

The Visual Basic Programming Language

Visual Basic is a high level language produced by Microsoft in 1991 as the first programming language that directly supported graphical user interfaces. From that time until now, there were many other versions released, each version having features that increased the power of the language. Visual Basic allows the programmer to use objects (tools) to accomplish a program's goal. Programs written for the Windows environment typically use objects such as labels, check boxes, list boxes, and buttons.

The Visual Studio integrated development environment (IDE)

An **integrated development environment (IDE)** is an environment that contains all of the tools and features you need to create, run, and test your programs. You also will use the IDE to create graphical user interfaces for your programs. A **graphical user interface (or GUI)** is what the person using your program (referred to as the user) sees and interacts with while your program is running. The user interface and its program instructions are referred to as an **application**. The Visual Studio IDE contains many different windows, each with its own special purpose. The five windows you will use most often when designing your user interfaces are shown in Figure 1-1.

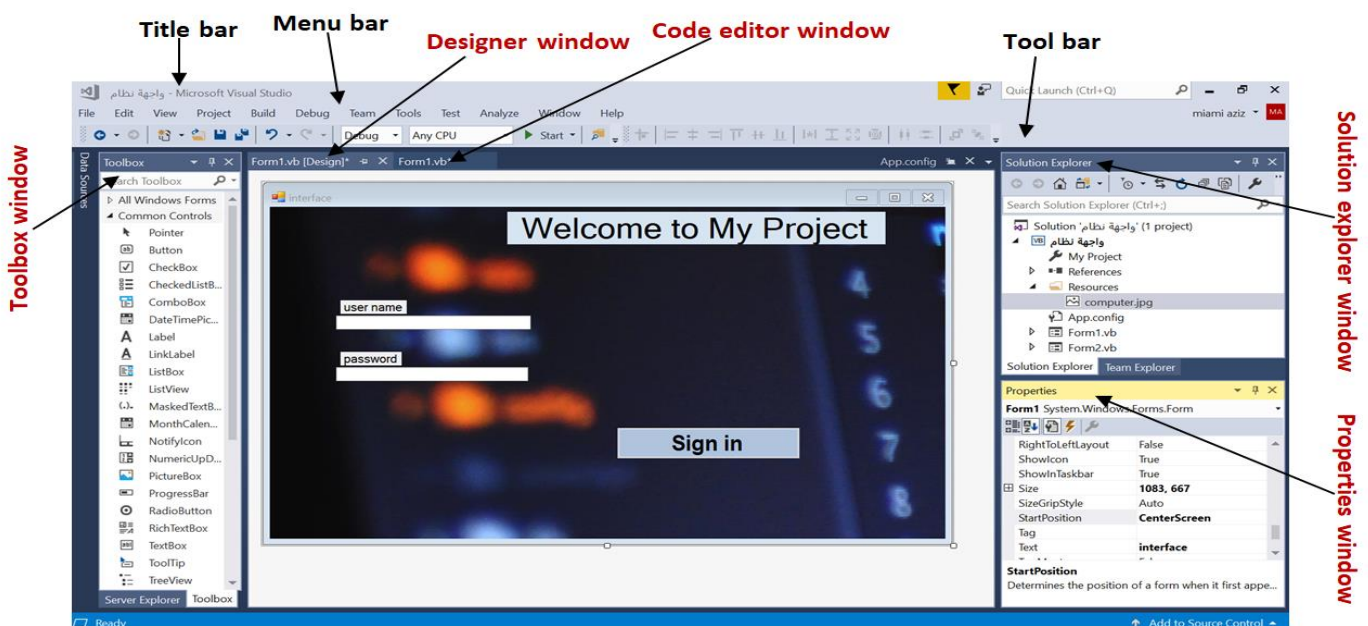


Figure 1-1 Visual Studio IDE

The **designer window** is where you create (or design) your application's GUI. A Windows Form object, or **form**, appears in the designer window shown in Figure 1-1. A **form** is the foundation for the user interface in an application created for the Windows environment. The form is automatically instantiated for you when you create a Windows Forms application in Visual Basic.

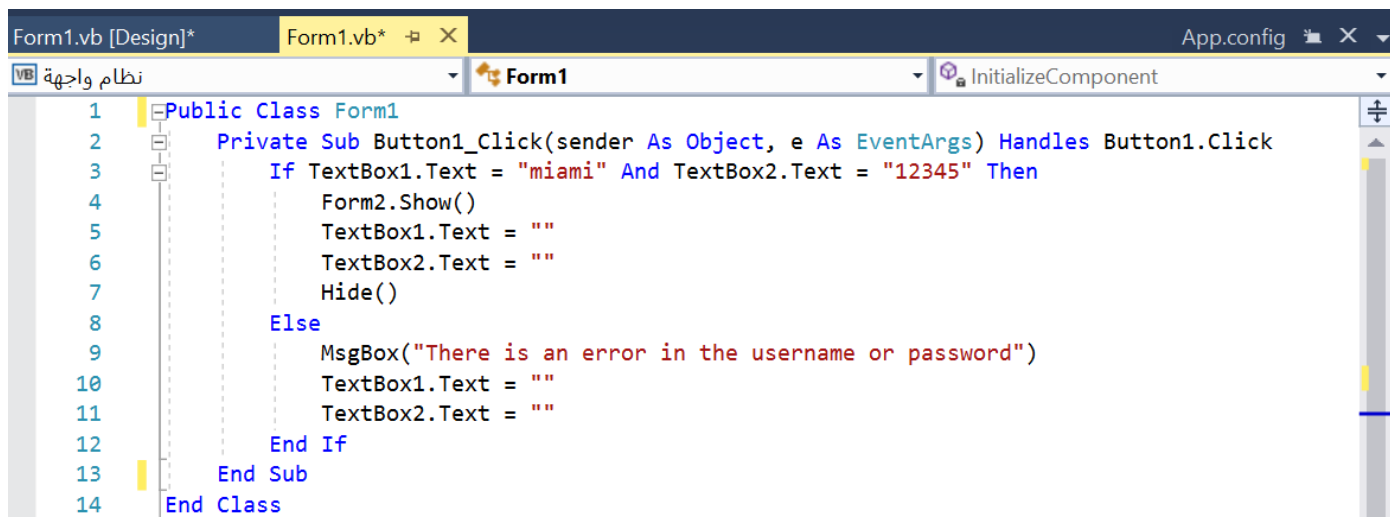
You use the **Toolbox window** to add other objects, called **controls (tools)**, to the form. You add a tool by clicking its corresponding tool picture in the toolbox and then dragging it with your mouse pointer to the form. For example, the two textbox objects shown in Figure 1-1 were instantiated (created) by dragging the TextBox tool from the toolbox to the form. Similarly, the button was instantiated using the Button tool.

Each tool has a set of attributes that determine its appearance and behavior. The attributes, called **properties**, are listed in the **Properties window** when the tool is selected in the designer window. In Figure 1-1, the form is selected, and the names of its properties (such as StartPosition and Text), along with their values (CenterScreen and interface), appear in the Properties window. You can use the Properties window to change the value of an tool's property. For example, you can use it to change the form's Text property, which appears in the form's title bar, from *interface* to *user interface*.

Windows applications in Visual Basic are composed of solutions, projects, and files. A solution is a container that stores the projects and files for an entire application. A project is also a container, but it stores only the files associated with that particular project. The **Solution Explorer** window displays a list of the projects contained in the current solution and the items contained in each project. The Solution Explorer window shown in Figure 1-1 indicates that the "واجهة النظام" Solution contains the "واجهة النظام" Project, which contains several items:

- ✓ The computer.jpg item is the name of the image that appear in the form background.
- ✓ The Form1.vb file stores the program instructions (**code**) that tell the button how to respond when the user click it.

You enter the code in the **Code Editor window**, which is shown in Figure 1-1. Code Editor window appears as soon as you click on any tool inside the form or click on the form itself. At this point, we are not expected to understand the contents of the Code Editor window in Figure 1-2; you will understand this code in the Upcoming lectures.



```
1 Public Class Form1
2     Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3         If TextBox1.Text = "miami" And TextBox2.Text = "12345" Then
4             Form2.Show()
5             TextBox1.Text = ""
6             TextBox2.Text = ""
7             Hide()
8         Else
9             MsgBox("There is an error in the username or password")
10            TextBox1.Text = ""
11            TextBox2.Text = ""
12        End If
13    End Sub
14 End Class
```

Figure 1-2 Code Editor Window

Create a Windows Forms Application

Our application will be a Windows Forms application, which is an application that has a Windows user interface and runs on a laptop (or desktop) computer.

To create a Windows Forms application:

1. Click **File** on the menu bar and then click **New Project** to open the New Project dialog box. If necessary, click the **Visual Basic** node in the Installed Templates list, and then click **Windows Forms App (.NET Framework)** in the middle column of the dialog box.
2. Change the name entered in the Name box to **any name you want**.
3. Click the **Browse** button to open the Project Location dialog box. Locate and then click the **wanted** folder which you want to use it in saving. Click the **Select Folder** button to close the Project Location dialog box.
4. If necessary, select the **Create directory for solution** check box in the New Project dialog box. Change the name entered in the Solution name box to **any name (if you want)**. Figure 1-3 shows the completed New Project dialog box in Visual Studio Community 2017.

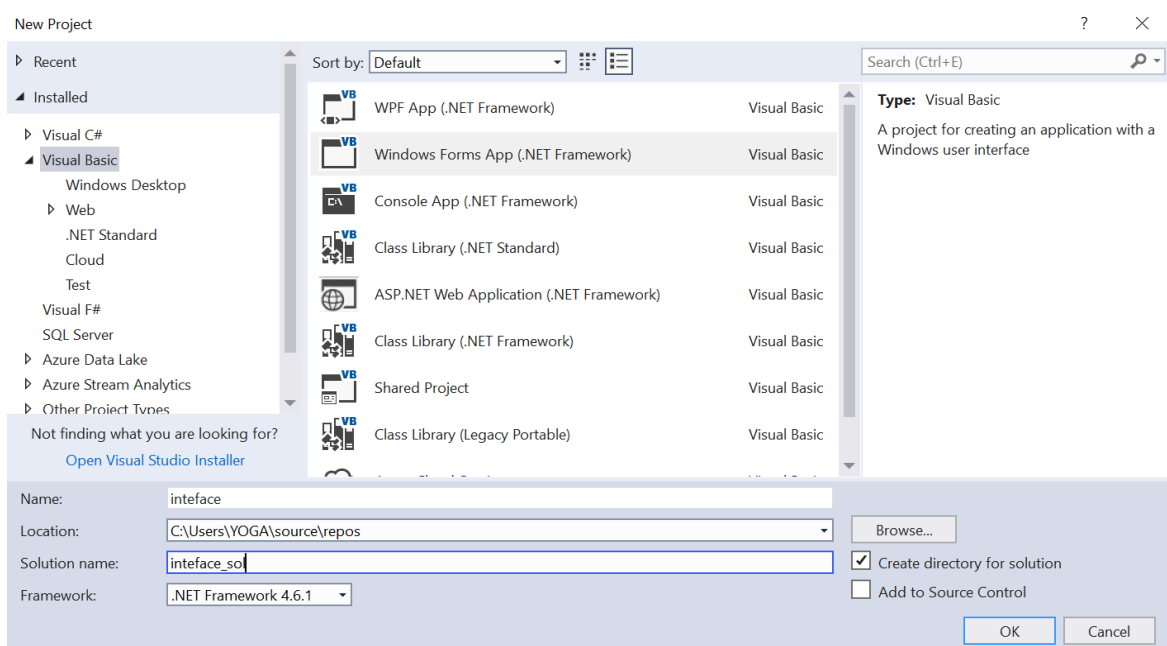


Figure 1-3 Completed new project dialog box

5. Click the **OK** button to close the New Project dialog box. The computer creates a solution and adds a Visual Basic project to the solution. The names of the solution and project, along with other information pertaining to the project, appear in the Solution Explorer window. Visual Basic also automatically creates a form object, which appears in the designer window. See Figure 1-4.

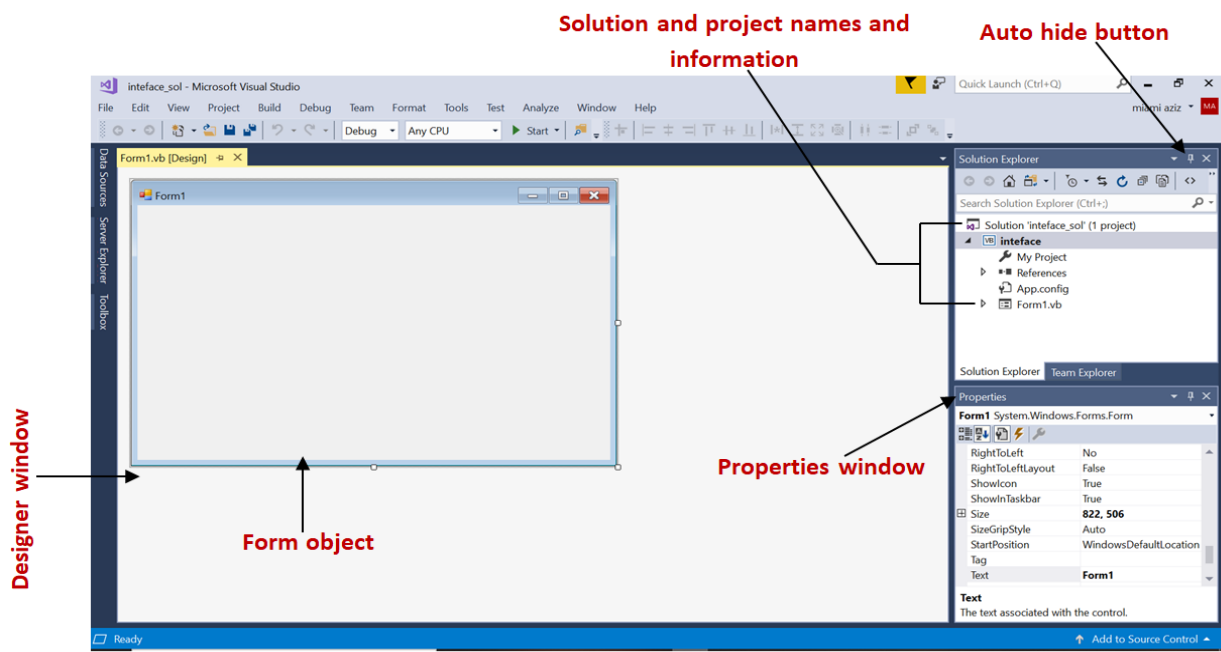


Figure 1-4 Solution and Visual Basic project

Manage the Windows in the IDE

In most cases, you will find it easier to work in the IDE if you either close or auto-hide the windows you are not currently using. You close an open window by clicking the Close button on its title bar. You will find the options for opening a closed window on the View menu. You auto-hide a window by using its Auto Hide button, which is located on the window's title bar. The Auto Hide button is a toggle button: Clicking it once activates it, and clicking it again deactivates it. The Toolbox and Data Sources windows are auto-hidden windows.

To close, open, auto-hide, and display windows in the IDE:

1. Click the **Close** button on the Properties window's title bar to close the window. Then click **View** on the menu bar and click **Properties Window** to open the window.
2. Click the **Auto Hide** (vertical pushpin) button on the Solution Explorer window. The Solution Explorer window now appears as a tab on the edge of the IDE.
3. To permanently display the ToolBox window, click the ToolBox tab and then click the Auto Hide (horizontal pushpin) button on the window's title bar.
4. To temporarily display the Solution Explorer window, click the **Solution Explorer** tab. Notice that the Auto Hide button is now a horizontal pushpin rather than a vertical pushpin. To return the Solution Explorer window to its auto-hidden state, click the **Solution Explorer** tab again.
5. To permanently display the Solution Explorer window, click the **Solution Explorer** tab and then click the **Auto Hide** (horizontal pushpin) button on the window's title bar. The vertical pushpin replaces the horizontal pushpin on the button.
6. In order to reset the window layout to the original shape, we go to the **Window** on the menu bar and click **reset window layout**