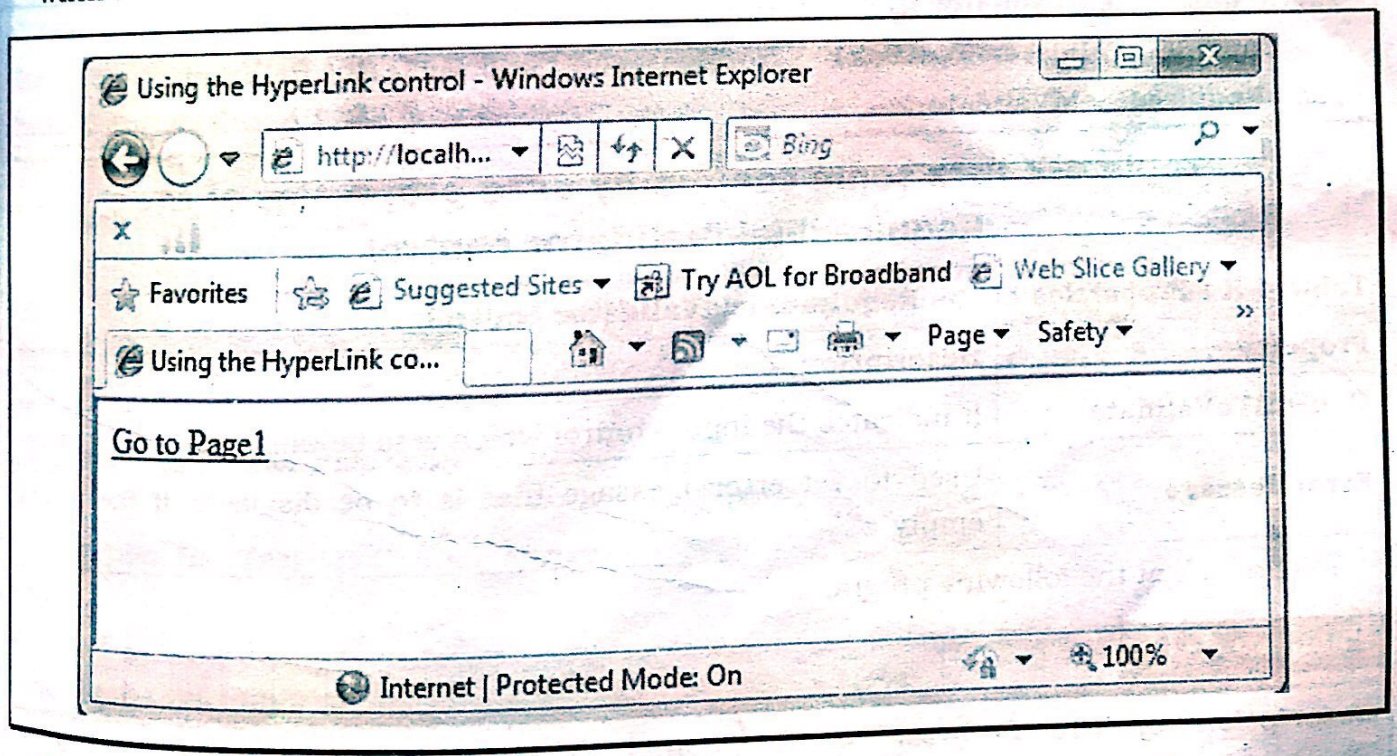
Step-5: Press **F5**. The output will be as follows:



When the user clicks on Hyperlink the output will be as follows:



Here's, the end of the **Program 4-14.**

4.8 VALIDATION CONTROLS

A web page may ask a user for some information and then store it in a back-end database. In almost every case, this information must be validated to ensure that invalid and

VALIDATION CONTROLS :- A webpage may ask a user for some information and then store it in a back-end database. In almost every case, this information must be validated to ensure that invalid and unauthenticated data do not get stored. Ideally, the validation of user input take place on the client side so that the user is immediately informed that there is something wrong with the input before the form is posted back to the server. To do so, ASP.NET provides a set of Validation controls that also reduce the headache of writing lengthy code to validate user input.

Validation controls are used to validate form fields before a form is submitted to the web server. The value of a field is compared with the given set of rules. So, if the value of all the fields satisfies their given set of rule only then the form is post back to the server, otherwise, it may display an error message. It means that validation controls prevent users from submitting the wrong type of data into a database table. For example, a user can not submit the value "123" for an Email ID field.

More than one validation control can also be used for the same input control. ASP.NET provides following validation controls.

4.8.1 Using the RequiredFieldValidator Control

RequiredFieldValidator control used to check whether a user entered a value for a required input field and if the input field has no value then it will display an error message set by **ErrorMessage** property of the **RequiredFieldValidator** control. It is generally tied to a text box. The basic syntax to add this control to a web form using **source code** is as follows:

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
    ControlToValidate="TextBox1" ErrorMessage="Field Must Not Be Empty">
</asp:RequiredFieldValidator>
```

Table 4-K lists some commonly used properties of the RequiredFieldValidator control.

Table 4-K : Properties of the RequiredFieldValidator control	
Property	**Description**
ControlToValidate	It indicates the input control which is to be validated.
ErrorMessage	Used to set error message that is to be displayed if the field is empty.

Take a look at the following program.

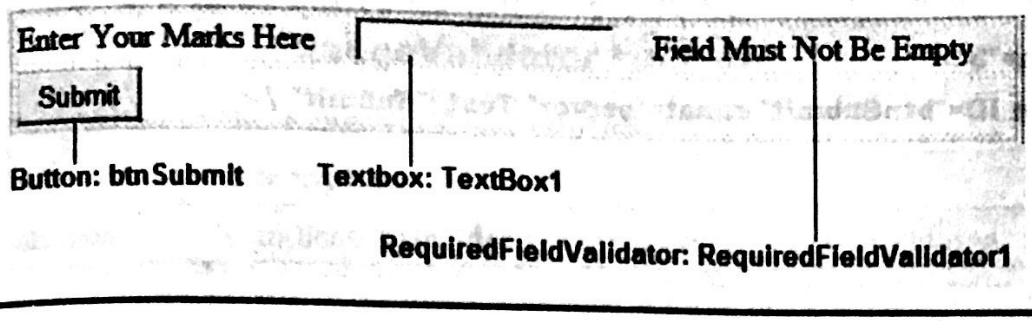
PROGRAM 4-15

Description of the Program 4-15: *In this program, a textbox is validated using RequiredFieldValidator. If textbox is empty and user clicks on submit button an error message will be display to inform the user that the field must not be empty. Follow the steps given below.*

Step-1: Create a new **Web Site** named **"ValidationControls"**.

Step-2: Add a **Web Form** to the application named **"RequiredFieldDemo.aspx"**.

Step-3: Create the following design.



Step-4: Set the following properties:

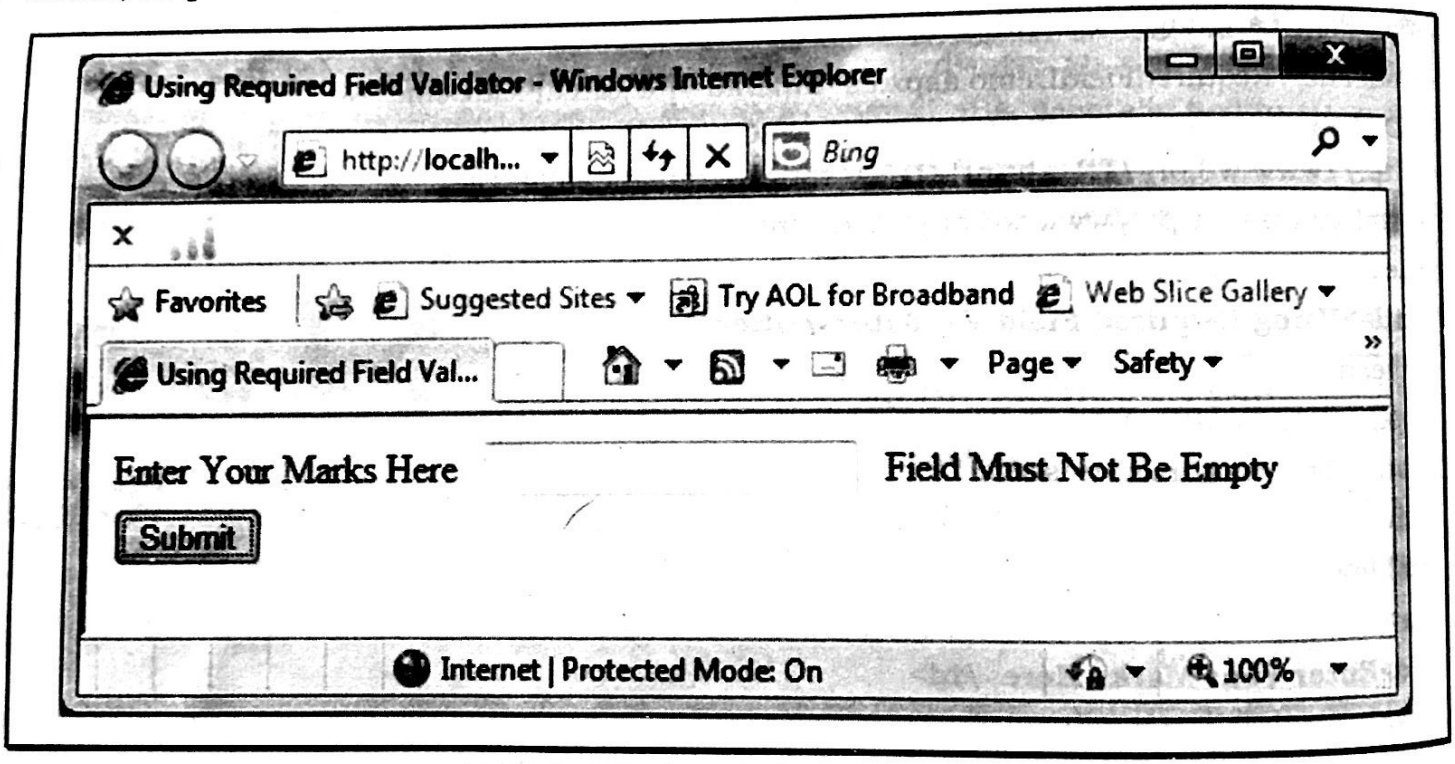
Control	Porperty	Value
RequiredFieldValidator1	ControlToValidate	TextBox1
	ErrorMessage	Field Must Not Be Empty

Source will be as follows:

```
<%@ Page Language="C#" AutoEventWireup="true"
 CodeFile="RequiredFieldDemo.aspx.cs" Inherits="RequiredFieldDemo" %>
<!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>Using Required Field Validator</title>
</head>
<body style="width: 492px">
<form id="form1" runat="server">
<div>
<table>
<tr>
<td>Enter Your Marks Here</td>
<td>
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
</td>
<td>
<asp:RequiredFieldValidator ID="RequiredFieldValidator1"
                            runat="server"
                               ControlToValidate="TextBox1"
                            ErrorMessage="Field Must Not Be Empty">
    </asp:RequiredFieldValidator>
</td>
</tr>
```

```
<tr>
<td colspan="3">
<asp:Button ID="btnSubmit" runat="server" Text="Submit" />
</td>
</tr>
</table>
</div>
</form>
</body>
</html>
```

Step-4: Press **F5**. When **TextBox1** is left empty by the user and the user clicks on **submit** button, **RequiredFieldValidator1** will generate an error message. The output will be as follows:



Here's, the end of the **Program 4-15.**

4.8.2 Using the RangeValidator Control

The **RangeValidator** control verifies that the input value falls within a certain minimum value and maximum value. The basic syntax to add this control to a web form using **source code** is as follows:

```
<asp:RangeValidator ID="RangeValidator1" runat="server"
    ControlToValidate="TextBox1" ErrorMessage="Enter Value in Range (1-100)"
    MaximumValue="100" MinimumValue="1" Type="Integer">
</asp:RangeValidator>
```

Table 4-L lists some commonly used properties of the RangeValidator control.

Table 4-L : Properties of the RangeValidator control	
Property	**Description**
ControlToValidate	It indicates the input control which is to be validated.
ErrorMessage	Used to set error message that is to be displayed if the field value does not falls within the given range.
MaximumValue	It specifies the maximum value of the range.
MinimumValue	It specifies the minimum value of the range.
Type	It specifies the type of data in which the range is to be checked.

Take a look at the following program.

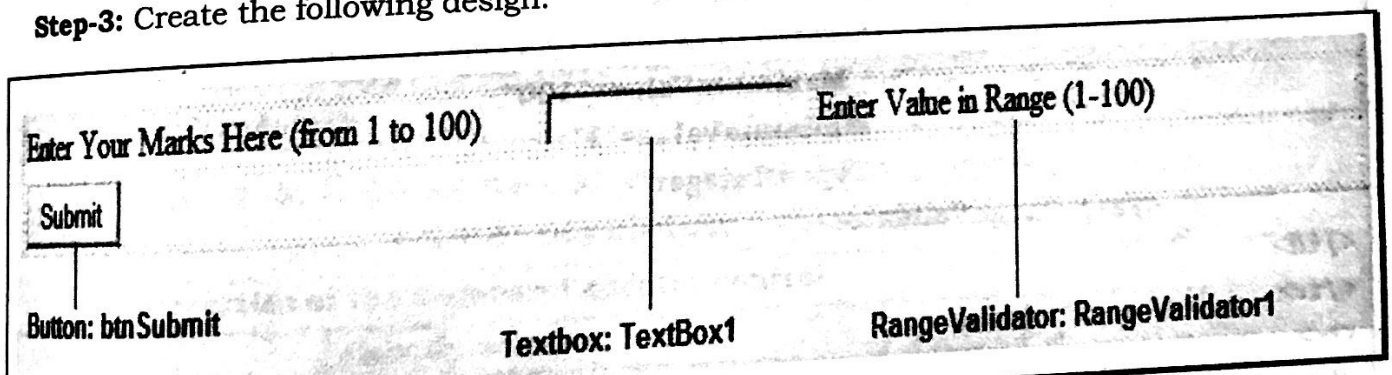
PROGRAM 4-16

Description of the Program 4-16: *In this program, a textbox is validated using RangeValidator. If value entered by the user is out of specified range and user clicks on submit button an error message will be display to inform the user that the entered value is out of specified range. Follow the steps given below.*

Step-1: Create a new **Web Site** named **"ValidationControls"**.

Step-2: Add a **Web Form** to the application named **"RangeValidatorDemo.aspx"**.

Step-3: Create the following design.



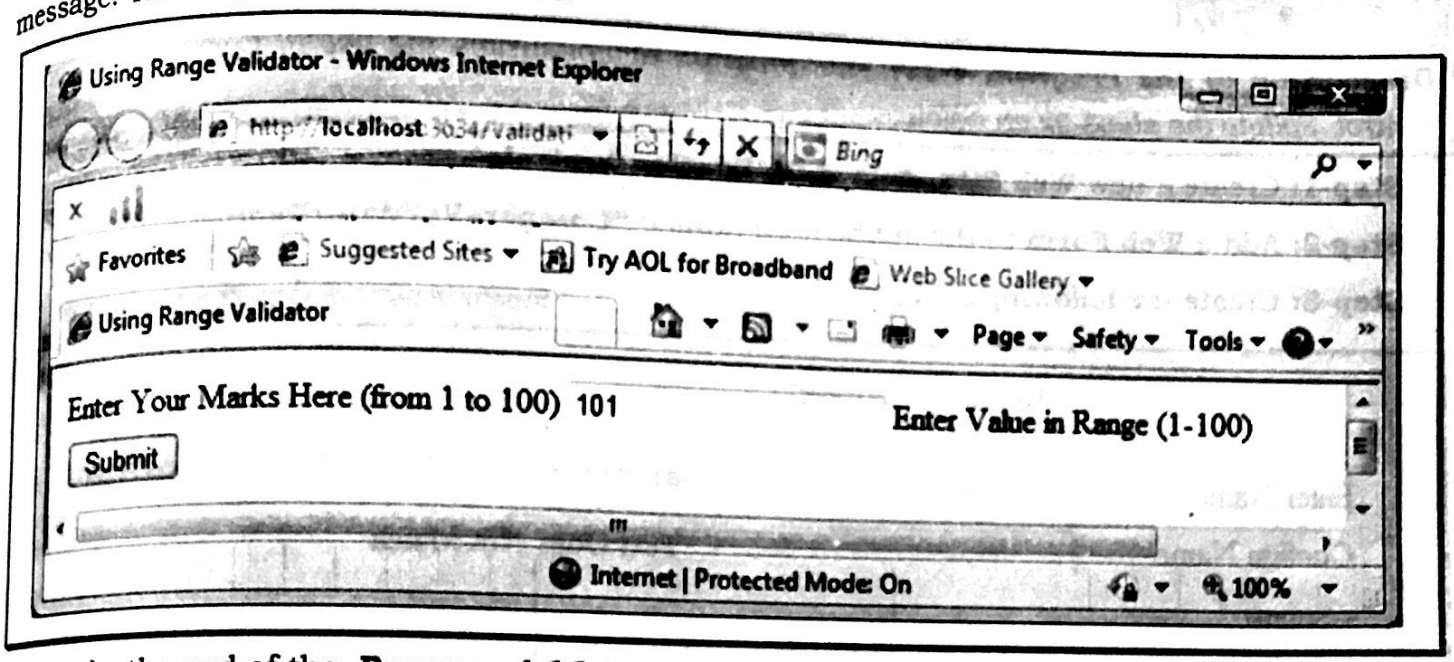
Step-4: Set the following properties:

Control	Porperty	Value
RangeValidator1	ControlToValidate	TextBox1
	ErrorMessage	Enter Value in Range (1-100)
	MaximumValue	100
	MinimumValue	1
	Type	Integer

Source will be as follows:

```
<%@ Page Language="C#" AutoEventWireup="true"
        CodeFile="RangeValidatorDemo.aspx.cs" Inherits="RangeValidatorDemo" %>
<!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>Using Range Validator</title>
</head>
<body style="width: 724px; height: 76px;">
<form id="form1" runat="server">
<div>
<table>
<tr>
<td>Enter Your Marks Here (from 1 to 100)</td>
<td>
<asp:TextBox ID="TextBox1" runat="server" style="margin-left: 0px">
    </asp:TextBox>
</td>
<td>
<asp:RangeValidator ID="RangeValidator1" runat="server"
                ControlToValidate="TextBox1"
                        ErrorMessage="Enter Value in Range (1-100)"
                          MaximumValue="100"
                        MinimumValue="1"
                        Type="Integer">
            </asp:RangeValidator>
</td>
</tr>
<tr>
<td colspan="3">
<asp:Button ID="btnSubmit" runat="server" Text="Submit" />
</td>
</tr>
</table>
</div>
</form>
</body>
</html>
```

Step-5: Press **F5**. When user enters a value in **TextBox1** which is out of range and incompatible type and clicks on submit button, **RangeValidator1** will generate an error message. The output will be as follows:



Here's, the end of the **Program 4-16.**

4.8.3 Using the CompareValidator Control

The **CompareValidator** control compares a value in one control with a fixed value, or, a value in another control. The basic syntax to add this control to a web form using **source code** is as follows:

```
<asp:CompareValidator ID="CompareValidator1" runat="server"
                ControlToCompare="TextBox1" ControlToValidate="TextBox2"
                ErrorMessage="Text Does Not Match">
</asp:CompareValidator>
```

Table 4-M lists some commonly used properties of the CompareValidator control.

Table 4-M : Properties of the CompareValidator control	
Property	**Description**
ControlToCompare	It specifies the input control to compare with.
ControlToValidate	It indicates the input control which is to be validated.
ErrorMessage	Used to set error message that is to be displayed if the field value does not match with the value of the input control **(ControlToCompare)**or the constant value **(ValueToCompare)**.
Operator	It specifies the comparison operator, the available operators are: Equal, NotEqual, GreaterThan, GreaterThanEqual, LessThan, LessThanEqual and DataTypeCheck.
ValueToCompare	It specifies the constant value to compare with.

Take a look at the following program.

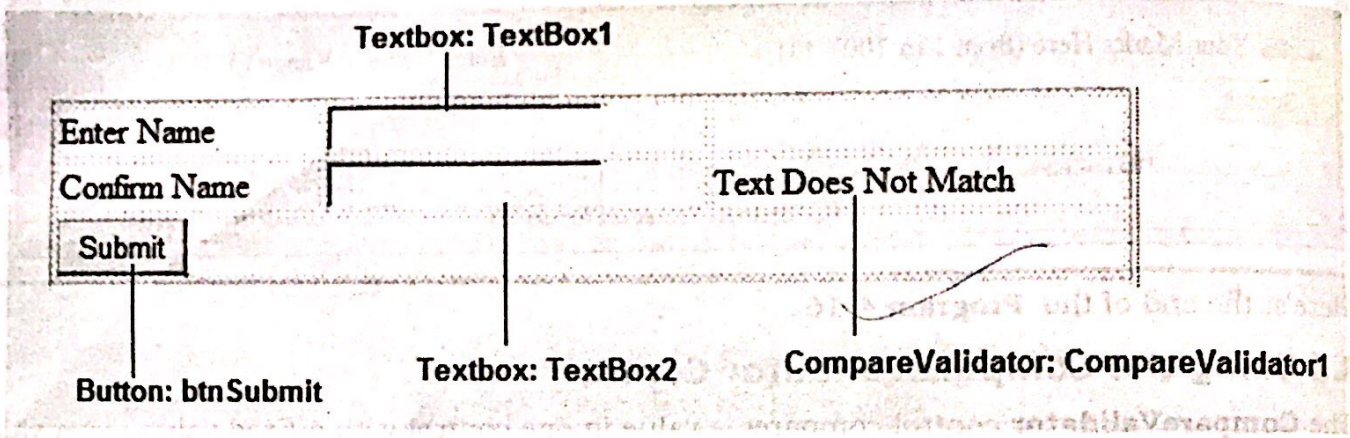
PROGRAM 4-17

Description of the Program 4-17: *This program demonstrates the use of CompareValidator control. Follow the steps given below.*

Step-1: Create a new **Web Site** named **"ValidationControls"**.

Step-2: Add a **Web Form** to the application named **"CompareValidatorDemo.aspx"**.

Step-3: Create the following design.



Step-4: Set the following properties:

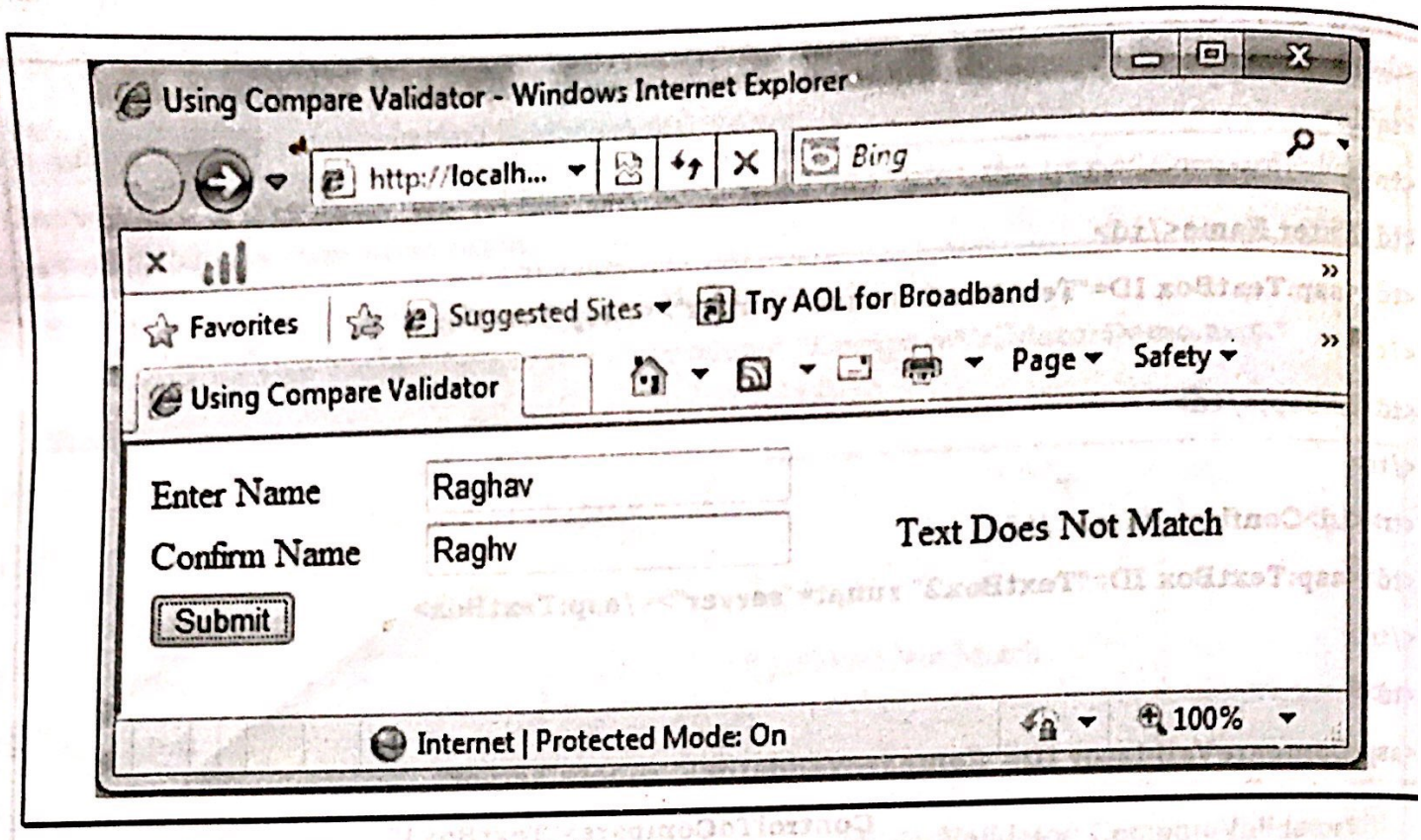
Control	Porperty	Value
CompareValidator1	ControlToCompare	TextBox1
	ControlToValidate	TextBox2
	ErrorMessage	Text Does Not Match

Source of this is as follows:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="CompareValidatorDemo.aspx.cs" Inherits="CompareValidatorDemo" %>
<!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>Using Compare Validator</title>
</head>
<body style="width: 492px">
<form id="form1" runat="server">
```

```
<div>
<table>
<tr>
<td>Enter Name</td>
<td><asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
</td>
<td> </td>
</tr>
<tr><td>Confirm Name</td>
<td><asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
</td>
<td>
<asp:CompareValidator ID="CompareValidator1" runat="server"
                      ControlToCompare="TextBox1"
                    ControlToValidate="TextBox2"
                      ErrorMessage="Text Does Not Match">
    </asp:CompareValidator>
</td>
</tr>
<tr>
<td colspan="3">
<asp:Button ID="btnSubmit" runat="server" Text="Submit"
onclick="btnSubmit_Click" />
</td>
</tr>
</table>
</div>
</form>
</body>
</html>
```

Step-5: Press **F5**. If user enters different values in **TextBox1** and **TextBox2** and clicks on submit button then **CompareValidator1** will generate an error message. The output will be as follows:



Here's, the end of the **Program 4-17.**

4.8.4 Using the RegularExpressionValidator Control

The **RegularExpressionValidator** allows validating the input text by matching it against a regular expression. There is a set of regular expression in the **ValidationExpression** property. The basic syntax to add this control to a web form using **source code** is as follows:

```
<asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"
      ControlToValidate="TextBox1" ErrorMessage="Regular Expression Does Not Match"
      ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*">
</asp:RegularExpressionValidator>
```

Here's the ValidationExpression is for Internet E-Mail address.

Table 4-N lists some commonly used properties of the RegularExpressionValidator control.

Table 4-N : Properties of the RegularExpressionValidator control	
Property	**Description**
ControlToValidate	It indicates the input control which is to be validated.
ErrorMessage	Used to set error message that is to be displayed if the field value expression does not match with the specified **ValidationExpression.**
ValidationExpression	Used to specify the pattern to be matched with the pattern of input control value.

Take a look at the following program.

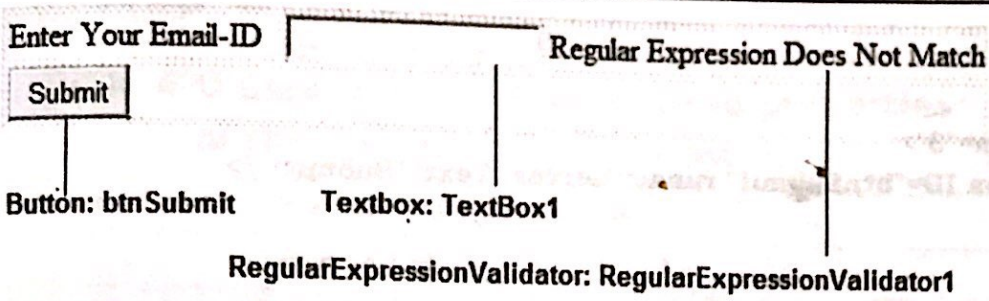
PROGRAM 4-18

Description of the Program 4-18: *This program demonstrates the use of RegularExpressionValidator control. Follow the steps given below.*

Step-1: Create a new **Web Site** named **"ValidationControls"**.

Step-2: Add a **Web Form** to the application named **"RegularExpressionDemo.aspx"**.

Step-3: Create the following design.



Step-4: Set the following properties:

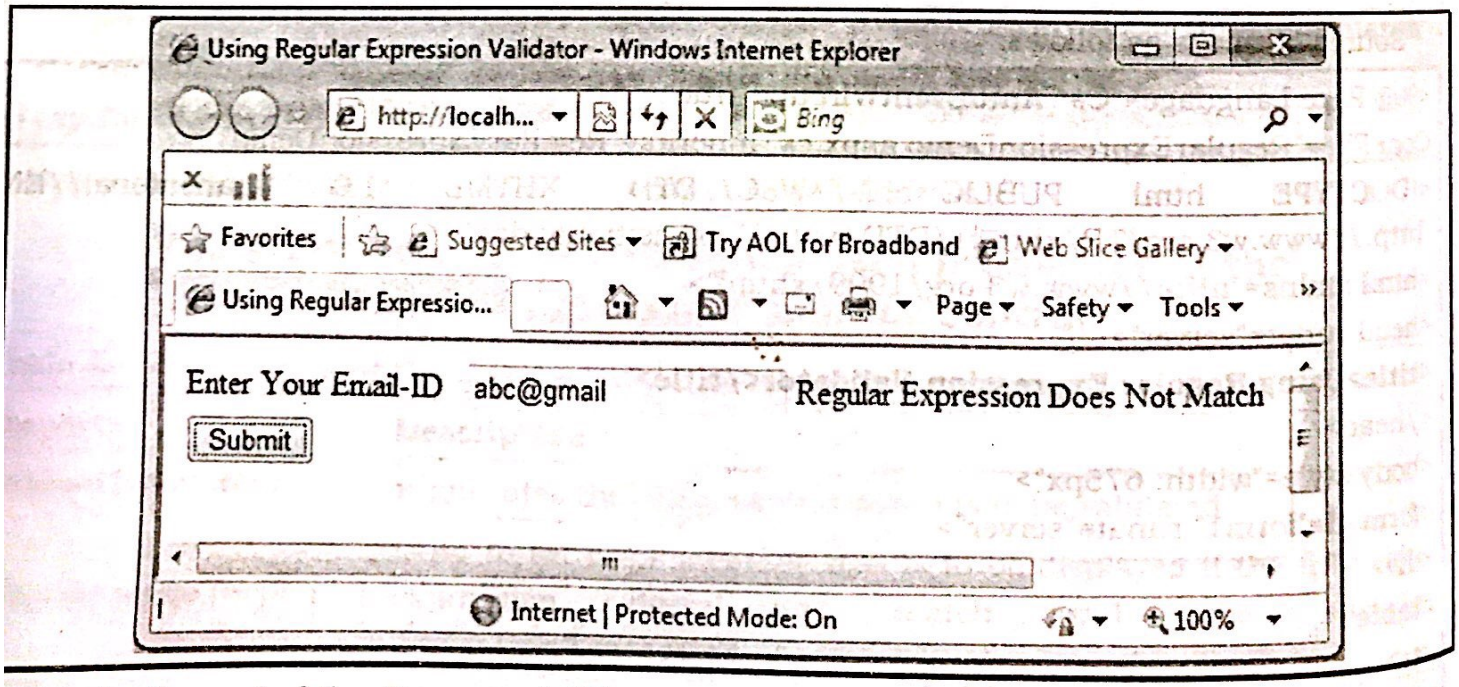
Control	Porperty	Value
RegularExpression	ControlToValidate	TextBox3
	ErrorMessage	Regular Expression Does Not Match
	ValidationExpression	\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)
		Internet e-mail address

Source of this is as follows:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="RegularExpressionDemo.aspx.cs" Inherits="RegularExpressionDemo" %>
<!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>Using Regular Expression Validator</title>
</head>
<body style="width: 675px">
<form id="form1" runat="server">
<div>
<table>
<tr>
<td>Enter Your Email-ID  </td>
<td>
```

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
</td>
<td>
<asp:RegularExpressionValidator ID="RegularExpressionValidator1"
            runat="server"
            ControlToValidate="TextBox1"
            ErrorMessage="Regular Expression Does Not Match"
            ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*">
            </asp:RegularExpressionValidator>
</td>
</tr>
<tr>
<td colspan="3">
<asp:Button ID="btnSubmit" runat="server" Text="Submit" />
</td>
</tr>
</table>
</div>
</form>
<p> </p>
</body>
</html>
```

Step-4: Press **F5**. If user enters an invalid **Email-ID** in **TextBox1** and clicks on submit button then **RegularExpressionValidator1** will generate an error message. The output will be as follows:



Here's, the end of the **Program 4-18.**

4.8.5 Using the CustomValidator Control

The **CustomValidator** control allows writing application specific custom validation routines for both the client side and the server side validation. The basic syntax to add this control to a web form using **source code** is as follows:

```
<asp:CustomValidator id="CustomValidator1" ControlToValidate="TextBox1"
    ErrorMessage="Custom Validation Does Not Meet"
    ClientValidationFunction="Function_ClientValidate"
    OnServerValidate="Function_ServerValidate" runat="server">
</asp:CustomValidator>
```

Table 4-O lists some commonly used properties of the CustomValidator control.

Table 4-O : Properties of the CustomValidator control

Property	Description
ControlToValidate	It indicates the input control which is to be validated.
ErrorMessage	Used to set error message that is to be displayed if the field value does not satisfies the client side validation or server side validation
ClientValidationFunction	Access function of client side validation.
OnServerValidate	Access function of server side validation.

Take a look at the following program.

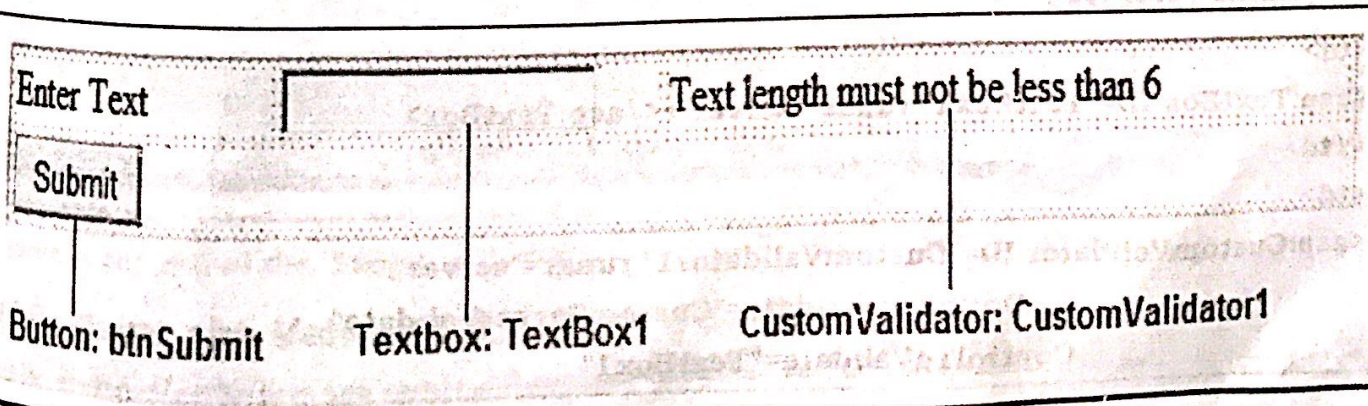
PROGRAM 4-19

Description of the Program 4-19: *This program demonstrate the use of CustomValidatorControl. Follow the steps given below.*

Step-1: Create a new **Web Site** named **"ValidationControls"**.

Step-2: Add a **Web Form** to the application named **"CustomValidationDemo.aspx"**.

Step-3: Create the following design.



Step-4: Set the following properties:

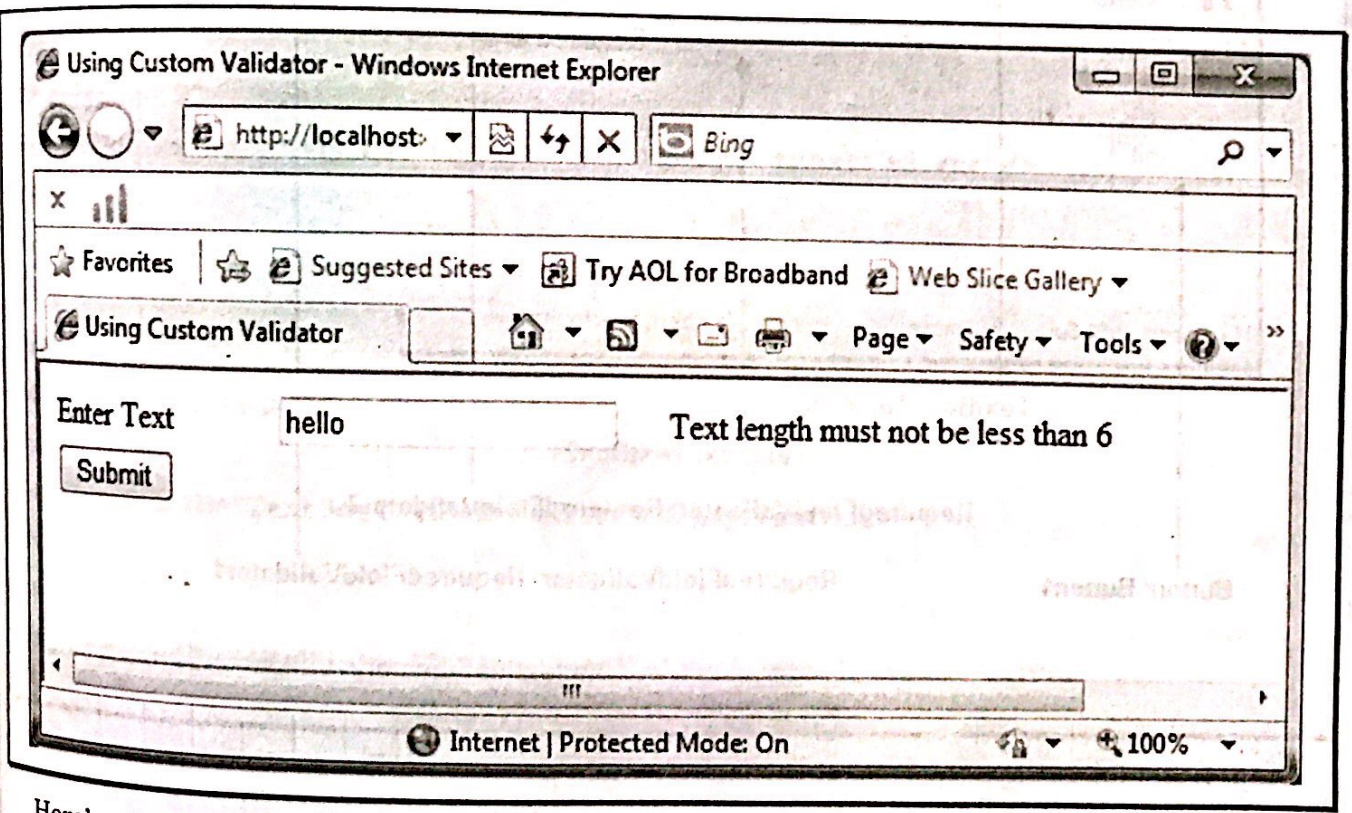
Control	Porperty	Value
CustomValidator1	ControlToValidate	TextBox1
	ErrorMessage	Text length must not be less than 6
	ServerValidate **Note:** It's an event	CustomServerValidate

Source will be as follows:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="CustomValidationDemo.aspx.cs"
Inherits="CustomValidationDemo" %>
<!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>Using Custom Validator</title>
<script runat="server">
void CustomServerValidate(Object source, ServerValidateEventArgs args)
   {
if (args.Value.Length < 6)
        args.IsValid = false;
else
         args.IsValid = true;
   }
</script>
</head>
<body style="width: 675px">
<form id="form1" runat="server">
<div>
<table>
<tr>
<td>Enter Text</td>
<td>
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
</td>
<td>
<asp:CustomValidator ID="CustomValidator1" runat="server"
                    OnServerValidate="CustomServerValidate"
                   ControlToValidate="TextBox1"
```

```
            ErrorMessage="Text length must not be less than 6" >
        </asp:CustomValidator>
</td>
</tr>
<tr>
<td colspan="3">
<asp:Button ID="btnSubmit" runat="server" Text="Submit" />
</td>
</tr>
</table>
</div>
</form>
</body>
</html>
```

Step-5: Press **F5**. When user enter text in **TextBox1** whose length is less than 6, and the user clicks on **submit** button, **CustomValidator1** will generate an error message. The output will be as follows:



Here's, the end of the **Program 4-19.**

4.8.6 Using the ValidationSummary Control

The **ValidationSummary** control does not perform any validation but shows a summary of all

errors in the page. The basic syntax to add this control to a web form using **source code** is as follows:

```
<asp:ValidationSummary ID="ValidationSummary1" runat="server" />
```

Take a look at the following program.

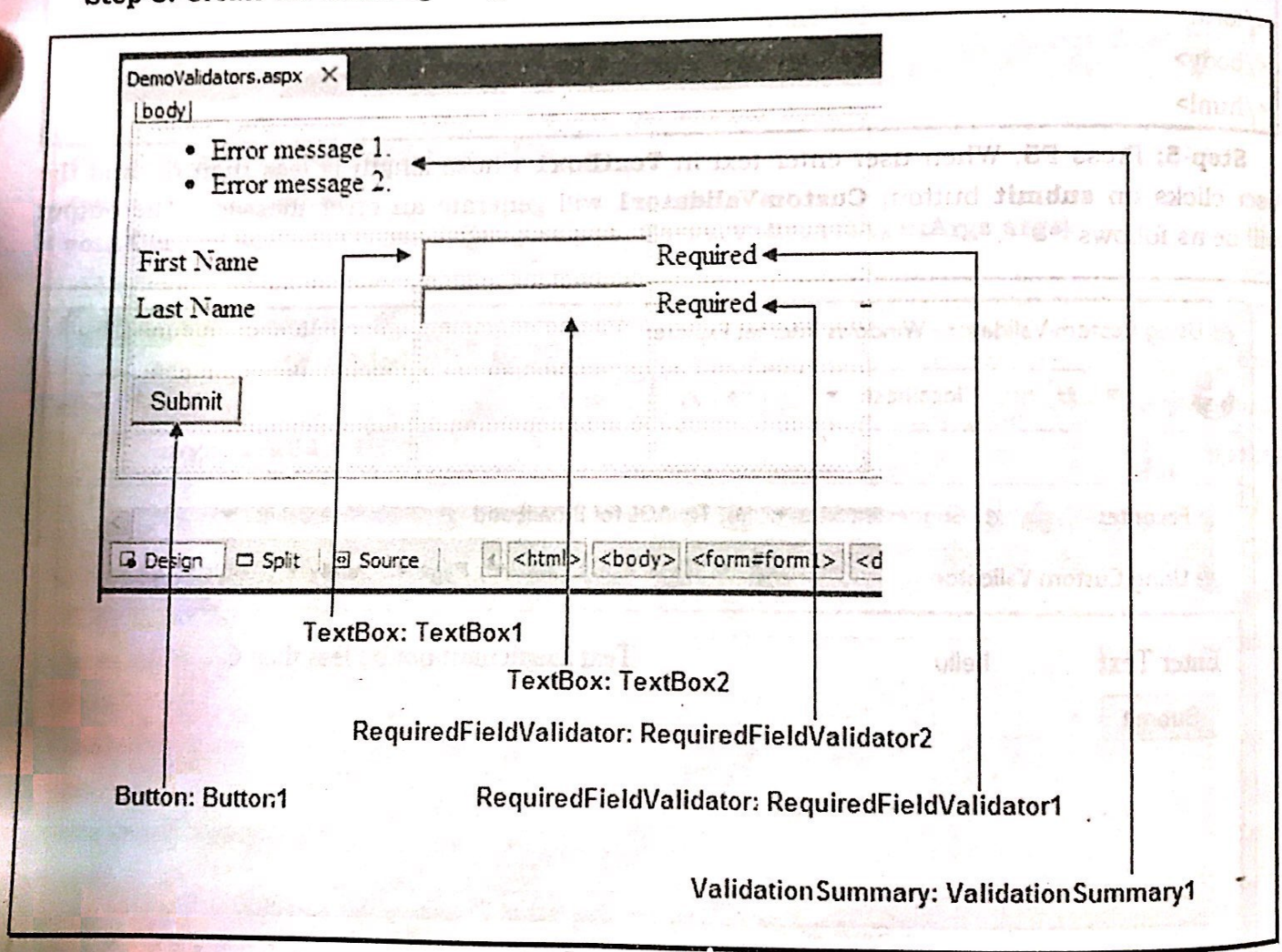
PROGRAM 4-20

Description of the Program 4-20: *This program demonstrates the use of ValidationSummary controls. Follow the steps given below.*

Step-1: Create a new **Web Application** named **"ValidationControls"**.

Step-2: Add a **Web Form** to the application named **"SummaryValidatorDemo.aspx"**.

Step-3: Create the following design.



Step-4: Set following properties of various Controls.

Control	Porperty	Value
RequiredFieldValidator1	ControlToValidate	TextBox1
	ErrorMessage	First Name Required
	Text	Required

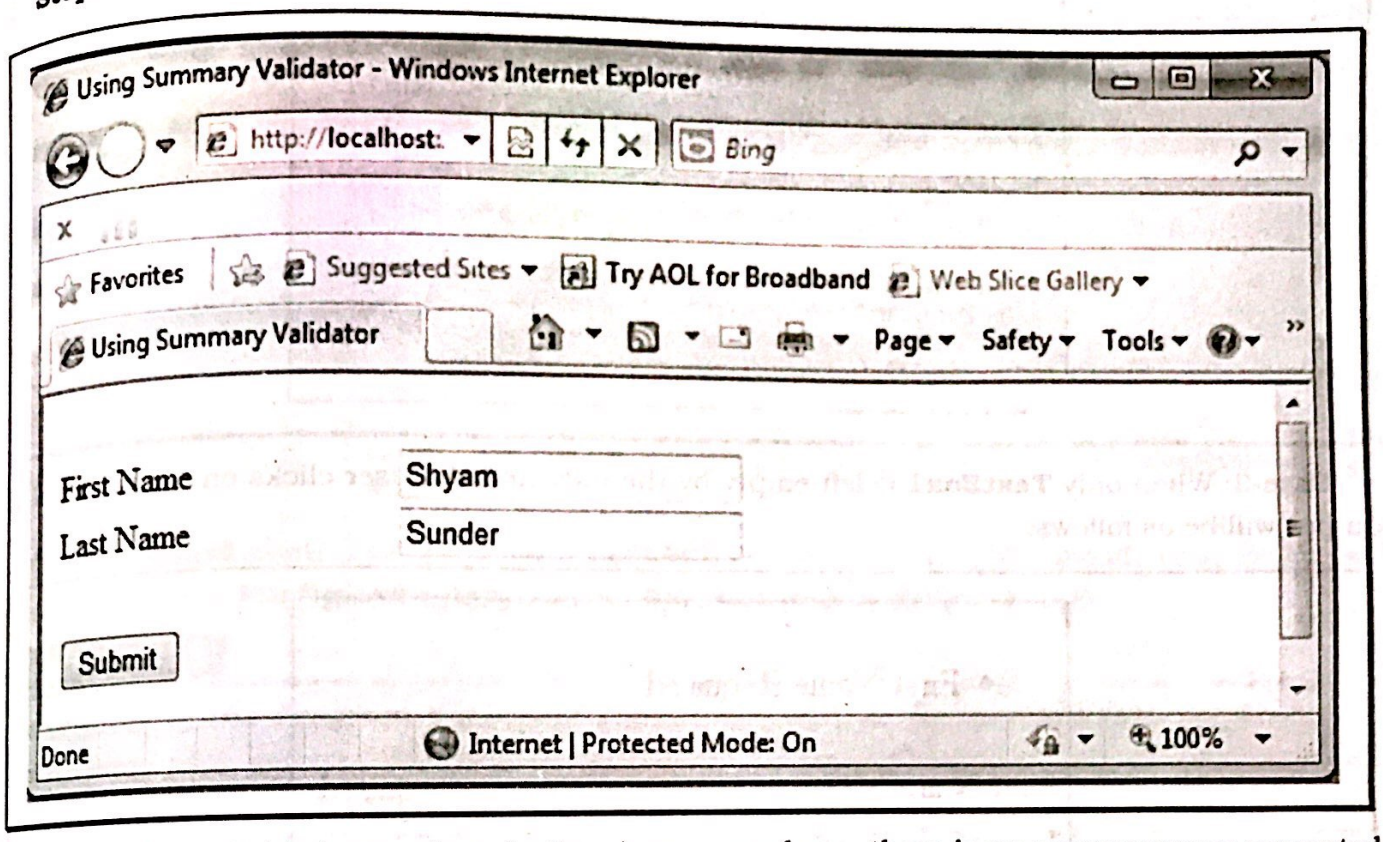
RequiredFieldValidator2	ControlToValidate	TextBox2
	ErrorMessage	Last Name Required
	Text	Required
Button1	Text	Submit

Source will be as follows:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="SummaryValidatorDemo.aspx.cs" Inherits="_Default" %>
<!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>Using Summary Validator</title>
</head>
<body>
<form id="form1" runat="server">
<div>
<table>
<tr>
<td colspan="2">
<asp:ValidationSummary ID="ValidationSummary1" runat="server" />
</td>
<td> </td>
</tr>
<tr>
<td>First Name</td>
<td>
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
</td>
<td>
<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
                            ControlToValidate="TextBox1"
                           ErrorMessage="First Name Required">Required
      </asp:RequiredFieldValidator>
</td>
</tr>
```

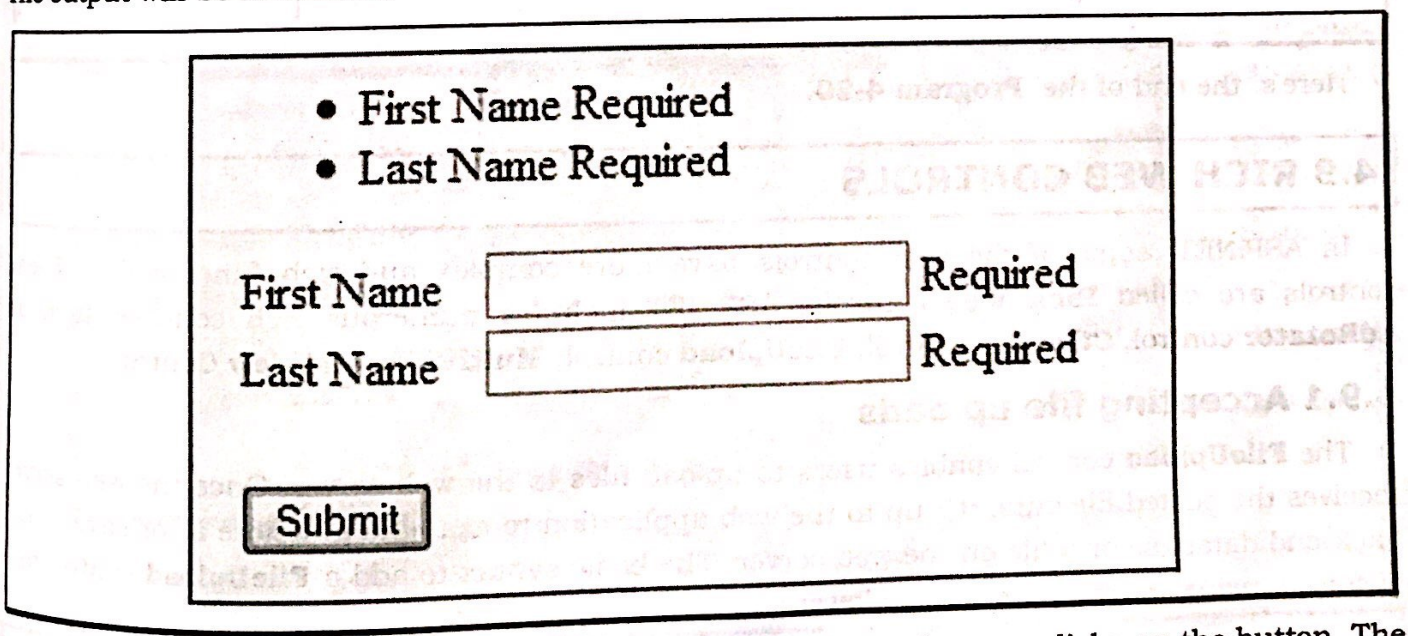
```
<tr>
<td>Last Name</td>
<td>
            <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
</td>
<td>
<asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
                            ControlToValidate="TextBox2"
                        ErrorMessage="Last Name Required">Required
      </asp:RequiredFieldValidator>
</td>
</tr>
<tr>
<td></td>
<td></td>
<td> </td>
</tr>
<tr>
<td>
<asp:Button ID="Button1" runat="server" Text="Submit" />
</td>
<td> </td>
<td> </td>
</tr>
<tr>
<td> </td>
<td> </td>
<td> </td>
</tr>
</table>
</div>
</form>
</body>
</html>
```

Step-5: Press **F5**. The output will be as follows:

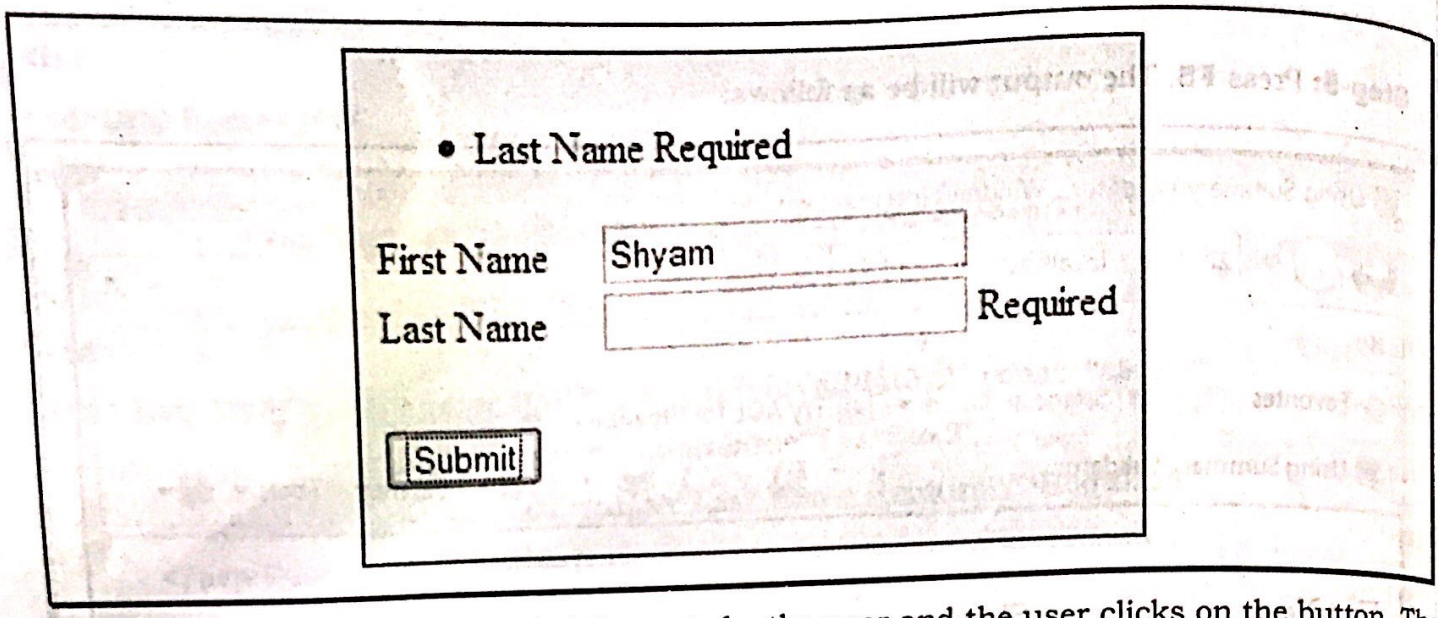


Since both the fields have values in the given example so, there is no error message generated by any of the validation control. Now, we are going to discuss some cases when one or both the fields are left empty by the user. These are as follows:

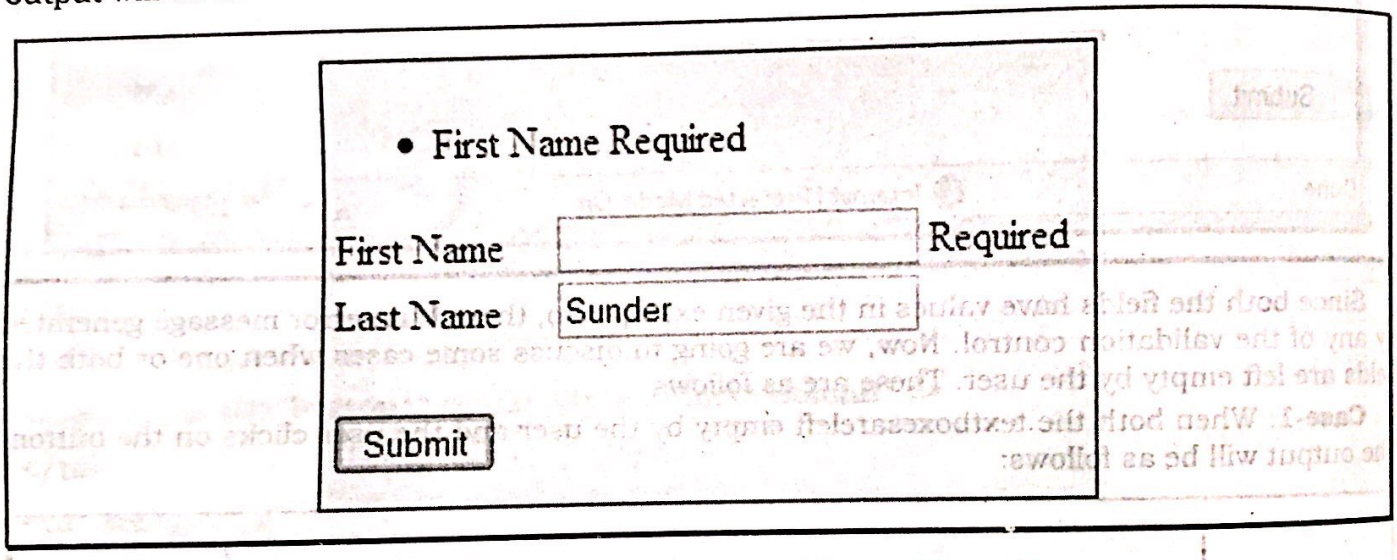
Case-1: When both the textboxesareleft empty by the user and the user clicks on the button. The output will be as follows:



Case-2: When only **TextBox2** is left empty by the user and the user clicks on the button. The output will be as follows:



Case-3: When only **TextBox1** is left empty by the user and the user clicks on the button. The output will be as follows:



Here's, the end of the **Program 4-20.**

4.9 RICH WEB CONTROLS

In ASP.NET, some of the web controls have more complex and rich functionality. These controls are called **Rich Web** controls. ASP.NET includes numerous rich controls such as **AdRotator** control, **Calendar** control, **FileUpload** control, **MultiView** and**View** Control.

4.9.1 Accepting file uploads

The **FileUpload** control enables users to upload files to the web server. Once the web server receives the posted file data, it's up to the web application to examine it, ignore it, or save it to a back-end database or a file on the web server. The basic syntax to add a **FileUpload** control to a web form using **source** code is as follows:

```
<asp:FileUpload ID="FileUpload1" runat="server" />
```

Table 4-P lists some commonly used properties of the FileUpload control.

Table 4-P : Properties of the Calendar control	
Property	**Description**
Enabled	Used to enable or disable the **FileUpload** control.
FileBytes	Used to get the uploaded file contents as a byte array.
FileContent	Used to get the uploaded file contents as a stream.
FileName	Used to get the name of the file uploaded.
HasFile	Returns True when a file has been uploaded.
PostedFile	Used to get the uploaded file wrapped in the **HttpPostedFile** object.

The **FileUpload** control also supports **SaveAs()** method to save the uploaded file to the file system. The use of **FileUpload** control can be demonstrated by **Program 4-21**

PROGRAM 4-21

Description of the Program 4-21: *In this program, a file is selected using FileUpload control. On a button click selected file is to be uploaded into the web application's directory named UploadedFiles. Follow the steps given below.*

Step-1: Create a new **Web Site** named "**RichControlsApp**".

Step-2: Add a **Web Form** to the application named **"FileUploadControl.aspx"**.

Step-3: Add a folder to the application named "**UploadedFiles**" where all the images will be uploded.