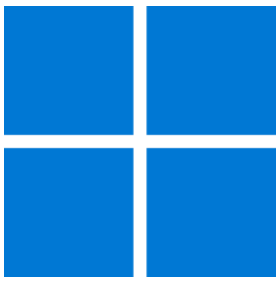
tutorial

Windows App SDK



Tic Tac Toe

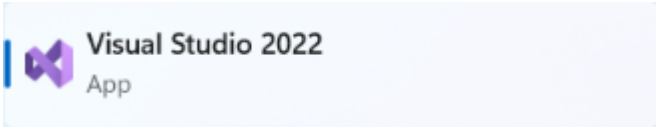
Tic Tac Toe

Tic Tac Toe shows how you can create the game also known as **Noughts and Crosses** displayed with emoji and with a toolkit from **NuGet** using the **Windows App SDK**.

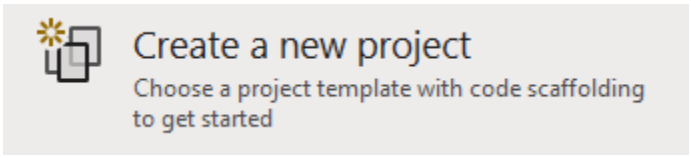
Step 1

Follow **Setup and Start** on how to get **Setup** and **Install** what you need for **Visual Studio 2022** and **Windows App SDK**.

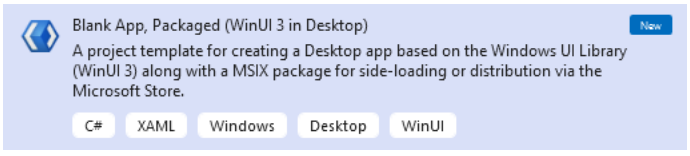
In **Windows 11** choose **Start** and then find or search for **Visual Studio 2022** and then select it.



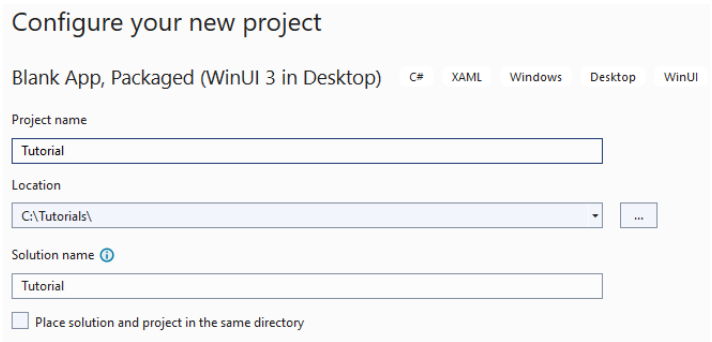
Once **Visual Studio 2022** has started select **Create a new project**.



Then choose the **Blank App, Packages (WinUI in Desktop)** and then select **Next**.

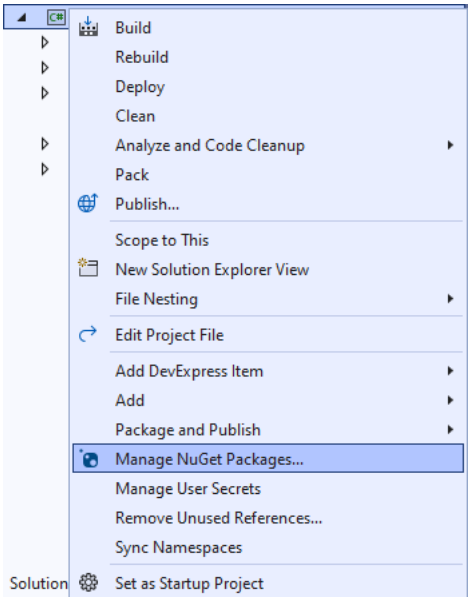


After that in **Configure your new project** type in the **Project name** as *TicTacToe*, then select a Location and then select **Create** to start a new **Solution**.



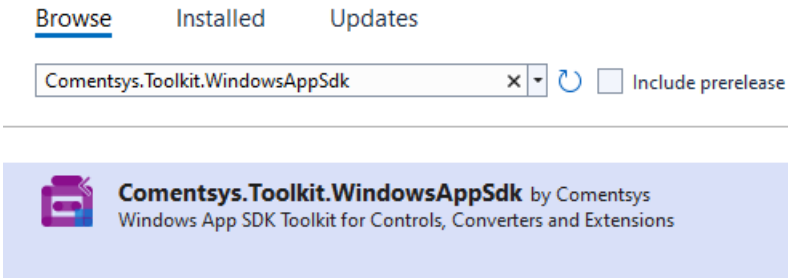
Step 2

Then in **Visual Studio** within **Solution Explorer** for the **Solution**, right click on the **Project** shown below the **Solution** and then select **Manage NuGet Packages...**



Step 3

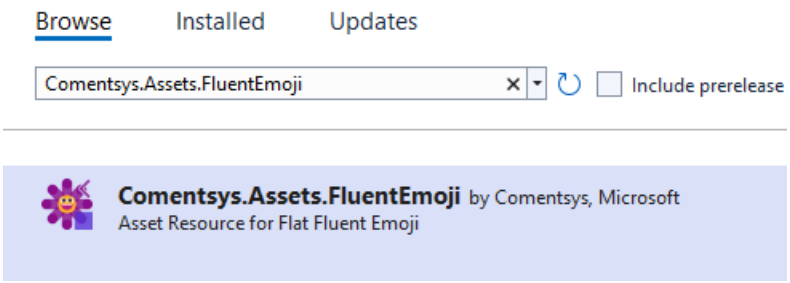
Then in the **NuGet Package Manager** from the **Browse** tab search for **Comentsys.Toolkit.WindowsAppSdk** and then select **Comentsys.Toolkit.WindowsAppSdk by Comentsys** as indicated and select **Install**



This will add the package for **Comentsys.Toolkit.WindowsAppSdk** to your **Project**. If you get the **Preview Changes** screen saying **Visual Studio is about to make changes to this solution. Click OK to proceed with the changes listed below.** You can read the message and then select **OK** to **Install** the package.

Step 4

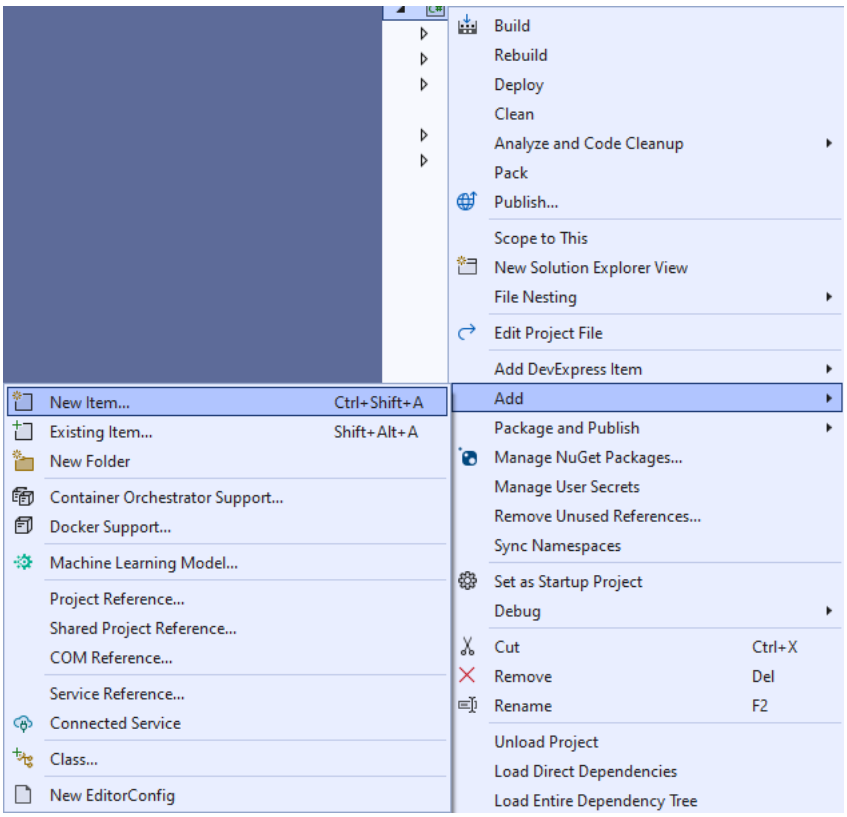
Then while still in the **NuGet Package Manager** from the **Browse** tab search for **Comentsys.Assets.FluentEmoji** and then select **Comentsys.Assets.FluentEmoji by Comentsys** as indicated and select **Install**



This will add the package for **Comentsys.Assets.FluentEmoji** to your **Project**. If you get the **Preview Changes** screen saying **Visual Studio is about to make changes to this solution. Click OK to proceed with the changes listed below.** You can read the message and then select **OK** to **Install** the package, then you can close the **tab** for **Nuget: TicTacToe** by selecting the **x** next to it.

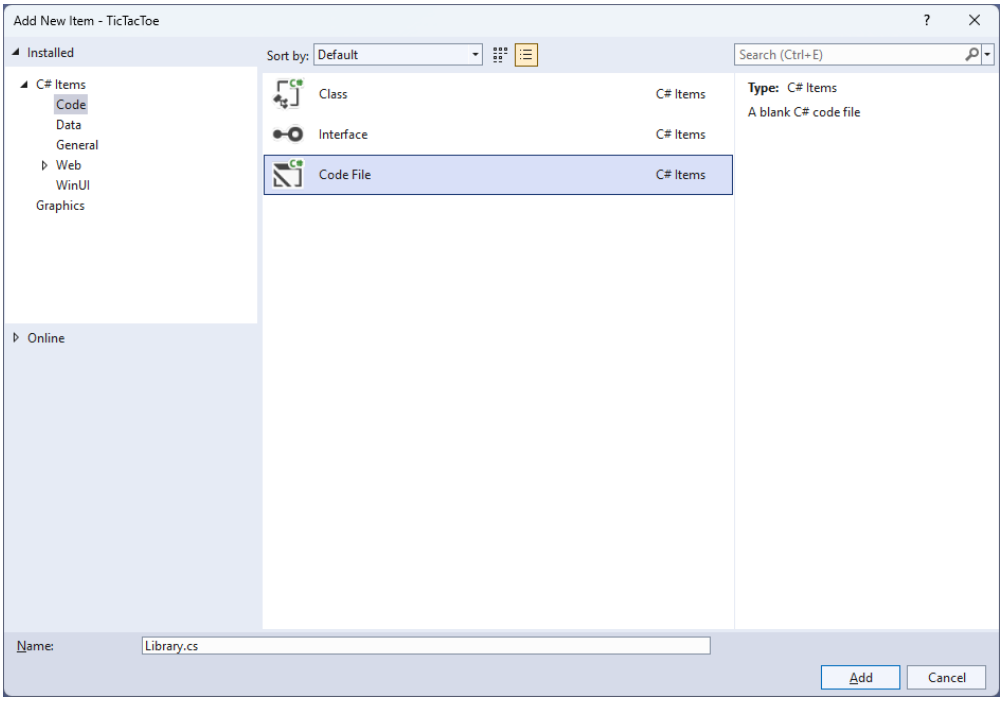
Step 5

Then in **Visual Studio** within **Solution Explorer** for the **Solution**, right click on the **Project** shown below the **Solution** and then select **Add** then **New Item...**



Step 6

Then in **Add New Item** from the **C# Items** list, select **Code** and then select **Code File** from the list next to this, then type in the name of *Library.cs* and then **Click** on **Add**.



Step 7

You will now be in the **View** for the **Code** of *Library.cs*, within this first type the following **Code**:

```
using Comentsys.Assets.FluentEmoji;
using Comentsys.Toolkit.WindowsAppSdk;
using Microsoft.UI.Xaml;
using Microsoft.UI.Xaml.Controls;
using Windows.UI;

public class Library
{
    private const string title = "Tic Tac Toe";
    private const string blank = " ";
    private const string nought = "O";
    private const string cross = "X";
    private const int size = 3;
    private readonly string[,] _board = new string[size, size];
    private readonly Color _red = Color.FromArgb(255, 249, 47, 96);
    private readonly Color _blue = Color.FromArgb(255, 0, 166, 237);
    private Dialog _dialog;
    private bool _won = false;
    private string _piece = string.Empty;

    // Winner & Drawn

    // Asset

    // Add

    // Layout & New
}
```

The **`Class`** that has been defined in so far *Library.cs* has **`using`** for the packages that were added of **`Comentsys.Assets.FluentEmoji`** and **`Comentsys.Toolkit.WindowsAppSdk`** amongst others needed. There are also some **`const`** and **`readonly`** values for parts of the game and to represent the board and some colours. Then there are some **Variables** including another control of **`Dialog`** which can be used to show messages.

Step 8

Still in the **`Class`** for *Library.cs* after the **Comment** of **`// Winner & Drawn`** type the following **Methods**:

```
private bool Winner() =>
    (_board[0, 0] == _piece && _board[0, 1] ==
    _piece && _board[0, 2] == _piece) ||
    (_board[1, 0] == _piece && _board[1, 1] ==
    _piece && _board[1, 2] == _piece) ||
    (_board[2, 0] == _piece && _board[2, 1] ==
    _piece && _board[2, 2] == _piece) ||
    (_board[0, 0] == _piece && _board[1, 0] ==
    _piece && _board[2, 0] == _piece) ||
    (_board[0, 1] == _piece && _board[1, 1] ==
    _piece && _board[2, 1] == _piece) ||
    (_board[0, 2] == _piece && _board[1, 2] ==
    _piece && _board[2, 2] == _piece) ||
    (_board[0, 0] == _piece && _board[1, 1] ==
    _piece && _board[2, 2] == _piece) ||
    (_board[0, 2] == _piece && _board[1, 1] ==
    _piece && _board[2, 0] == _piece);

private bool Drawn() =>
    _board[0, 0] != blank && _board[0, 1] !=
    blank && _board[0, 2] != blank &&
    _board[1, 0] != blank && _board[1, 1] !=
    blank && _board[1, 2] != blank &&
    _board[2, 0] != blank && _board[2, 1] !=
    blank && _board[2, 2] != blank;
```

`Winner` will work out the winning positions the **`_piece`** is in and will be **`true`** if the positions result in a complete set of either noughts or crosses or **`false`** and **`Drawn`** will be used to check if the game is a draw.

Step 9

While still in the **`Class`** for *Library.cs* after the **Comment** of **`// Asset`** type in the following **Method**:

```
private Viewbox Asset() => new()
{
    Child = new Asset
    {
        AssetResource =
        _piece switch
        {
            nought => FlatFluentEmoji.Get(
                FluentEmojiType.HollowRedCircle,
                _red.AsDrawingColor(),
                _blue.AsDrawingColor()),
            _ => FlatFluentEmoji.Get(
                FluentEmojiType.CrossMark)
        }
    }
};
```

Step 10

While still in the **`Class`** for *Library.cs* after the **Comment** of **`// Add`** type the following **Method**:

```
private void Add(Grid grid, int row, int column)
{
    Button button = new()
    {
        Width = 75,
        Height = 75,
        Margin = new Thickness(10)
    };
    button.Click += (object sender, RoutedEventArgs e) =>
    {
        if (!_won)
        {
            button = (Button)sender;
            if (button.Content == null)
            {
                button.Content = Asset();
                _board[(int)button.GetValue(Grid.RowProperty),
                (int)button.GetValue(Grid.ColumnProperty)] = _piece;
            }
            if (Winner())
            {
                _won = true;
                _dialog.Show($"{_piece} wins!");
            }
            else if (Drawn())
            {
                _dialog.Show("Draw!");
            }
            else
            {
                // Swap Players
                _piece = _piece == cross ? nought : cross;
            }
        }
        else
        {
            _dialog.Show("Game Over!");
        }
    };
    button.SetValue(Grid.ColumnProperty, column);
    button.SetValue(Grid.RowProperty, row);
    grid.Children.Add(button);
}
```

`Add` will be used to create a **`Button`** which will have **Event Handler** of **`Click`** set which will use **`Asset`** to set the **Content** to the nought with *Hollow Red Circle* which has been modified to be blue instead of red or the *Red Cross Mark* for cross. This **Method** will also check if the game has been won with **`Winner`** or **`Drawn`** and will display a message if the game is over.

Step 11

While still in the **`Class`** for *Library.cs* after the **Comment** of **`// Layout & New`** type the following **Methods**:

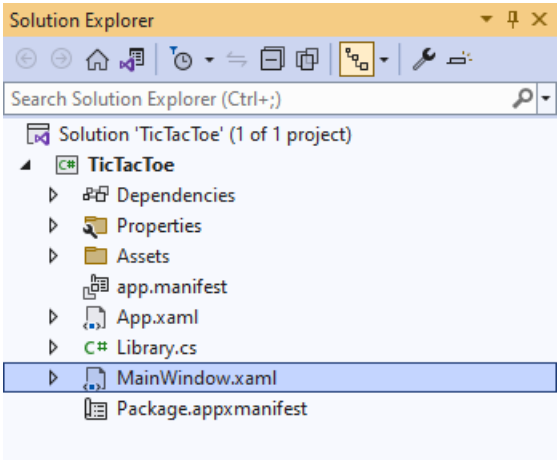
```
private void Layout(Grid grid)
{
    grid.Children.Clear();
    grid.RowDefinitions.Clear();
    grid.ColumnDefinitions.Clear();
    // Setup Grid
    for (int index = 0; index < size; index++)
    {
        grid.RowDefinitions.Add(new RowDefinition());
        grid.ColumnDefinitions.Add(new ColumnDefinition());
    }
    // Setup Board
    for (int row = 0; row < size; row++)
    {
        for (int column = 0; column < size; column++)
        {
            Add(grid, row, column);
            _board[row, column] = blank;
        }
    }
}

public async void New(Grid grid)
{
    _won = false;
    _dialog = new Dialog(grid.XamlRoot, title);
    _piece = await _dialog.ConfirmAsync("Who goes First?", nought, cross) ?
        nought : cross;
    Layout(grid);
}
```

`Layout` will be used with **`Add`** to setup what the game will look like and **`New`** will start a game, it will setup the **`Dialog`** and will show a message to choose which player should go first and then uses **`Layout`**.

Step 12

Then from **Solution Explorer** for the **Solution** double-click on **MainWindow.xaml** to see the **XAML** for the **Main Window**.



Step 13

In the **XAML** for **MainWindow.xaml** there be some **XAML** for a **`StackPanel`**, this should be **Removed** by removing the following:

```
<StackPanel Orientation="Horizontal"
HorizontalAlignment="Center" VerticalAlignment="Center">
    <Button x:Name="myButton" Click="myButton_Click">Click Me</Button>
</StackPanel>
```

Step 14

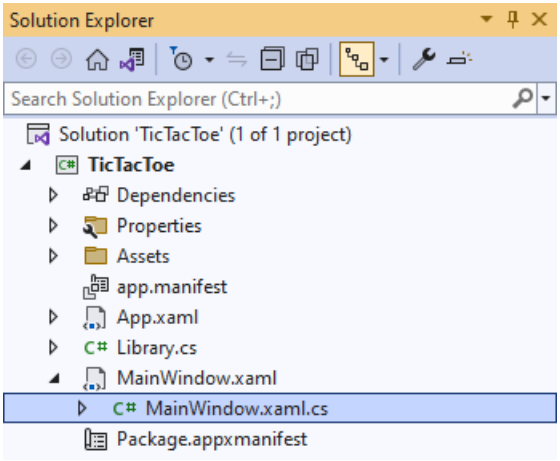
While still in the **XAML** for **MainWindow.xaml** above **`</Window>`**, type in the following **XAML**:

```
<Grid>
    <Viewbox>
        <Grid Margin="50" Name="Display"
        HorizontalAlignment="Center"
        VerticalAlignment="Center" Loaded="New"/>
    </Viewbox>
    <CommandBar VerticalAlignment="Bottom">
        <AppBarButton Icon="Page2" Label="New" Click="New"/>
    </CommandBar>
</Grid>
```

This **XAML** contains a **`Grid`** with a **`Viewbox`** which will scale a **`Grid`**. It has a **`Loaded`** event handler for **`New`** which is also shared by the **`AppBarButton`**.

Step 15

Then, within **Solution Explorer** for the **Solution** select the arrow next to **MainWindow.xaml** then double-click on **MainWindow.xaml.cs** to see the **Code** for the **Main Window**.



Step 16

In the **Code** for **MainWindow.xaml.cs** there be a **Method** of **`myButton_Click(...)`** this should be **Removed** by removing the following:

```
private void myButton_Click(object sender, RoutedEventArgs e)
{
    myButton.Content = "Clicked";
}
```

Step 17

Once **`myButton_Click(...)`** has been removed, type in the following **Code** below the end of the **Constructor** of **`public MainWindow() { ... }`**:

```
private readonly Library _library = new();

private void New(object sender, RoutedEventArgs e) =>
    _library.New(Display);
```

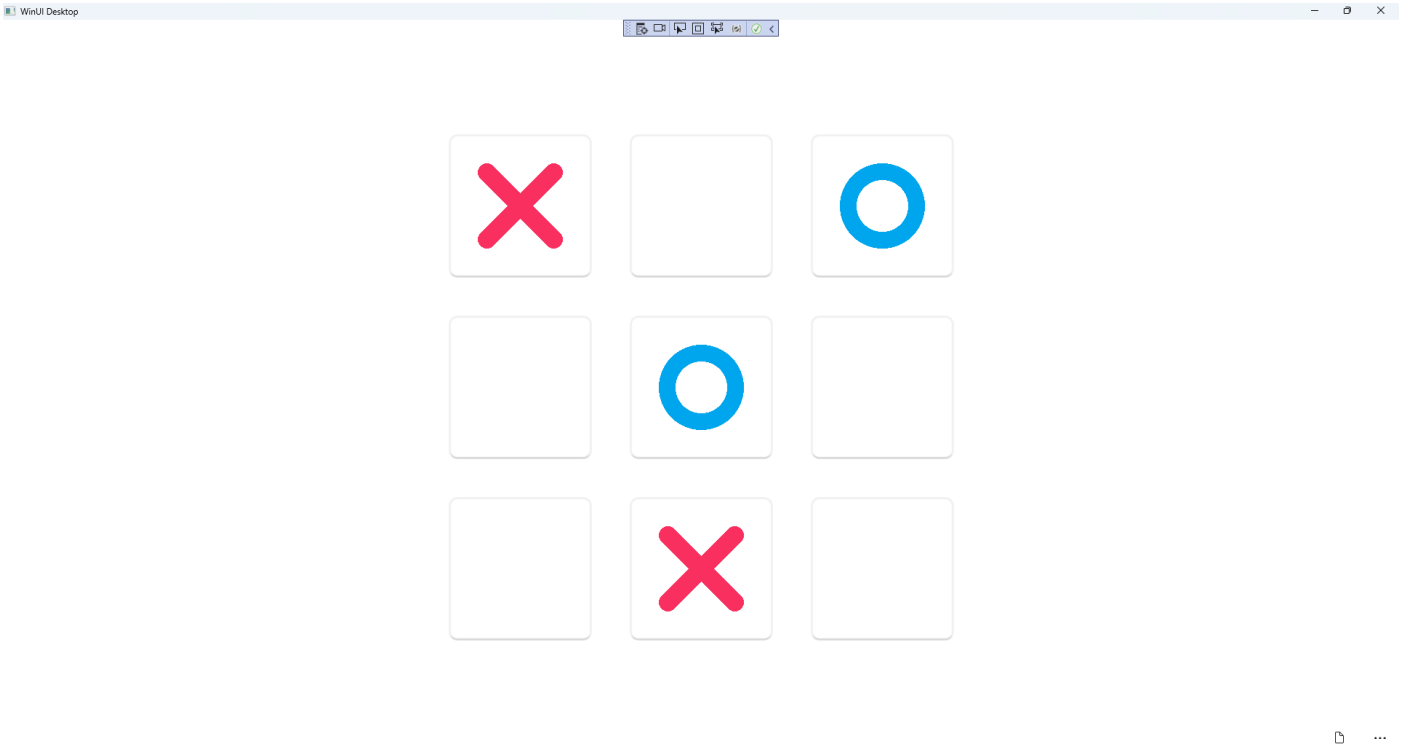
Here an **Instance** of the **Class** of **`Library`** is created then below this is the **Method** of **`New`** that will be used with **Event Handler** from the **XAML**, this **Method** uses Arrow Syntax with the **`=>`** for an Expression Body which is useful when a **Method** only has one line.

Step 18

That completes the **Windows App SDK** application. In **Visual Studio 2022** from the **Toolbar** select **TicTacToe (Package)** to **Start** the application.

Step 19

Once running you should see a **Message** asking *Who goes First?* You can then choose *X* or *O* then **Click** on any **`Button`** to take your turn in the game, with three noughts or crosses in any horizontal, vertical or diagonal direction to win the game! You can restart the game by selecting *New*.



Step 20

To **Exit** the **Windows App SDK** application, select the **Close** button from the top right of the application as that concludes this **Tutorial** for **Windows App SDK** from tutorialr.com!