LABORATORY ACTIVITY

No.2

CSCI 1113
Mobile Application Development

Lesson:
Basics of Data Binding in .NET MAUI

Objectives

After completing this activity, you will be able to:

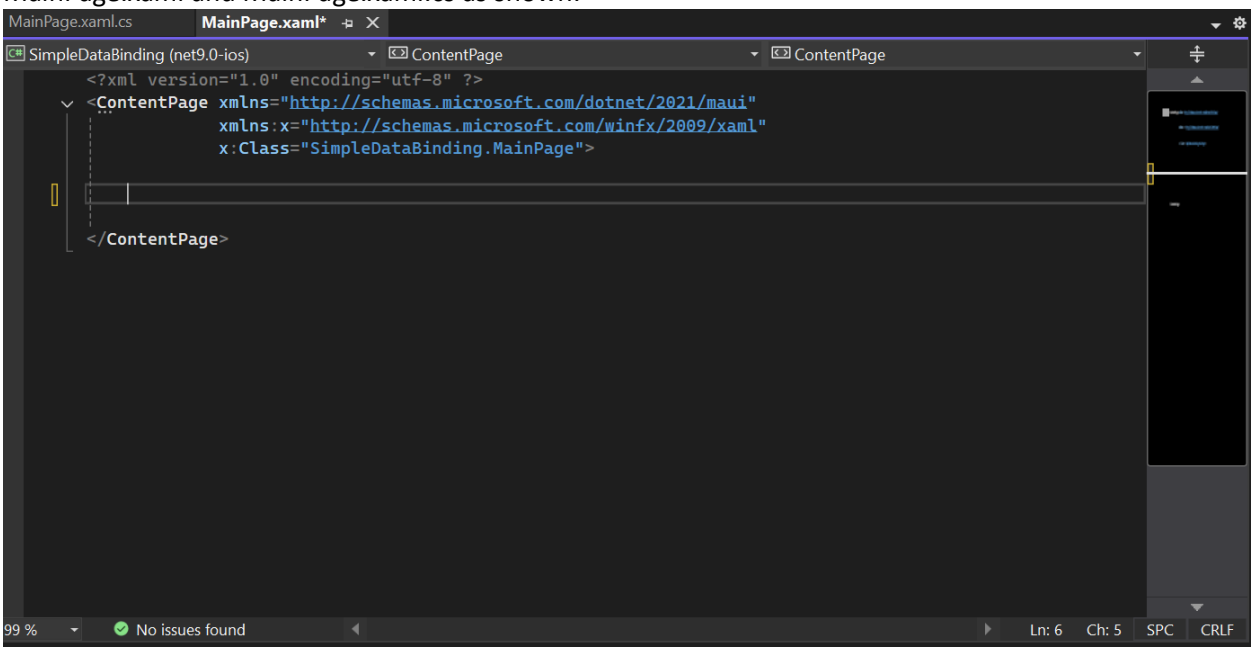
- Bind data from a user input to a ListView
- Create a simple Add Item application using .NET MAUI.
- Run and test the application on different platforms.

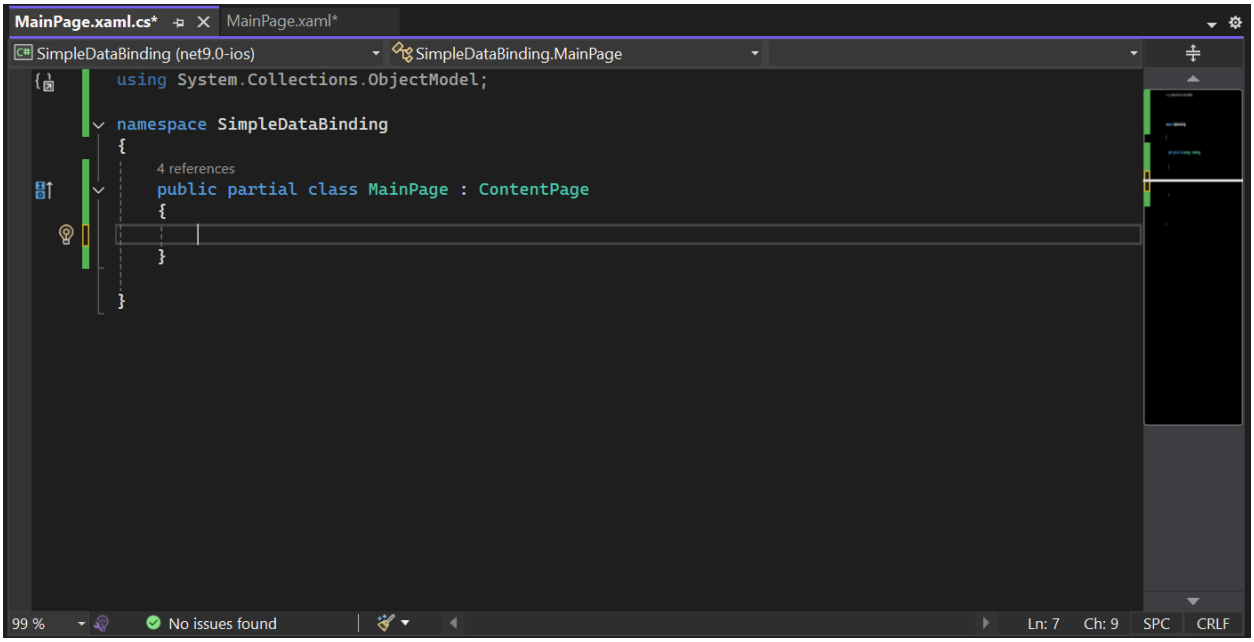
Duration

1hr 30 mins

********** START HERE **********

1) Create a new .Net MAUI app and remove the code of the default application in both MainPage.xaml and MainPage.xaml.cs as shown:





2) Now lets create a simple UI for an application where the user can input items in a list. Starting with an <Entry> tag:

```
<Entry x:Name="ItemEntry" Placeholder="Enter item here" />
```

This creates an input field where users can type their items. The x:Name attribute assigns a name (ItemEntry) to the element so it can be referenced in the code-behind (C#). The Placeholder attribute provides a gray text hint when the field is empty.

3) Next a <Button> Tag to allows users to add items to the list. The Text attribute sets the label to "Add." The Clicked event is tied to a method in the code-behind (OnAddButtonClicked), which defines what happens when the button is pressed.

```
<Button Text="Add" Clicked="OnAddButtonClicked" />
```

4) Then a <ListView> display the scrollable list of items.

```
<ListView x:Name="ItemsListView">
    <ListView.ItemTemplate>
        <DataTemplate>
            <TextCell Text="{Binding}" />
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>
```

- x:Name="ItemsListView" assigns a name to reference it in the code-behind.
- ItemTemplate defines how each item in the list appears. A TextCell is used here to display plain text.
- Text="{Binding}" binds each item in the Items collection (defined in the code-behind) to the TextCell. This means each item will automatically be shown in the list.

5) After that, go over to the MainPage.xaml.cs for the code behind. Create an ObservableCollectionn inside “public partial class MainPage : ContentPage” to hold the items. An ObservableCollection is a dynamic collection that notifies the UI whenever items are added, removed, or changed. This ensures that the ListView updates automatically whenever the Items collection is modified.

```
private ObservableCollection<string> Items { get; set; } = new ObservableCollection<string>();
```

6) The MainPage() constructor initializes the page and its components:
   - InitializeComponent() sets up the UI by processing the XAML file.
   - ItemsListView.ItemsSource = Items; binds the Items collection to the ListView. This tells the ListView to display the contents of Items.

```
public MainPage()
{
    InitializeComponent();
    ItemsListView.ItemsSource = Items;
}
```

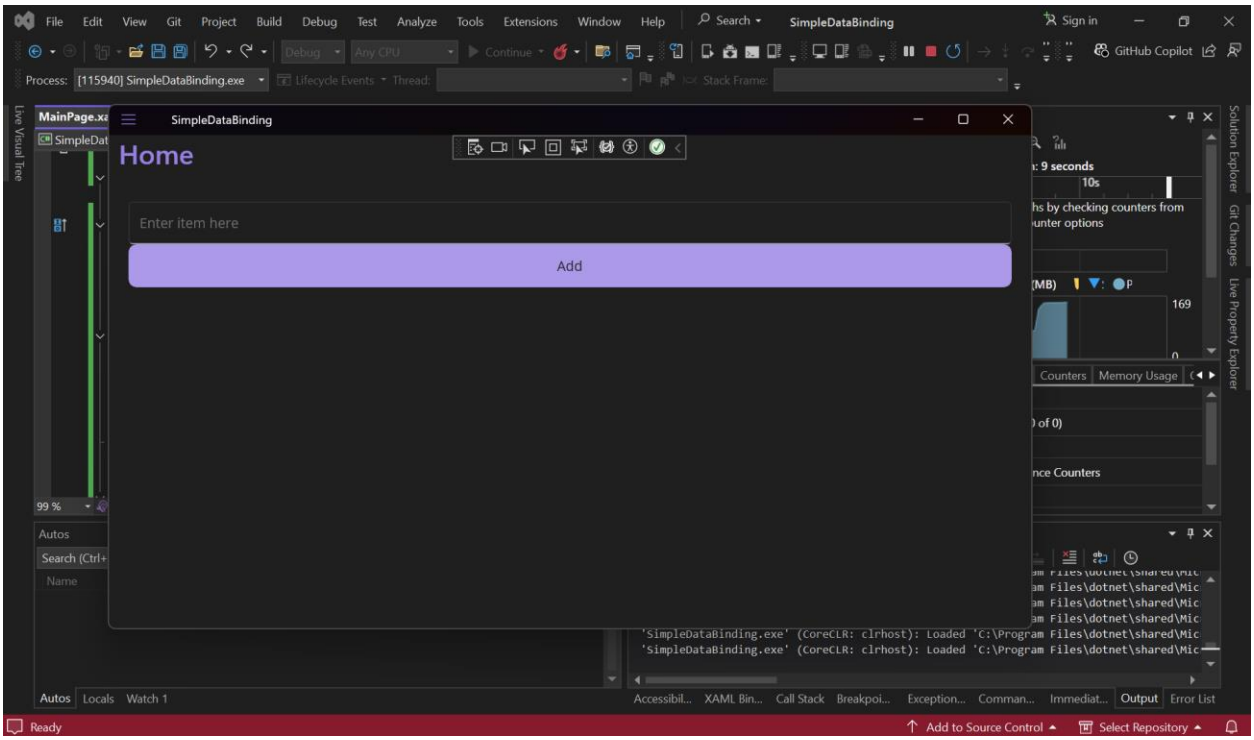
7) Finally, add a button event handler method for adding items in the list:

```
private void OnAddButtonClicked(object sender, EventArgs e)
{
    if (!string.IsNullOrWhiteSpace(ItemEntry.Text))
    {
        Items.Add(ItemEntry.Text);
        ItemEntry.Text = string.Empty;
    }
}
```

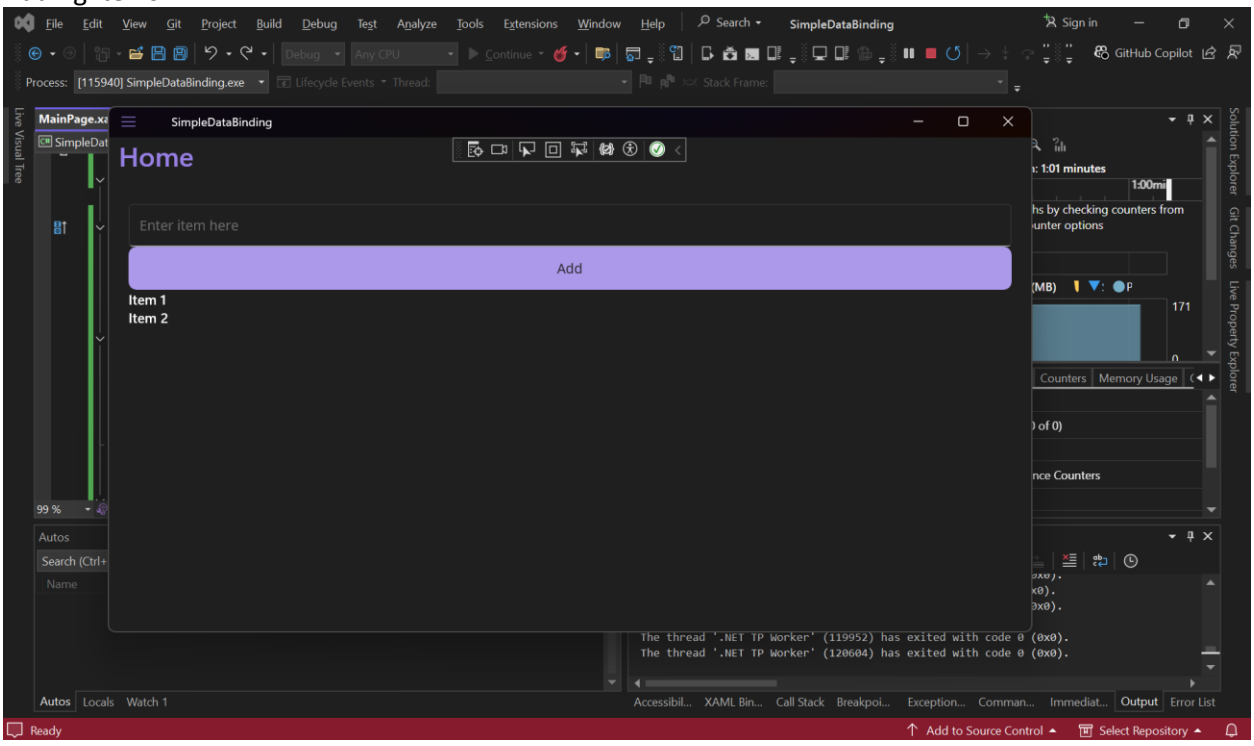
This method is triggered when the "Add" button is clicked:

- if (!string.IsNullOrWhiteSpace(ItemEntry.Text)) checks if the input field is not empty or just whitespace.
- Items.Add(ItemEntry.Text) adds the text from the input field (ItemEntry.Text) to the Items collection. Since Items is bound to the ListView, the UI updates automatically.
- ItemEntry.Text = string.Empty; clears the input field after the item is added.

8) Build the code and run it in windows machine:



Adding Items:



Additional Task:

Modify the code so that when an item is selected, it will be removed from the list.

(Hint: Put an ItemSelected Event in Listview and create a corresponding method in code-behind.)