Form Validation

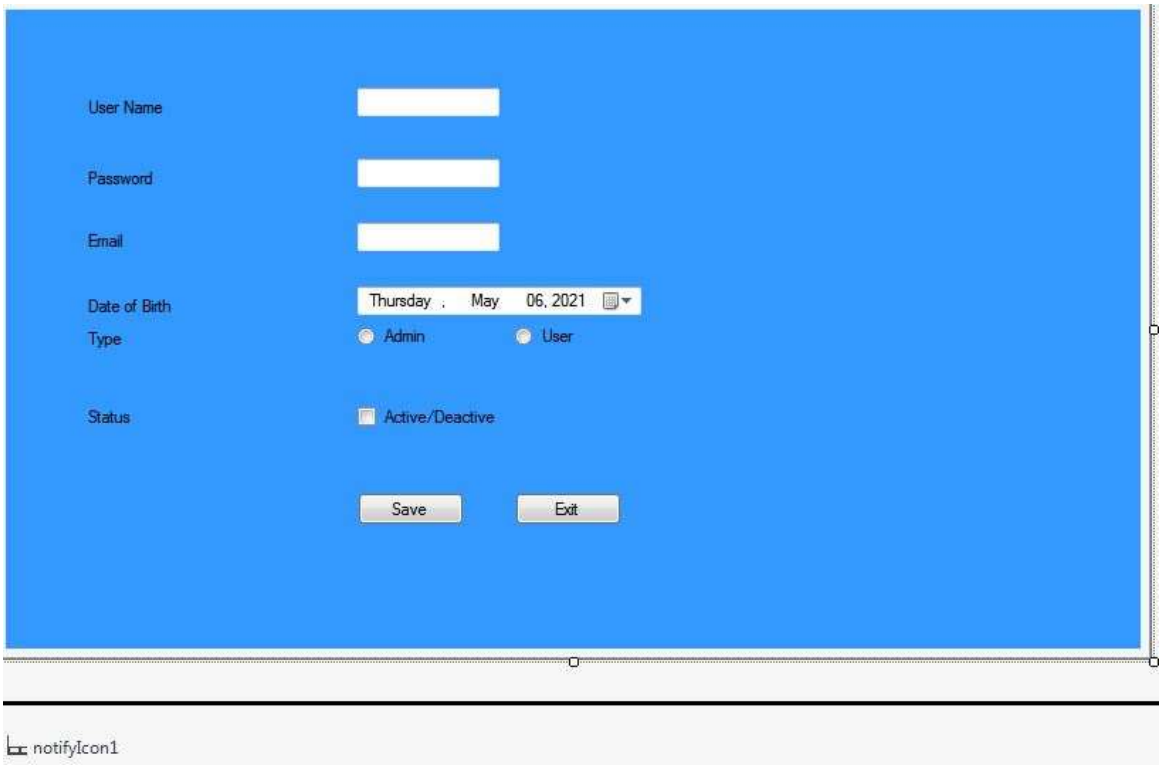
Form validation is a "technical process where a desktop-form checks if the information provided by a user is correct."

The form will either alert the user that they messed up and need to fix something to proceed, or the form will be validated and the user will be able to continue with their registration process. For example, Twitter won't let me use an email address that is incorrectly formatted, and gives me an error message when I try to do so:

But when I correctly enter my email address, it works, and the error message disappears:

Basically, validation makes sure that the provided text is in the right format (e.g., for email, user@example.com), and if the text fits the qualifications for a suitable entry (e.g., the email isn't already registered, or the password fits the criteria).

Now let's look at an example given below:



In this example we used three textboxes, one check box and one radio button, and date time picker, as these objects are mostly used in desktop applications. The message to the user is shown in two ways. The first one is to use the notifyIcon and the second one is the MessageBox.

notifyIcon:

The Windows Forms NotifyIcon component is typically used to display icons for processes that run in the background and do not show a user interface much of the time. An example would be a virus protection program that can be accessed by clicking an icon in the status notification area of the taskbar.

MessageBox:

MessageBox is a class in C# and Show is a method that displays a message in a small window in the center of the Form. MessageBox is used to provide confirmations of a task being done or to provide warnings before a task is done. We perform validation on these objects in different ways. The coding and basic concept of different ways of validation is as:

1. Leave:

Leave event occurs when the input focus leaves the control.

a. Display the output using notifyIcon.

```
private void textBox1_Leave(object sender, EventArgs e)
```

```
        {
            if (textBox1.Text.Length == 0)
            {
                notifyIcon1.ShowBalloonTip(200, "Name is not enter", " Please Enter
name", ToolTipIcon.Error);
                textBox1.Focus();

            }
        }
        private void textBox2_Leave(object sender, EventArgs e)
        {
            if (textBox2.Text.Length == 0)
            {
                notifyIcon1.ShowBalloonTip(200, "Password is not enter", " Please Enter
Password", ToolTipIcon.Error);
                textBox2.Focus();

            }
        }
```

b. Display the output using MessageBox.

```
private void textBox1_Leave(object sender, EventArgs e)
        {
            if (textBox1.Text.Length == 0)
            {
                MessageBox.Show("Please Enter your name");
                textBox1.Focus();

            }
        }
private void textBox2_Leave(object sender, EventArgs e)
        {
            if (textBox2.Text.Length == 0)
            {
                MessageBox.Show("Please Enter your password");
                textBox2.Focus();

            }
        }
```

2. Enter:

Enter event occurs when the control is entered.

```
private void textBox1_Enter(object sender, System.EventArgs e)
{
    // If the TextBox contains text, change its foreground and background colors.
    if (!string.IsNullOrEmpty(textBox1.Text))
    {
        textBox1.ForeColor = Color.Red;
        textBox1.BackColor = Color.Black;
        // Move the selection pointer to the end of the text of the control.
        textBox1.Select(textBox1.Text.Length, 0);
    }
}
```

3. KeyPress:

KeyPress occurs when the keys from the keyboard is pressed while the control has focus on the object.

```
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
        {
            if ((e.KeyChar >= '0' && e.KeyChar <= '9'))
            {
                e.Handled = true;
                notifyIcon1.ShowBalloonTip(200, "Numbers are not allowed", " Please Enter
alphabet", ToolTipIcon.Error);
            }

        }

private void textBox2_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!(e.KeyChar >= '0' && e.KeyChar <= '9'))
            {
                e.Handled = true;
                notifyIcon1.ShowBalloonTip(200, "Alphabets are not allowed", " Please
Enter number", ToolTipIcon.Error);
            }


        }
```

Save Button coding to control the radio button, check box, datetimepickerand validating email id.

```
private void button1_Click(object sender, EventArgs e)
        {

            if ((dateTimePicker1.Value.DayOfWeek == DayOfWeek.Sunday) ||
            (dateTimePicker1.Value.DayOfWeek == DayOfWeek.Saturday))
            {
                notifyIcon1.ShowBalloonTip(200, "Appointment cannot be scheduled in the
weekend.", "Please select a weekday", ToolTipIcon.Error);
            }
            if (!radioButton1.Checked && !radioButton2.Checked)
            {
                MessageBox.Show("Please check one radio button!");
            }
            if (!checkBox1.Checked)
            {
                MessageBox.Show("Please select a value from comboBox!");
            }
           if(!(textBox3.Text.IndexOf("@") > -1 && textBox3.Text.IndexOf(".") > -1))
            {
                    MessageBox.Show("enter a valid email address.");

                }


        }
```