Contents

Overview

MAUI is a .Net technology for creating a multi-platform GUI.

Applications generally require a graphical user interface (GUI). For a browser-based application there is HTML, but sometimes end users want a GUI which is 'like' other 'native' applications in their environment.

MAUI is an open-source technology for creating an application GUI with a single code base which has the 'look and feel' of a 'native' application GUI for the Windows, Android and Apple environments. The single MAUI code base will present a 'native' GUI on all device targets for these operating systems.

See here for more information.

This document presents the step-by-step instructions to create a MAUI solution which will present a 'native' GUI for these operating systems. The MAUI solution illustrates how to support algorithms and business rules supported by an APL64 cross-platform component within the solution.

Download the Example Source Code

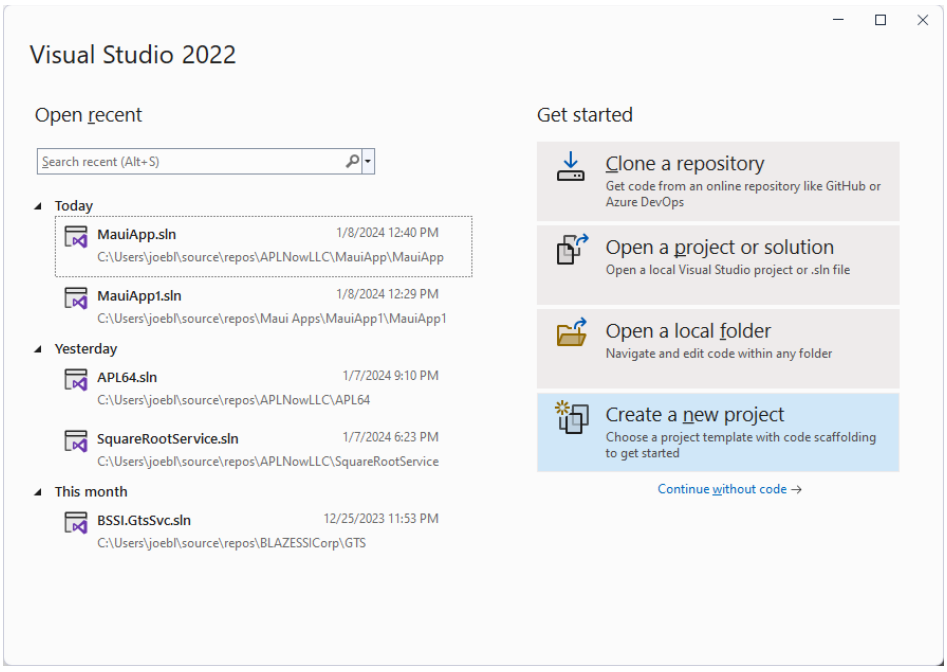
Download example source code here:

http://apl2000.com/APL64/UserDocumentation/APL64_MAUI_GUI.zip

Extract the files to 'c:\'

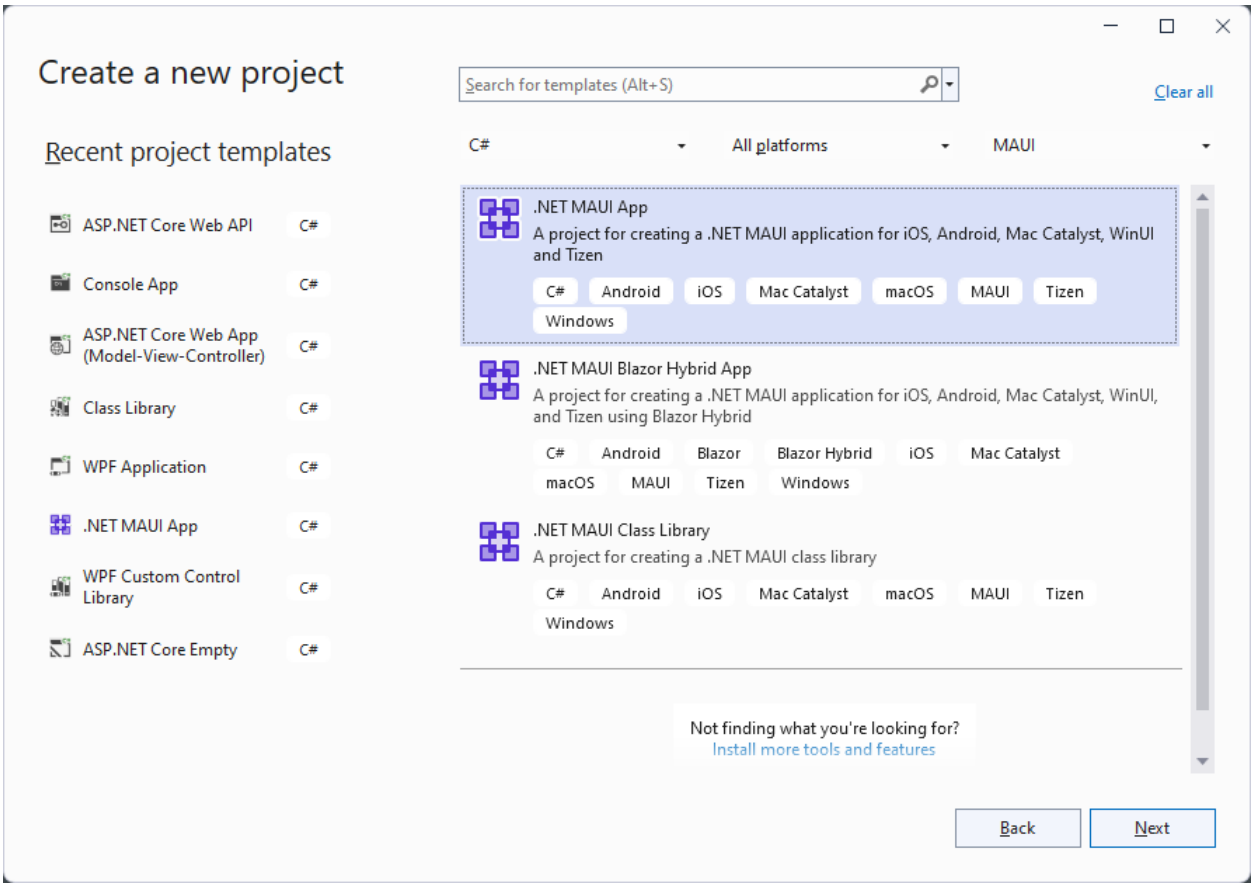
Create the Visual Studio Solution

Open VS2022 and Create a New Project



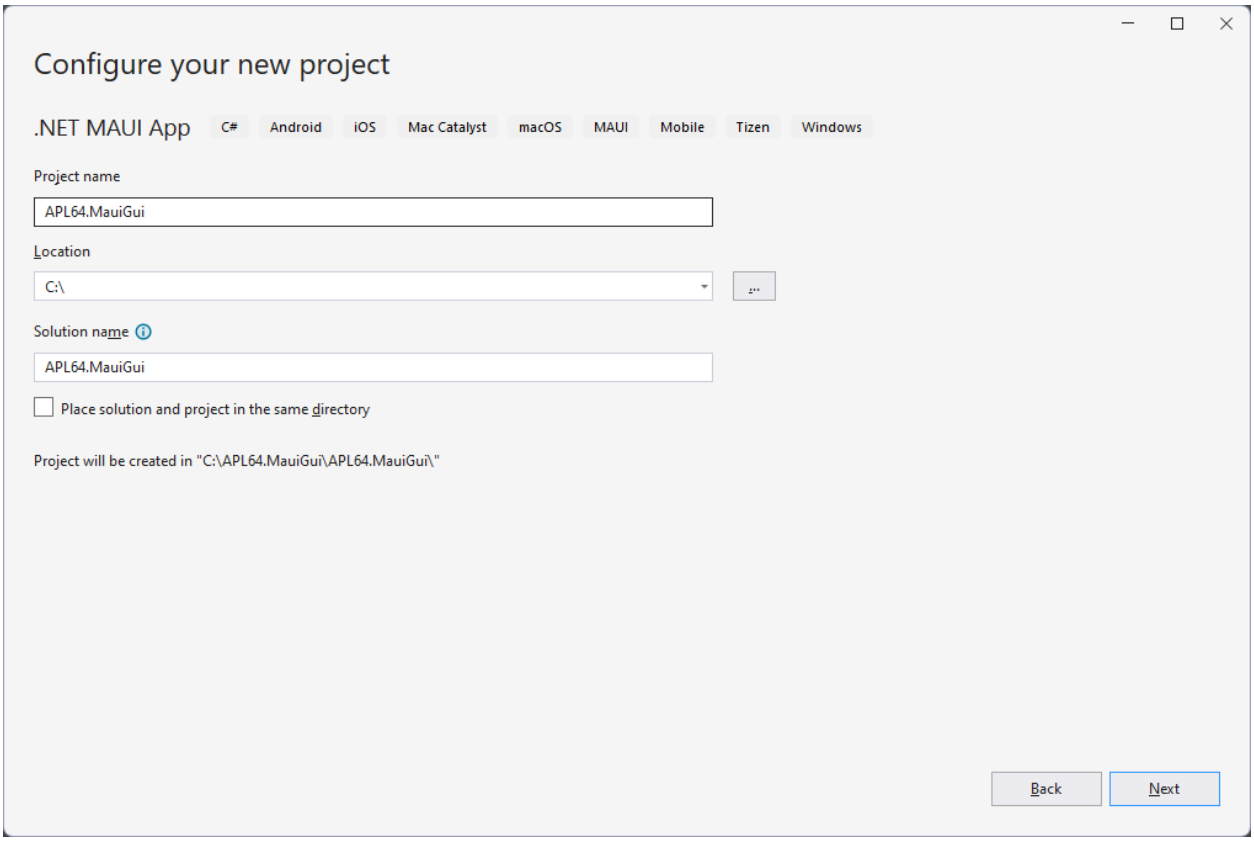
If the above dialog is not visible, use the VS2022 | File | New | Project menu.

Select the MAUI Template

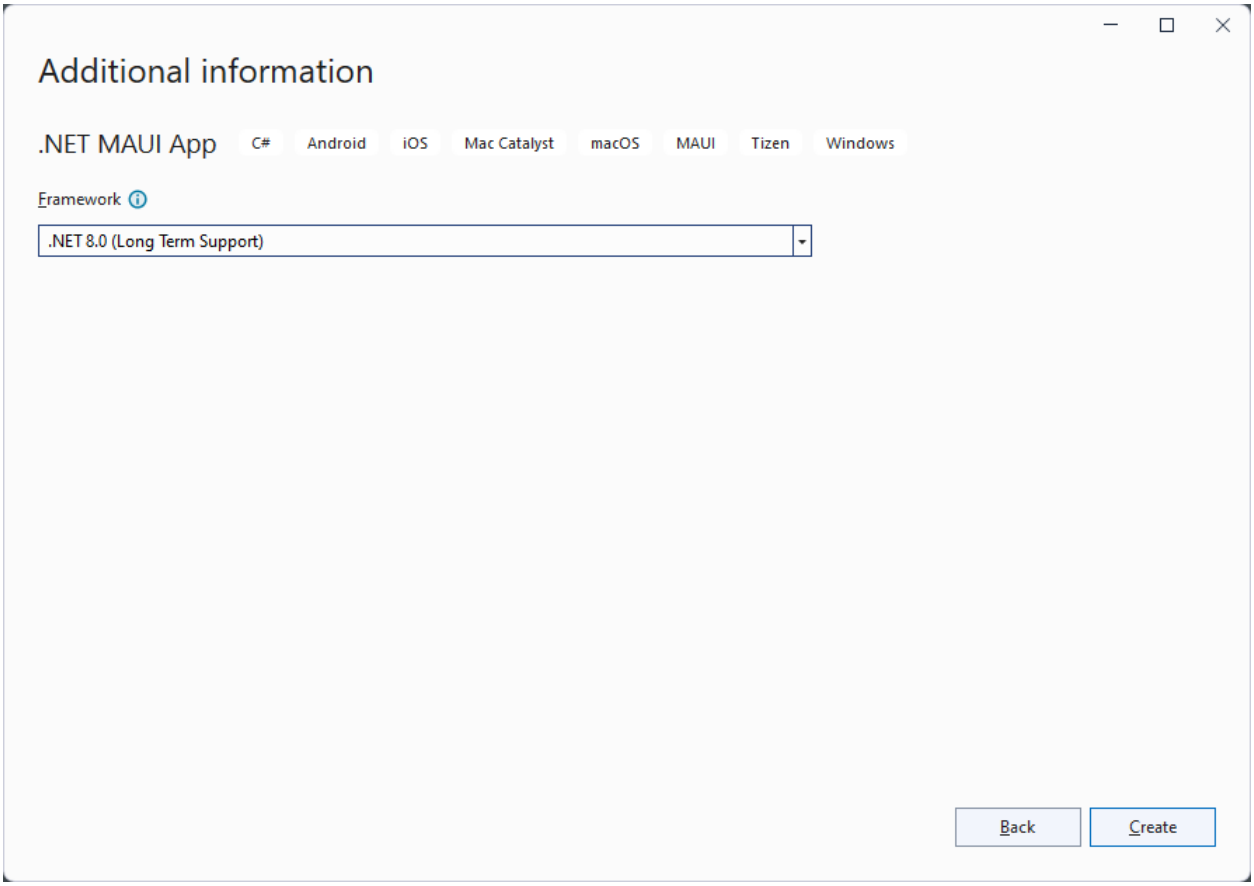


If the .Net MAUI App template is not available, use the Visual Studio installer to add the MAUI templates to Visual Studio on the workstation. Visual Studio 2022 v17.11.6 or greater is recommended.

Specify the Solution Name and Local Repository



Select the .Net Target Version and click the Create button:

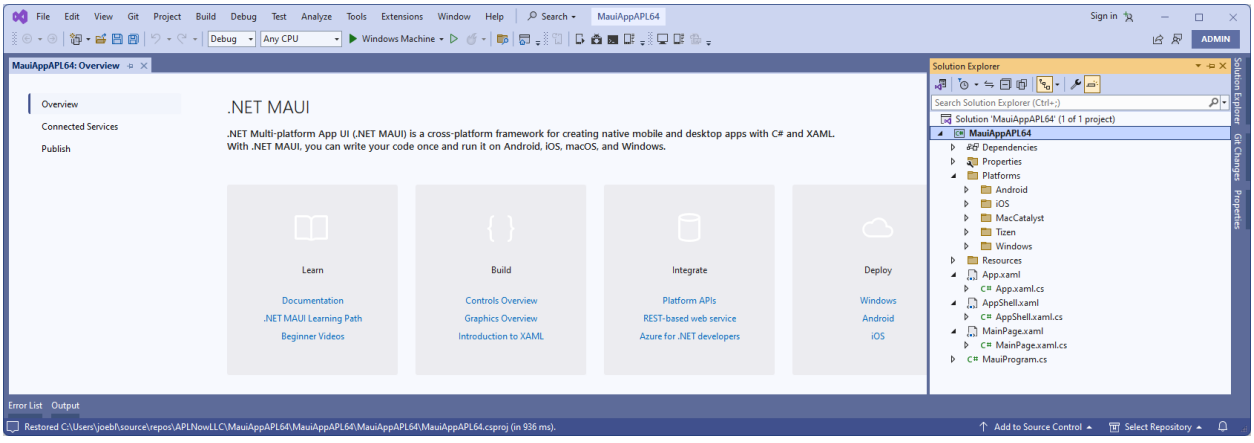


Edit the Solution Files

Close the 'Overview' Tab

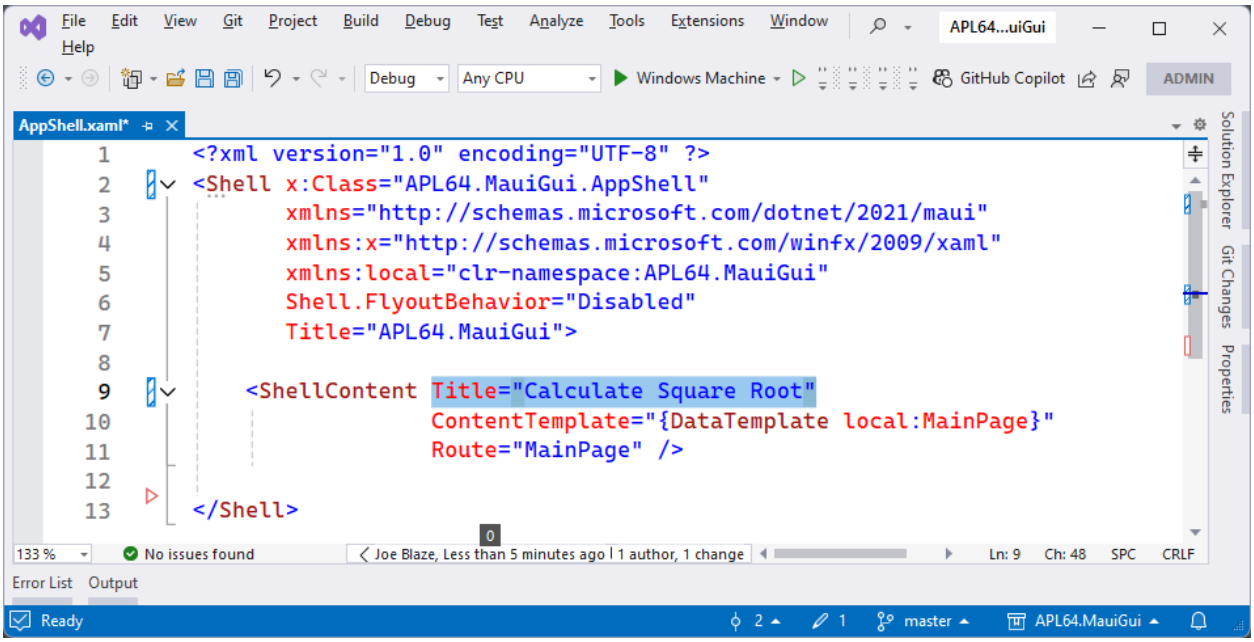
The 'Overview' tab may be closed as it is not needed for the application.

The Solution Explorer will be used to select the application components in the template for editing to complete the example application:



Edit the AppShell.xaml file

Change the Title property:

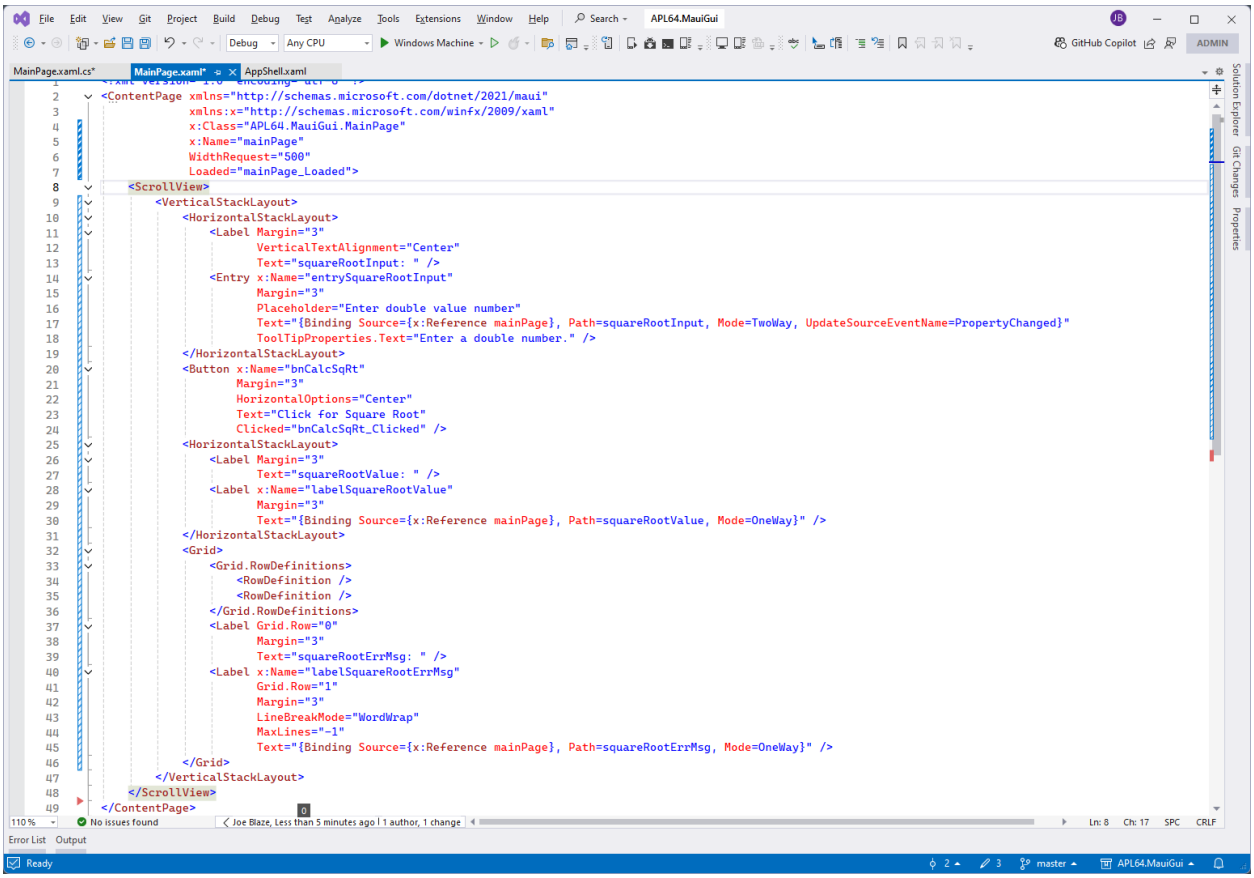


Edit the MainPage.xaml file

Replace the existing code in the MainPage.xaml file with this code:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="APL64.MauiGui.MainPage"
             x:Name="mainPage"
             WidthRequest="500"
             Loaded="mainPage_Loaded">
    <ScrollView>
        <VerticalStackLayout>
            <HorizontalStackLayout>
                <Label Margin="3"
                       VerticalTextAlignment="Center"
                       Text="squareRootInput: " />
                <Entry x:Name="entrySquareRootInput"
                       Margin="3"
                       Placeholder="Enter double value number"
                       Text="{Binding Source={x:Reference mainPage},
Path=squareRootInput, Mode=TwoWay, UpdateSourceEventName=PropertyChanged}"
                       ToolTipProperties.Text="Enter a double number." />
            </HorizontalStackLayout>
            <Button x:Name="bnCalcSqRt"
                    Margin="3"
                    HorizontalOptions="Center"
                    Text="Click for Square Root"
                    Clicked="bnCalcSqRt_Clicked" />
            <HorizontalStackLayout>
                <Label Margin="3"
                       Text="squareRootValue: " />
```

```
                <Label x:Name="labelSquareRootValue"
                       Margin="3"
                       Text="{Binding Source={x:Reference mainPage},
Path=squareRootValue, Mode=OneWay}" />
            </HorizontalStackLayout>
            <Grid>
                <Grid.RowDefinitions>
                    <RowDefinition />
                    <RowDefinition />
                </Grid.RowDefinitions>
                <Label Grid.Row="0"
                       Margin="3"
                       Text="squareRootErrMsg: " />
                <Label x:Name="labelSquareRootErrMsg"
                       Grid.Row="1"
                       Margin="3"
                       LineBreakMode="WordWrap"
                       MaxLines="-1"
                       Text="{Binding Source={x:Reference mainPage},
Path=squareRootErrMsg, Mode=OneWay}" />
            </Grid>
        </VerticalStackLayout>
    </ScrollView>
</ContentPage>
```



Observe the XML GUI controls on the MainPage:

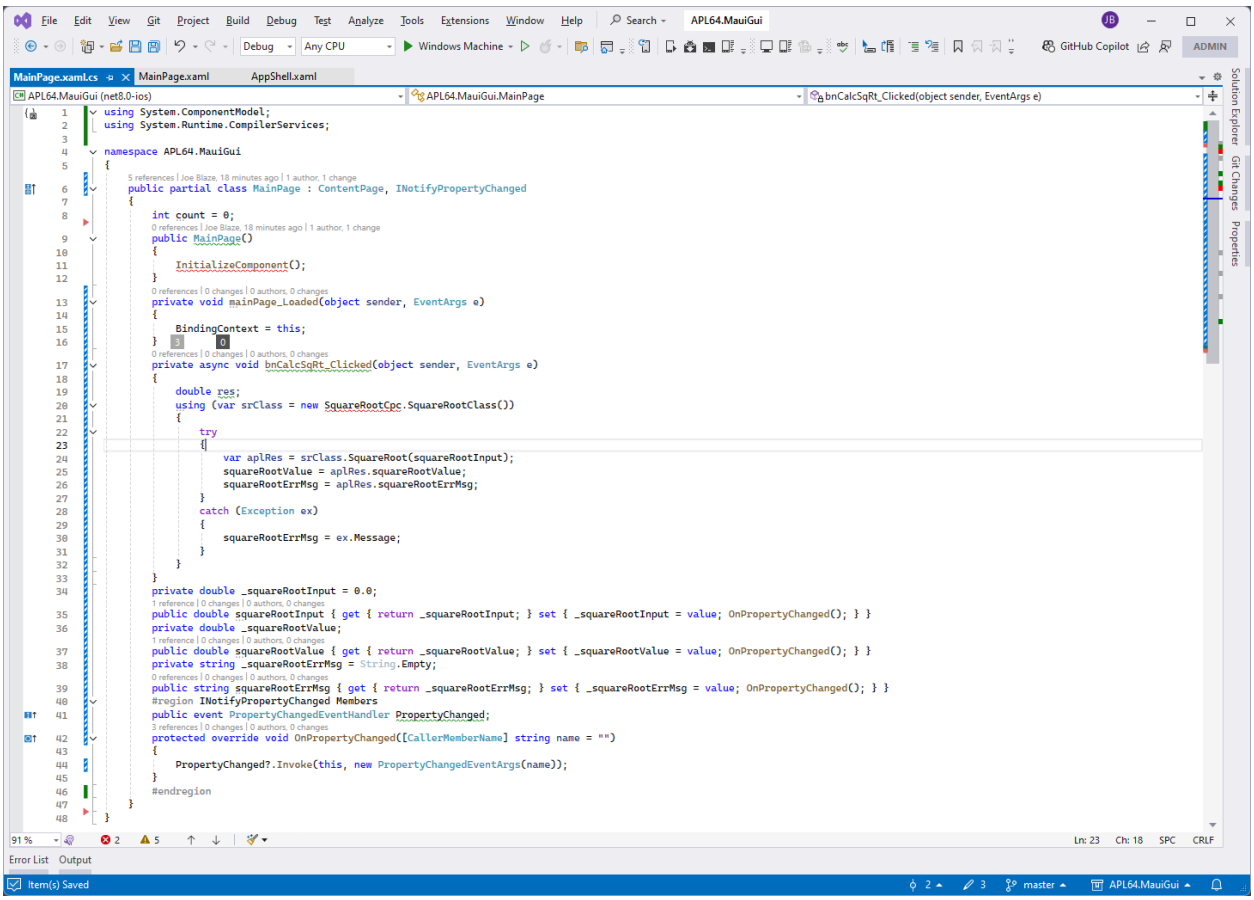
- Picker: User option to select the Calculation Source: Local or Server
- Entry: User entry field for double, numeric value
- Button: User option to calculate the square root of the value in the Entry control
- Labels: Display of result or error message

Edit the MainPage.xaml.cs file

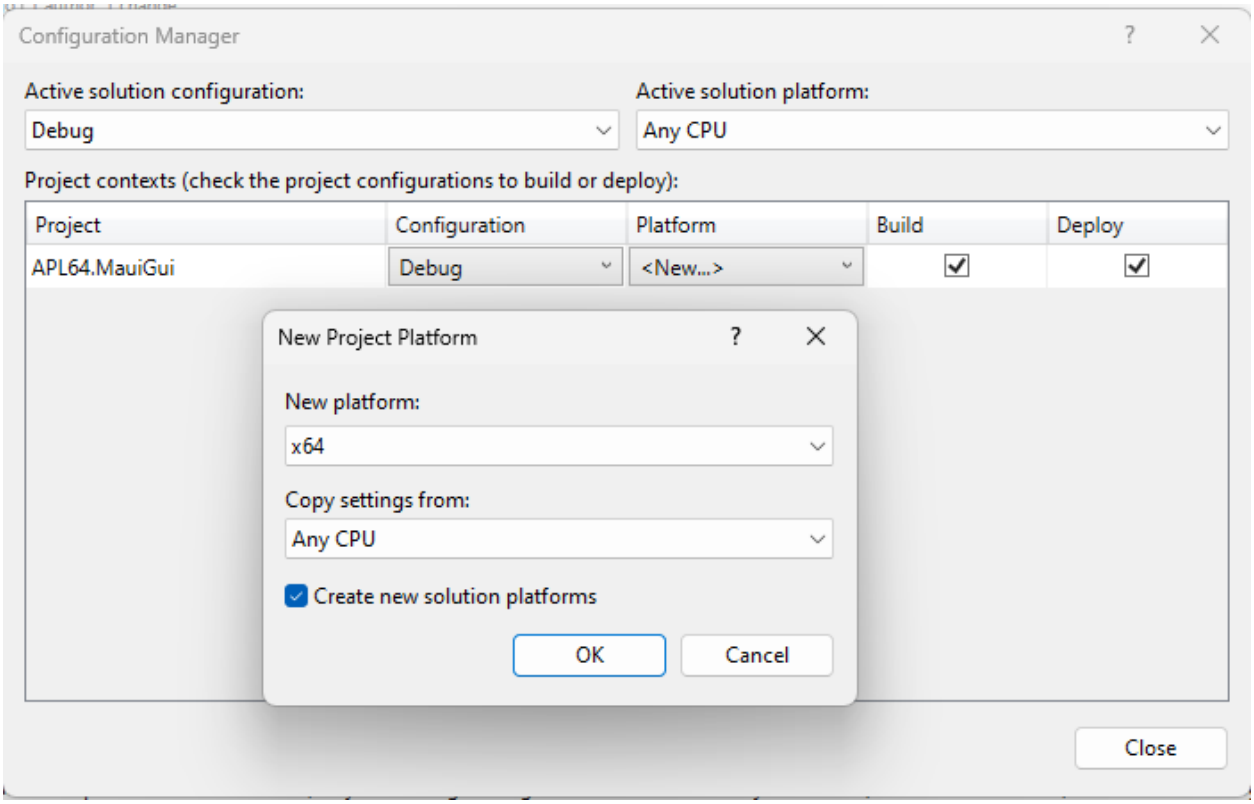
Replace the existing code in the MainPage.xaml.cs file with this code:

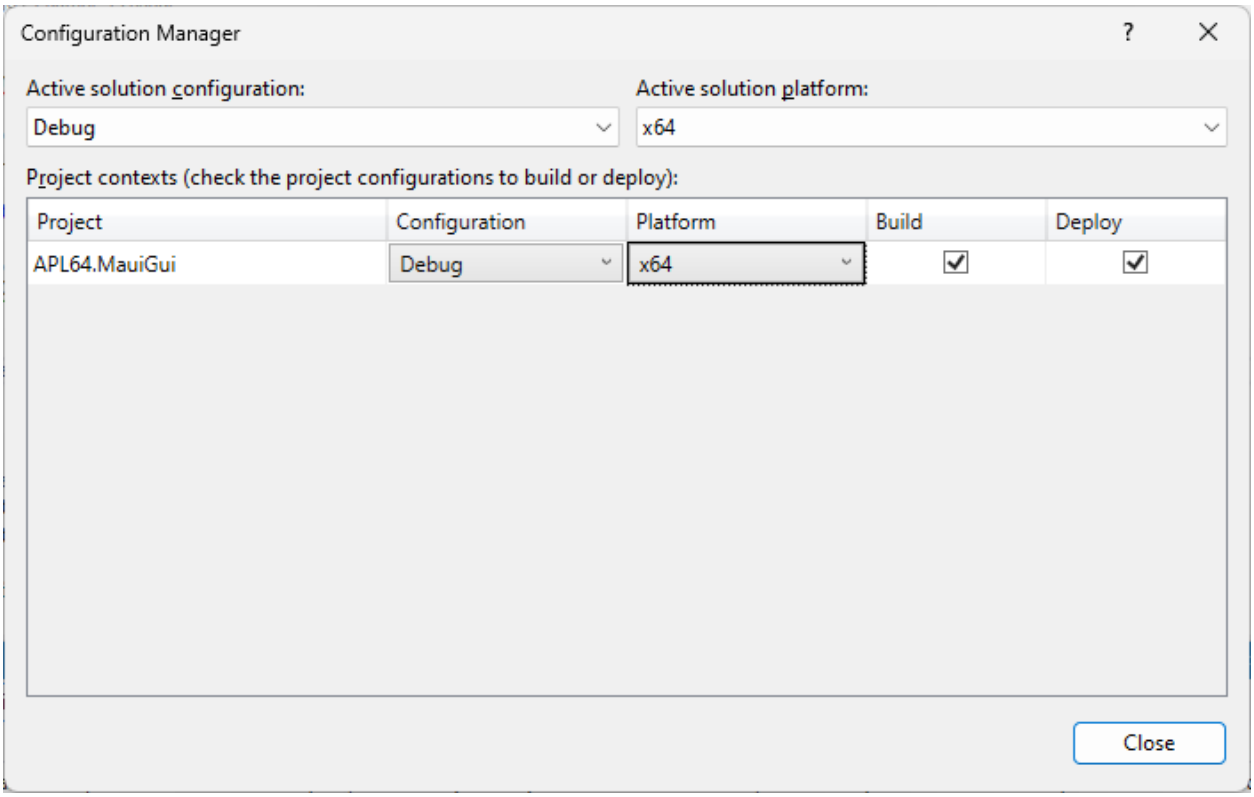
```
using System.ComponentModel;
using System.Runtime.CompilerServices;

namespace APL64.MauiGui
{
   public partial class MainPage : ContentPage, INotifyPropertyChanged
   {
      int count = 0;
      public MainPage()
      {
         InitializeComponent();
      }
      private void mainPage_Loaded(object sender, EventArgs e)
      {
         BindingContext = this;
      }
      private async void bnCalcSqRt_Clicked(object sender, EventArgs e)
      {
         double res;
         using (var srClass = new SquareRootCpc.SquareRootClass())
         {
            try
            {
               var aplRes = srClass.SquareRoot(squareRootInput);
               squareRootValue = aplRes.squareRootValue;
               squareRootErrMsg = aplRes.squareRootErrMsg;
            }
            catch (Exception ex)
            {
               squareRootErrMsg = ex.Message;
            }
         }
      }
      private double _squareRootInput = 0.0;
      public double squareRootInput { get { return _squareRootInput; } set { _squareRootInput = value;
OnPropertyChanged(); } }
      private double _squareRootValue;
```

```
    public double squareRootValue { get { return _squareRootValue; } set { _squareRootValue = value;
OnPropertyChanged(); } }
    private string _squareRootErrMsg = String.Empty;
    public string squareRootErrMsg { get { return _squareRootErrMsg; } set { _squareRootErrMsg =
value; OnPropertyChanged(); } }
    #region INotifyPropertyChanged Members
    public event PropertyChangedEventHandler PropertyChanged;
    protected override void OnPropertyChanged([CallerMemberName] string name = "")
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));
    }
    #endregion
  }
}
```



Set the Solution Build Configuration to x64

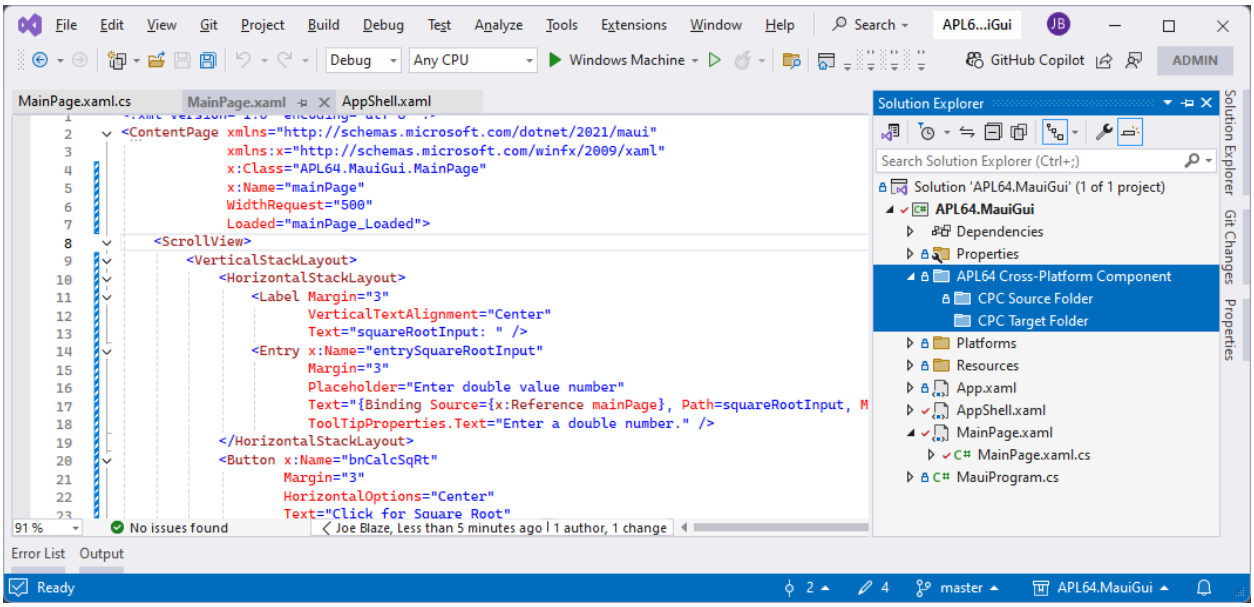




Implement the Local APL64 Cross-platform Component

For purposes of this document the local APL64 CPC implementation files are contained in the 'APL64 Cross-Platform Component' folder of the Visual Studio solution. This folder contains the 'CPC Source Folder' and the 'CPC Target Folder'.

Prepare the APL64 CPC Folders in the Solution



Prepare an APL64 Workspace with an APL64 Public Function:

An APL64 programmer-defined, public function will support the square root algorithm in the App when the 'Calculation Source' is user-selected as 'APL64 CPC Local'. The 'APL64 Cross-Platform Component\CPC Source Folder\SquareRootCpc.ws64' workspace contains the required public function.

The APL64 public function, SquareRoot, has a single right argument, inputDouble, specified as a Double numeric value and three result values:

- squareRootValue, specified as a Double numeric value
- squareRootHasError, specifies as a Boolean value
- squareRootErrorMsg, specified as a String value

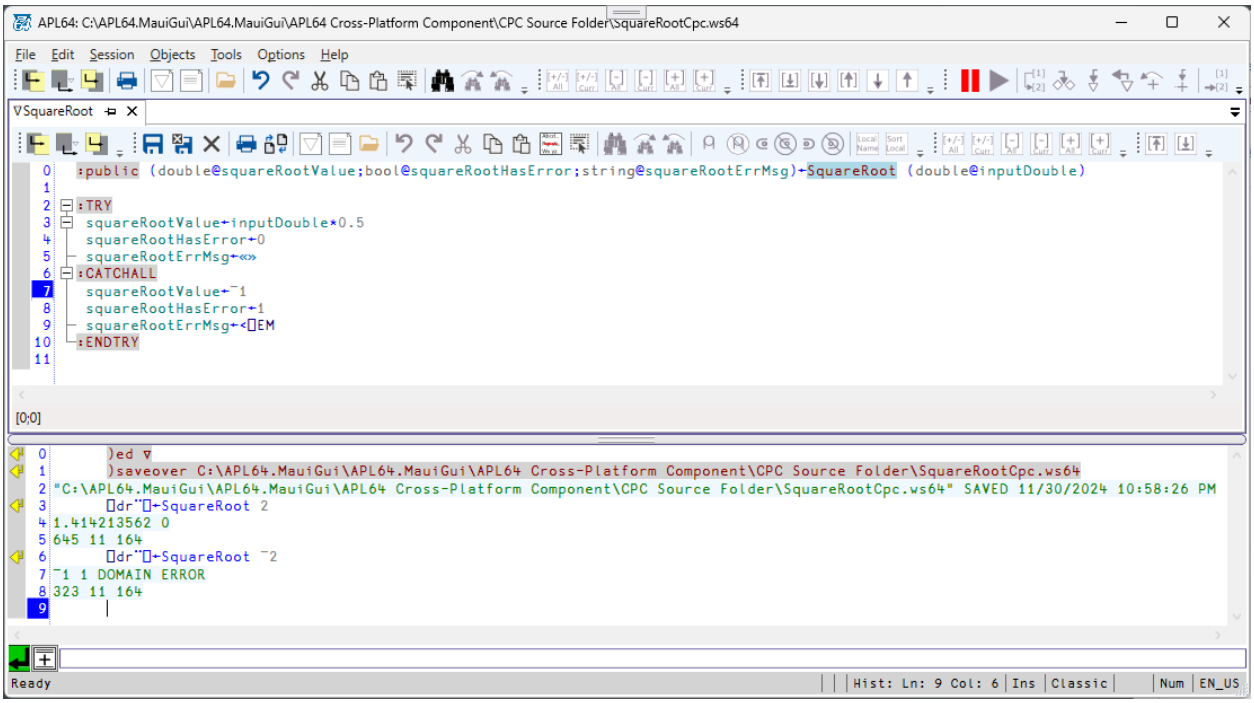
When the calculation of the squareRootValue occurs without an exception, the squareRootErrMsg is assigned an empty String, «». If the calculation of the squareRootValue results in an exception, the squareRootErrMsg is assigned the string value of the APL64 error message system variable, <⎕EM.

```
:public
(double@squareRootValue;bool@squareRootHasError;string@squareRootErrMsg)←SquareRoot
(double@inputDouble)

:TRY
 squareRootValue←inputDouble*0.5
```

```
 squareRootHasError←0
 squareRootErrMsg←«»
:CATCHALL
 squareRootValue←¯1
 squareRootHasError←1
 squareRootErrMsg←<□EM
:ENDTRY
```

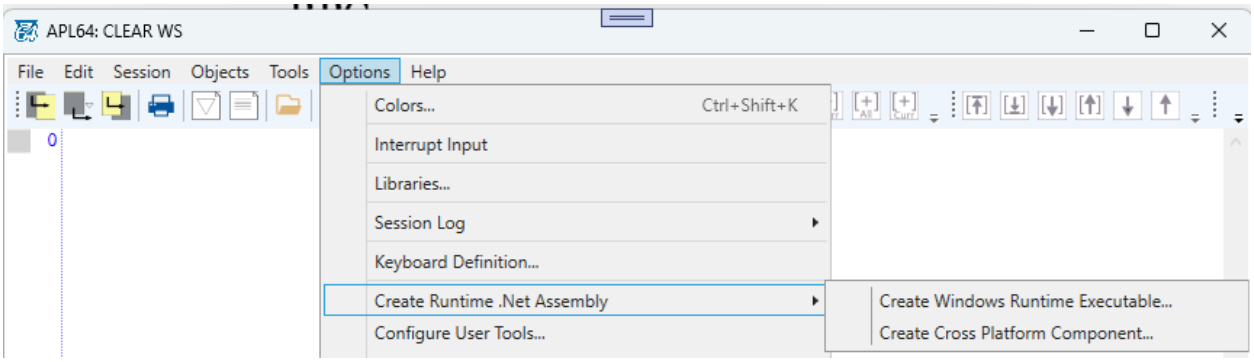
The workspace name is case sensitive because it is part of the name of the .Net Nuget package for the APL64 CPC.

Test the APL64 Public Function



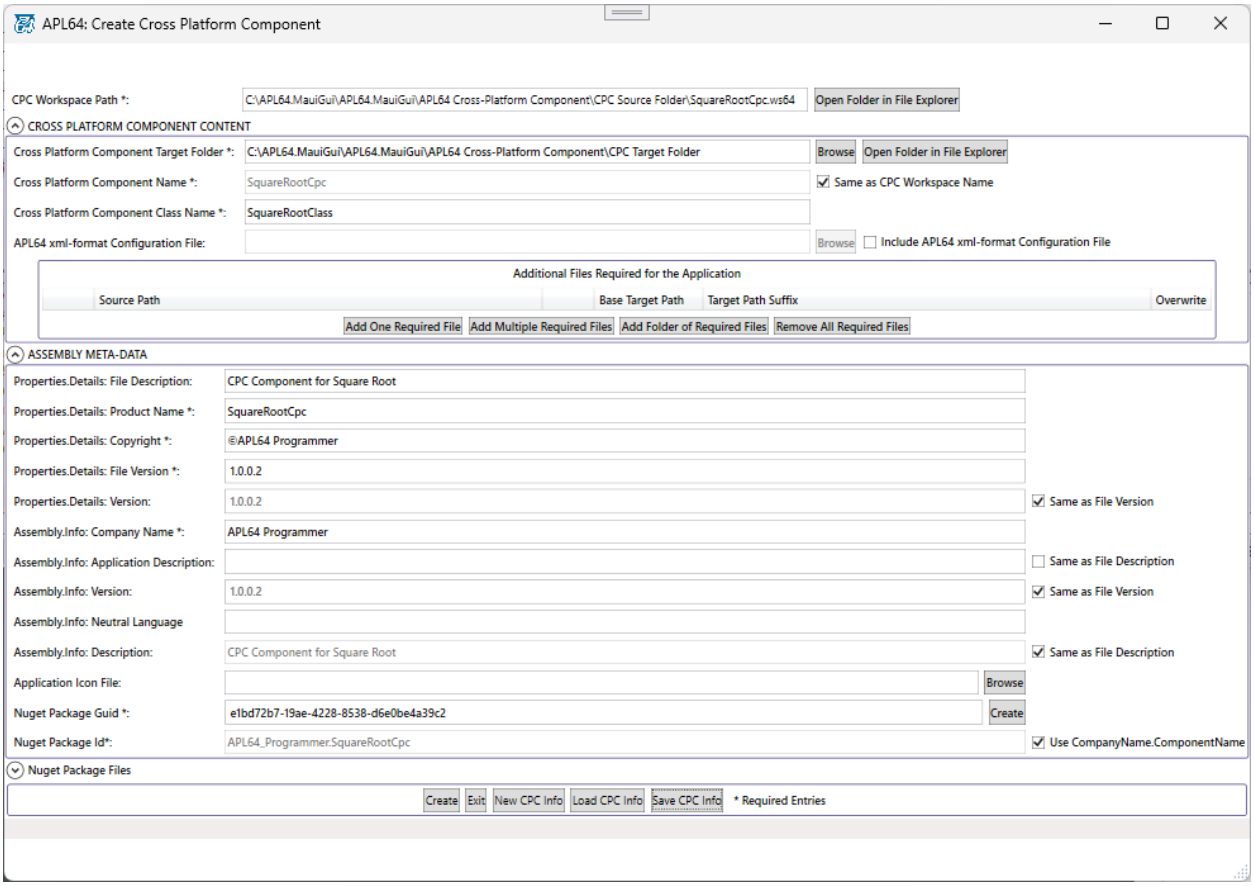
Use the APL64 Cross-platform Component Utility

Use Options | Create Runtime .Net Assembly | Create Cross-Platform Component utility in the APL64 Developer version:

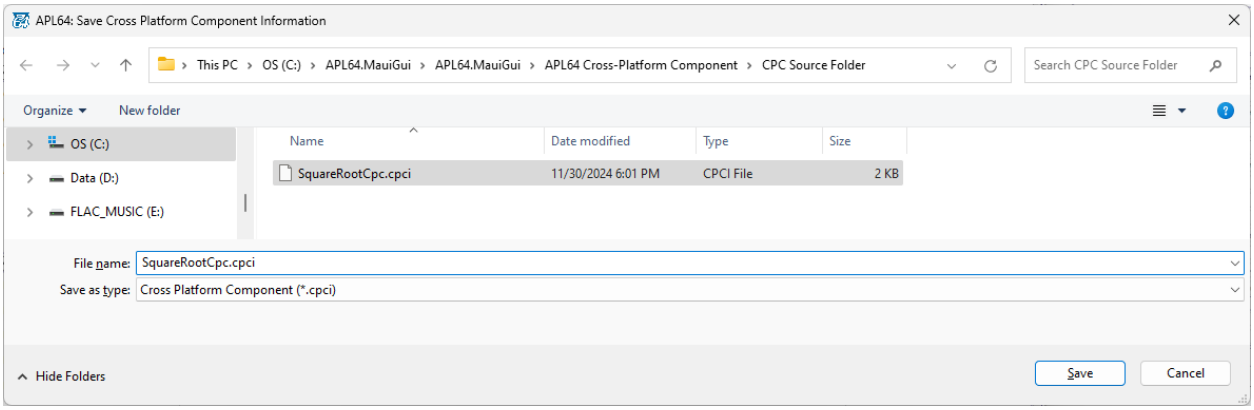


Complete the CPCI Fields Appropriately

These fields include the ‘CPC Workspace Path’, ‘Cross Platform Component Target Folder’, ‘Nuget Package Id’ and the ‘File Version’ Entries.

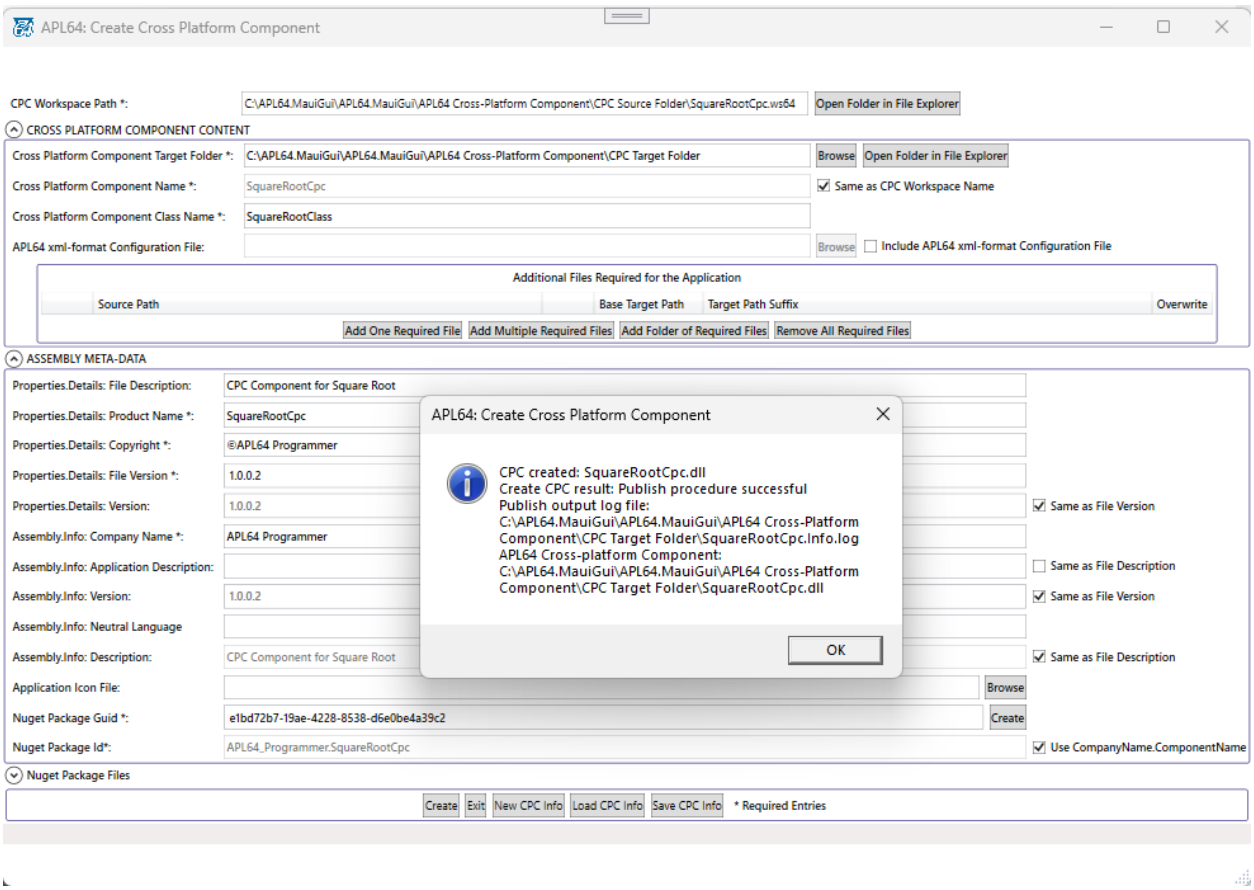


Save the CPC Configuration File



Create the Cross-platform Component

Click the Create button and observe the successful creation of the CPC:

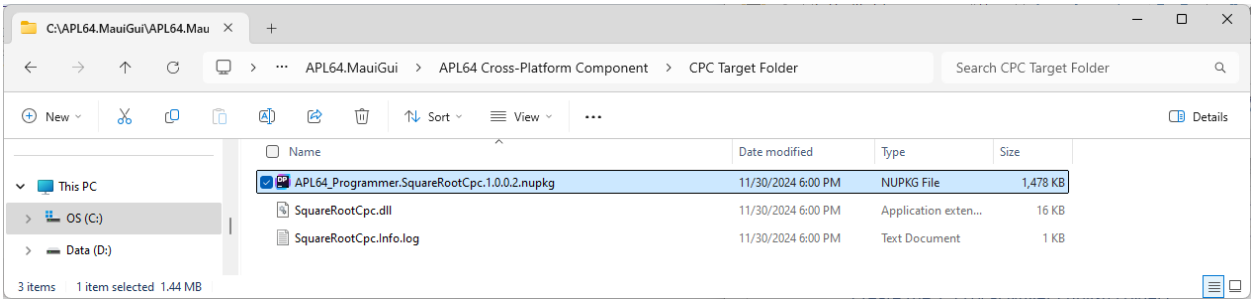


Create the 'C:\APL64 Nuget Packages\'

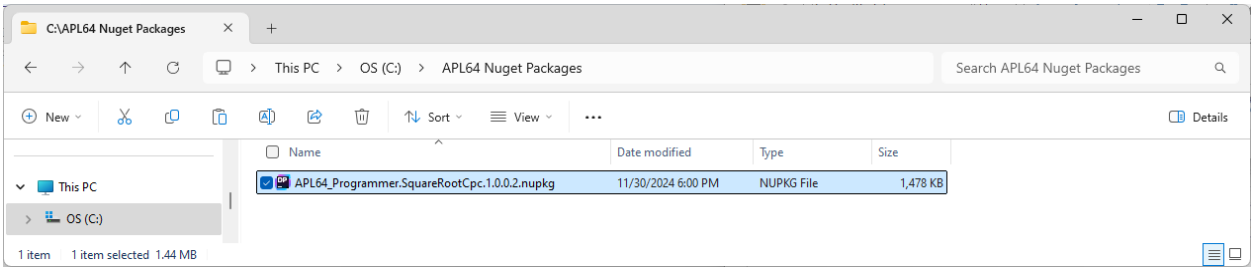
This folder should not be in the MauiAppAPL64 solution folder.

Copy the APL64 CPC Nuget Package to a Local Nuget Publish Folder

Copy the 'APL64Programmer.SquareRootCpc.1.0.0.nupkg' Nuget package from the CPC Target Folder:

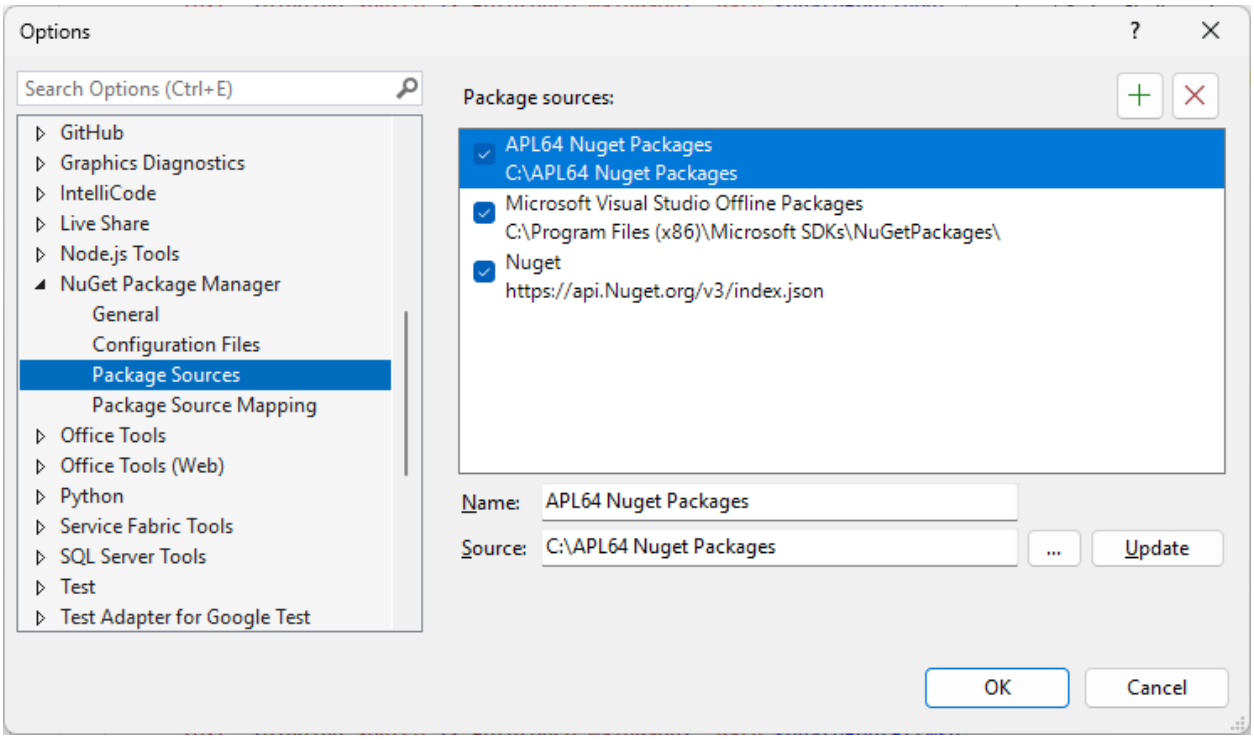


Paste the 'APL64Programmer.SquareRootCpc.1.0.0.nupkg' Nuget package to the 'c:\ Local Nuget Publish Folder\':



In VS2022 Recognize the Local Nuget Publish Folder

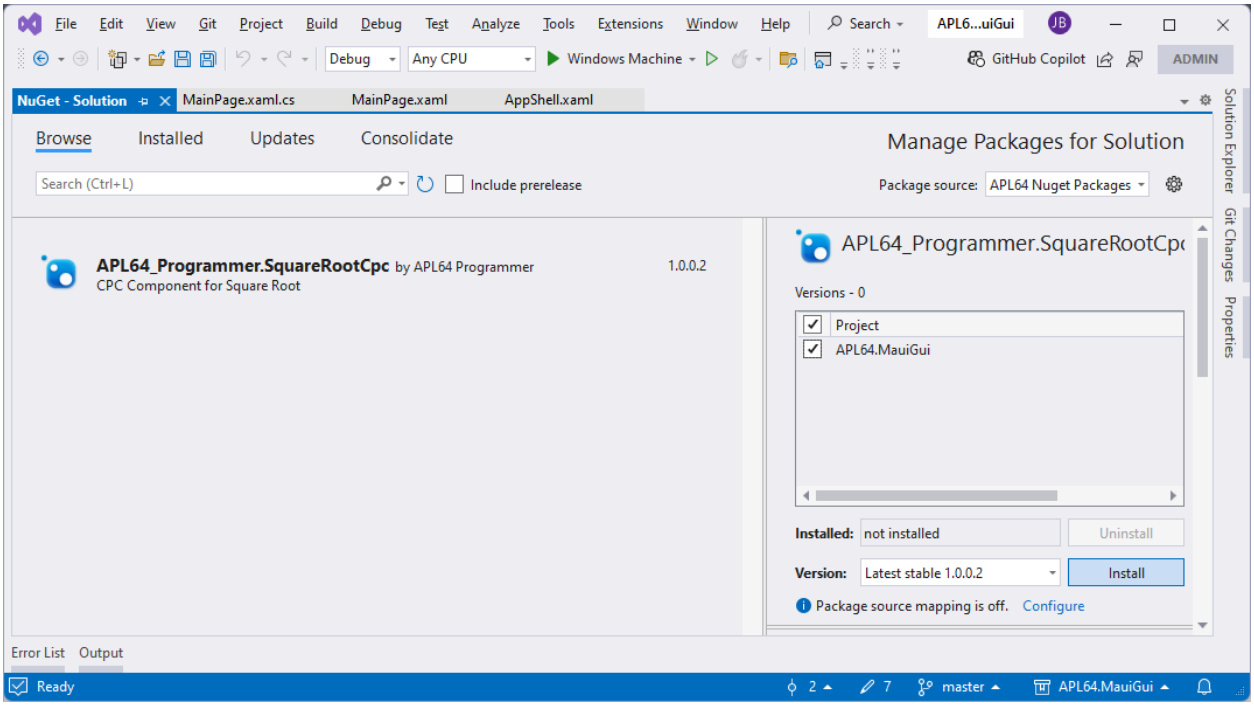
In Visual Studio use: Tools | Nuget Package Manager | Package Manager Settings | Package Sources | + (Add package source) to specify the Local Nuget Publish folder:



Incorporate the Local APL64 CPC in the Application

Reference the APL64 CPC

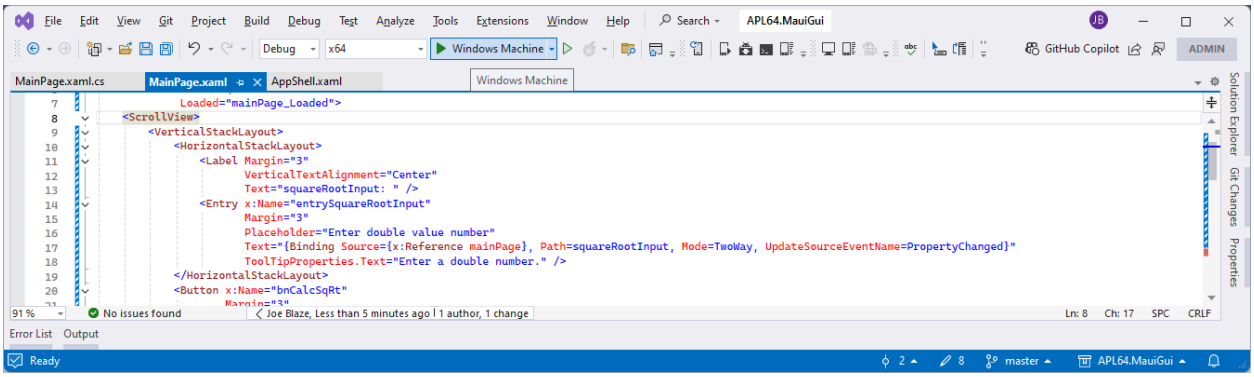
Use the VS2022 Tools | Nuget Package Manager | Nuget Package Manager for Solution dialog to add a reference to the APL64 CPC:

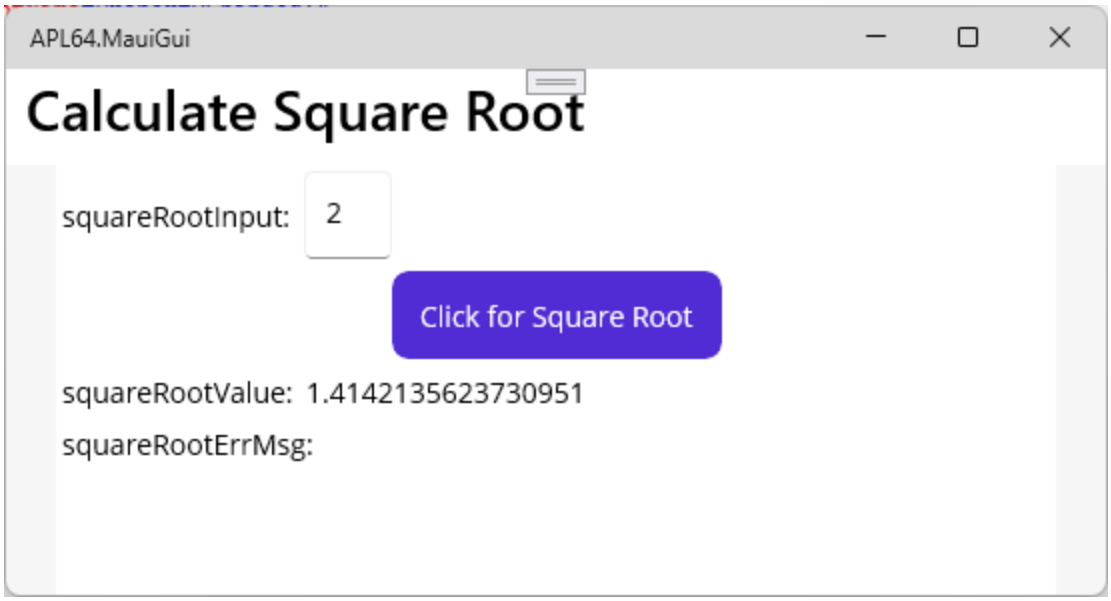


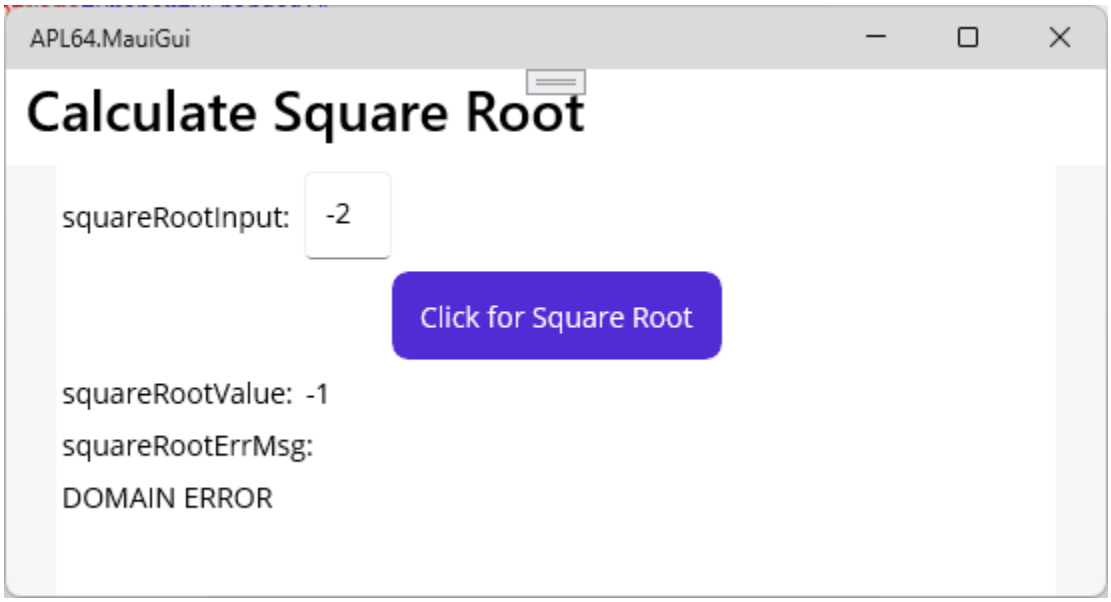
Test the Application

Test in Windows

Select the Windows Machine debug option in Visual Studio 2022:



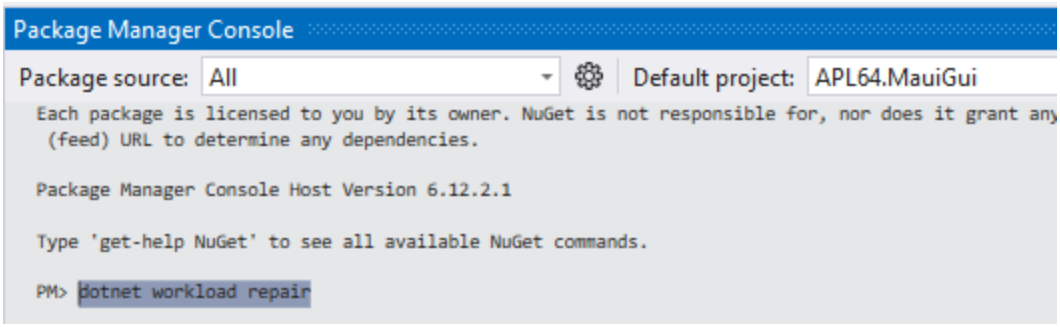




Test in Android

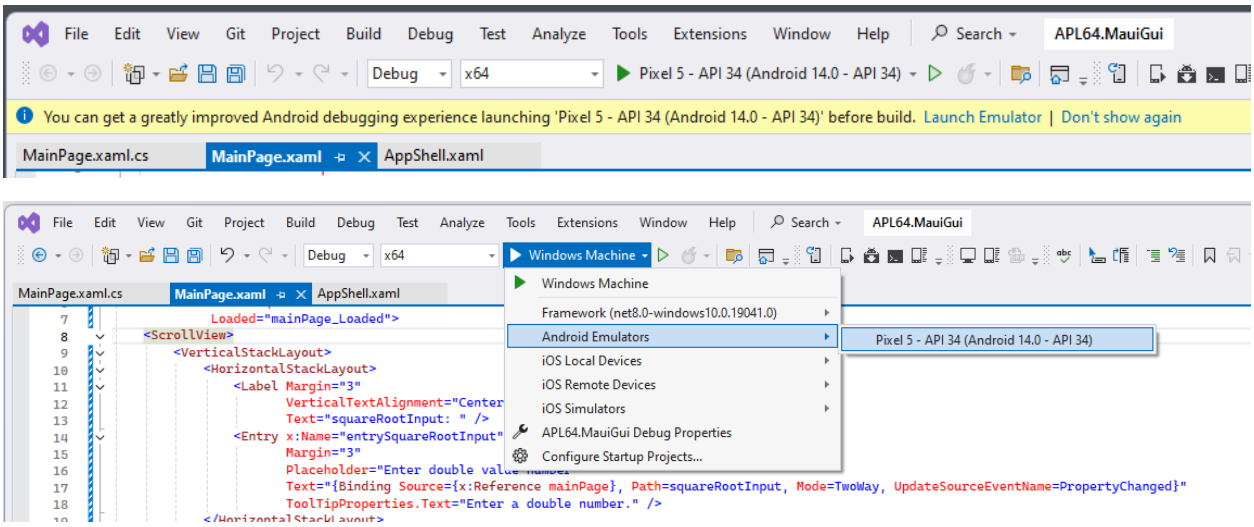
It may be necessary to [configure the programmer's workstation for Hyper-V] to run an Android emulator.

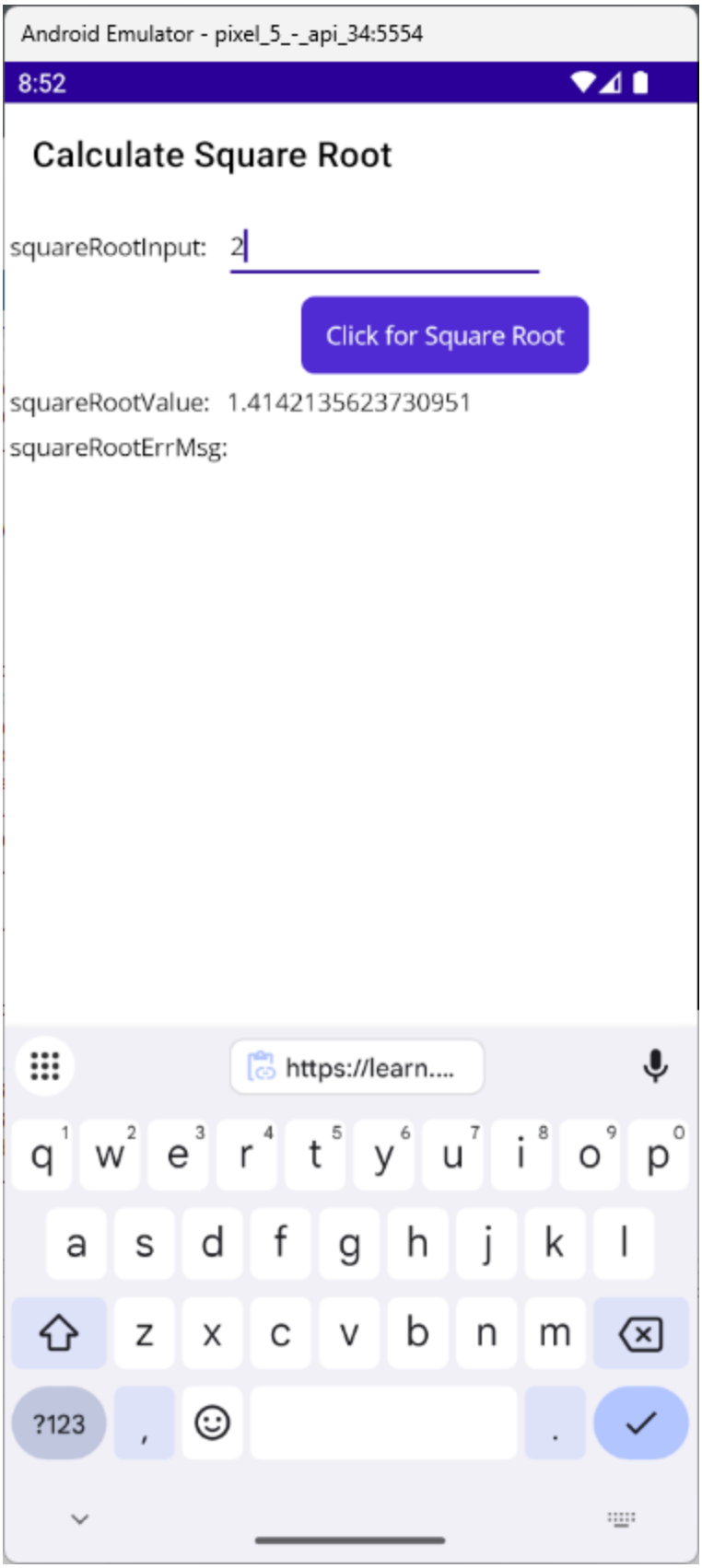
It may be necessary to repair the Visual Studio 'workloads' for Android emulators, by running the 'dotnet workload repair' command in the Visual Studio Tools | NuGet Package Manager | Package Manager Console.



Select an appropriate Android version in Visual Studio 2022.

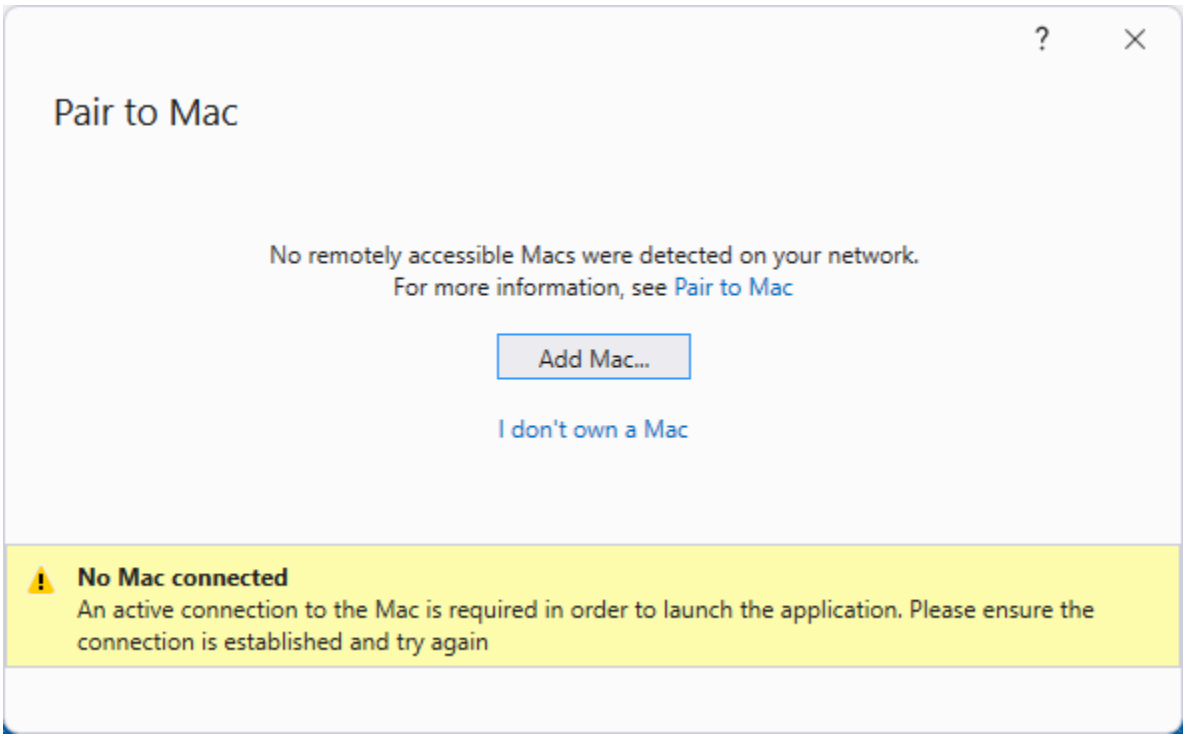
Depending on the programming environment, start up of the emulator can take considerable time. Visual Studio may provide an option to launch the Android emulator before running the application to improve testing performance:





Test in iOS

After selecting a Mac/Apple option to debug, Visual Studio may present the dialog to connect a Mac/Apple device to the programming workstation:



If a Mac/Apple device is not available to connect to the programming workstation, click the 'I don't own a Mac' link in the dialog. It may be possible to setup a virtual iOS on the programming workstation, but this procedure is beyond the scope of this example.

More Information

MAUI Documentation

MAUI Tutorial

MAUI Videos

MAUI GUI Controls

MAUI Graphics

MAUI XAML

Access to Platform API

Deployment: Windows, Android, iOS