# BCIT

**Comp 3951 Topics in Application Development**

**Technical Programming Option**

# Option Head Mirela Gutica

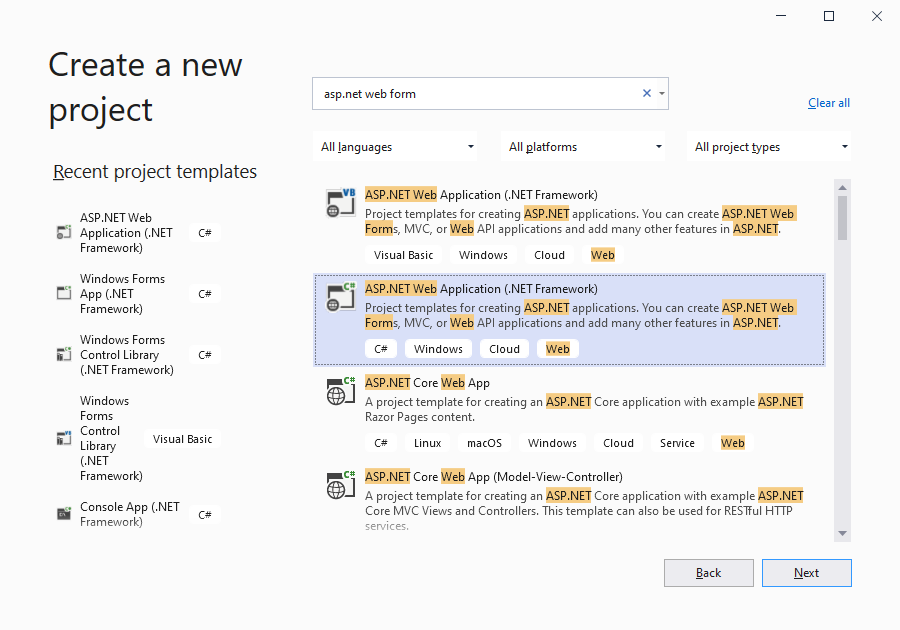
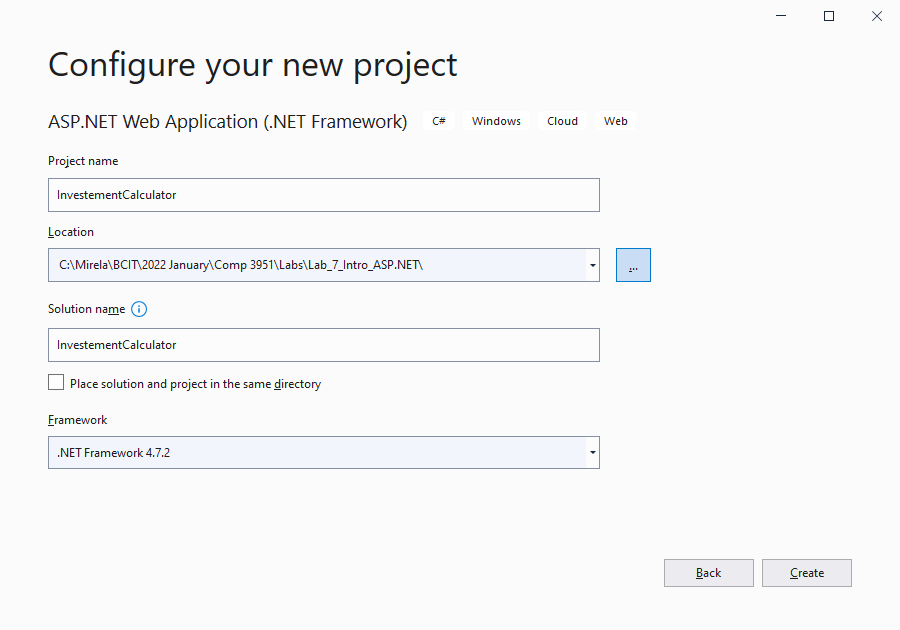
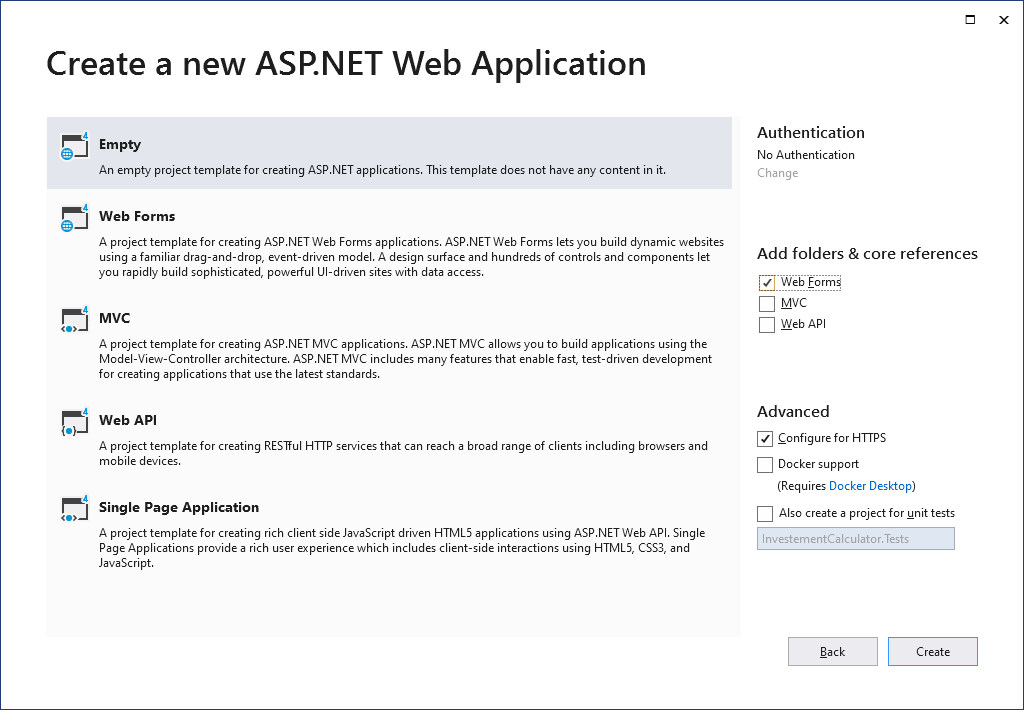
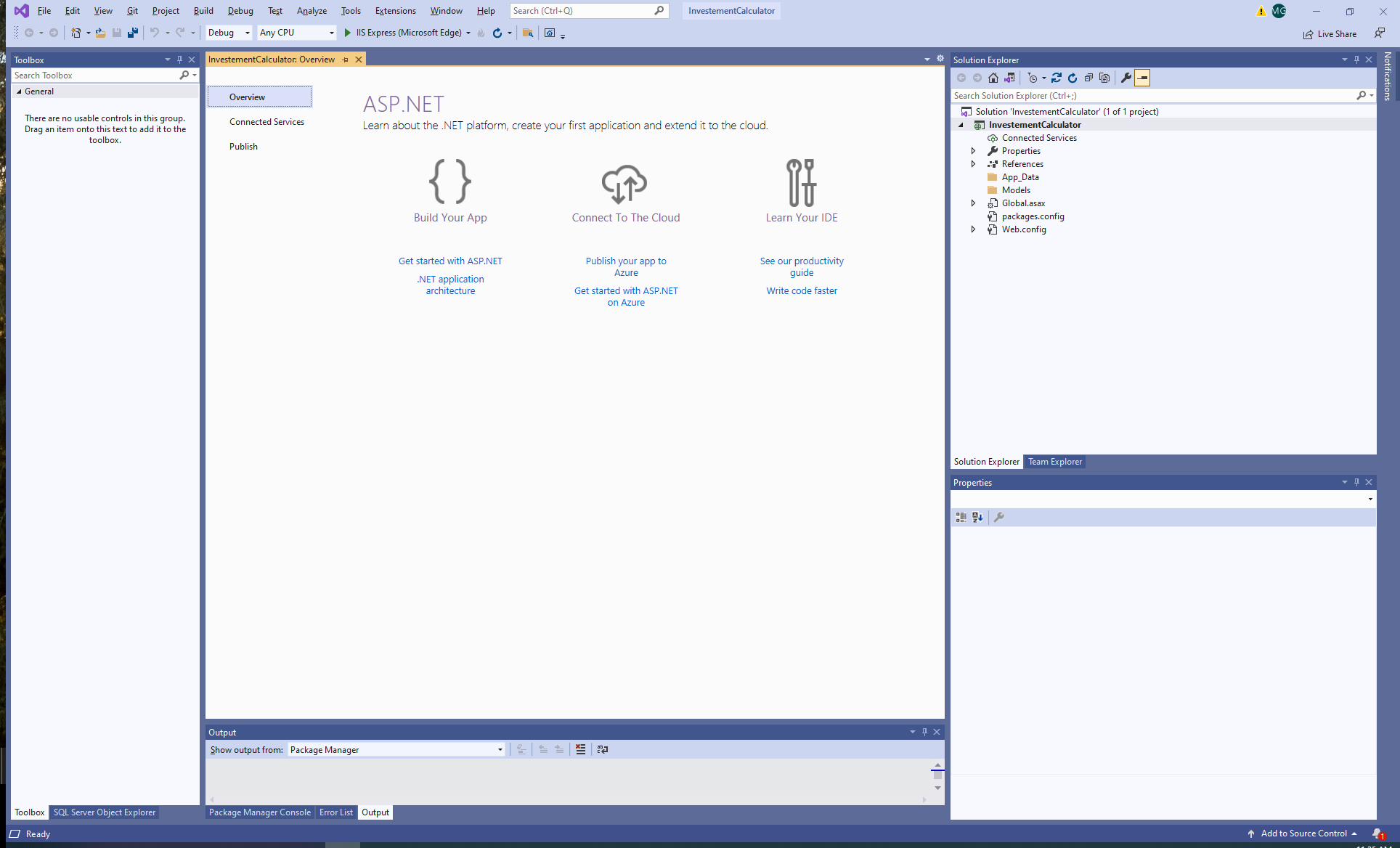
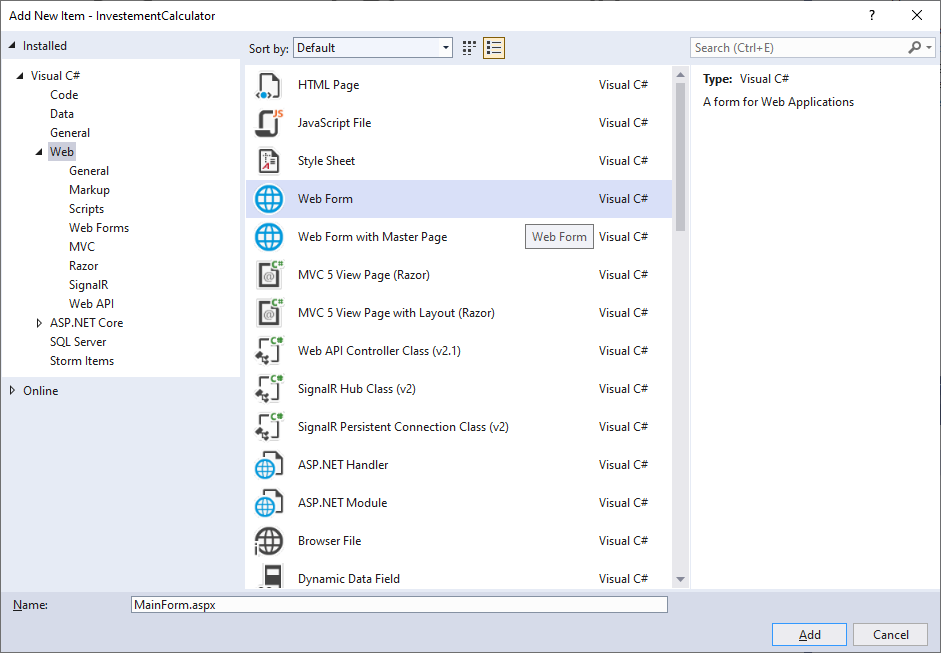
**Spring 2022**

Mark: \_\_\_\_\_\_\_\_ /100

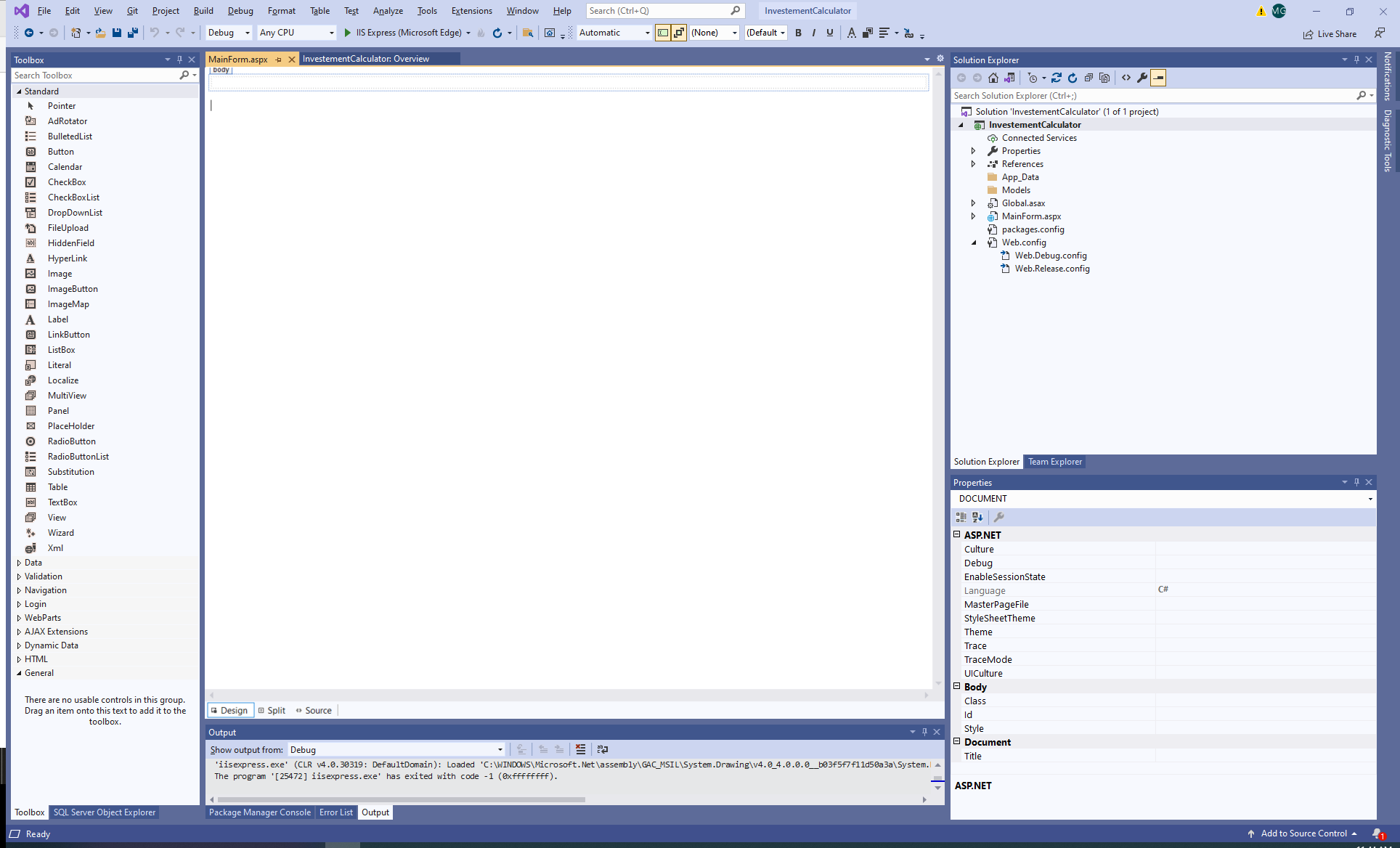
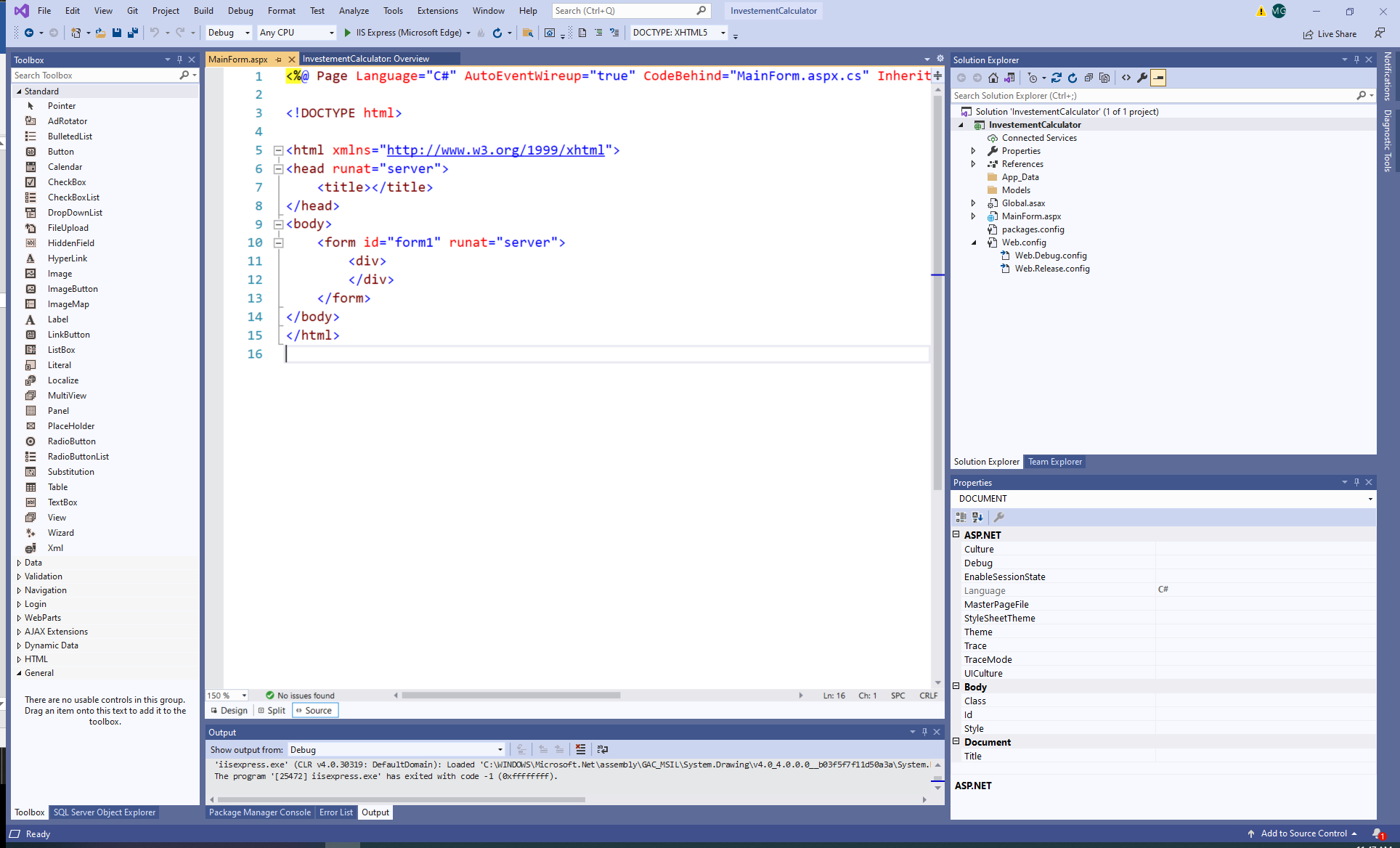
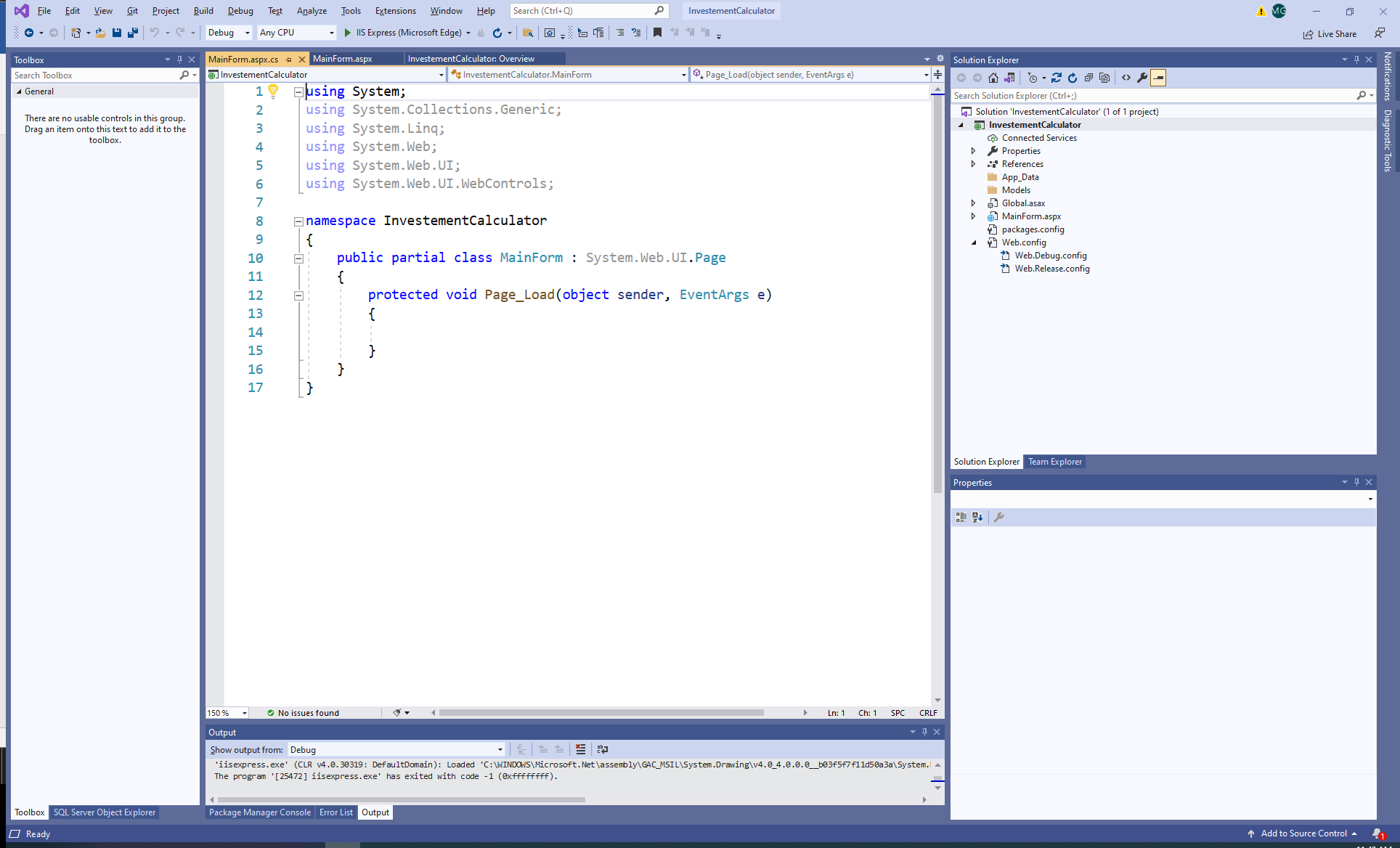
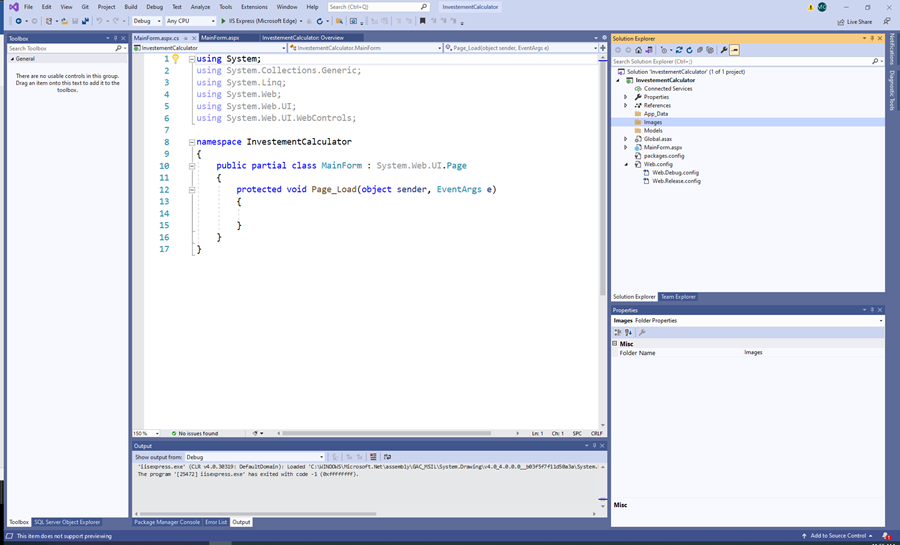
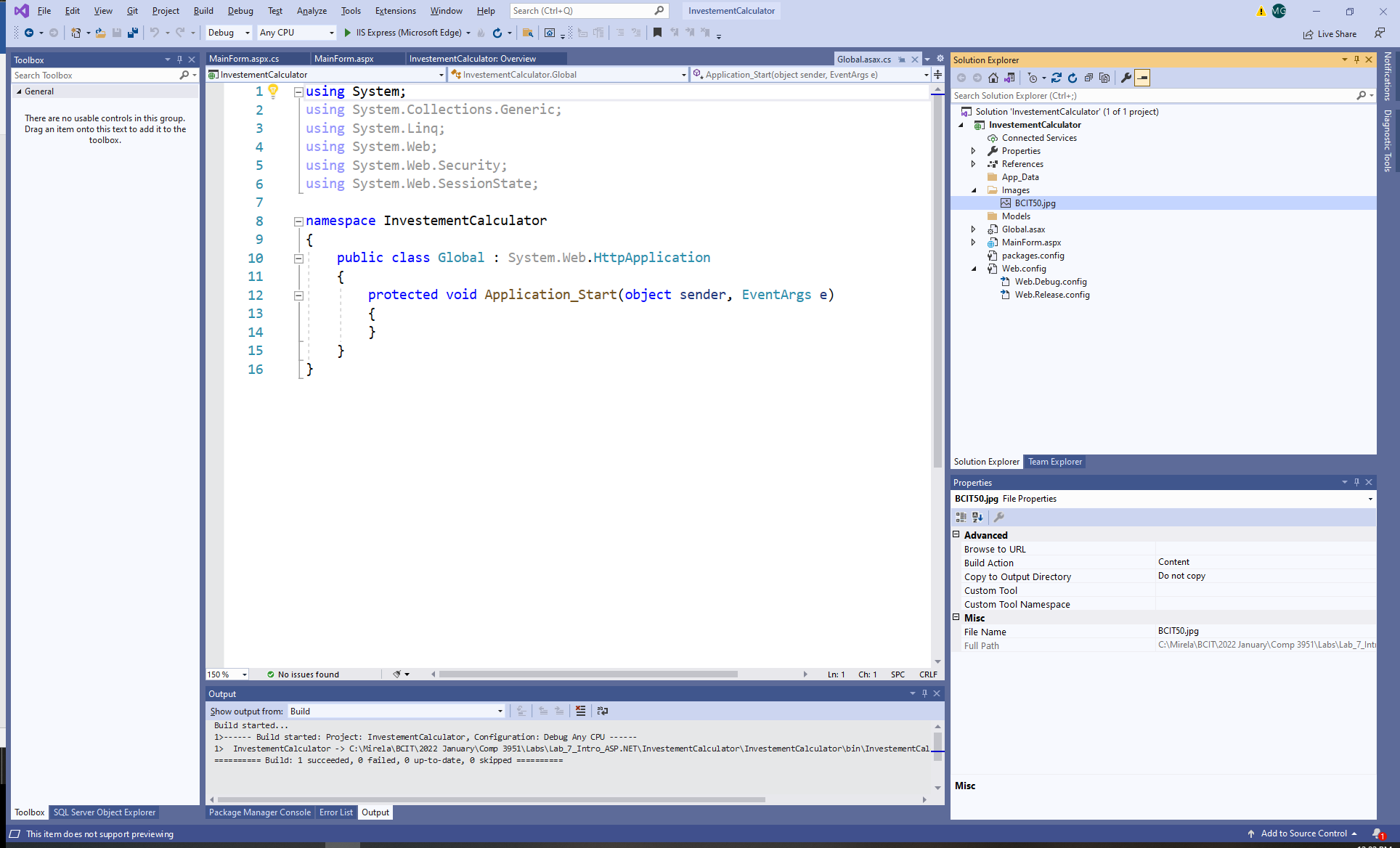
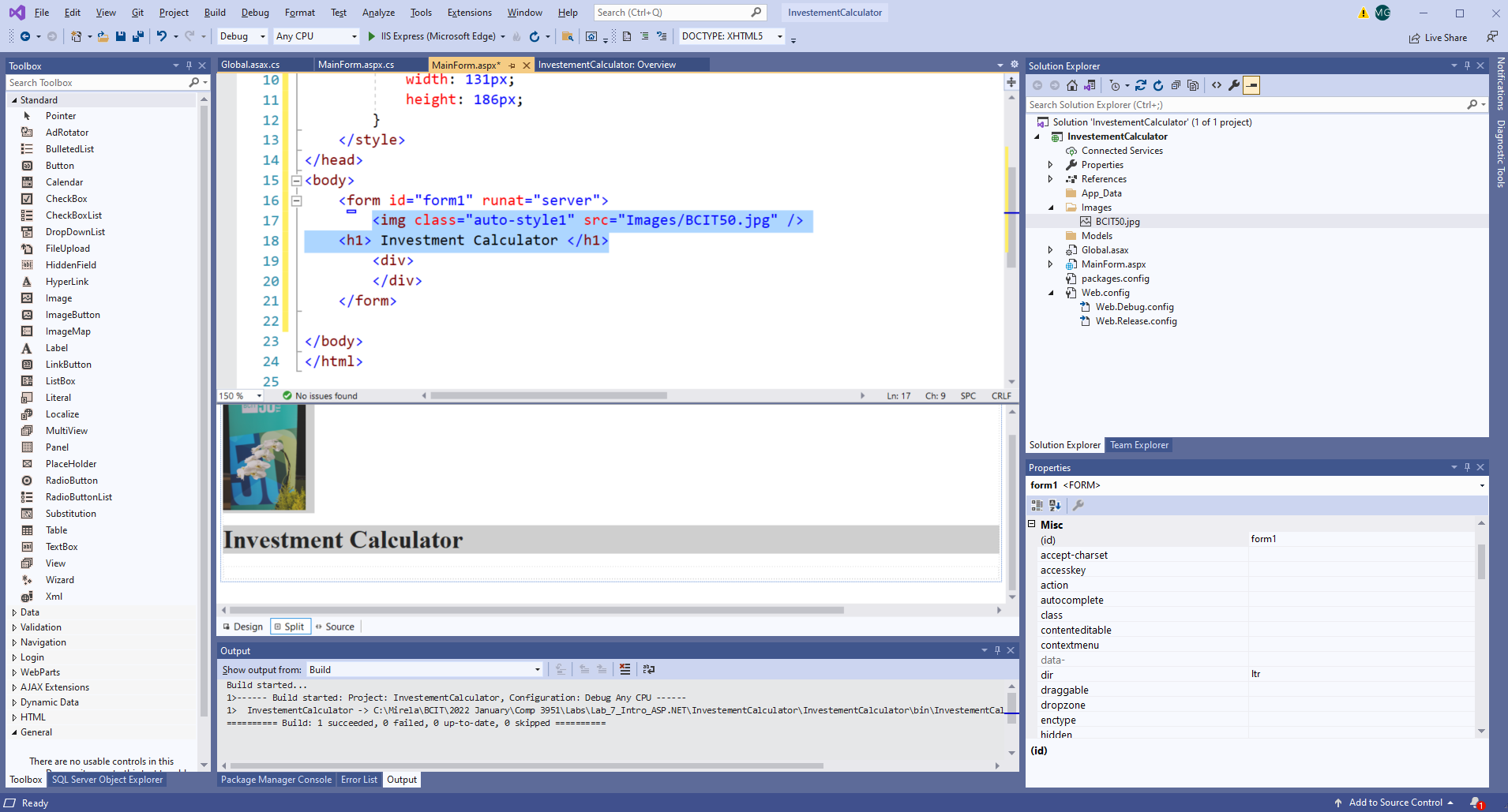
Lab 7

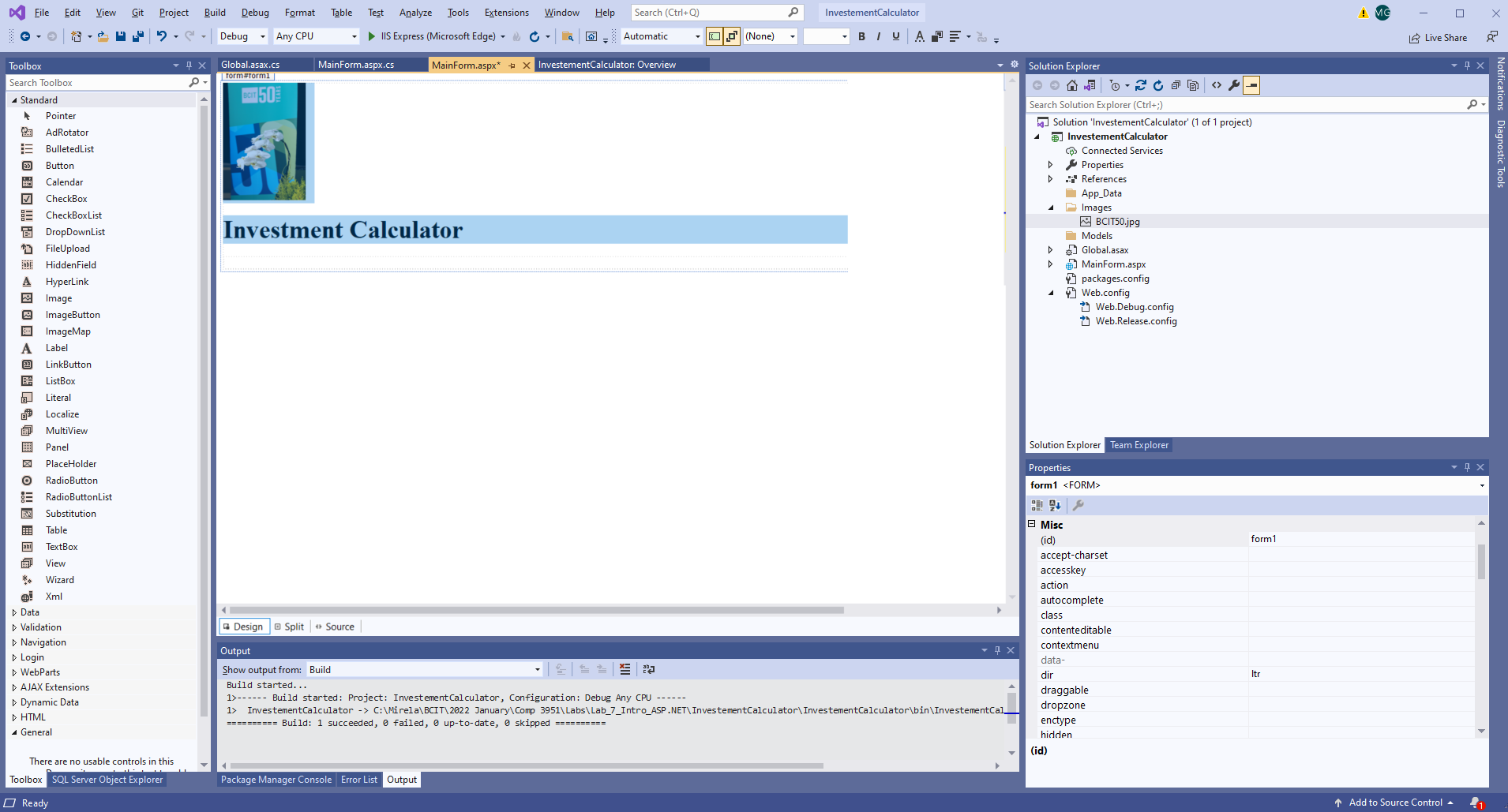
**Creating a Web Application with Web Forms in ASP.NET**

**Requirements**:

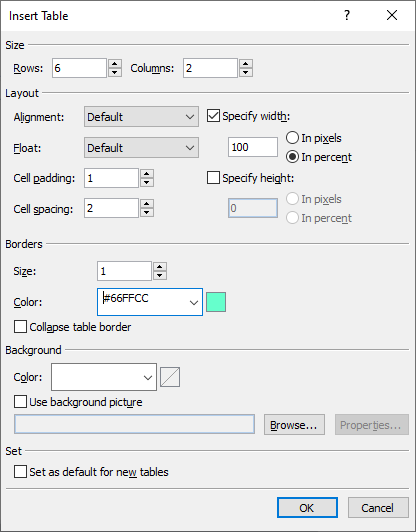
1. You will design and implement a simple Web Form website that will expose you to introductory aspects of ASP.NET development, data flow and web server controls.
2. The application calculates the future value of investments based on user input. You will add validation and style.
3. Start a new C# project “ASP.NET Web Application”.
4. 
5. Select “Empty” and check Web Forms. The “Web Forms” choice (the second one) is ok, but the framework will create some ‘default” pages for you.
6. 
7. When you create a project/website you can select a target framework: By default, 4.7.2 is used for ASP.NET Web Forms by Visual Studio 2019/2021. You can see/change later the target platform: select Debug/Properties/ InvestmentCalculator under the Build menu.
8. Observe that the new created website. If you run it, it will not work as there is no web page created.
9. 
10. Add a Web Form to your website. Use the Project/Add New Item dialog, and Web Form. 
11. The web form has a basic functionality. The Design is basic (a white form). Click on the “Source” (bottom left corner). Observe the designer, the source code and the code behind.

Run. Now the web site will run an empty page.

1. 
2. 
3. 
4. Add a folder to the website. On the Solution Explorer on the Investment Calculator, right mouse click, select Add, New Folder. Name it “Images”.
5. Add an image/logo of your choice to the folder. Then right click on the folder Images and use Add Existing Item to add the image to the project.
6. 
7. On the MainForm, click on the “Source”. Observe the keyword: runat="server". The controls (the form is a control) with this attribute run on the server (are server-side controls).
8. You can add HTLM elements to the form (two ways):
   1. In the Designer: e.g., drag a control from the ToolBox to the designer.
   2. In the Source type “<” and the intelliSense will help you enter snippets of HTML code, tags, attributes.
9. Add an image to the form. You can drag the image to the designer or you can write html code in the Source. Also, add a title. You can see both the designer and the Source by using the Split view:
10. 

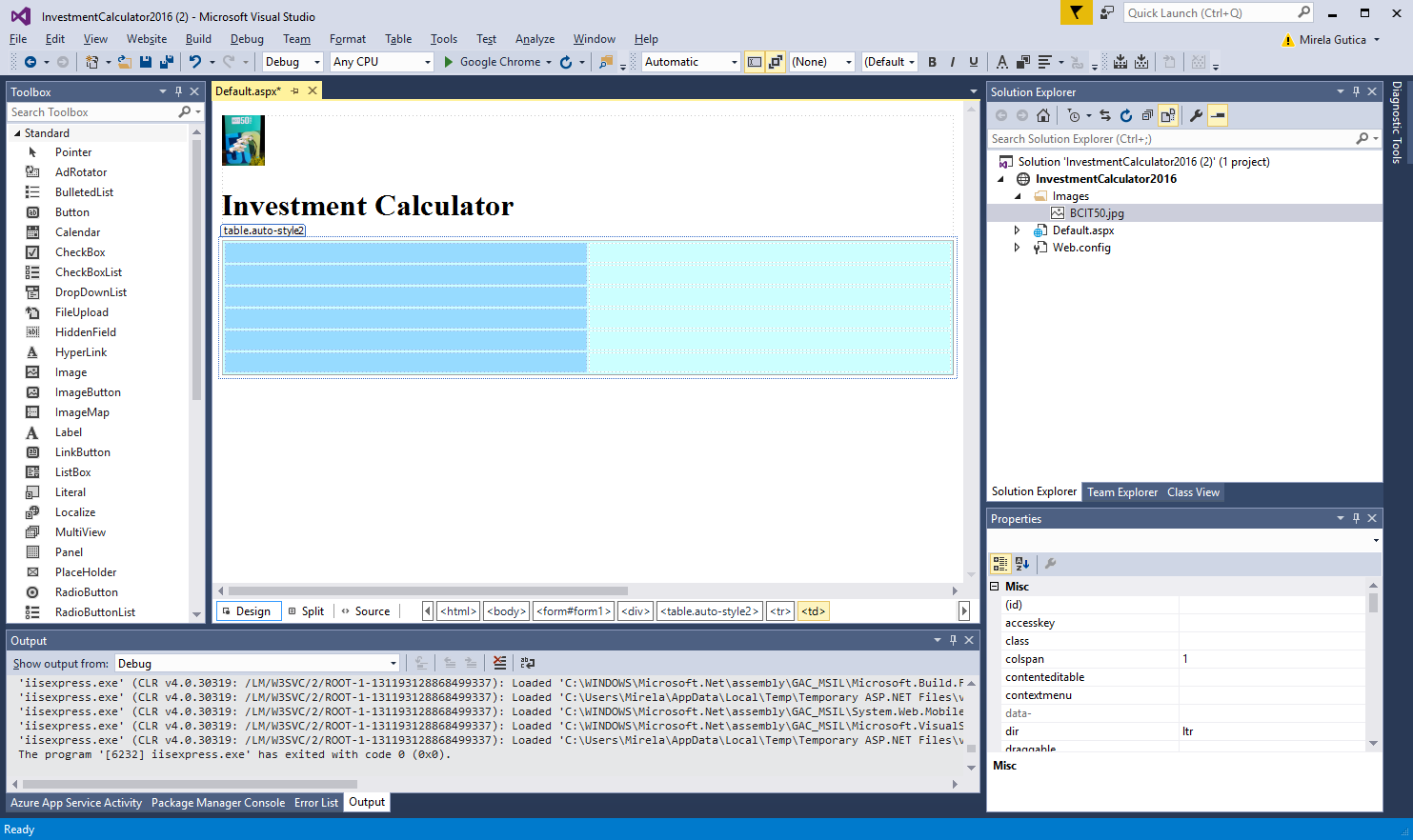


1. Add a table to your form. By default, the settings for a table are set to “flow layout” (text and controls are positioned from left to right and from top to bottom). A table provides the flow control. Another way to achieve flow control is to use CSSs.
   1. In Designer, note that the menu bar has a menu item “Table”. Use the Table menu and select “Insert Table” to create the table (an alternative choice is to use the Table control from the Toolbox).
   2. Use the Insert Table dialog to define the table’s properties:

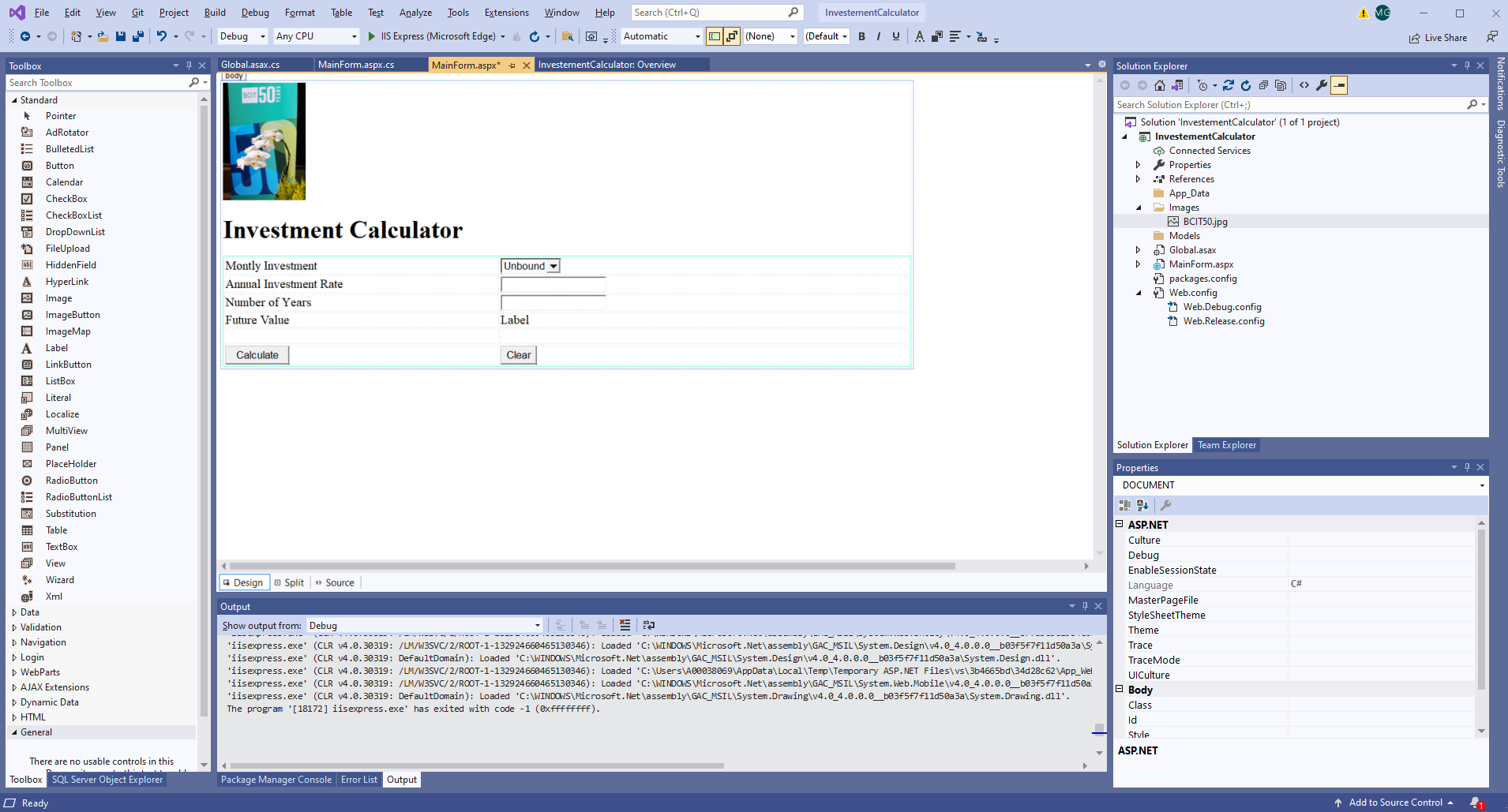


* 1. You can format the table after you create it. You can resize rows/columns by dragging theirs borders (bottom and right, respectively).

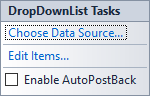
1. Add web server controls to the web form.
   1. Observe in the Toolbox the categories of controls available in ASP.NET Web Forms.
   2. Web controls can be (1) dragged to the form/table in designer or (2) inserted as a snippet of code in html.
   3. As in C# Windows Forms, use the Properties window to set properties and events to the web controls.



* 1. Many web server controls have smart tag menus that provide options for performing common tasks and setting common properties.
  2. Drag on your table the following controls: one dropdownlist, two textboxes, five labels and two buttons.
  3. Name the controls properly.



* 1. Observe the smart tag menu for the dropdownlist web control and how it can be connected to a data source. Leave the control unbound for now.



1. Add functionality in code-behind:
   * 1. In the Page\_Load event load the monthly investment choices.

if (!IsPostBack){

// add items to the DropDownListMontlyInvestment

}

* + 1. In the ButtonCalculate event add your investment calculations.

if (IsValid){

// calculations

}

* + 1. In the ButtonClear clear the content to the default values.

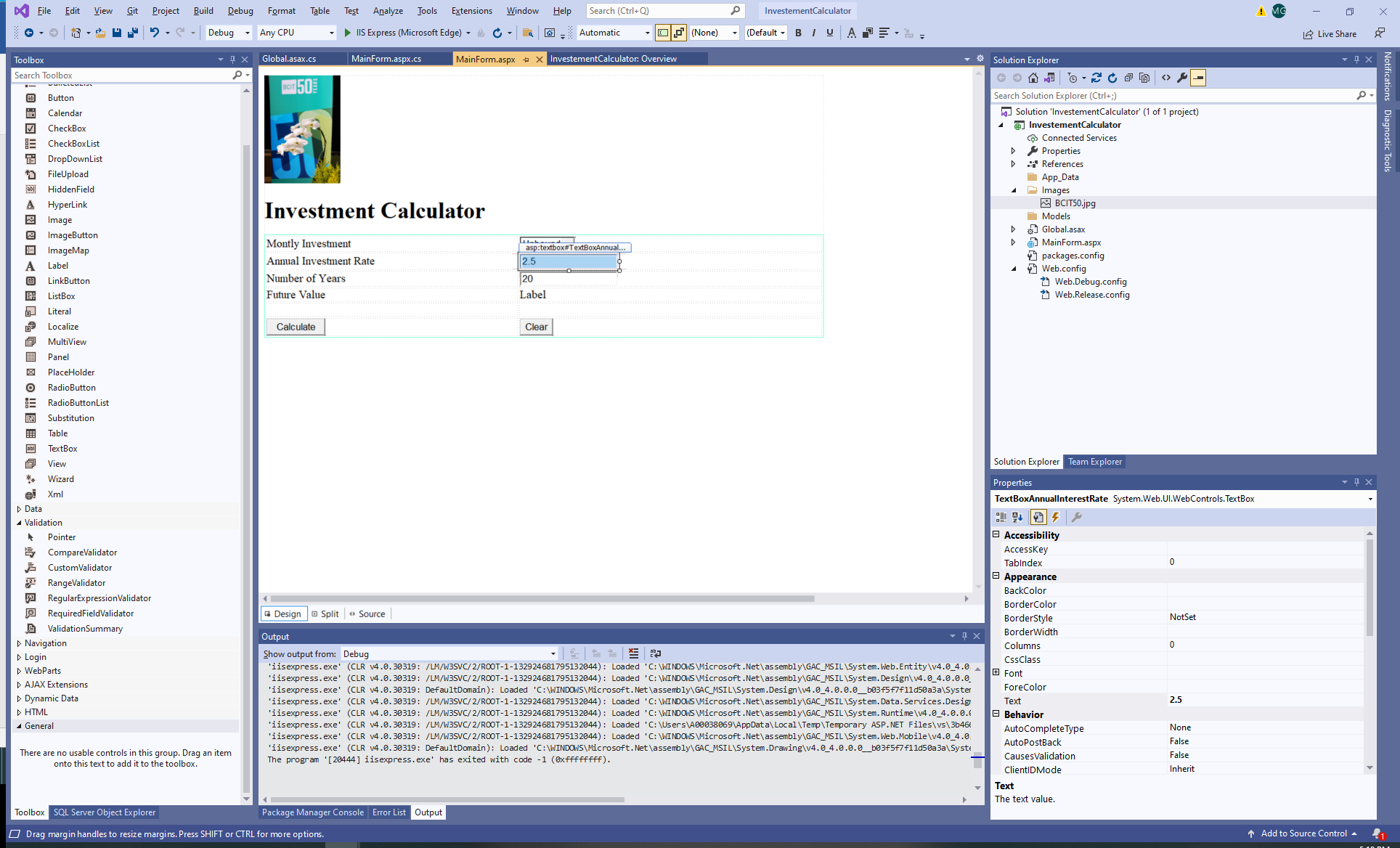
1. Calculate the “future value” with the formula:

future value = (future value + investment per month)

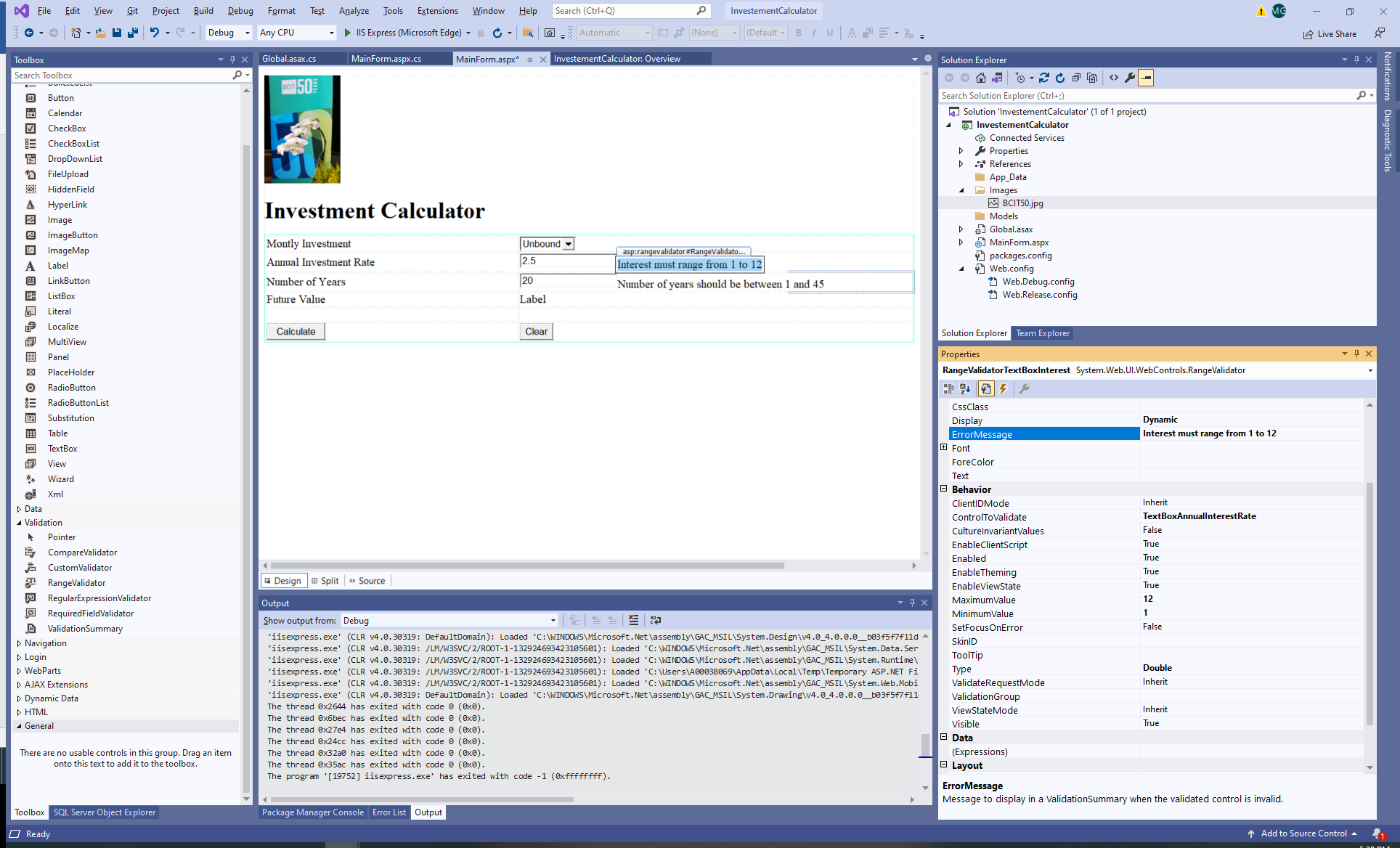
\* (1 + monthly interest rate)

(or another formula of your choice)

1. Give some initial values to the Annual Investment Rate and Number of Years.

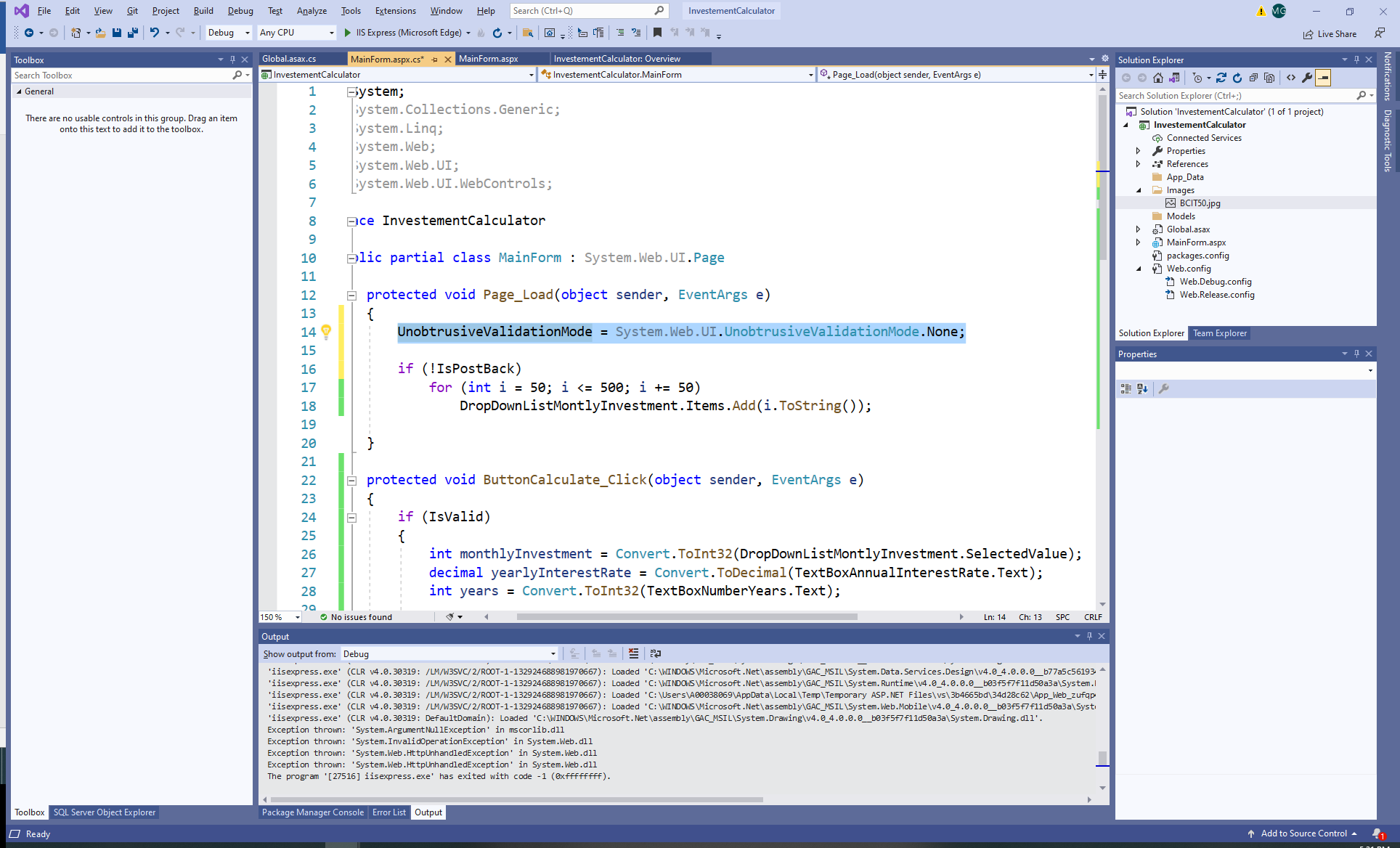


1. Run the website. Observe the need for validation **as users can input any values** in textboxes.
2. Add validation. Note that ASP.NET has a validation controls (very useful).
   1. Add field validators to your textbox controls.
      1. Add range validators
         1. Set up the relevant properties in the properties window: display (dynamic), error message, control to validate, range, type, id, min and max value.

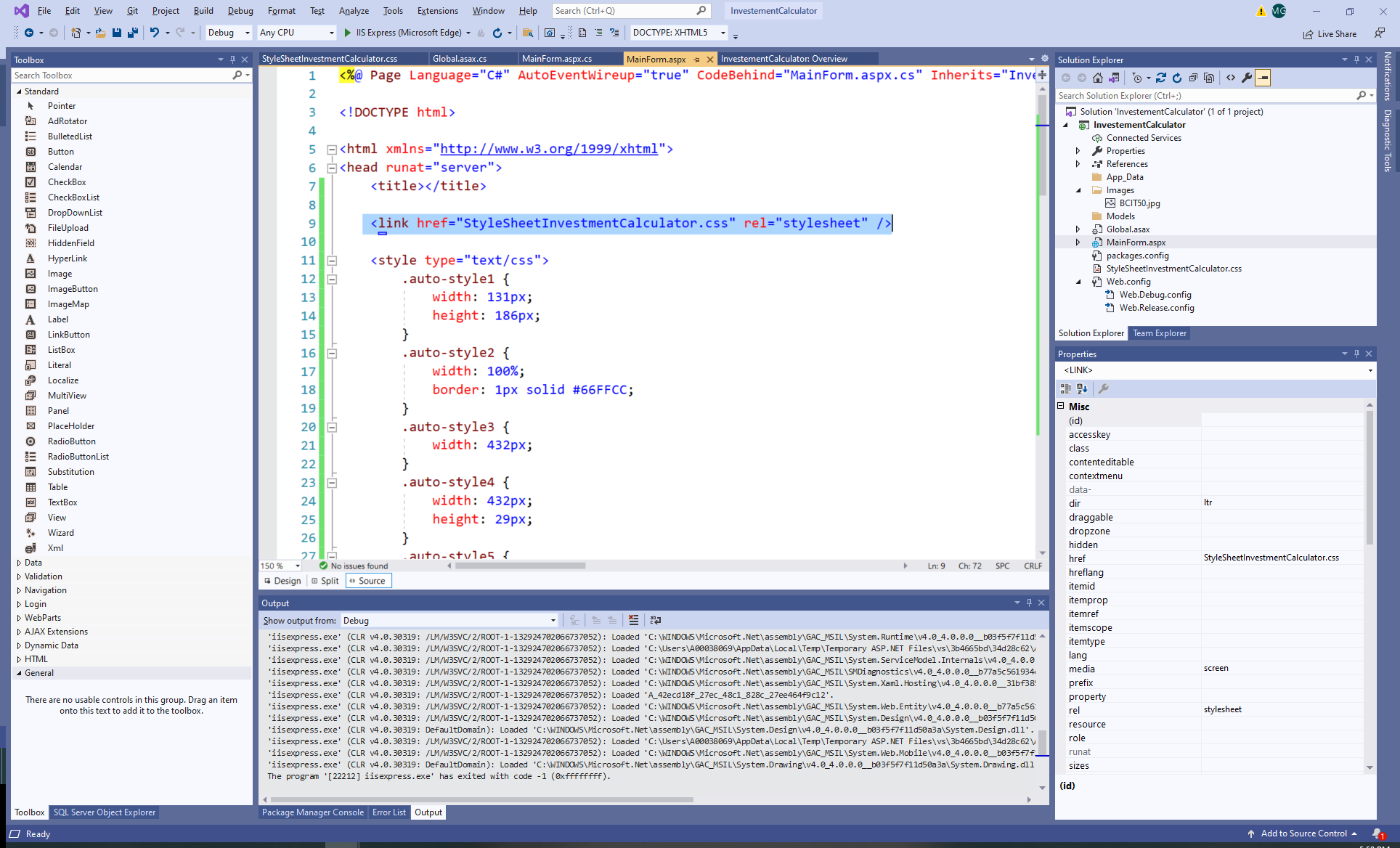


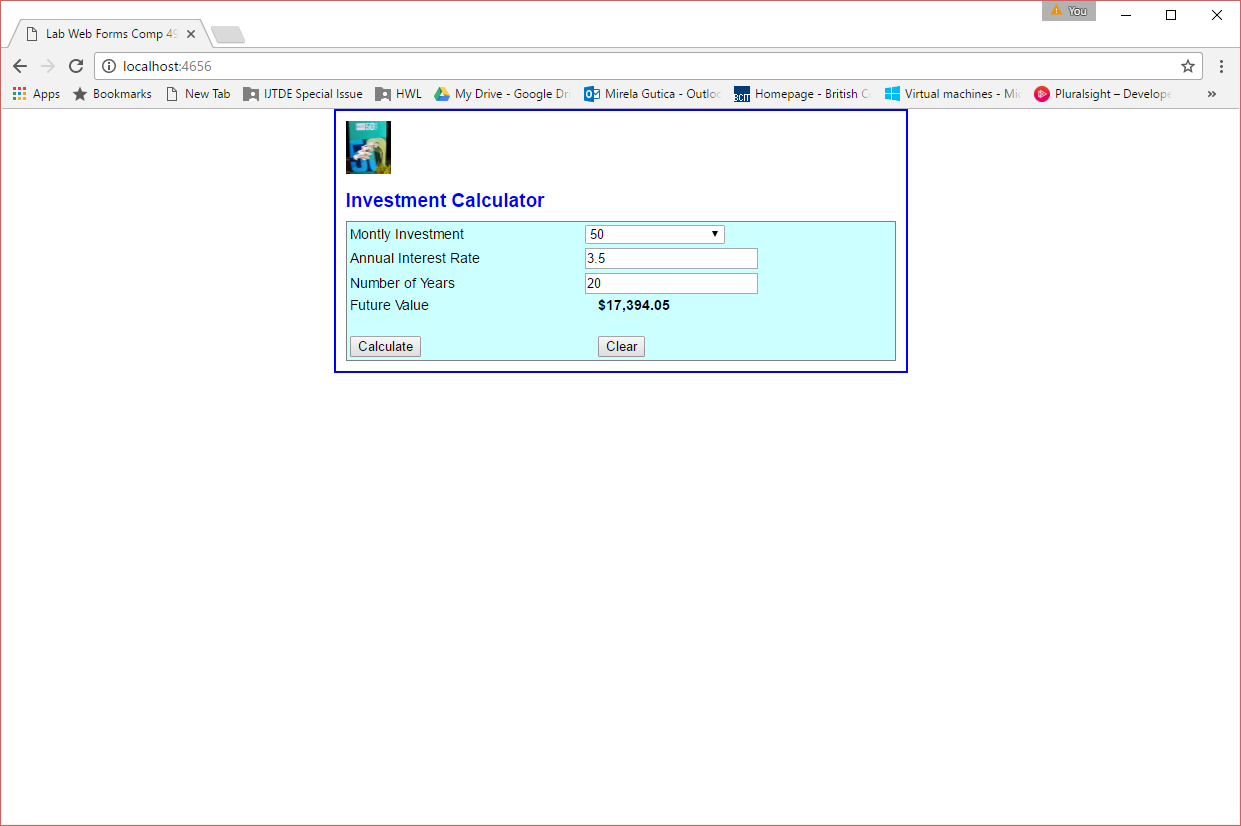
1. In Page\_Load add this line of code:

UnobtrusiveValidationMode = System.Web.UI.UnobtrusiveValidationMode.None;



1. Add CSS style.
   1. Add the CSS style of your choice such that the interface design is improved. Right click on the Investment Calculator solution, then Add/Add New Item/StyleSheet. Add the corresponding CSS class to each element of the interface.
   2. Add the link to the StyleSheet in the Source:





1. Test your application.
2. If you have Azure accounts: upload on your azure virtual server
3. To test your website functionality use: <http://localhost/Name_of_folder/default.aspx>

