Appendix A
Web Application Development

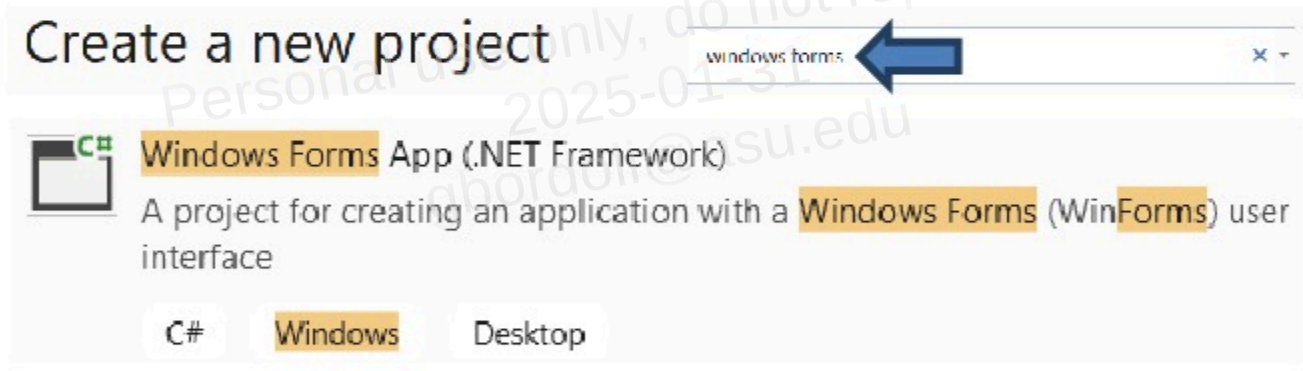
This appendix discusses the graphical user interface design and application development using web services discovered on the Internet.

A.1 Design of Graphical User Interface

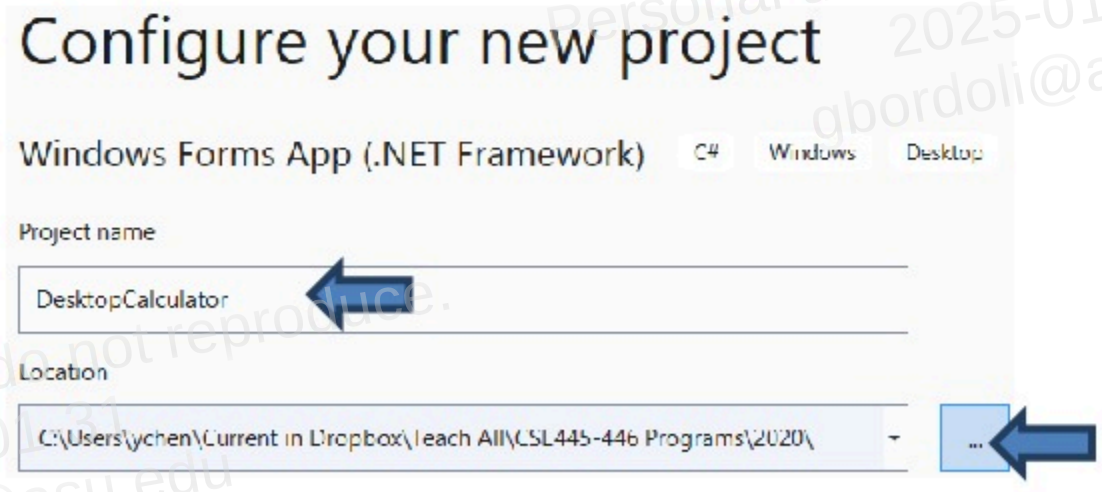
The difference between an application and a service or a software component is that the former has a human user interface, also known as GUI (Graphical User Interface) if graphical representation is used in the interface design; and the latter has a programming interface, also known as Application Programming Interface (API). This section discusses GUI design in .Net.

In Visual Studio programming environment, you can create an application with text-based user interface (Console Application) and with GUI (e.g., Windows Forms Application). In both cases, the application will run on a Windows computer. You can also create a web application or website application. In this case, the application will be deployed to a web server, such as a Windows server with IIS. Chapter 3, Section 3.4.1 discussed the process of creating and deploying a website Application. This section will discuss the Windows Forms Application.

To create a Windows Forms Application, start Visual Studio (2019) and create a new project by searching "windows forms":



Click "Next" button and choose a project name and a location for the project. Note, do not use the default location is Visual Studio folder, which can cause a permission problem when you run the program.



Then, a Form or a Design surface that can be used for creating a GUI will be opened, as shown in Figure A.1.

In Figure A.1, the following components and tools are available for you to build your Windows applications:

- **GUI/Code Editor:** This is a shared window that allows you to create your GUI, as well as enter your program code that is linked to the GUI.
  (1) GUI: When you click the "Form1.cs" in the "Solution Explorer," you will see the blank Designer Surface, on which you can draw your interface items. You can drag-and-drop the items in the toolbox on the left of the window to the Design surface.
  (2) When you double-click any of the items that you have drawn, the code editor behind the item will be opened, with the code prototype (the first line of the method) generated, and you can enter your C# program in the code editor. All Visual C# programs have the .cs extension.
- **Solution Explorer:** It offers you an organized view of your project, its files, and programs created by the system and by you, as well as ready access to the commands that pertain to them. You can add new projects into the solution. This is particularly convenient to organize multiple projects in one solution, if the programs in the solution need to communicate with each other. For example, if you create a number of web services and an application that uses the services, the application can access the services without having to deploy the services to IIS or a server. You can deploy the entire solution, which becomes a namespace, whereas the classes in the projects become the classes in the namespace.
- **Toolbox:** On the left side of the window, you can move your mouse over the Toolbox tab to open the Toolbox window if it is not open. Click the plus (+) sign next to the Common Controls. You will see a list of form controls that are seen in a Windows application. From the Toolbox, you can find the components (items) needed in your GUI, such as Button, RadioButton, Label, TextBox, CheckBox, and so on.
- **Properties:** They allow you to customize each of the GUI components (items) that you have drawn. This includes the name, color, text, font type, and font size.

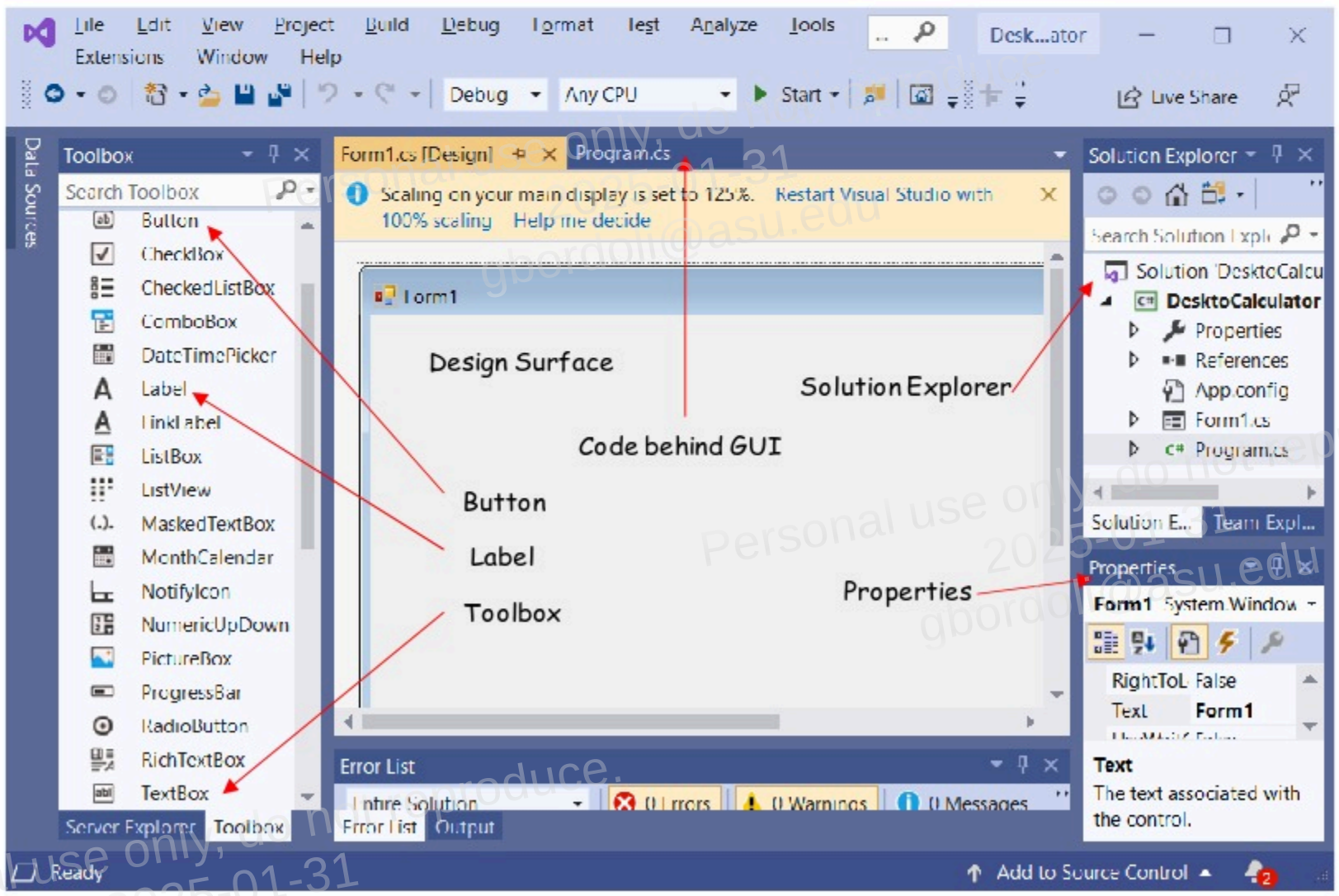


Figure A.1. Design panel of a GUI application

Create your own calculator

As an example, we will create a simple calculator, as shown in Figure A.2, which consists of two TextBoxes, four Buttons, and two Labels. From the two text boxes, you can enter two numbers. When you click one of the buttons, the operation on the button, +, -, *, and / will be performed on the two numbers. There are two Labels used in the GUI. The Label marked "=" simply displays the information. The second Label marked "Show result here" holds the place for displaying the result of the calculation.

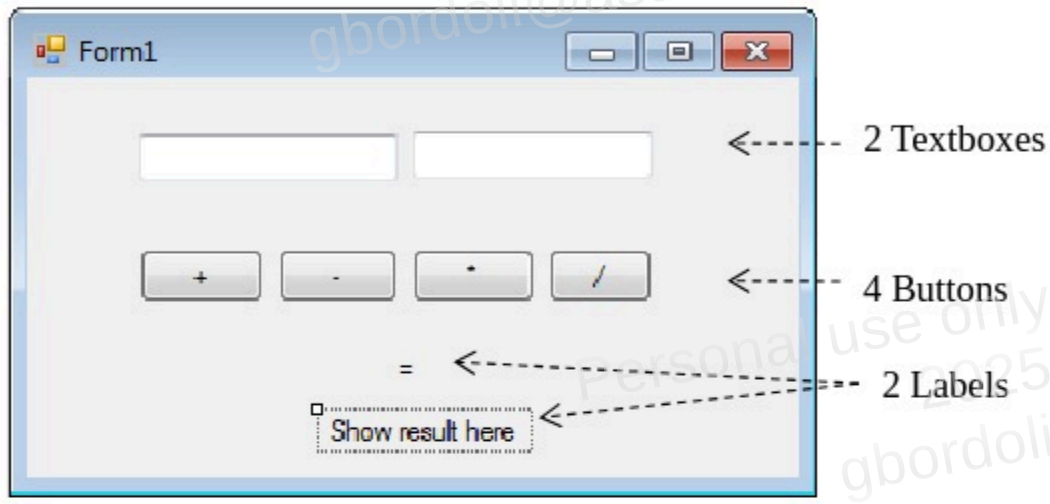


Figure A.2. GUI design of a simple calculator

Now we will write C# code behind the GUI to perform the computation needed. Since each button represents a computing task, we will link a method behind each button. By double-clicking on a button, the prototype of the method will be opened in the Code Editor, shown as follows, assuming "+" button is double-clicked.

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

Now, you can add your code in the prototype to perform the required operation. The operands come from the two Textboxes. By default, they are named textBox1 and textBox2. You can find the names in the Properties field of the project. Thus, the operation should look like:

```
result = textBox1 + textBox2
```

The problem is that anything entered from a textbox will have a type of string. Thus, we need to convert them to the integer type, as shown in the code below:

```
Int32 number1 = Convert.ToInt32(textBox1.Text);
Int32 number2 = Convert.ToInt32(textBox2.Text);
Int32 result = number1 + number2;
```

Next, we want to display the result to the human user. There are different ways to display the result. First, we can use a MessageBox to pop up a window to show the result. Thus, we can use the following code to implement this solution:

```
private void button1_Click(object sender, EventArgs e)
{
     Int32 number1 = Convert.ToInt32(textBox1.Text);
     Int32 number2 = Convert.ToInt32(textBox2.Text);
     Int32 result = number1 + number2;
     MessageBox.Show("The sum of " + textBox1.Text + " and " +
          textBox2.Text + " is " + result.ToString());
}
```

If 23 and 47 are entered and button "+" is clicked, the result will be given in a pop up window shown in Figure A.3.

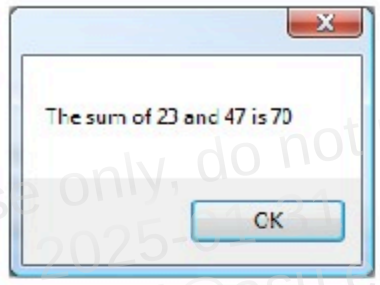


Figure A.3. Using a MessageBox to display result

If we want to display the result in the GUI in the place where the Label is marked "Show result here," we need to double-click the Label, which creates a prototype shown in the last three lines of the code below. This operation is necessary to link a GUI item drawn on the Design surface to the code behind the GUI. Next, we look up the name of the Label in the Properties field, which is label2. Then, we change the output code at line 6, and we can send the result to label2.Text.

```
private void button1_Click(object sender, EventArgs e)
{
     Int32 number1 = Convert.ToInt32(textBox1.Text);
     Int32 number2 = Convert.ToInt32(textBox2.Text);
     Int32 result = number1+number2;
     label2.Text = result.ToString();
}
private void label2_Click(object sender, EventArgs e)
{
}
```

With the changes, the result will be displayed in the GUI, as shown in Figure A.4.

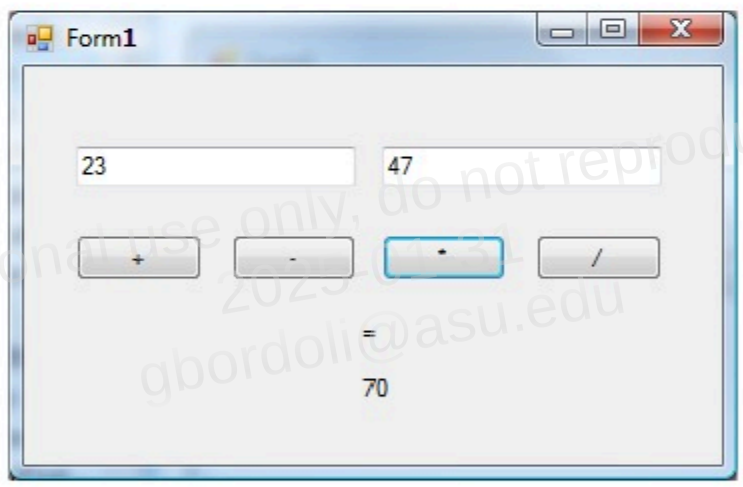


Figure A.4. Display the result in GUI marked a label

We still need to program the other three buttons -, *, and / to make a complete calculator application. The code that follows shows the complete program. Although the entire program is organized in a sequence of methods, you cannot simply copy the entire code into the code editor. The GUI items will not be able to find the methods linked to them. You must click each button and use the system-generated prototype to start with your method.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
```

```
using System.Windows.Forms;
namespace myGUI {
    public partial class Form1 : Form {
        public Form1(){ InitializeComponent(); }
        private void button1_Click(object sender, EventArgs e) {
            Int32 number1 = Convert.ToInt32(textBox1.Text);
            Int32 number2 = Convert.ToInt32(textBox2.Text);
            Int32 result = number1 + number2;
            label2.Text = result.ToString();
        }
        private void button2_Click(object sender, EventArgs e) {
            Int32 number1 = Convert.ToInt32(textBox1.Text);
            Int32 number2 = Convert.ToInt32(textBox2.Text);
            Int32 result = number1 - number2;
            label2.Text = result.ToString();
        }
        private void button3_Click(object sender, EventArgs e) {
            Int32 number1 = Convert.ToInt32(textBox1.Text);
            Int32 number2 = Convert.ToInt32(textBox2.Text);
            Int32 result = number1 * number2;
            label2.Text = result.ToString();
        }
        private void button4_Click(object sender, EventArgs e) {
            Int32 number1 = Convert.ToInt32(textBox1.Text);
            Int32 number2 = Convert.ToInt32(textBox2.Text);
            Int32 result = number1 / number2;
            label2.Text = result.ToString();
        }
    }
}
```

You can create an executable file by choosing “Build” → “Batch build.” Then, check the “release” box. This option of the compilation will generate the .exe file in the bin folder. The executable file can run on a different computer. Now, you can generate the executable for your calculator and send it to your friend for testing.

Create your own web browser

What browser do you use? Edge, Chrome, or Firefox? How about creating your own web browser? Now we show you another application that allows you to create your own web browser in a few simple steps.

1. Start Visual Studio by clicking Start → All Programs → Microsoft Visual Studio.Net.
2. Create a new project by clicking on File → New → Project.
3. From the New Project window, create a new Visual C# Windows Forms Application (.Net Framework) and name the application “JohnDoesBrowser.” You can use your name.
4. On the new created project, select the “Form1”, in the Properties window on the right side and modify the following properties using the following values:
    a. Text → John Doe’s Browser
    b. Size → 720, 640 (Width, Height)

    As shown in Figure A.5.

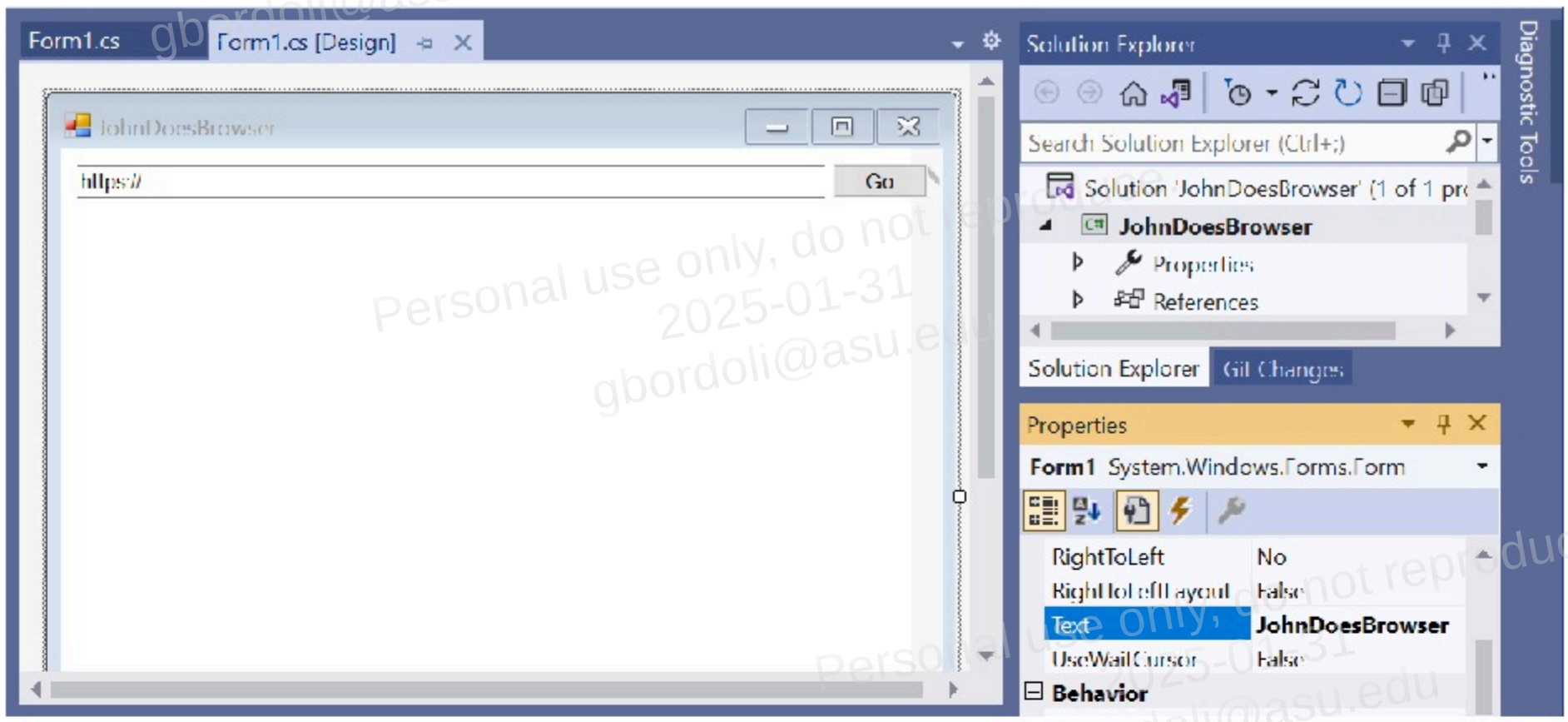


Figure A.5. Web browser GUI design

5. From the Toolbox, drag-and-drop the GUI item "WebBrowser" onto the design surface. The web browser control will fill the design surface completely. If you do not want the content area of your browser to fill the entire browser window, you can click the smart tag located on the top-right corner of the web browser control and select "Undock in parent container." Then, you can select the web browser control and expand the area so that it occupies almost the entire designer space. Make sure to leave room at the bottom for the URL address and the Go button, as shown in Figure A.6.

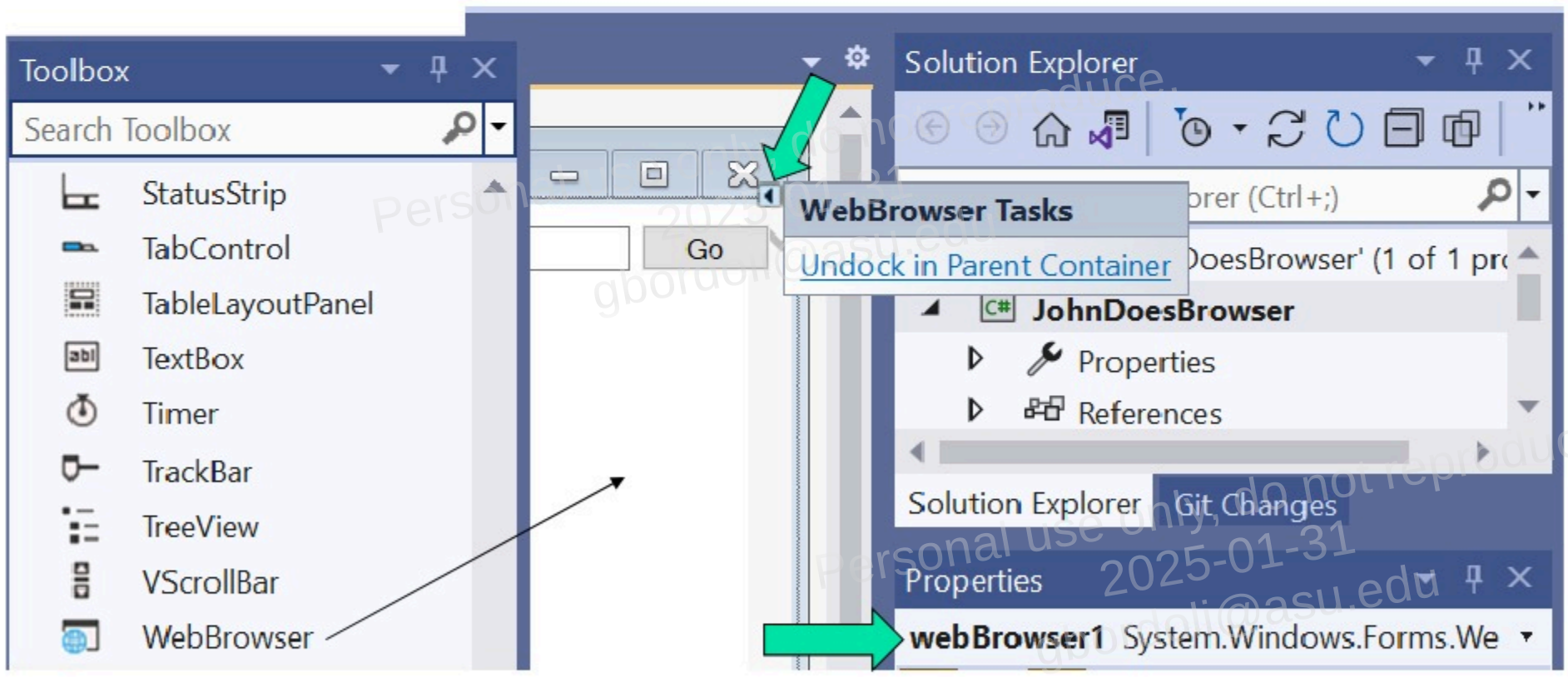


Figure A.6. After adding WebBrowser control, unlock in Parent Container

6. Drag-and-drop a Textbox and a Button from the Toolbox onto the Design surface. The textbox will be used to enter the URL for your browser, and the button will be used for invoking the web page. Place them on the top or bottom, as you wish. Change the properties of the controls using the following values:

a. Textbox: (Name) → txtUrl

b. Textbox: Text: http://

c. Size the Textbox wide enough, so that it can fit in most URLs

d. Button: Text → Go, and (name) → btnGo

7. Now, you can link the code behind the button "Go," by double-clicking on the button and it will take you to the code area. Add one line of code (highlighted) in the prototype, shown as follows:

```
        private void btnGo_Click(object sender, EventArgs e) {
webBrowser1.Navigate(txtUrl.Text); // Add this line of code
        }
```

8. Compile and execute your application by pressing Ctrl+F5 (or use menu command). Your own web browser is ready to take you to any URL you enter, as shown in Figure A.7

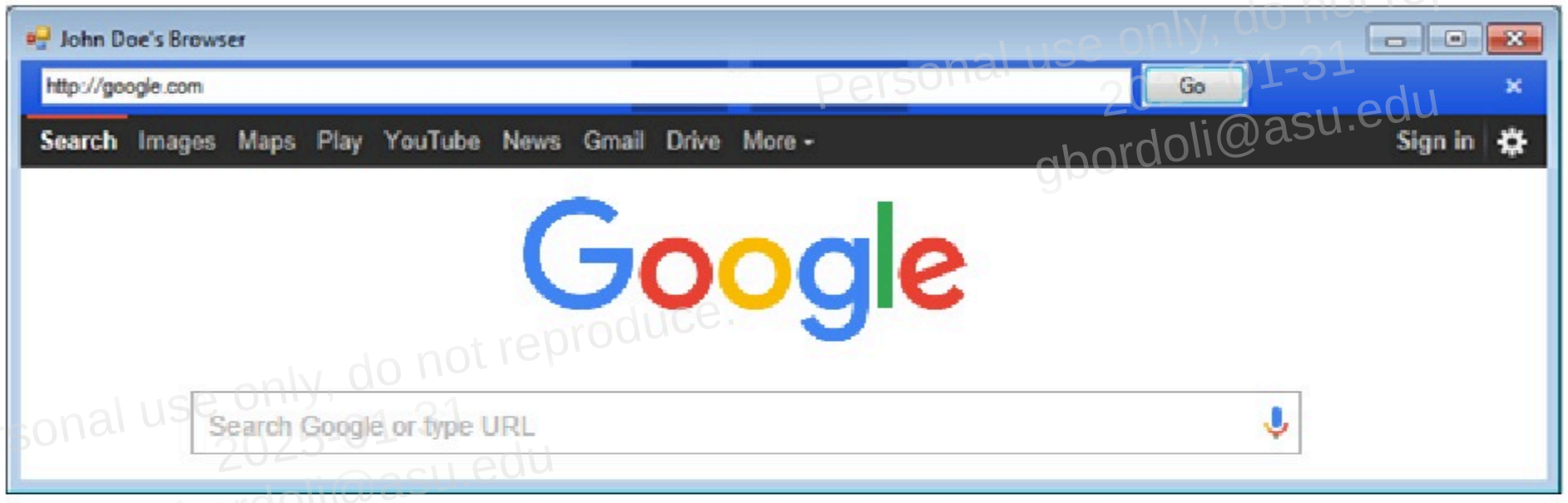


Figure A.7. John Doe's web browser GUI

Choose "Build" → "Batch build," and check the "release" box to generate the .exe file to run on a different computer. Send your browser to your friend for testing.

You can add many other features to your browser. For example, you build a simple calculator in your browser, which allows you to do calculation while reading the web page. In the later sections, we will show you how to add live weather forecast, currency exchange rate, and so on, into your Windows applications such as the web browser.

A.2 Discovering Web Services Available Online

Before we start to develop a software application, we first review the SOC development process shown in Figure A.8 from the application builder's perspective. Instead of trying to develop all the components, the application builder tries to discover the available services from the service brokers. There are many websites that publish services for the public to use, including:

- The government services. The U.S. government maintain and offer many types of web services, including:
  - The U.S. Government Open Data Services: https://data.gov/
  - National Digital Forecast Database (NDFD): https://graphical.mdl.nws.noaa.gov/xml/
  - National Geophysical Data Center Web Services: https://graphical.weather.gov/xml/
- Amazon Web Services for developers (https://aws.amazon.com/developer/): including services for education: https://aws.amazon.com/education.

- Google Web Services (http://www.google.com/apis/index.html): including search services and maps services.
- Microsoft web services, including:
  - SOAP services http://msdn.microsoft.com/en-us/library/cc966738.aspx and
  - RESTful services http://msdn.microsoft.com/en-us/library/ff701722.aspx.
- GoeNames web services (https://www.geonames.org/export/ws-overview.html), including different types of geographical services.
- Rapid API services: https://rapidapi.com/, including SOAP and RESTful services related to city data, flight data, sport data, medical data, and weather data.
- Flight booking services: http://ws.51book.com:8000/ltips/services/

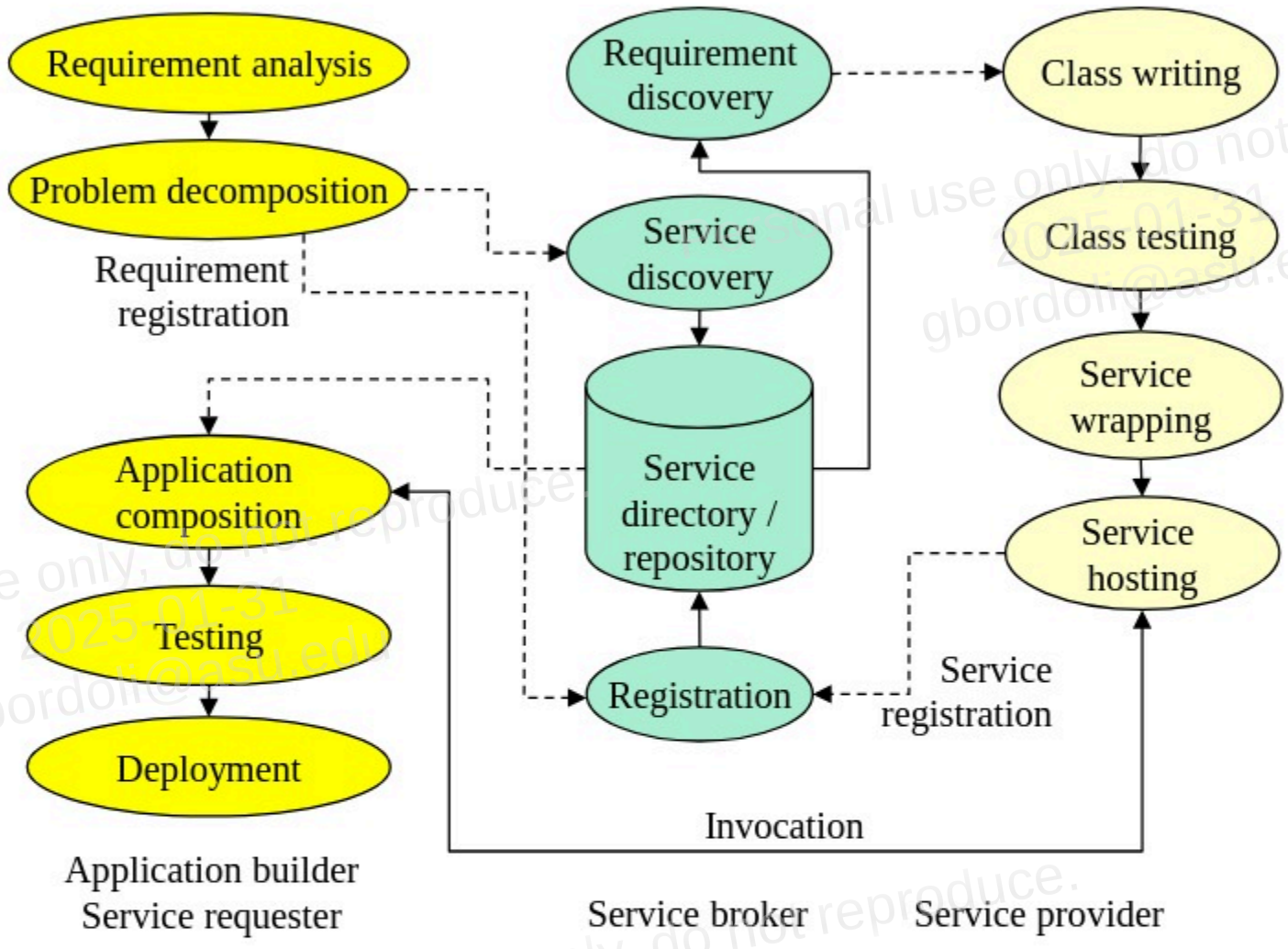


Figure A.8. Service-oriented development process with three independent parties

A.3 Access Web Services in Your Application: ImageService

This section shows how an application builder makes use of the remote services to create a website application that provides a GUI for accessing the web services in a human user-friendly interface.

In this section, we create an application program that allows a human user to enter an integer as the length of a random image string, and then, the user must type the same string shown in the image. The application will validate if the entered string matches the image showing in the image. This is a typical image verifier to prevent the denial of service.

Open Visual Studio and create a New Project. Select “Visual C#” and “Web” as your template. Select “ASP .Net Web Application” as your project type and then further select “Empty” so that unnecessary files are not added into the project. Name this project “ImageServiceTryIt” and then click OK, as shown in Figure A.9

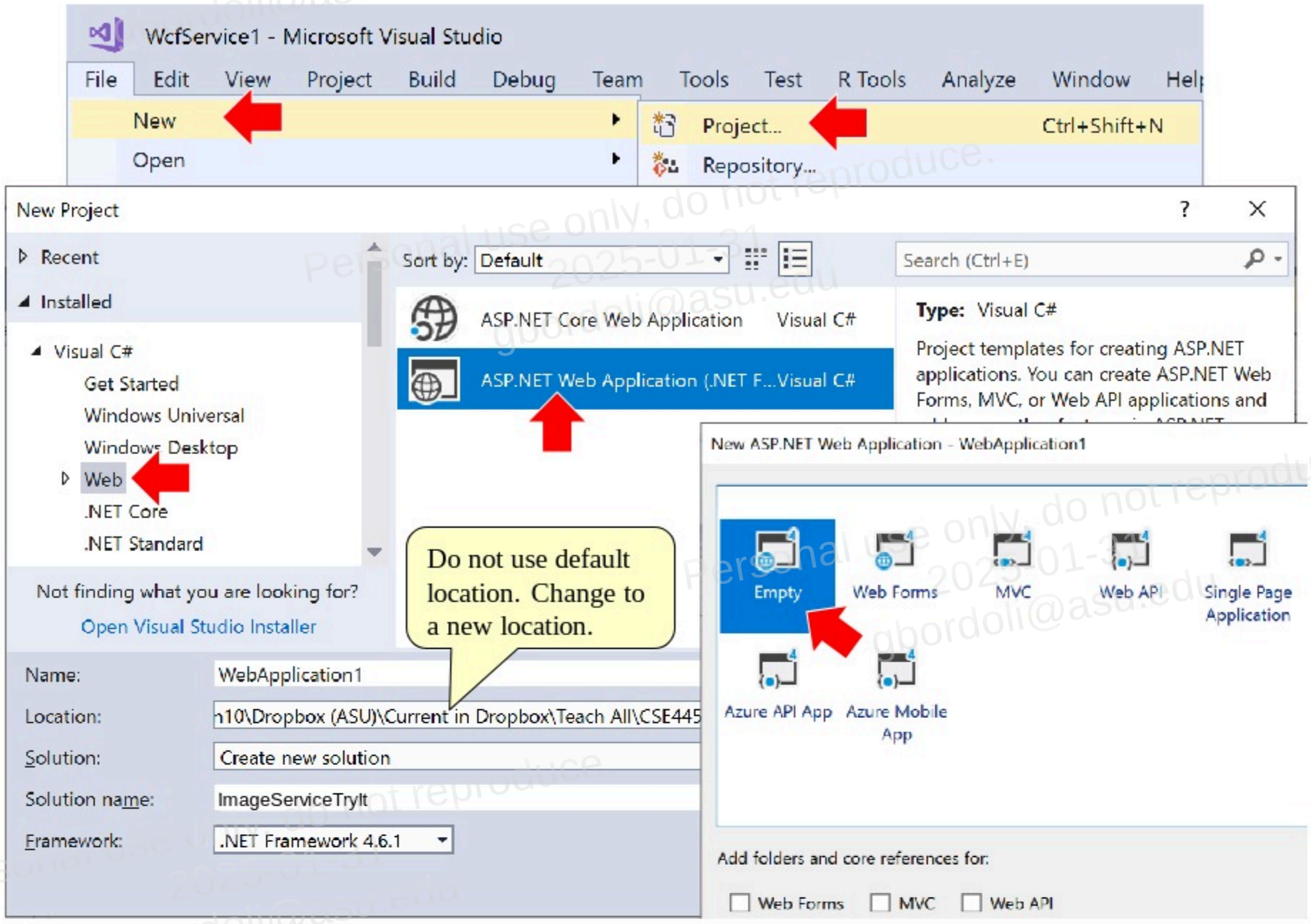


Figure A.9. Creating an ASP .Net Empty application

Then, right-click the project name and chose Add New Item. Then, select C# - Web and Web Form, as shown in Figure A.10.

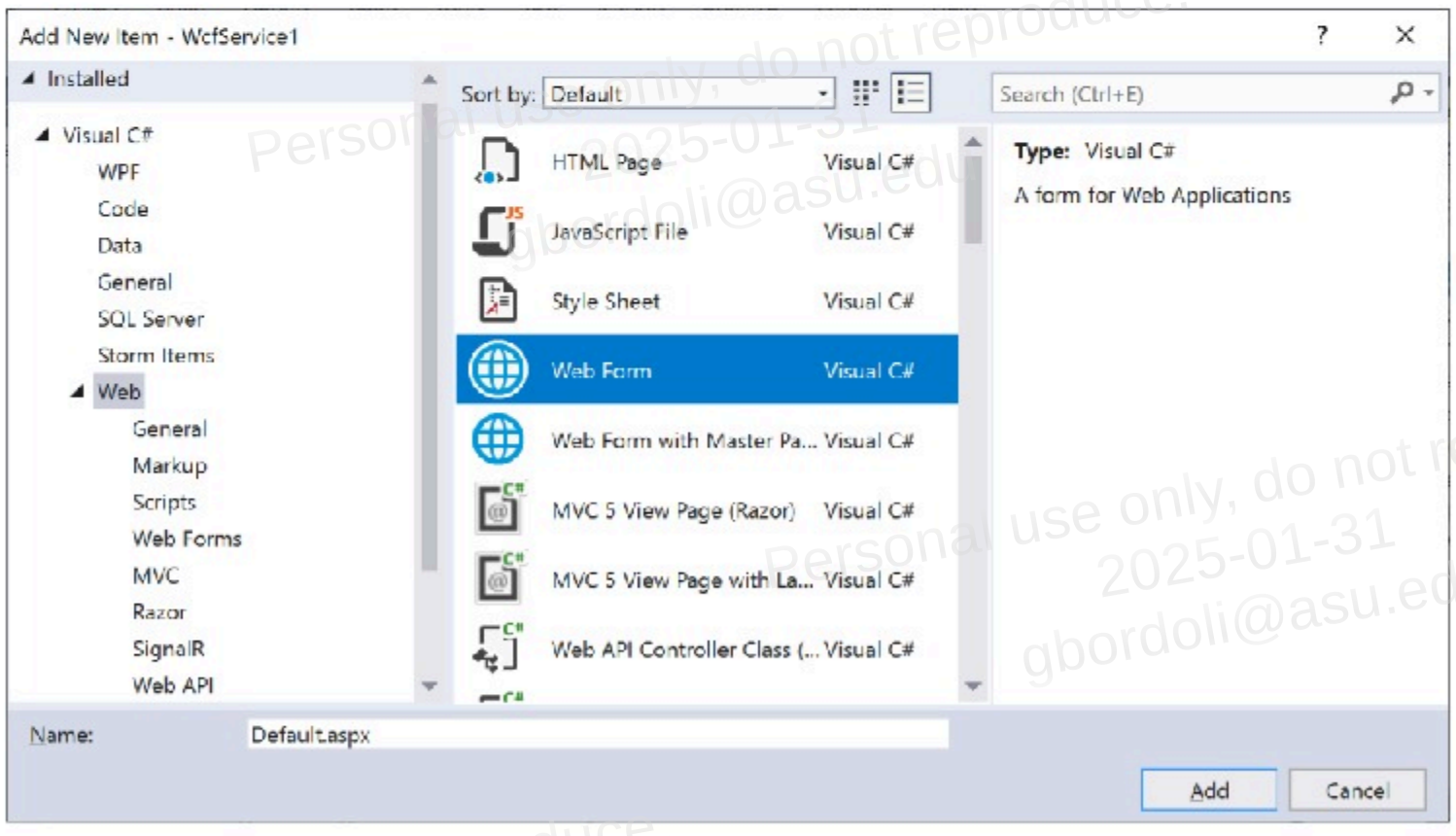


Figure A.10. Adding a Web Form into a project

The Default.aspx page is added, which provides a design surface for GUI design, as shown in Figure A.11. One can use the GUI components in the "Toolbox" to draw different type of GUI components, including Textbox for input, RichTextBox for output, Label for showing text on the GUI, RadioButton for selection, ComBox for a dropdown selection list, and "Button" for mouse click input; for example, invoking a function.

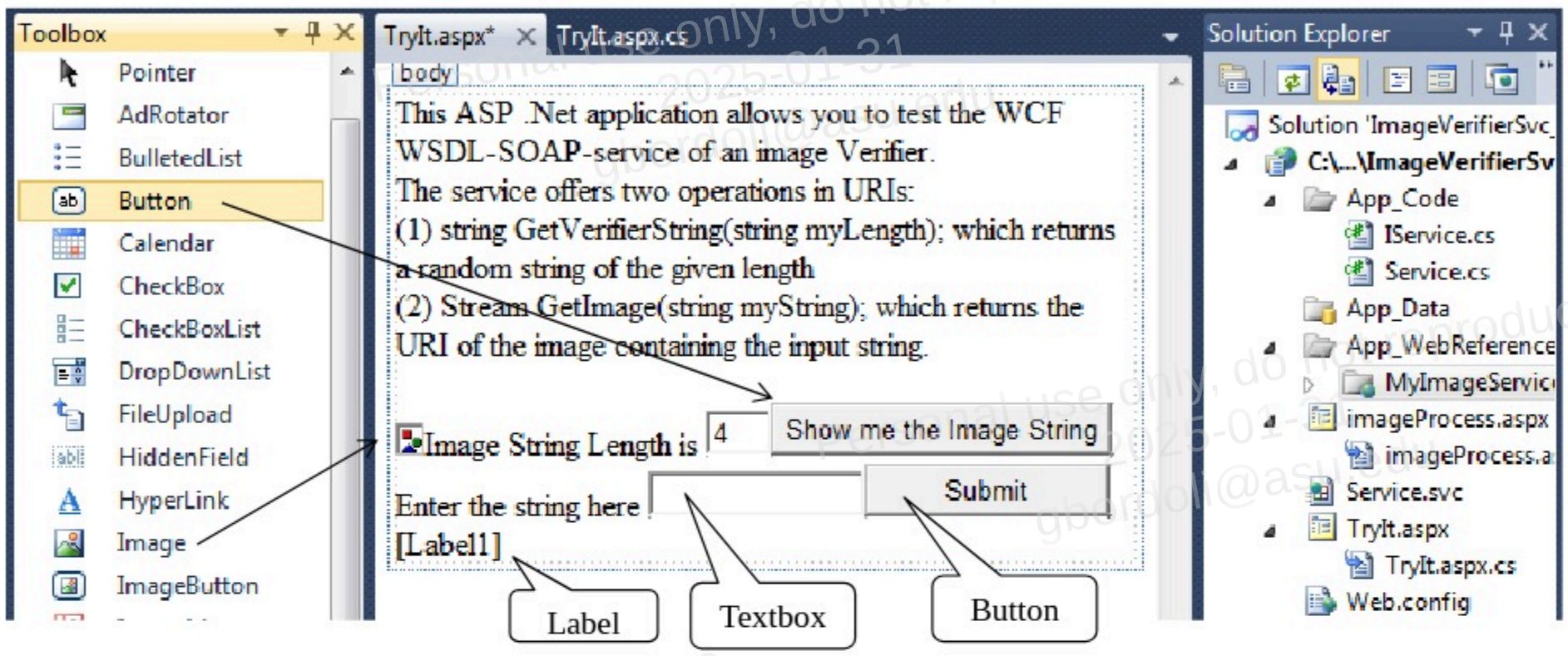


Figure A.11. Design the GUI using items in Toolbox

Before we continue to design the GUI and write C# code to link the GUI to the functions needed, we first locate the image service that generates a random string and then generate an image from the string. We have developed and deployed the service into ASU Services and Applications repository in Table C.2. Once the service is found, you can add WSDL file's URL as web service into your application, so that your program can access the web service remotely from the program. In most cases, the service is not located in the service directory's location. It is located in the service provider's site. Note that many services listed public service directory, such as Xmethods.net, are not free of charge. If charge is required, you will be asked to sign up the service before using it. The examples we use in the text are all free services.

Step 1: Locating the image web service

1) To begin, go to text Append C or http://venus.sod.asu.edu/WSRepository/;
2) Copy the service address from the page, which is:

   https://venus.sod.asu.edu/WSRepository/Services/ImageVerifierService/Service.svc

   You will need this address to bind and to invoke the web service.

Step 2: Adding web services to your application.

1) Return to Visual Studio .NET environment. Choose the menu "View" and choose the Solution Explorer. Right-click the folder "Reference" and choose "Add Service References… " A window will pop up, allowing you to enter the URL of the web service, as shown in Figure A.12.
2) Paste the WSDL URL of the movie service into the Address part, and click "Go." Once the service has been found, rename the Namespace to "proxyTheaterLocator" and click "OK."

3) Verify that the reference has been added successfully by opening the Solution Explorer (press Ctrl+Alt+L) and viewing the project node. You should see a Service References directory with a reference called "MyImageService."

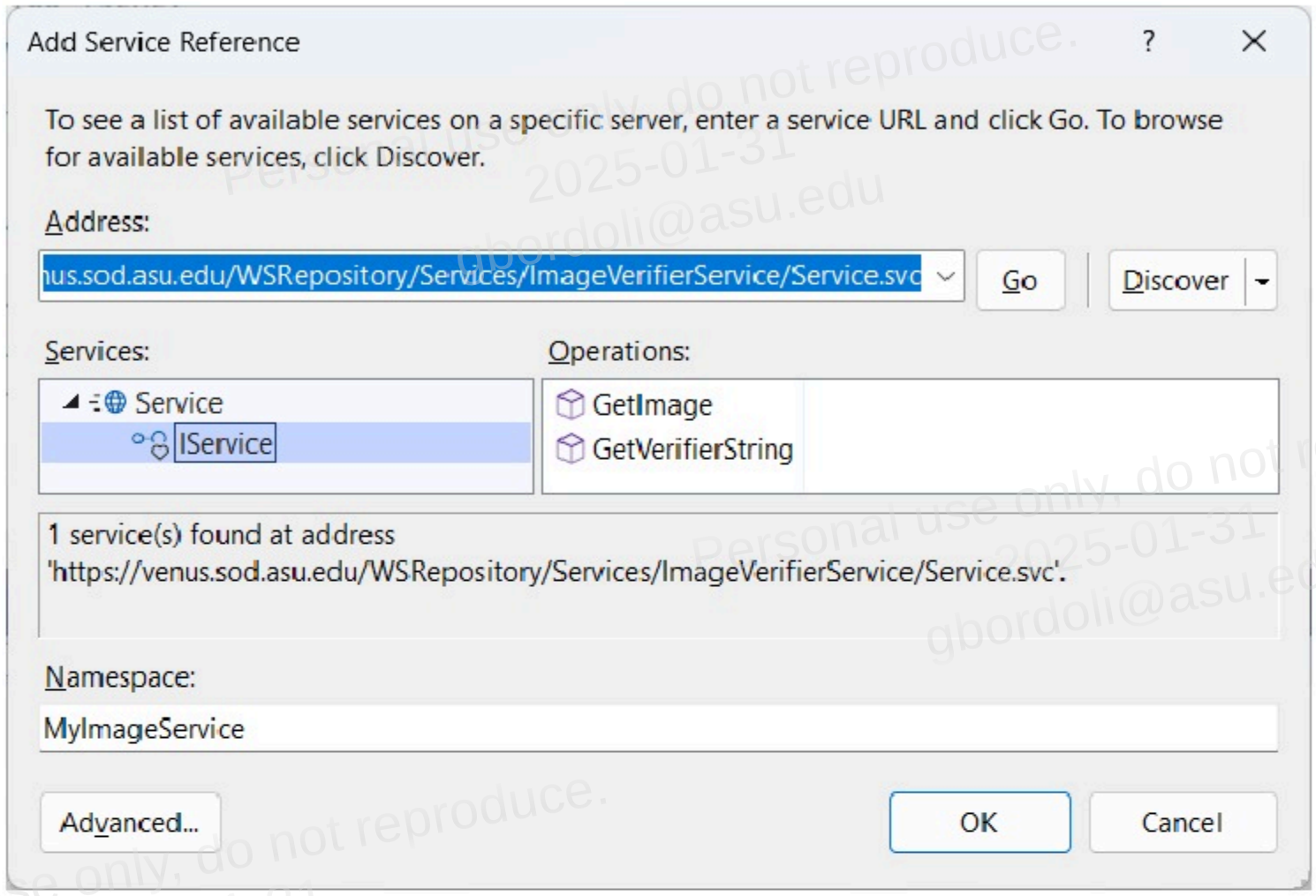


Figure A.12. Adding a service reference into the application

Step 3: Now let us follow Figure A.11 to design the GUI using the items in the toolbox: Image, Textbox, Label, and Button. After you have designed the GUI, you can switch the "Design" (graphic) view to the "Source" view. You can edit the design from both the Design view and the Source view, as shown in Figure 13.

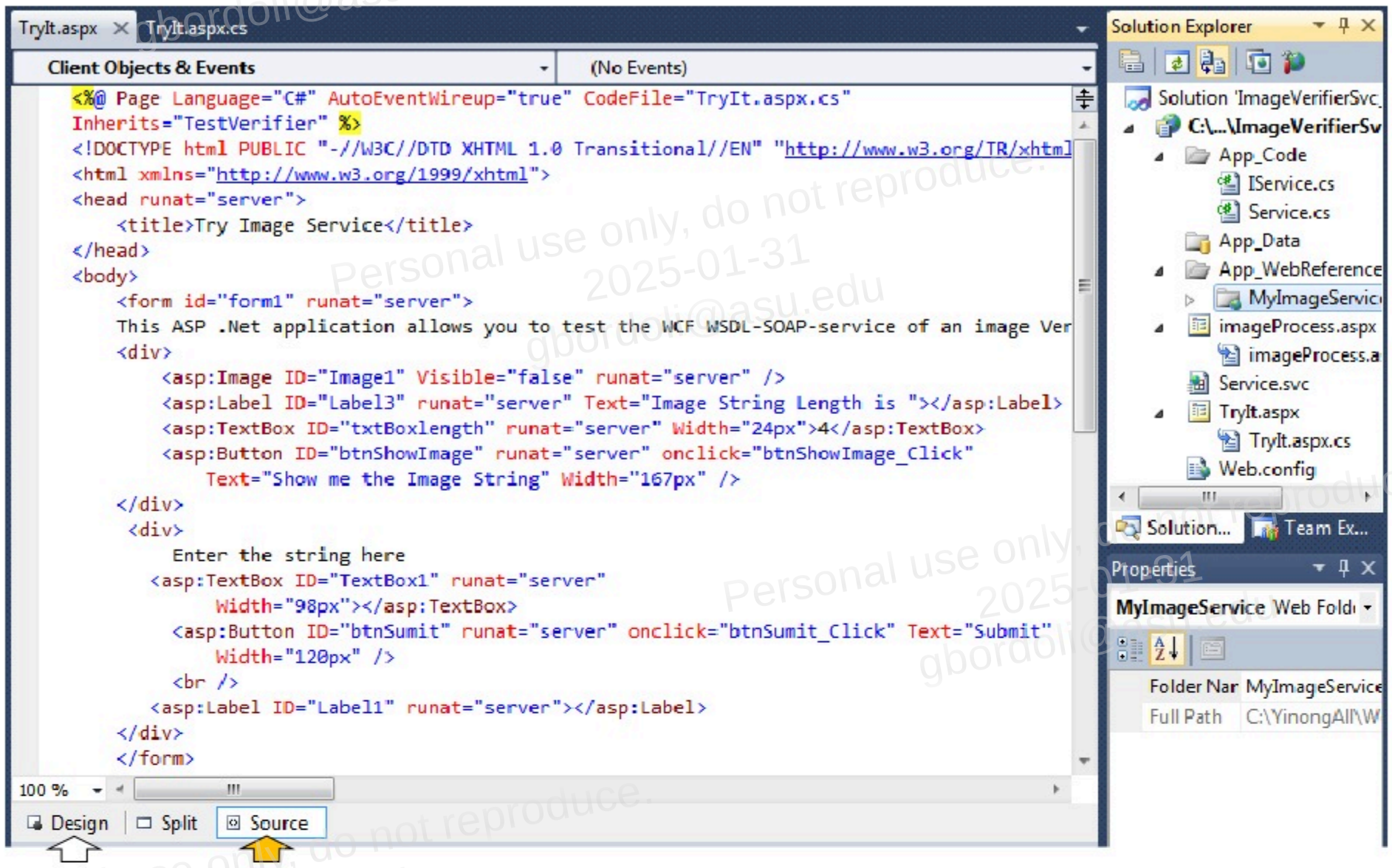


Figure A.13. Source view of the GUI design

Step 4: Adding code to the buttons. Double click the button "Show me the Image String," so that the code template is generated. Return to the Design view and then double click the button "Submit," so that the code template is generated. Note that it is important to use "double click" to generate the code templates. If you type the code templates, they will not be linked to the buttons. Your code should look as follows:

```
public partial class TestVerifier : System.Web.UI.Page {
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    protected void btnSumit_Click(object sender, EventArgs e)
    {
    }
    protected void btnShowImage_Click(object sender, EventArgs e)
    {
    }
}
```

Now, you can add your code piece by piece into the templates. After adding the code, it should look as follows:

```
public partial class TestVerifier : System.Web.UI.Page {
    protected void Page_Load(object sender, EventArgs e) {
        Image1.ImageUrl = "~/imageProcess.aspx";
    }
    protected void btnSumit_Click(object sender, EventArgs e) {
        if (Session["generatedString"].Equals(TextBox1.Text)) {
            Label1.Text = "Congratulation. The code you entered is correct!";
        }
        Else {
```

```
        Label1.Text = "I am sorry, the string you entered does not match the
image. Please try again!";
        }
    }
    protected void btnShowImage_Click(object sender, EventArgs e) {
        MyImageService.ServiceClient fromService = new
MyImageService.ServiceClient();  // create proxy to the remote service
        string userLength = txtBoxlength.Text;
        Session["userLength"] = userLength;
        string myStr = fromService.GetVerifierString(userLength);
        Session["generatedString"] = myStr;
btnShowImage.Text = "Show Me Another Image String";
        Image1.Visible = true;
    }
}
```

In the page protected void Page_Load(object sender, EventArgs e), we use the following statement to call another page “imageProcess.aspx”:

```
Image1.ImageUrl = "~/imageProcess.aspx";
```

To create this page, you can right-click the project name and choose “Add New Item.” Choose “Web Form” and name the form imageProcess.aspx. In this page, you can add the following code to call the web service to generate the image.

```
using System;
using System.IO;
using System.Drawing.Imaging;
public partial class imageProcess : System.Web.UI.Page {
    protected void Page_Load(object sender, EventArgs e) {
        Response.Clear();
        MyImageService.ServiceClient fromService = new
MyImageService.ServiceClient();
        string myStr, userLen;
        if (Session["generatedString"] == null) {
            if (Session["userLength"] == null)
                userLen = "3";
            else
                userLen = Session["userLength"].ToString();
            myStr = fromService.GetVerifierString(userLen);
            Session["generatedString"] = myStr;
        }
        else {
            myStr = Session["generatedString"].ToString();
        }
        Stream myStream = fromService.GetImage(myStr);
        System.Drawing.Image myImage = System.Drawing.Image.FromStream(myStream);
        Response.ContentType = "image/jpeg";
        myImage.Save(Response.OutputStream, ImageFormat.Jpeg);
    }
}
```

In this code, the proxy to the service is generated by the statement:

```
MyImageService.ServiceClient fromService = new MyImageService.ServiceClient();
```

The necessity of creating the module is to have the returned image assigned to the “image” control (Toolbox item). If we put the code in the page imageProcess.aspx directly in the TryIt.aspx page, the returned image will use the entire page, instead of the designated area of the page.

Now, you can test program. The input and output are shown in Figure A.14.

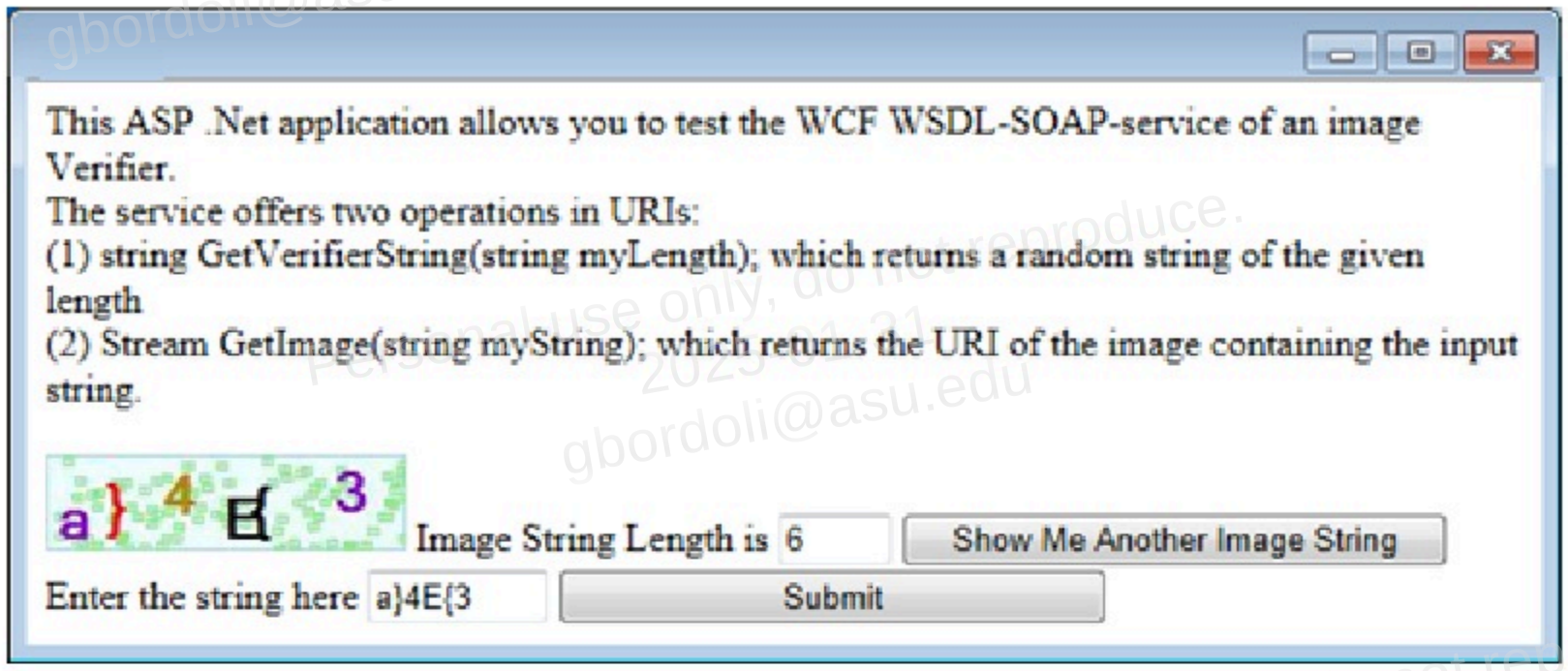


Figure A.14. Test and results of the program

In the code, you may add a print statement to print variable values for debugging purposes:

```
System.Diagnostics.Debug.WriteLine();
```

which will print the result in the debug console in the bottom of visual studio.

The application is deployed and can be tested at:

http://venus.sod.asu.edu/WSRepository/Services/ImageVerifierSvc/TryIt.aspx