Web Development Assignment 1

**Datatypes used:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Textbox** | **Variable** | **Datatype** | **Reason** |
| TextBoxFirstName | firstName | string | Value will contain only letters |
| TextBoxLastName | lastName | string | Value will contain only letters |
| TextBoxAddress | address | string | Value will contain both letters and numbers |
| TextBoxCity | city | string | Value will contain only letters |
| TextBoxProvince | province | string | Value will contain only letters |
| TextBoxPostalCode | postalCode | string | Value will contain both letters and numbers |
| TextBoxAge | age | integer | Value will contain only numbers |
| TextBoxPassword | password | string | Value will contain characters of all variety |
| TextBoxAPassword | confirmPassword | string | Value will contain characters of all variety |
| TextBoxEmail | email | string | Value will contain both letters and numbers and a @ symbol |
| TextBoxAEmail | altEmail | string | Value will contain both letters and numbers and a @ symbol |
| TextBoxPhone | phone | string | Value will contain mostly numbers but will also include slashes or spaces |

**Questions**

1. **What kind of Directives did you use? Why?**

I only used one directive, which being the page language being set to C#. I chose C# over Visual Basic as I have much more experience with C# and I feel it can accomplish more than Visual Basic.

1. **Explain the role of View State in a few sentences.**

View State is a way of maintaining a state, from across multiple PostBacks to the web server. In a nutshell, it is a technique used to keep changes to the state of a Web Form for PostBacks.

1. **What kind of logical expression did you use? Explain about it.**

I used a boolean called “keepReg”. I required this bool as it is used to allow the for loop to loop or not. If it is true, it will loop, if it is false, it will not run.

1. **What kind of loop did you use? Why not the other type?**

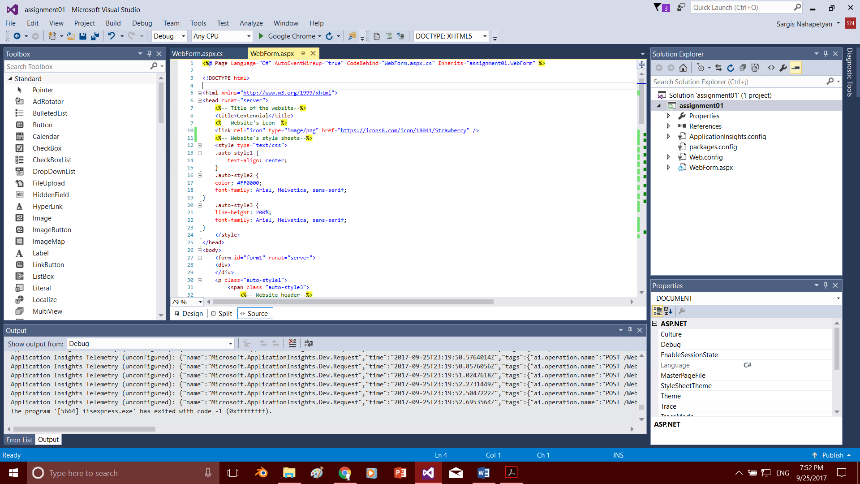
I used a “for” loop in order to continue the registration. I chose to sue for as I only needed the loop to run once. If I used a “while” loop, it would constantly run without stopping and even prevent the page from loading due to constantly looping.

1. **What is the meaning of robustness? Did you use this feature in your project?**

Robustness is the ability of a computer program to handle errors and large input. I would say my program is quite robust as it handles many variables and values. It also disallows errors in the textboxes, making it error proof.

**Code** **Segments:**

**HTML:**

 **Header:**

Contains style sheets, the title of the website and the website’s icon. The website’s title will be displayed on the top. The icon is a link which takes an image from the link and puts it beside the title. The style sheets are used to give graphic properties to the text and textboxes.

<head runat="server">

<%-- Title of the website--%>

<title>Centennial</title>

<%-- Website's icon--%>

<link rel="icon" type="image/png" href="https://icons8.com/icon/18041/Strawberry" />

<%-- Website's style sheets--%>

<style type="text/css">

.auto-style1 {

text-align: center;

}

.auto-style2 {

color: #FF0000;

font-family: Arial, Helvetica, sans-serif;

}

.auto-style3 {

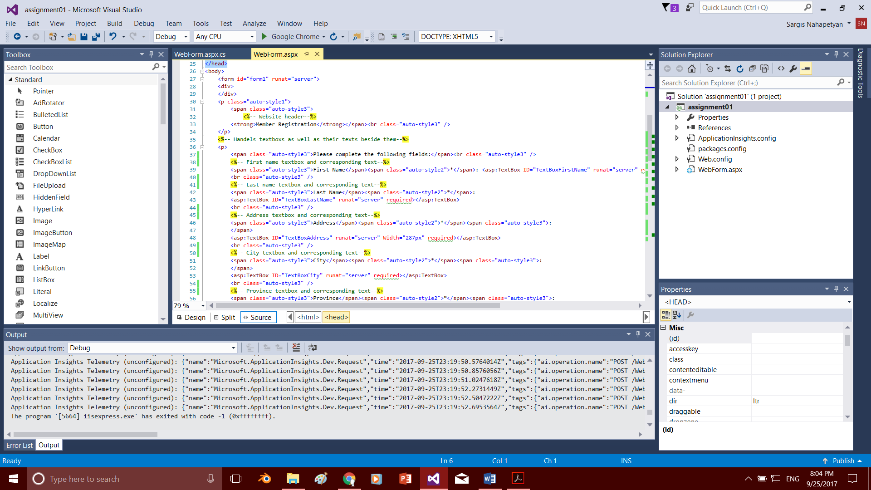
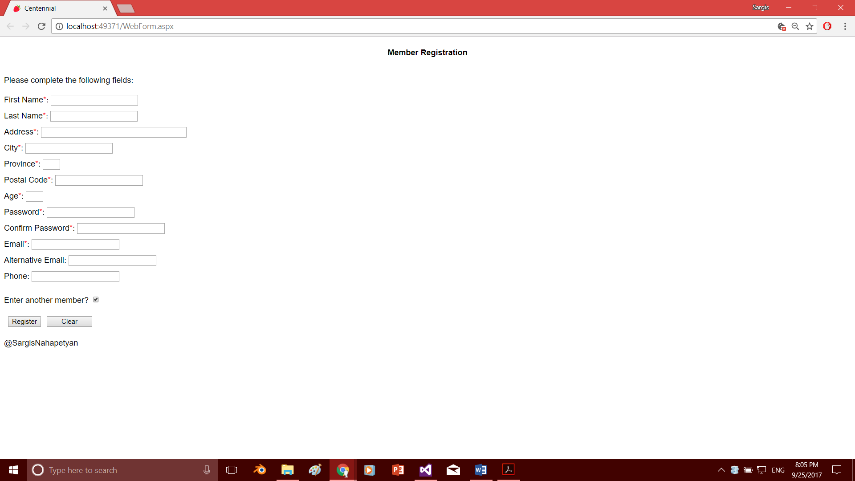
line-height: 200%;

font-family: Arial, Helvetica, sans-serif;

}

</style>

</head>

 **Body:**

Contains text, textboxes, buttons, a checkbox, and all their requirements. Each field enables a new button, checkbox, and textbox to be displayed as well as their text beside them. The fields with “required” in them means that the browser will not allow them to continue until it is filled. “Pattern” means that the textbox requires a certain input in order to be made valid. “Runat” specifics the server they need to run onto. “Class” specifics the style sheet the field uses. Finally, “type” specifics the type of contains in the textbox, such as “email” should be an email and “password” will be displayed hidden.

<body>

<form id="form1" runat="server">

<div>

</div>

<p class="auto-style1">

<span class="auto-style3">

<%-- Website header--%>

<strong>Member Registration</strong></span><br class="auto-style3" />

</p>

<%-- Handels textboxs as well as their texts beside them--%>

<p>

<span class="auto-style3">Please complete the following fields:</span><br class="auto-style3" />

<%-- First name textbox and corresponding text--%>

<span class="auto-style3">First Name</span><span class="auto-style2">\*</span>: <asp:TextBox ID="TextBoxFirstName" runat="server" required></asp:TextBox>

<br class="auto-style3" />

<%-- Last name textbox and corresponding text--%>

<span class="auto-style3">Last Name</span><span class="auto-style2">\*</span>:

<asp:TextBox ID="TextBoxLastName" runat="server" required></asp:TextBox>

<br class="auto-style3" />

<%-- Address textbox and corresponding text--%>

<span class="auto-style3">Address</span><span class="auto-style2">\*</span><span class="auto-style3">:

</span>

<asp:TextBox ID="TextBoxAddress" runat="server" Width="287px" required></asp:TextBox>

<br class="auto-style3" />

<%-- City textbox and corresponding text--%>

<span class="auto-style3">City</span><span class="auto-style2">\*</span><span class="auto-style3">:

</span>

<asp:TextBox ID="TextBoxCity" runat="server" required></asp:TextBox>

<br class="auto-style3" />

<%-- Province textbox and corresponding text--%>

<span class="auto-style3">Province</span><span class="auto-style2">\*</span><span class="auto-style3">:

</span>

<asp:TextBox ID="TextBoxProvince" runat="server" Width="31px" style="text-transform:uppercase" pattern="[A-Za-z]{2}" maxlength="2" required></asp:TextBox>

<br class="auto-style3" />

<%-- Postal code textbox and corresponding text--%>

<span class="auto-style3">Postal Code</span><span class="auto-style2">\*</span><span class="auto-style3">:

</span>

<asp:TextBox ID="TextBoxPostalCode" pattern="[A-Za-z][0-9][A-Za-z][0-9][A-Za-z][0-9]" title="Must follow the a0a0a0 format." runat="server" required></asp:TextBox>

<br class="auto-style3" />

<%-- Age textbox and corresponding text--%>

<span class="auto-style3">Age</span><span class="auto-style2">\*</span><span class="auto-style3">:

</span>

<asp:TextBox ID="TextBoxAge" runat="server" Width="31px" maxlength="3" type="number" min="18" max="120" required></asp:TextBox>

<br class="auto-style3" />

<%-- Password textbox and corresponding text--%>

<span class="auto-style3">Password</span><span class="auto-style2">\*</span><span class="auto-style3">:

</span>

<asp:TextBox ID="TextBoxPassword" type="password" pattern="^(?=.\*[A-Z])(?=.\*\d)[A-Za-z\d]{6,}$" title="Must have at least 6 characters and contain at least one digit

and one upper-case character." runat="server" required></asp:TextBox>

<br class="auto-style3" />

<%-- Confirm password textbox and corresponding text--%>

<span class="auto-style3">Confirm Password</span><span class="auto-style2">\*</span><span class="auto-style3">:

</span>

<asp:TextBox ID="TextBoxAPassword" type="password" pattern="^(?=.\*[A-Z])(?=.\*\d)[A-Za-z\d]{6,}$" title="Must have at least 6 characters and contain at least one digit

and one upper-case character." runat="server" required></asp:TextBox>

<br class="auto-style3" />

<%-- Email textbox and corresponding text--%>

<span class="auto-style3">Email</span><span class="auto-style2">\*</span><span class="auto-style3">:

</span>

<asp:TextBox ID="TextBoxEmail" type="email" pattern="[a-z0-9.\_%+-]+@[a-z0-9.-]+\.[a-z]{2,4}" runat="server" required></asp:TextBox>

<br class="auto-style3" />

<%-- Alternative email textbox and corresponding text--%>

<span class="auto-style3">Alternative Email: </span>

<asp:TextBox ID="TextBoxAEmail" type="email" pattern="[a-z0-9.\_%+-]+@[a-z0-9.-]+\.[a-z]{2,4}" runat="server"></asp:TextBox>

<br class="auto-style3" />

<%-- Phone textbox and corresponding text--%>

<span class="auto-style3">Phone: </span>

<asp:TextBox ID="TextBoxPhone" runat="server"></asp:TextBox>

<br class="auto-style3" />

</p>

<p class="auto-style3">

Enter another member?

<%-- Keep reg checkbox--%>

<asp:CheckBox ID="CheckBoxAM" runat="server" OnCheckedChanged="CheckBoxAM\_CheckedChanged" />

</p>

&nbsp

<%-- Reg button--%>

<asp:Button ID="ButtonReg" runat="server" Text="Register" OnClick="ButtonReg\_Click" />

&nbsp

<%-- Clear button--%>

<asp:Button ID="ButtonClear" runat="server" Text="Clear" Width="90px" formnovalidate OnClick="ButtonClear\_Click"/>

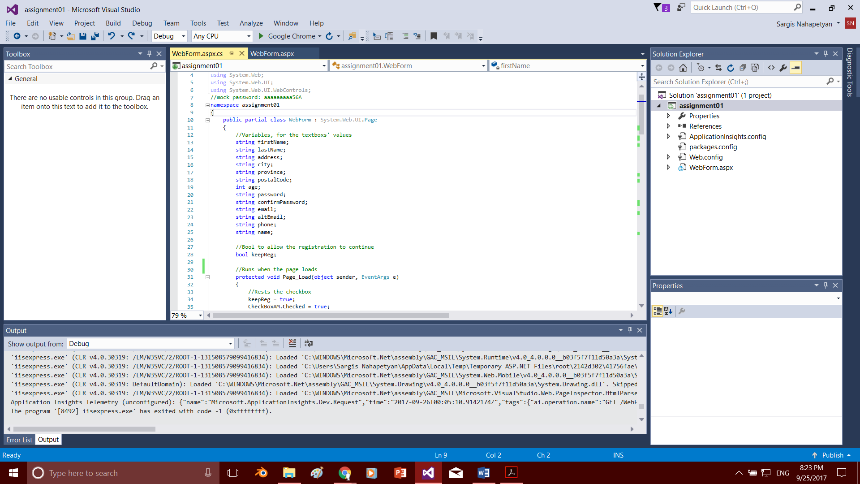
</form>

<%-- Codyright of website--%>

<p class="auto-style3">@SargisNahapetyan </p>

</body>

**C#:**

 **Global Variables:**

Contains all the variables that are used to store the textbox values. Each variable corresponds to a different textbox, such as “fristName” stores the contains of “TextBoxFirstName”. Most are a string datatype due to most of them hold values that have letters, however “age” only uses numbers so it uses an integer datatype. “name” and “keepReg” are special, as “name” is used to combine the first and last name, and “keepReg” is a boolean that is used to allow the registration to keep looping.

//Variables, for the textboxs' values

string firstName;

string lastName;

string address;

string city;

string province;

string postalCode;

int age;

string password;

string confirmPassword;

string email;

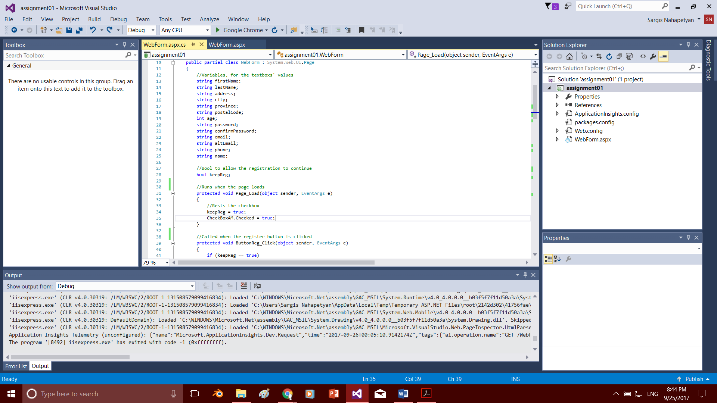
string altEmail;

string phone;

string name;

//Bool to allow the registration to continue

bool keepReg;

 **Page\_Load:**

A function that runs whenever the page is loaded. “keepReg = true” sets the registration boolean to true while “CheckBoxAM.Checked = true” states that the checkbox is checked. Both of these happen whenever the page is loaded.

//Runs when the page loads

protected void Page\_Load(object sender, EventArgs e)

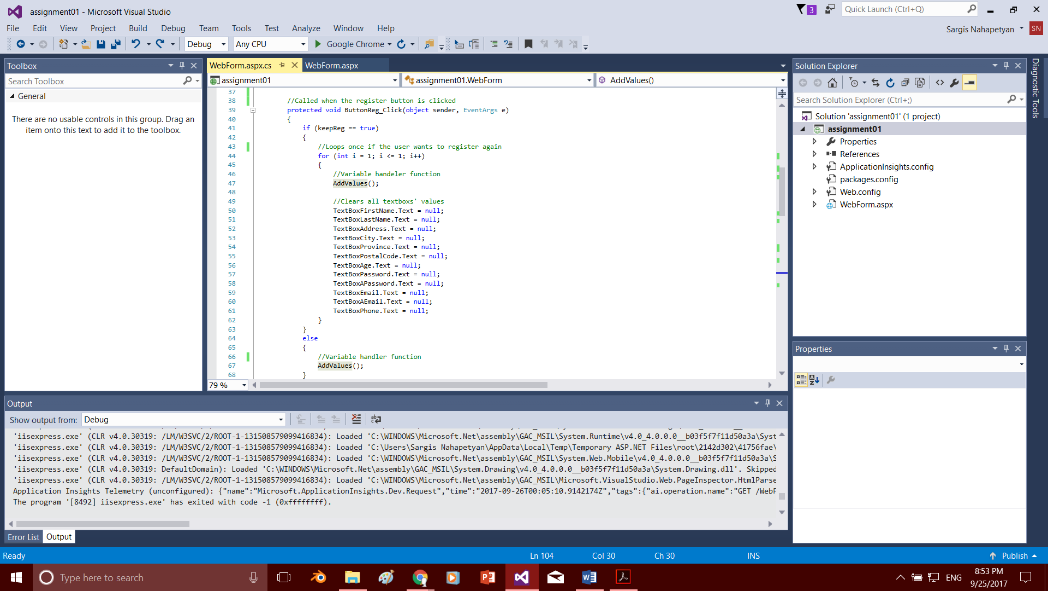
{

//Rests the checkbox

keepReg = true;

CheckBoxAM.Checked = true;

}

 **ButtonReg\_Click:**

Function is called whenever the register button is clicked. Once clicked, it will run the “AddValues” function, which adds all the textbox values into the variables. If “keepReg” is true, it will run “AddValues” and in addition, it will clear the textboxes and loop once. It will continue to loop as long as “keepReg” is true.

//Called when the register button is clicked

protected void ButtonReg\_Click(object sender, EventArgs e)

{

if (keepReg == true)

{

//Loops once if the user wants to register again

for (int i = 1; i <= 1; i++)

{

//Variable handeler function

AddValues();

//Clears all textboxs' values

TextBoxFirstName.Text = null;

TextBoxLastName.Text = null;

TextBoxAddress.Text = null;

TextBoxCity.Text = null;

TextBoxProvince.Text = null;

TextBoxPostalCode.Text = null;

TextBoxAge.Text = null;

TextBoxPassword.Text = null;

TextBoxAPassword.Text = null;

TextBoxEmail.Text = null;

TextBoxAEmail.Text = null;

TextBoxPhone.Text = null;

}

}

else

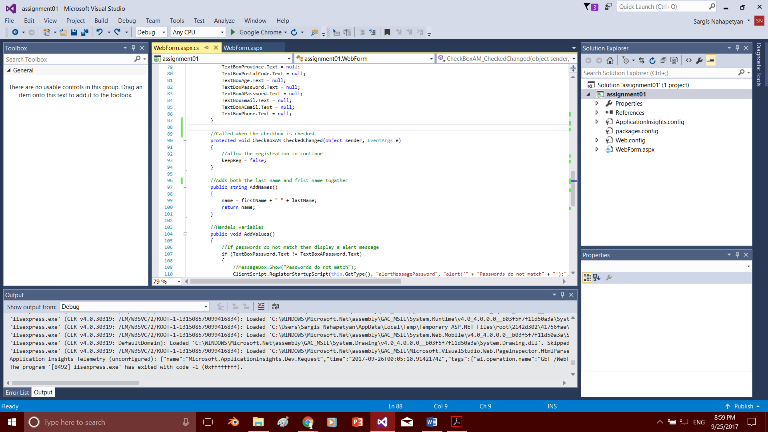
{

//Variable handler function

AddValues();

}

}

 **ButtonClear\_Click:**

The function is quite simple. The function is called whenever the clear button is clicked. Once clicked, all textbox values will be set to null. Which will wipe them clean and make them blank, erasing all values in them.

//Called when the clear button is clicked

protected void ButtonClear\_Click(object sender, EventArgs e)

{

//Clears all textboxs' values

TextBoxFirstName.Text = null;

TextBoxLastName.Text = null;

TextBoxAddress.Text = null;

TextBoxCity.Text = null;

TextBoxProvince.Text = null;

TextBoxPostalCode.Text = null;

TextBoxAge.Text = null;

TextBoxPassword.Text = null;

TextBoxAPassword.Text = null;

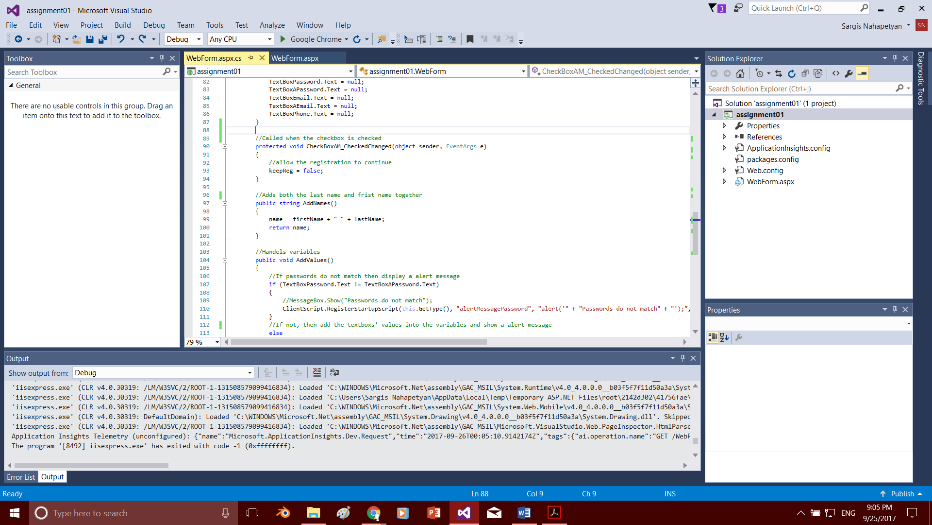
TextBoxEmail.Text = null;

TextBoxAEmail.Text = null;

TextBoxPhone.Text = null;

}

**CheckBoxAM\_CheckedChanged:**



The function is quite simple. The function is called whenever the checkbox is checked or unchecked, depending if it starts checked or not. The checkbox starts checked and will therefore be called whenever it is unchecked. Once unchecked it will set “keepReg” to false, which will disable the loop in ButtonReg\_Click and not allow it to keep registering.

//Called when the checkbox is checked

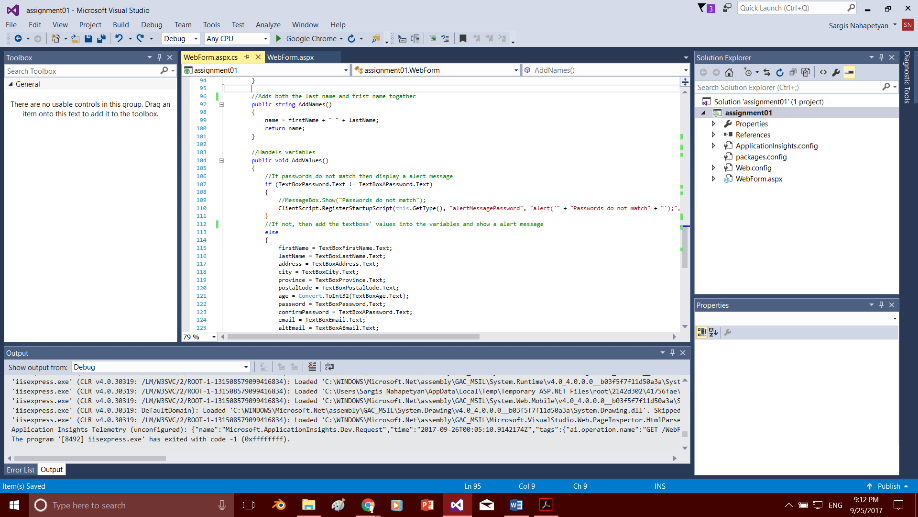
protected void CheckBoxAM\_CheckedChanged(object sender, EventArgs e)

{

//allow the registration to continue

keepReg = false;

}

 **AddNames:**

The function is only called once in the “AddValues” function in order to add the first and last name of the register together. The function takes “fristName” adds it to “lastName” and make them equal to “name”, “return” is responsible for returning the value. They are separated by a space in a middle to add a space in the name. The type of the function is a string, due to the return type also being a string. The function can also be used like a variable to make things simple.

//Adds both the last name and frist name togather

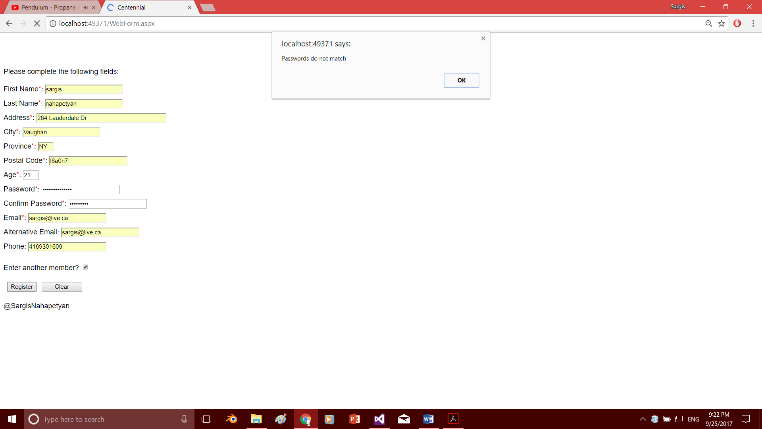
public string AddNames()

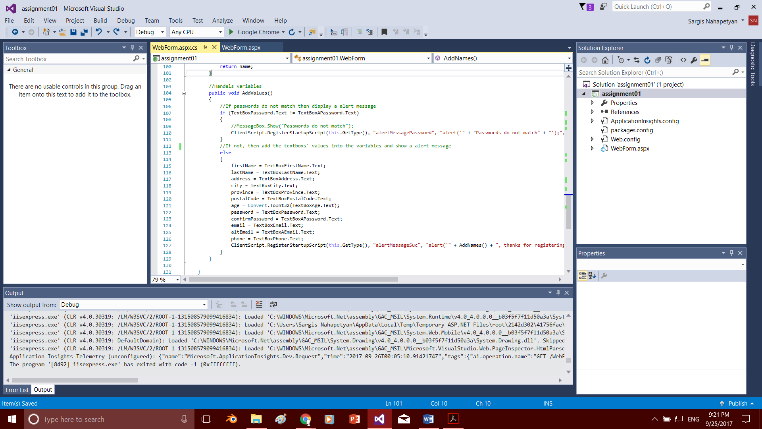
{

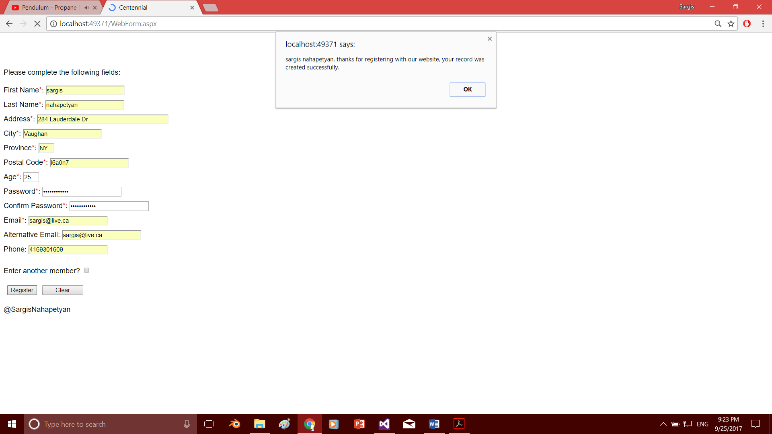
name = firstName + " " + lastName;

return name;

}

 **AddValues:**



The function is called by ButtonReg\_Click in order to validate the information and store them into variables. The code is in a function and is used twice as using the code itself will be too lengthy. It is also unnecessary to have the same unaltered code being used twice. When the function is called, it will first check if the values in “TextBoxPassword” and “TextBoxAPassword” are not the same. If so, it will display an alert message informing the user that the passwords do not match. If the values do match however, it will instead proceed to put the values in their proper variables. It will then display an alert message informing the user that the registration was successful, it will even use their full name thanks to “AddNames”.

//Handels variables

public void AddValues()

{

//If passwords do not match then display a alert message

if (TextBoxPassword.Text != TextBoxAPassword.Text)

{

//MessageBox.Show("Passwords do not match");

ClientScript.RegisterStartupScript(this.GetType(), "alertMessagePassword", "alert('" + "Passwords do not match" + "');", true);

}

//If not, then add the textboxs' values into the variables and show a alert message

else

{

firstName = TextBoxFirstName.Text;

lastName = TextBoxLastName.Text;

address = TextBoxAddress.Text;

city = TextBoxCity.Text;

province = TextBoxProvince.Text;

postalCode = TextBoxPostalCode.Text;

age = Convert.ToInt32(TextBoxAge.Text);

password = TextBoxPassword.Text;

confirmPassword = TextBoxAPassword.Text;

email = TextBoxEmail.Text;

altEmail = TextBoxAEmail.Text;

phone = TextBoxPhone.Text;

ClientScript.RegisterStartupScript(this.GetType(), "alertMessageSuc", "alert('" + AddNames() + ", thanks for registering with our website, your record was created successfully." + "');", true);

}

}