Unit code & name: SWE40006 – Software Deployment and Evolution

Student Name: Avish Kishore

Student ID: 104187075

Semester: Semester 2, 2025
Due date: 24 Aug 2025, 23:59
Submission date: 25 Aug 2025

Declaration of task level attempted

For this portfolio submission, I have attempted Task 1.1 (Pass), Task 1.2 (Credit), Task 1.3 (Distinction), and Task 1.4 (High Distinction).

Task 1.1 – Followed the WiX walkthrough to deploy a sample desktop application.

Task 1.2 – Extended the walkthrough by creating and deploying my own C# WinForms desktop app.

Task 1.3 – Deployed an application that included multiple DLLs (MathLib and HelpersLib) packaged together with the EXE, including shortcuts.

Task 1.4 – Explained in detail the process of preparing and deploying the application to the Microsoft Store using Visual Studio 2022 and Partner Center.

This declaration confirms that I have completed the full sequence of subtasks (1.1 to 1.4) in order to demonstrate deployment at increasing levels of complexity.

Overview

This report documents my completion of **WiX Toolset desktop deployment** using **Visual Studio Community 2022** and **WiX Toolset v3.11.2**. I followed the provided WiX walkthrough, created a sample WinForms desktop app, authored a WiX v3 installer, built an **MSI**, and verified installation on Windows 11. I also captured issues encountered and my fixes.

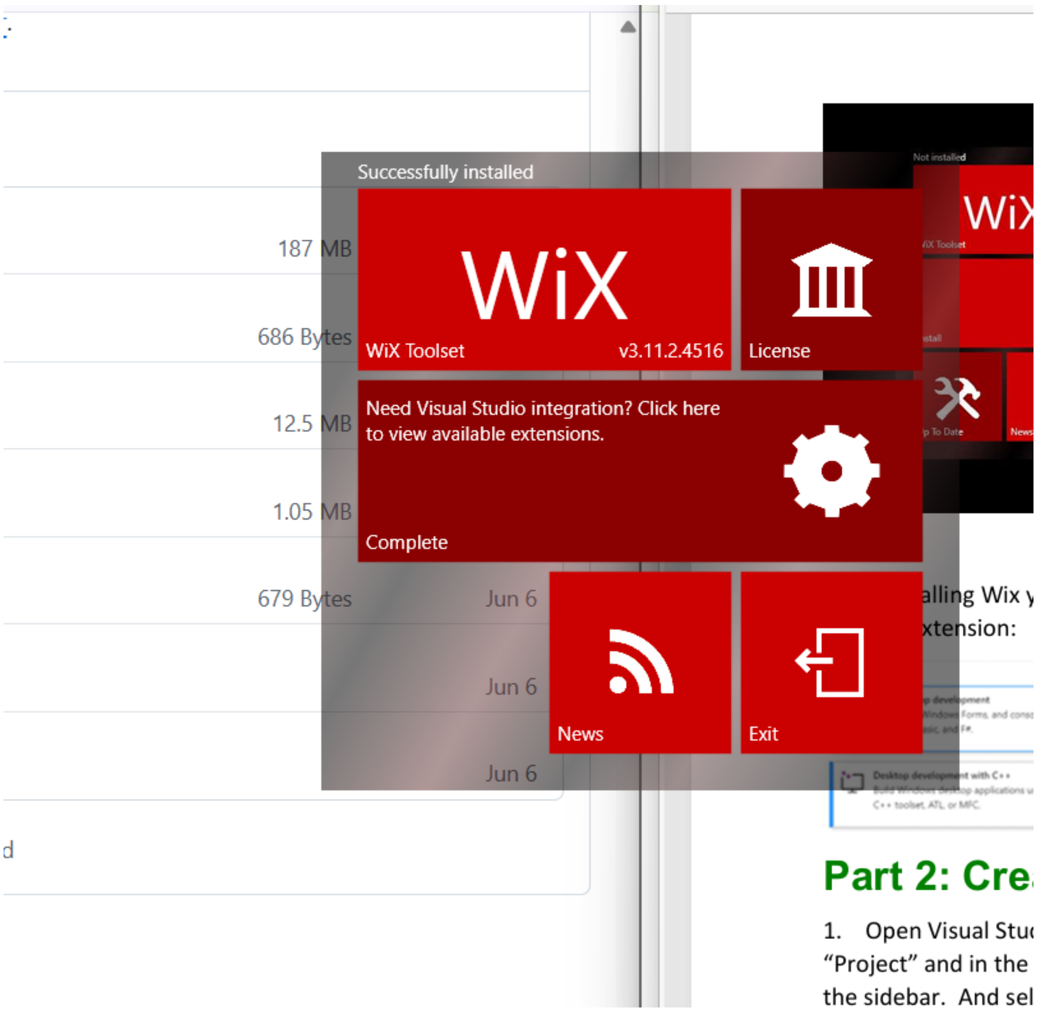
Environment

- **OS:** Windows 11 (x64)
- **IDE:** Visual Studio Community 2022
- **Workloads installed:** .NET desktop development; Desktop development with C++; Universal Windows Platform development
- **WiX Toolset:** v3.11.2 (runtime) + **WiX Toolset Visual Studio 2022 Extension** (Votive)
- **.NET Framework for app:** 4.6.2 (compatible)

TASK1.1

1) Install WiX Toolset 3.11.2 and VS extension

- Installed WiX Toolset runtime (**wix311**) and then the **WiX Toolset Visual Studio 2022 Extension** via *Extensions → Manage Extensions*.

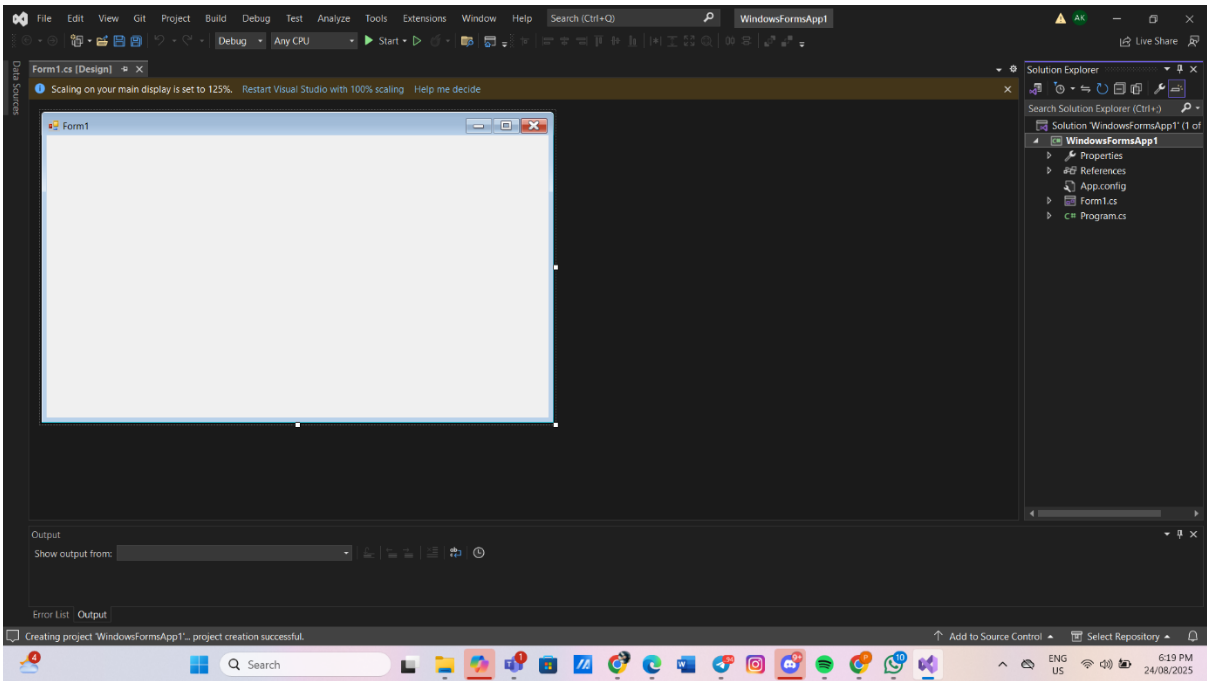


- **Evidence – Figure 1:** WiX Toolset 3.11.2 installer success screen.

2) Create a sample desktop application (WinForms)

- New project: **Windows Forms App (.NET Framework)** named ExampleDesktop targeting **.NET Framework 4.6.2**.

- Confirmed it builds and runs (blank form launched).



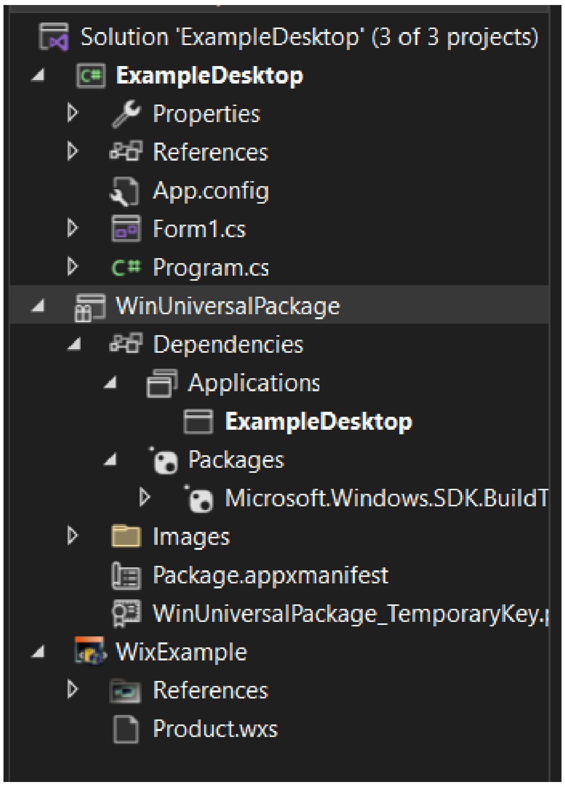
Scaling on your main display is set to 125%. Restart Visual Studio with 100% scaling Help me decide

Creating project 'WindowsFormsApp1'... project creation successful.

- **Evidence – Figure 2:** ExampleDesktop running (screenshot of window).

3) Add a WiX v3 Setup Project and reference the app

- Added a new project **Setup Project for WiX v3** named WiXExample.
- In WiXExample, added a **Project Reference** to ExampleDesktop.

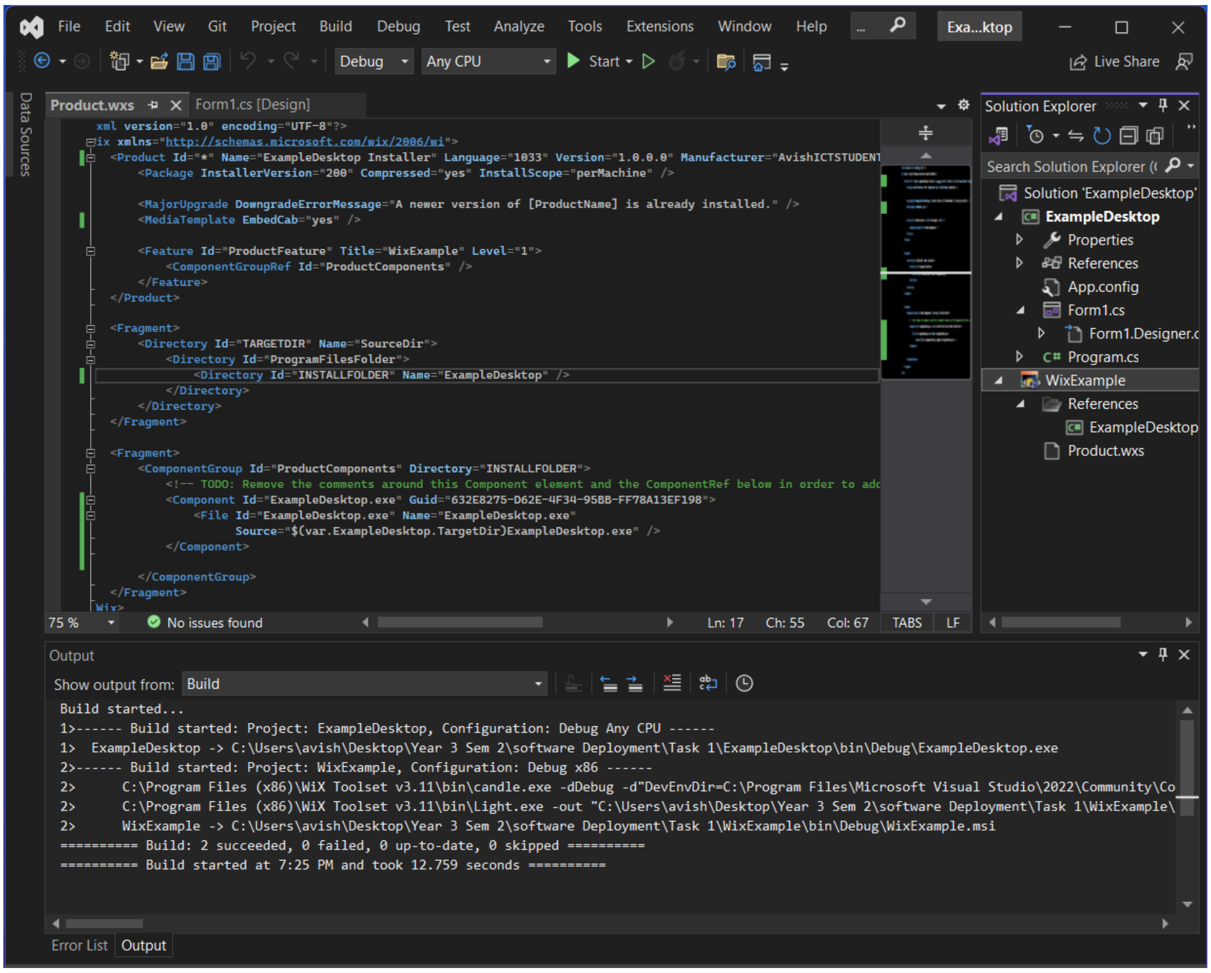


```
Solution 'ExampleDesktop' (3 of 3 projects)
◢ ExampleDesktop
    ▷ Properties
    ▷ References
      App.config
    ▷ Form1.cs
    ▷ Program.cs
◢ WinUniversalPackage
    ◢ Dependencies
        ◢ Applications
              ExampleDesktop
        ◢ Packages
            ▷ Microsoft.Windows.SDK.BuildT
    ▷ Images
      Package.appxmanifest
      WinUniversalPackage_TemporaryKey.
◢ WixExample
    ▷ References
      Product.wxs
```

- **Evidence – Figure 3:** Solution Explorer showing ExampleDesktop + WiXExample.

4) Author installer contents (Product.wxs)

I edited Product.wxs to include the built EXE and make a single-file MSI (embedded CAB). Key fragment:



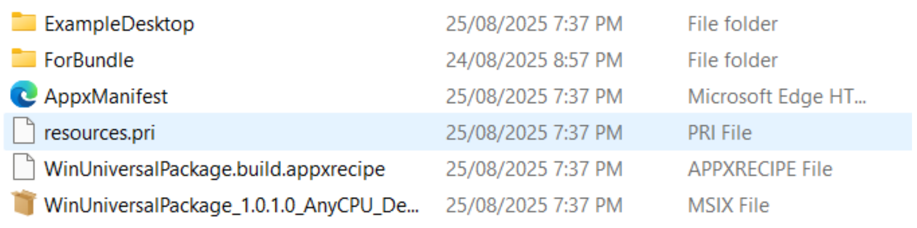
```
 xml version="1.0" encoding="UTF-8"?>
ix xmlns="http://schemas.microsoft.com/wix/2006/wi">
  <Product Id="*" Name="ExampleDesktop Installer" Language="1033" Version="1.0.0.0" Manufacturer="AvishICTSTUDENT
    <Package InstallerVersion="200" Compressed="yes" InstallScope="perMachine" />

    <MajorUpgrade DowngradeErrorMessage="A newer version of [ProductName] is already installed." />
    <MediaTemplate EmbedCab="yes" />

    <Feature Id="ProductFeature" Title="WixExample" Level="1">
      <ComponentGroupRef Id="ProductComponents" />
    </Feature>
  </Product>

  <Fragment>
    <Directory Id="TARGETDIR" Name="SourceDir">
      <Directory Id="ProgramFilesFolder">
        <Directory Id="INSTALLFOLDER" Name="ExampleDesktop" />
      </Directory>
    </Directory>
  </Fragment>

  <Fragment>
    <ComponentGroup Id="ProductComponents" Directory="INSTALLFOLDER">
      <!-- TODO: Remove the comments around this Component element and the ComponentRef below in order to adc
      <Component Id="ExampleDesktop.exe" Guid="632E8275-D62E-4F34-95BB-FF78A13EF198">
        <File Id="ExampleDesktop.exe" Name="ExampleDesktop.exe"
              Source="$(var.ExampleDesktop.TargetDir)ExampleDesktop.exe" />
      </Component>

    </ComponentGroup>
  </Fragment>
Wix>
```

```
Build started...
1>------ Build started: Project: ExampleDesktop, Configuration: Debug Any CPU ------
1>  ExampleDesktop -> C:\Users\avish\Desktop\Year 3 Sem 2\software Deployment\Task 1\ExampleDesktop\bin\Debug\ExampleDesktop.exe
2>------ Build started: Project: WixExample, Configuration: Debug x86 ------
2>      C:\Program Files (x86)\WiX Toolset v3.11\bin\candle.exe -dDebug -d"DevEnvDir=C:\Program Files\Microsoft Visual Studio\2022\Community\Co
2>      C:\Program Files (x86)\WiX Toolset v3.11\bin\Light.exe -out "C:\Users\avish\Desktop\Year 3 Sem 2\software Deployment\Task 1\WixExample\
2>      WixExample -> C:\Users\avish\Desktop\Year 3 Sem 2\software Deployment\Task 1\WixExample\bin\Debug\WixExample.msi
========== Build: 2 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
========== Build started at 7:25 PM and took 12.759 seconds ==========
```

- **Evidence – Figure 4:** Editor showing Product.wxs with the <File Source="$(var.ExampleDesktop.TargetDir)ExampleDesktop.exe" /> entry.

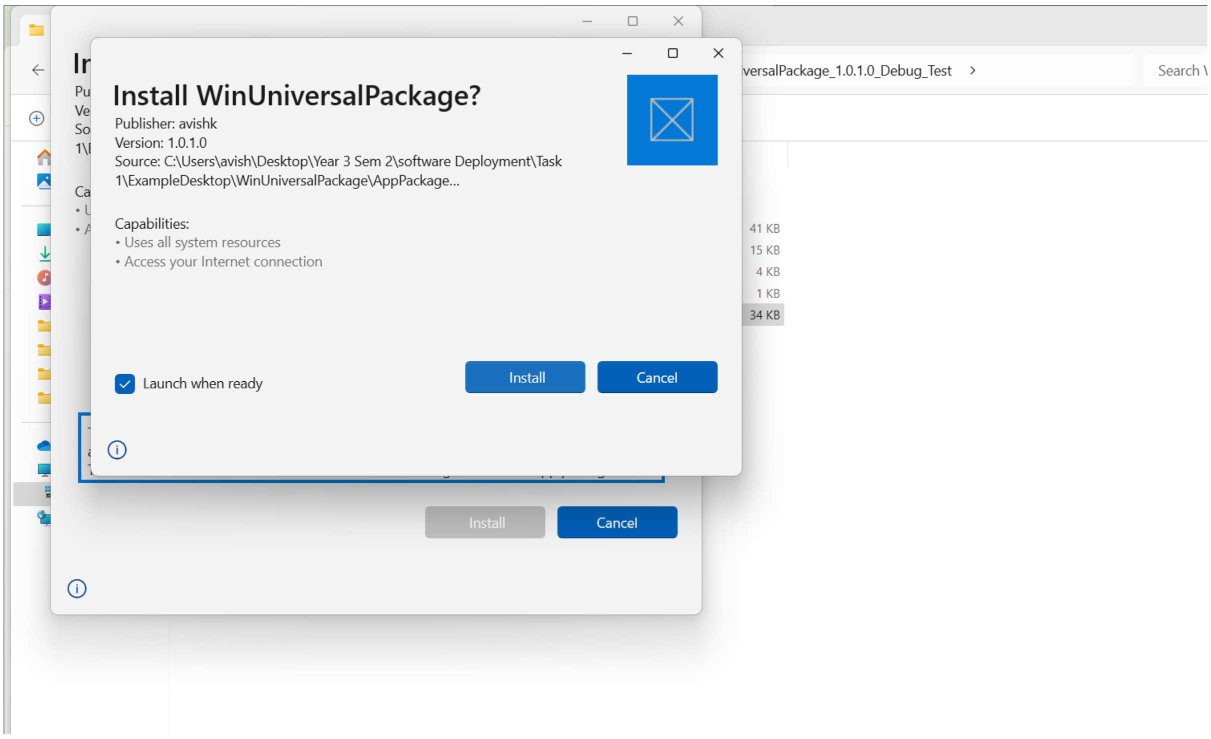
5) Build the MSI and verify installation

- Built the WiXExample project in **Release** configuration.
- Output MSI located at: WiXExample\bin\Release\WiXExample.msi.
- Executed the MSI → followed the install wizard → application installed to C:\Program Files (x86)\ExampleDesktop (default).
- Launched the installed app from **Start** menu / **Apps & Features** entry confirmed.



ExampleDesktop	25/08/2025 7:37 PM	File folder
ForBundle	24/08/2025 8:57 PM	File folder
AppxManifest	25/08/2025 7:37 PM	Microsoft Edge HT...
resources.pri	25/08/2025 7:37 PM	PRI File
WinUniversalPackage.build.appxrecipe	25/08/2025 7:37 PM	APPXRECIPE File
WinUniversalPackage_1.0.1.0_AnyCPU_De...	25/08/2025 7:37 PM	MSIX File

- **Evidence – Figure 5:** Build output showing MSI creation in bin\Release.

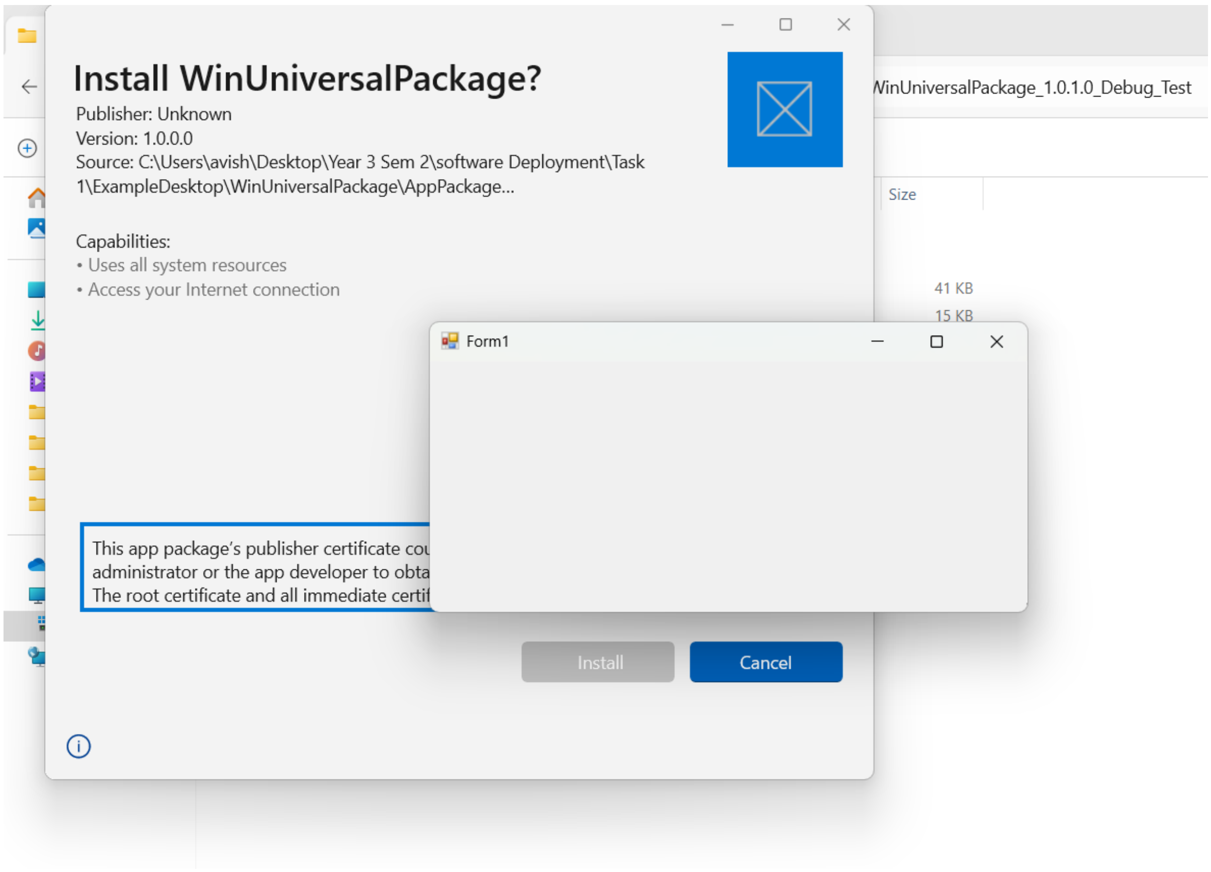


Install WinUniversalPackage?

Publisher: avishk
Version: 1.0.1.0
Source: C:\Users\avish\Desktop\Year 3 Sem 2\software Deployment\Task 1\ExampleDesktop\WinUniversalPackage\AppPackage...

Capabilities:
• Uses all system resources
• Access your Internet connection

- **Evidence – Figure 6:** Installer running / completed successfully.



Install WinUniversalPackage?

Publisher: Unknown
Version: 1.0.0.0
Source: C:\Users\avish\Desktop\Year 3 Sem 2\software Deployment\Task 1\ExampleDesktop\WinUniversalPackage\AppPackage...

Capabilities:
• Uses all system resources
• Access your Internet connection

This app package's publisher certificate cou
administrator or the app developer to obta
The root certificate and all immediate certi

- **Evidence – Figure 7:** Installed application visible/launching after install.

Error analysis and resolutions

I encountered several issues during setup and resolved them as follows.

1. **WiX v4 setup failed (.NET 3.5 requirement)**
   - **Symptom:** "WiX Toolset requires the .NET Framework 3.5.1 Windows feature to be enabled" and installation failed.
   - **Cause:** I tried WiX v4 initially; it still requires .NET 3.5 and differs from the v3 walkthrough.
   - **Resolution:** Uninstalled WiX v4, installed **WiX v3.11.2** (which aligns with the walkthrough) and installed the **VS 2022 WiX v3 extension**. After that, project templates appeared and builds succeeded.
2. **(During optional MSIX validation) Publisher certificate not trusted (0x800B010A)**
   - **Symptom:** MSIX installer showed "publisher certificate could not be verified".
   - **Resolution:** Created a **self-signed test certificate** in the packaging project, exported the public **.cer**, installed it to **Local Machine → Trusted People** (and Root), then rebuilt and installed the MSIX successfully.
   - **Outcome:** App installed and launched; kept this section for completeness though it's beyond 1.1.

These steps demonstrate analysis of console/UI errors and the fixes applied.

Brief explanation of scripts/config

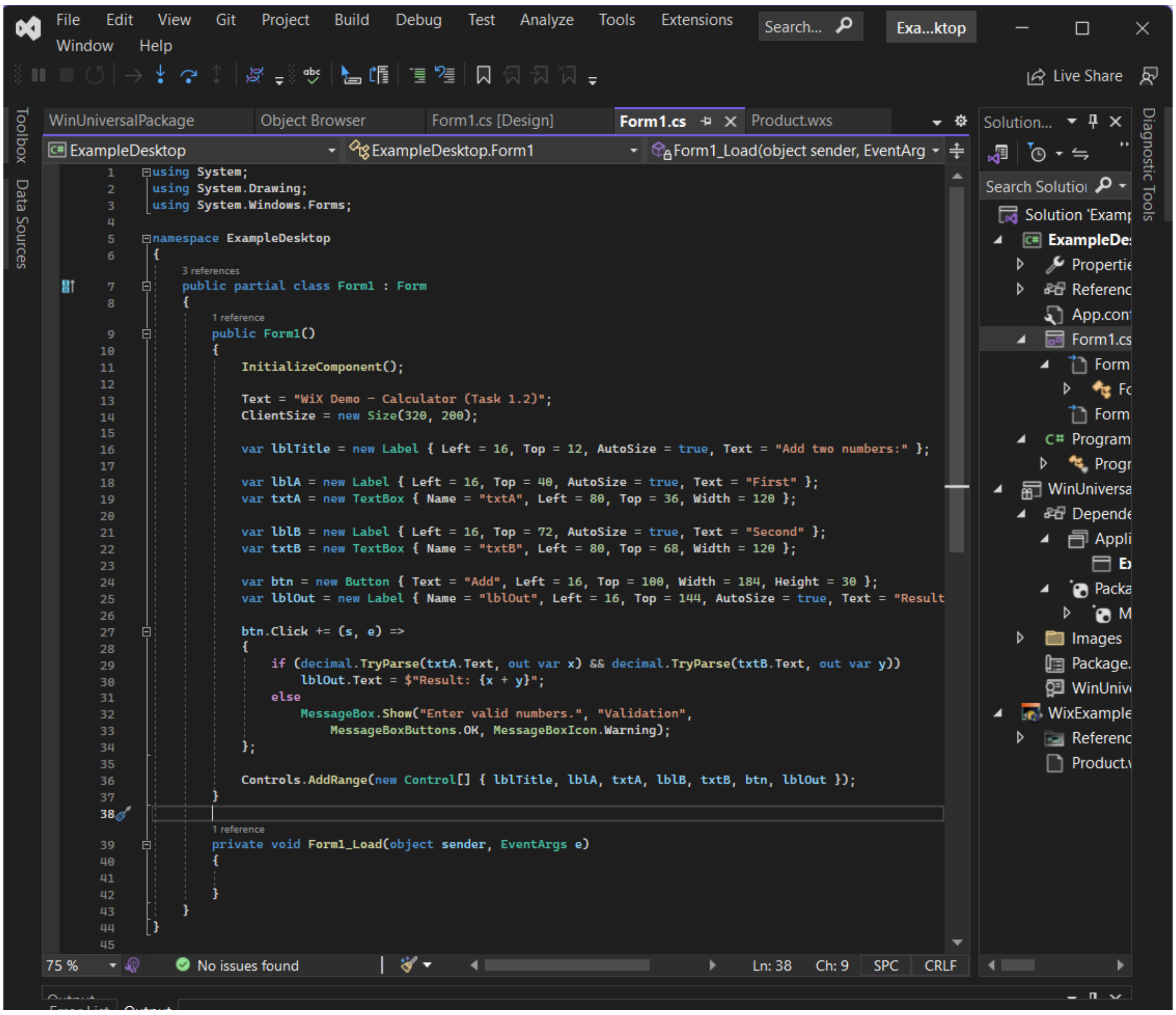
- **WiX authoring (Product.wxs)**: Declares product metadata, install directories, the app file to install, and a basic Feature. MediaTemplate EmbedCab="yes" creates a single-file MSI for easy submission/testing.
- **Build linkage**: $(var.ExampleDesktop.TargetDir) dynamically points to the app's build output so the MSI always picks up the latest EXE.

Task 1.2 - Credit: Deploy my own C# desktop application

1) App description

I implemented a simple **C# WinForms Calculator** inside the existing ExampleDesktop project to satisfy the "own desktop program" requirement. The UI contains two text boxes for numeric input, a **Calculate** button, and a label to display the result. This differs from the walkthrough sample and demonstrates my own coding and testing.

2) Key source code (Form1.cs)

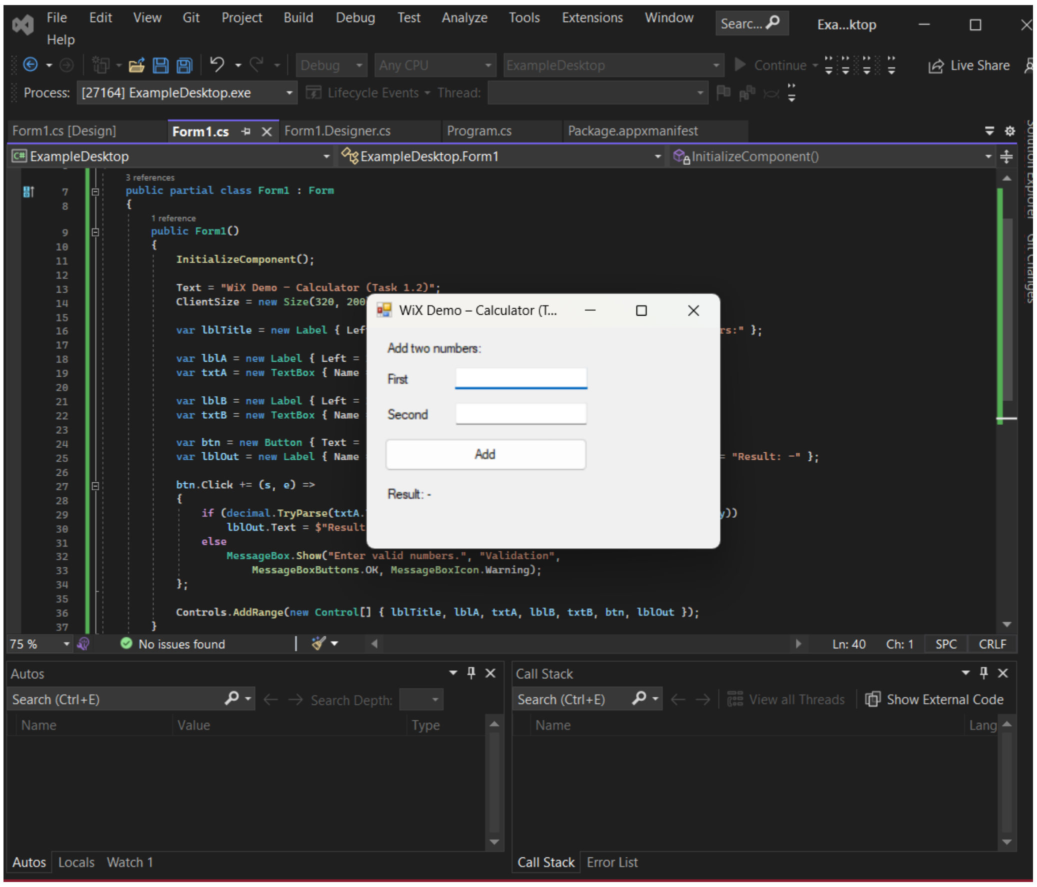


```csharp
using System;
using System.Drawing;
using System.Windows.Forms;

namespace ExampleDesktop
{
    3 references
    public partial class Form1 : Form
    {
        1 reference
        public Form1()
        {
            InitializeComponent();

            Text = "WiX Demo – Calculator (Task 1.2)";
            ClientSize = new Size(320, 200);

            var lblTitle = new Label { Left = 16, Top = 12, AutoSize = true, Text = "Add two numbers:" };

            var lblA = new Label { Left = 16, Top = 40, AutoSize = true, Text = "First" };
            var txtA = new TextBox { Name = "txtA", Left = 80, Top = 36, Width = 120 };

            var lblB = new Label { Left = 16, Top = 72, AutoSize = true, Text = "Second" };
            var txtB = new TextBox { Name = "txtB", Left = 80, Top = 68, Width = 120 };

            var btn = new Button { Text = "Add", Left = 16, Top = 100, Width = 184, Height = 30 };
            var lblOut = new Label { Name = "lblOut", Left = 16, Top = 144, AutoSize = true, Text = "Result

            btn.Click += (s, e) =>
            {
                if (decimal.TryParse(txtA.Text, out var x) && decimal.TryParse(txtB.Text, out var y))
                    lblOut.Text = $"Result: {x + y}";
                else
                    MessageBox.Show("Enter valid numbers.", "Validation",
                        MessageBoxButtons.OK, MessageBoxIcon.Warning);
            };

            Controls.AddRange(new Control[] { lblTitle, lblA, txtA, lblB, txtB, btn, lblOut });
        }

        1 reference
        private void Form1_Load(object sender, EventArgs e)
        {

        }
    }
}
```

3) WiX packaging

No schema changes were required beyond Task 1.1. The same Product.wxs references the app output via $(var.ExampleDesktop.TargetDir)ExampleDesktop.exe, so rebuilding the **WiXExample** project packages the updated EXE automatically. I rebuilt the MSI in **Release** and reinstalled to validate.

4) Evidence to include (Task 1.2)



```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();

        Text = "WiX Demo – Calculator (Task 1.2)";
        ClientSize = new Size(320, 200

        var lblTitle = new Label { Lef                                rs:" };

        var lblA = new Label { Left =
        var txtA = new TextBox { Name

        var lblB = new Label { Left =
        var txtB = new TextBox { Name

        var btn = new Button { Text =
        var lblOut = new Label { Name                               = "Result: –" };

        btn.Click += (s, e) =>
        {
            if (decimal.TryParse(txtA.                                y))
                lblOut.Text = $"Result
            else
                MessageBox.Show("Enter valid numbers.", "Validation",
                    MessageBoxButtons.OK, MessageBoxIcon.Warning);
        };

        Controls.AddRange(new Control[] { lblTitle, lblA, txtA, lblB, txtB, btn, lblOut });
    }
```

WiX Demo – Calculator (T...

Add two numbers:

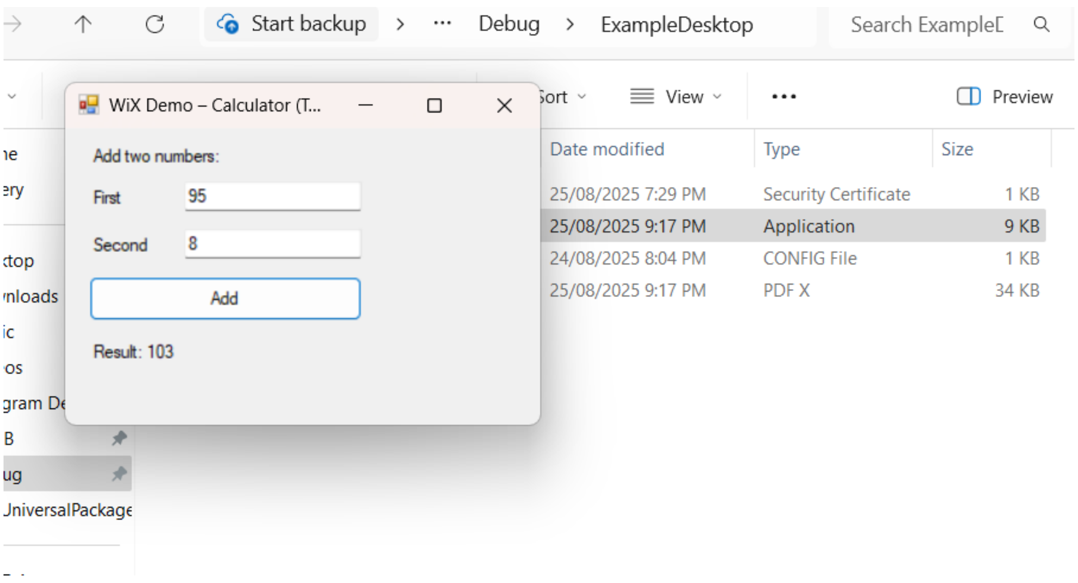
First

Second

Add

Result: -

- **Figure 8:** Calculator UI design/code visible in Visual Studio (Form1.cs).



WiX Demo – Calculator (T...

Add two numbers:

First 95

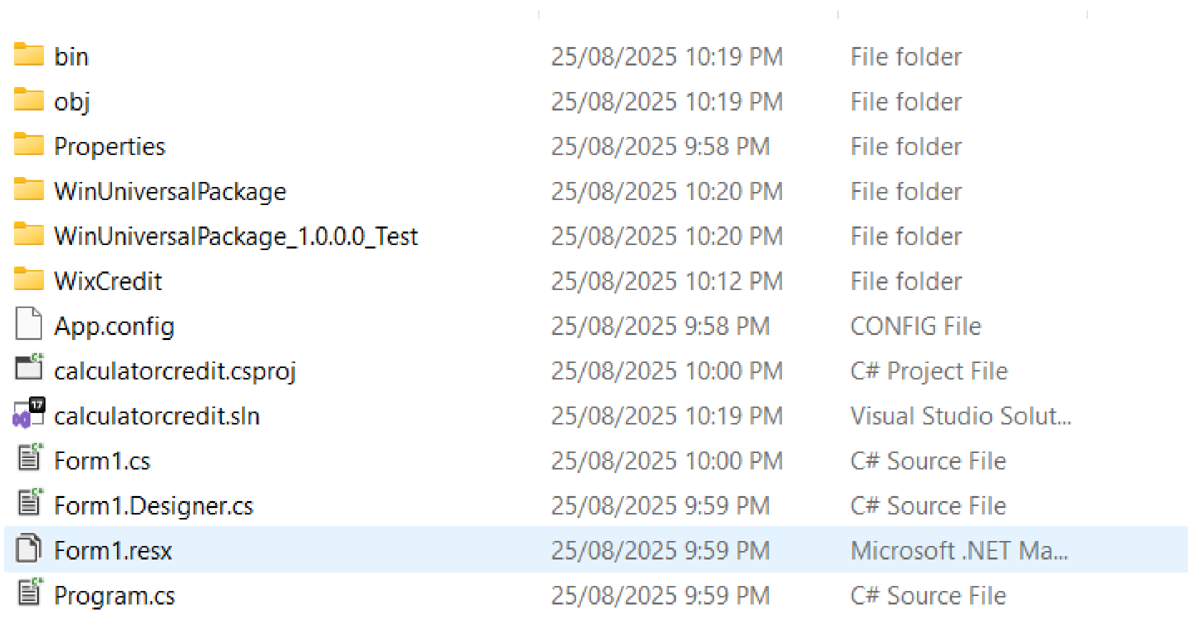
Second 8

Add

Result: 103

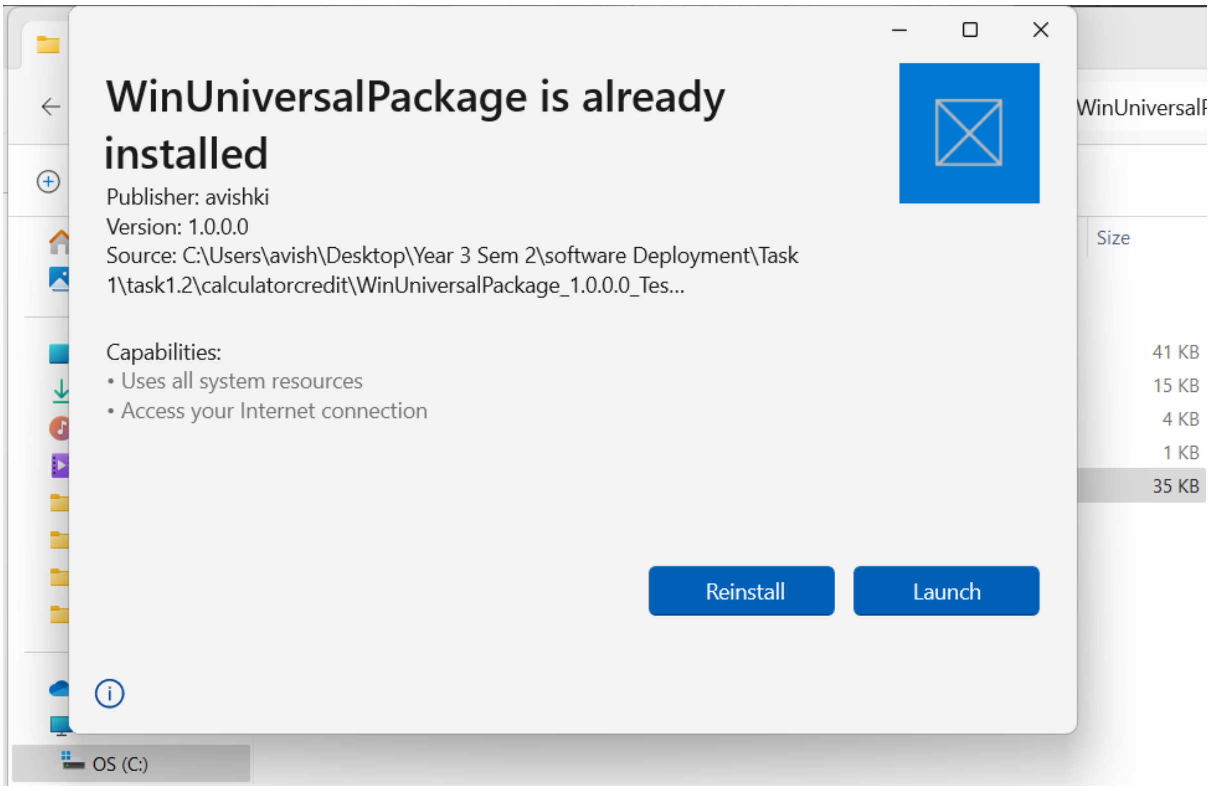
Date modified	Type	Size
25/08/2025 7:29 PM	Security Certificate	1 KB
25/08/2025 9:17 PM	Application	9 KB
24/08/2025 8:04 PM	CONFIG File	1 KB
25/08/2025 9:17 PM	PDF X	34 KB

- **Figure 9:** Calculator running with a sample addition



bin	25/08/2025 10:19 PM	File folder
obj	25/08/2025 10:19 PM	File folder
Properties	25/08/2025 9:58 PM	File folder
WinUniversalPackage	25/08/2025 10:20 PM	File folder
WinUniversalPackage_1.0.0.0_Test	25/08/2025 10:20 PM	File folder
WixCredit	25/08/2025 10:12 PM	File folder
App.config	25/08/2025 9:58 PM	CONFIG File
calculatorcredit.csproj	25/08/2025 10:00 PM	C# Project File
calculatorcredit.sln	25/08/2025 10:19 PM	Visual Studio Solut...
Form1.cs	25/08/2025 10:00 PM	C# Source File
Form1.Designer.cs	25/08/2025 9:59 PM	C# Source File
Form1.resx	25/08/2025 9:59 PM	Microsoft .NET Ma...
Program.cs	25/08/2025 9:59 PM	C# Source File

- **Figure 10:** WiX Release build after code change (timestamped).

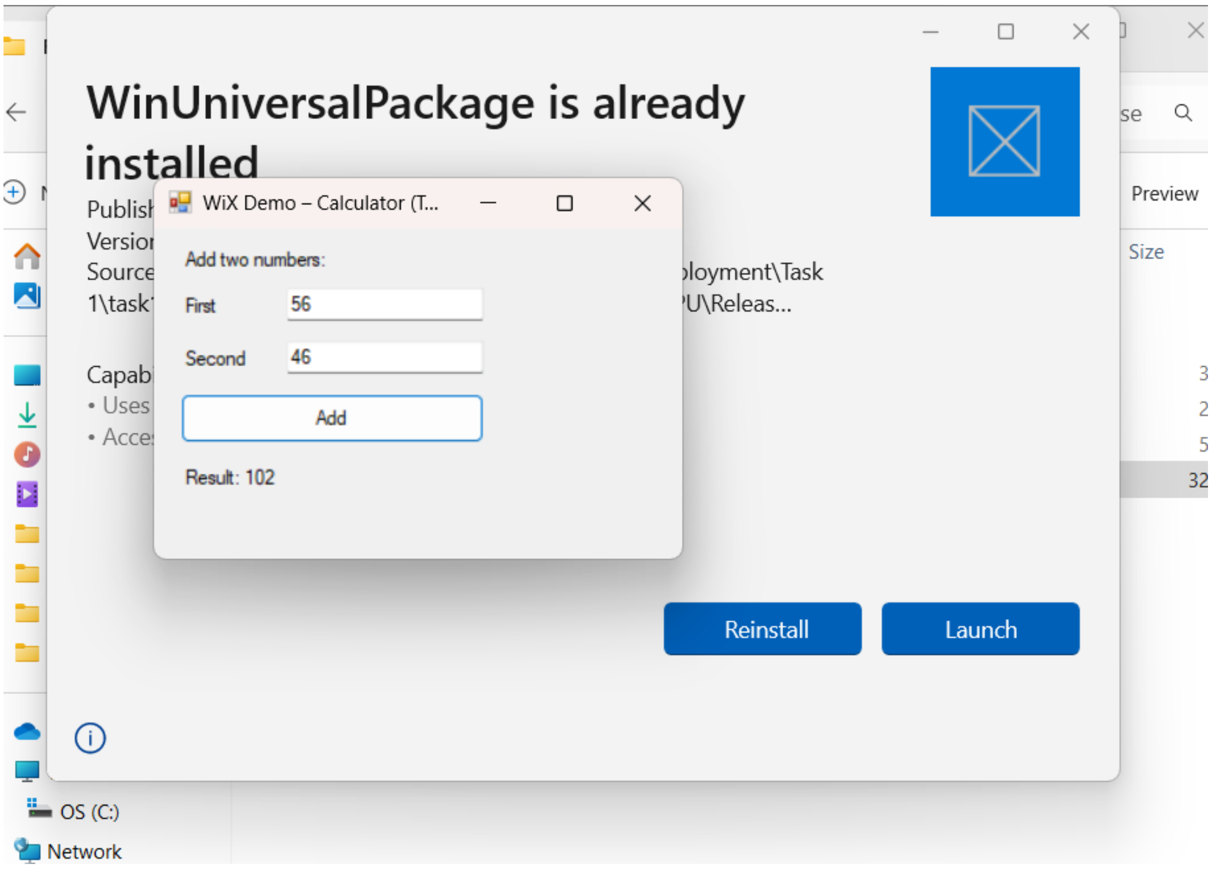


WinUniversalPackage is already installed

Publisher: avishki
Version: 1.0.0.0
Source: C:\Users\avish\Desktop\Year 3 Sem 2\software Deployment\Task 1\task1.2\calculatorcredit\WinUniversalPackage_1.0.0.0_Tes...

Capabilities:
• Uses all system resources
• Access your Internet connection

- **Figure 11:** MSI installation completing for the updated app.


WinUniversalPackage is already installed

Publish

Version

Source

1\task

ployment\Task

U\Releas...

Capab

• Uses

• Acce

WiX Demo – Calculator (T...

Add two numbers:

First 56

Second 46

Add

Result: 102

- **Figure 12:** Installed app launching from Start and performing a calculation.

5) Notes on differences vs Task 1.1

- Task 1.1 used a minimal form; Task 1.2 adds custom logic and UI authored by me.
- Packaging remains identical; WiX picks up the new build via the project reference.

Task1.3 - Distinction: Deploy my app with multiple DLLs

Summary

I developed a WinForms calculator (Task13App) that references two additional class library projects: MathLib (arithmetic operations) and HelpersLib (formatting). I then created a WiX v3 installer (Task13Installer) that deploys the EXE along with both DLLs, and sets up Start-menu and Desktop shortcuts. Additionally, I included a Task13Packaging project to test MSIX packaging alongside the MSI approach.

Solution structure

Task13App – WinForms project (.NET Framework 4.6.2)

MathLib – Class Library project (.NET Framework 4.6.2)

HelpersLib – Class Library project (.NET Framework 4.6.2)

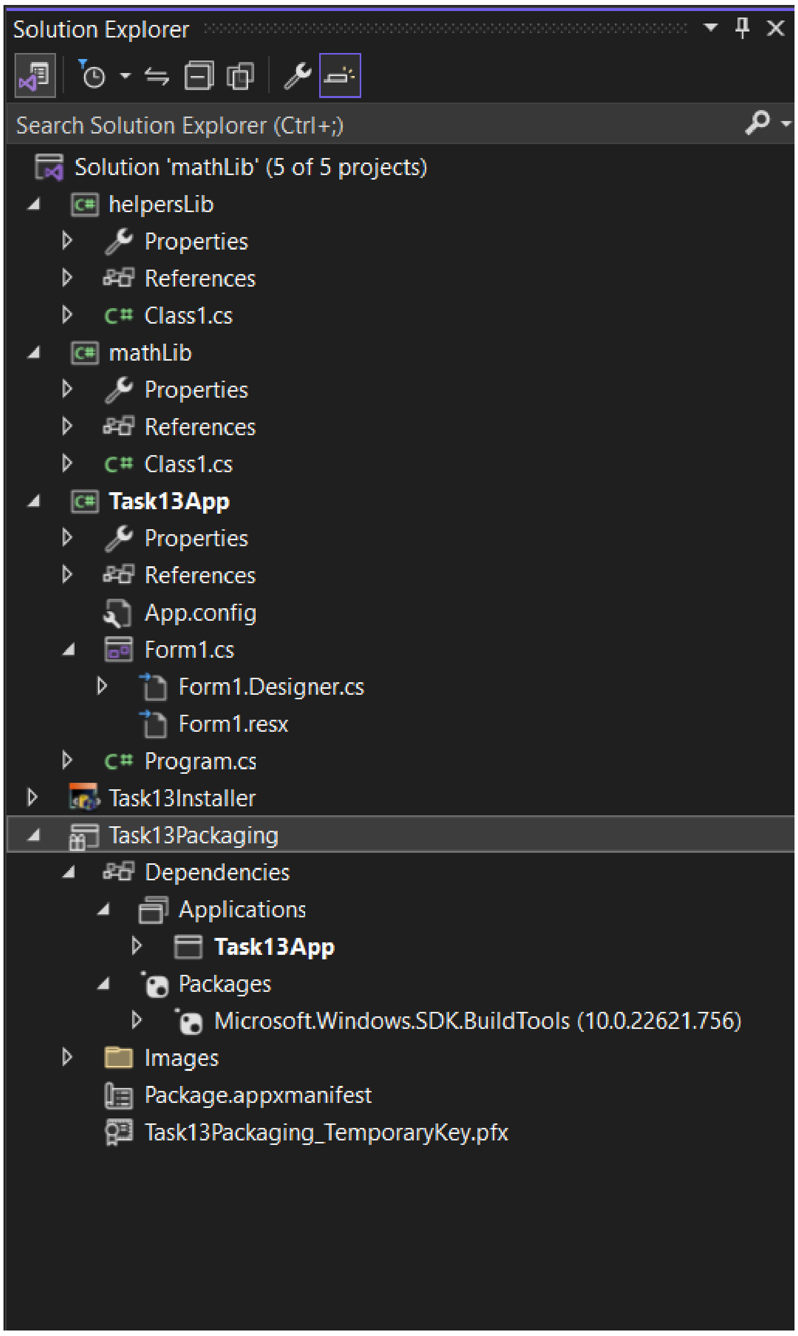
Task13Installer – WiX v3 Setup Project (with Product.wxs authoring)

Task13Packaging – Windows Application Packaging Project (used to prepare for Store deployment)

Build and install steps

1. Set solution configuration to **Release** in *Build → Configuration Manager*.
2. Rebuild MathLib, HelpersLib, and Task13App.
3. Rebuild Task13Installer → MSI generated in bin/Release.
4. Run the MSI → App and DLLs installed into C:\Program Files (x86)\\Task13Calculator.
5. Verified that Start-menu and Desktop shortcuts launched the calculator successfully.
6. Verified the **Task13Packaging** project produced an MSIX bundle for Store packaging.

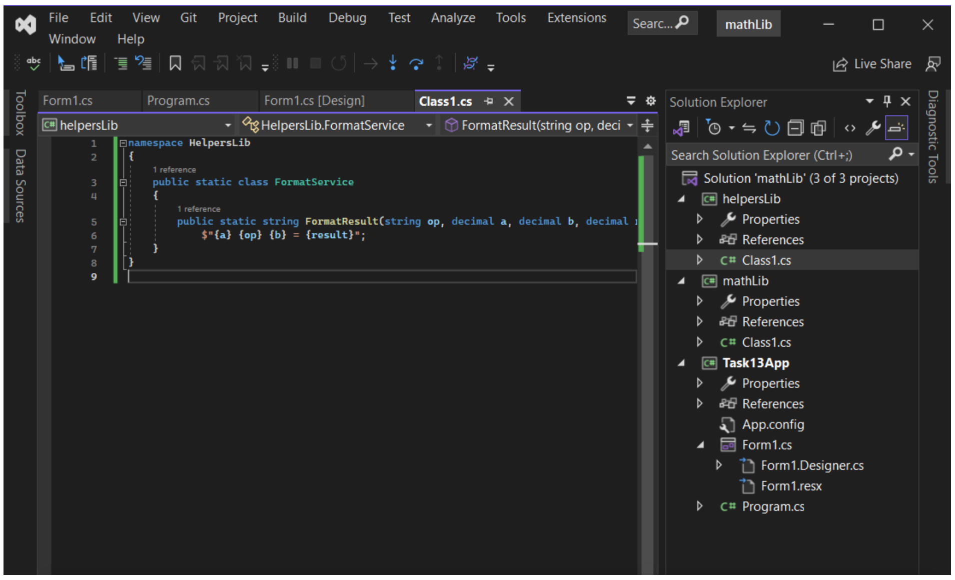
Evidence (to include as screenshots)


Solution Explorer

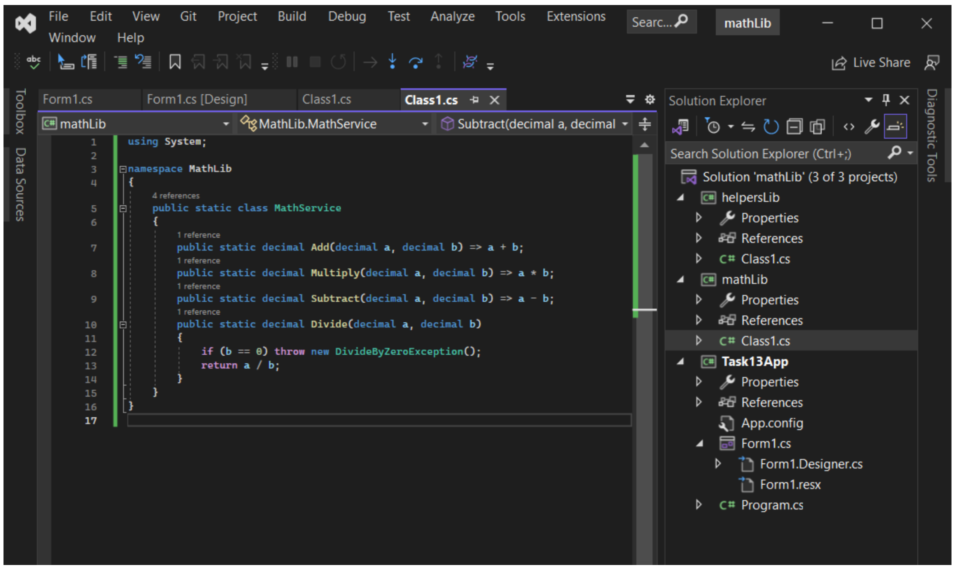
Search Solution Explorer (Ctrl+;)

```
Solution 'mathLib' (5 of 5 projects)
  helpersLib
    Properties
    References
    Class1.cs
  mathLib
    Properties
    References
    Class1.cs
  Task13App
    Properties
    References
    App.config
    Form1.cs
      Form1.Designer.cs
      Form1.resx
    Program.cs
  Task13Installer
  Task13Packaging
    Dependencies
      Applications
        Task13App
      Packages
        Microsoft.Windows.SDK.BuildTools (10.0.22621.756)
    Images
    Package.appxmanifest
    Task13Packaging_TemporaryKey.pfx
```

- Solution Explorer showing all five projects.

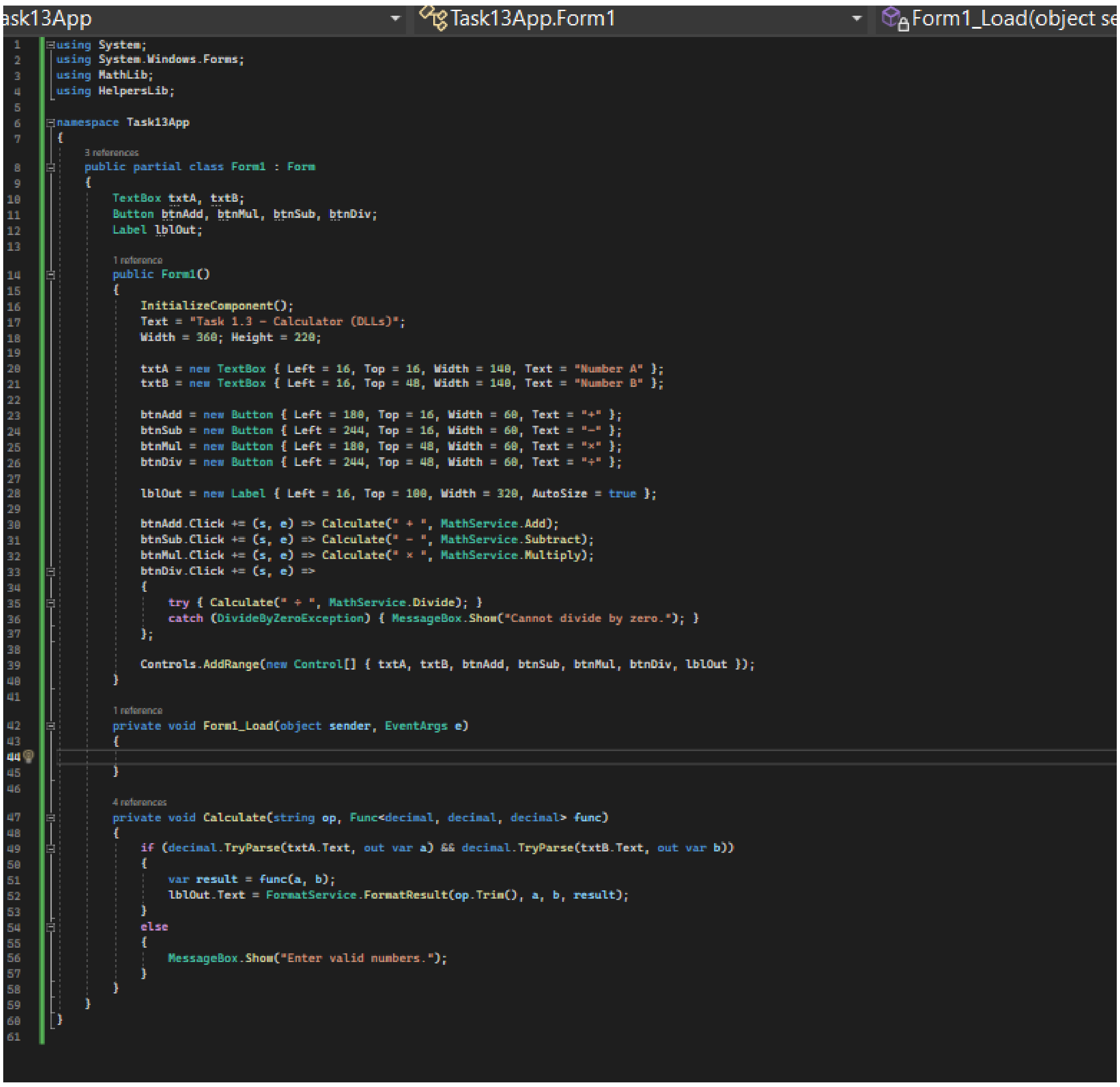


```
namespace HelpersLib
{
    1 reference
    public static class FormatService
    {
        1 reference
        public static string FormatResult(string op, decimal a, decimal b, decimal
            $"{a} {op} {b} = {result}";
    }
}
```



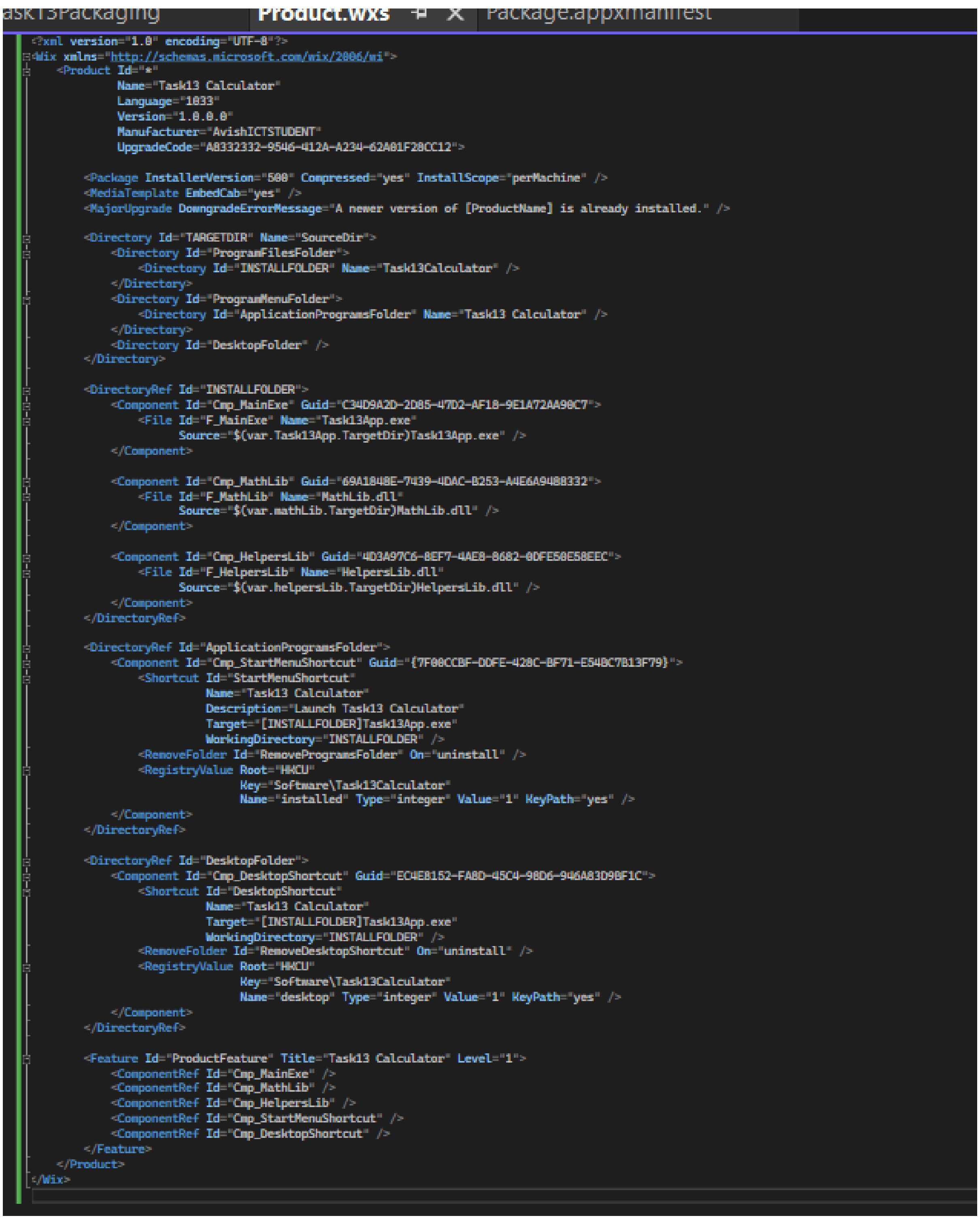
```
using System;

namespace MathLib
{
    4 references
    public static class MathService
    {
        1 reference
        public static decimal Add(decimal a, decimal b) => a + b;
        1 reference
        public static decimal Multiply(decimal a, decimal b) => a * b;
        1 reference
        public static decimal Subtract(decimal a, decimal b) => a - b;
        1 reference
        public static decimal Divide(decimal a, decimal b)
        {
            if (b == 0) throw new DivideByZeroException();
            return a / b;
        }
    }
}
```

- Code view of MathLib and HelpersLib.



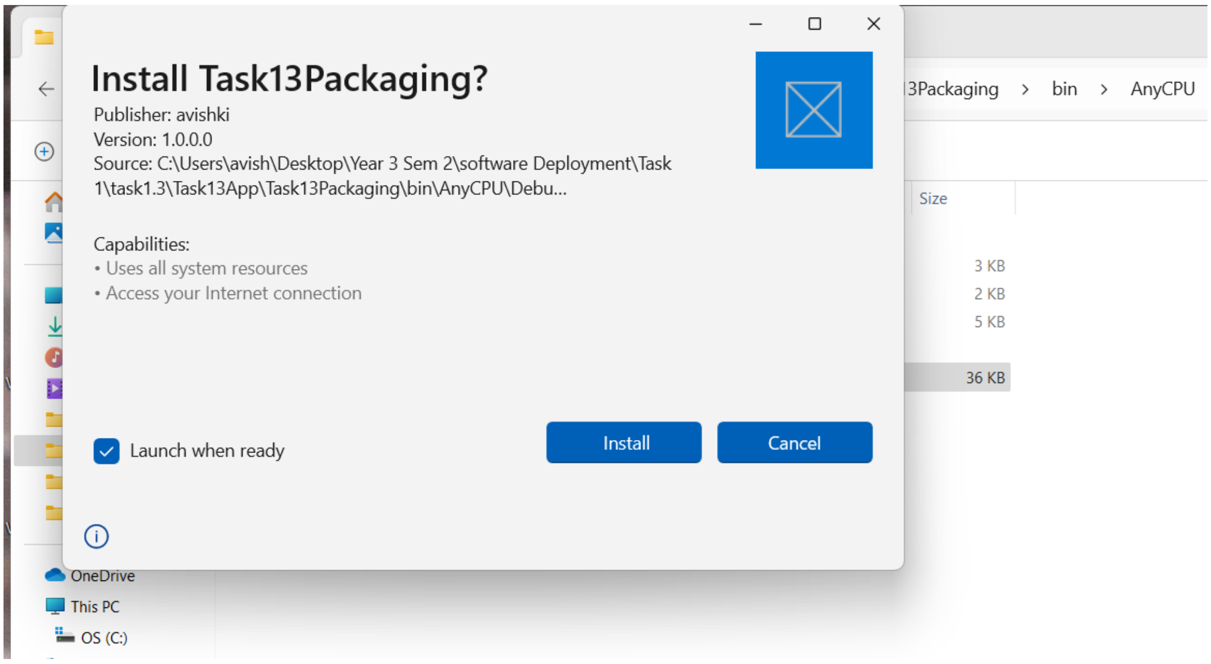
```csharp
using System;
using System.Windows.Forms;
using MathLib;
using HelpersLib;

namespace Task13App
{
    3 references
    public partial class Form1 : Form
    {
        TextBox txtA, txtB;
        Button btnAdd, btnMul, btnSub, btnDiv;
        Label lblOut;

        1 reference
        public Form1()
        {
            InitializeComponent();
            Text = "Task 1.3 – Calculator (DLLs)";
            Width = 360; Height = 220;

            txtA = new TextBox { Left = 16, Top = 16, Width = 140, Text = "Number A" };
            txtB = new TextBox { Left = 16, Top = 48, Width = 140, Text = "Number B" };

            btnAdd = new Button { Left = 180, Top = 16, Width = 60, Text = "+" };
            btnSub = new Button { Left = 244, Top = 16, Width = 60, Text = "−" };
            btnMul = new Button { Left = 180, Top = 48, Width = 60, Text = "×" };
            btnDiv = new Button { Left = 244, Top = 48, Width = 60, Text = "÷" };

            lblOut = new Label { Left = 16, Top = 100, Width = 320, AutoSize = true };

            btnAdd.Click += (s, e) => Calculate(" + ", MathService.Add);
            btnSub.Click += (s, e) => Calculate(" − ", MathService.Subtract);
            btnMul.Click += (s, e) => Calculate(" × ", MathService.Multiply);
            btnDiv.Click += (s, e) =>
            {
                try { Calculate(" ÷ ", MathService.Divide); }
                catch (DivideByZeroException) { MessageBox.Show("Cannot divide by zero."); }
            };

            Controls.AddRange(new Control[] { txtA, txtB, btnAdd, btnSub, btnMul, btnDiv, lblOut });
        }

        1 reference
        private void Form1_Load(object sender, EventArgs e)
        {

        }

        4 references
        private void Calculate(string op, Func<decimal, decimal, decimal> func)
        {
            if (decimal.TryParse(txtA.Text, out var a) && decimal.TryParse(txtB.Text, out var b))
            {
                var result = func(a, b);
                lblOut.Text = FormatService.FormatResult(op.Trim(), a, b, result);
            }
            else
            {
                MessageBox.Show("Enter valid numbers.");
            }
        }
    }
}
```

- Form1.cs file



```xml
<?xml version="1.0" encoding="UTF-8"?>
<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi">
    <Product Id="*"
             Name="Task13 Calculator"
             Language="1033"
             Version="1.0.0.0"
             Manufacturer="AvishICTSTUDENT"
             UpgradeCode="A8332332-9546-412A-A234-62A01F28CC12">

        <Package InstallerVersion="500" Compressed="yes" InstallScope="perMachine" />
        <MediaTemplate EmbedCab="yes" />
        <MajorUpgrade DowngradeErrorMessage="A newer version of [ProductName] is already installed." />

        <Directory Id="TARGETDIR" Name="SourceDir">
            <Directory Id="ProgramFilesFolder">
                <Directory Id="INSTALLFOLDER" Name="Task13Calculator" />
            </Directory>
            <Directory Id="ProgramMenuFolder">
                <Directory Id="ApplicationProgramsFolder" Name="Task13 Calculator" />
            </Directory>
            <Directory Id="DesktopFolder" />
        </Directory>

        <DirectoryRef Id="INSTALLFOLDER">
            <Component Id="Cmp_MainExe" Guid="C34D9A2D-2D85-47D2-AF18-9E1A72AA90C7">
                <File Id="F_MainExe" Name="Task13App.exe"
                      Source="$(var.Task13App.TargetDir)Task13App.exe" />
            </Component>

            <Component Id="Cmp_MathLib" Guid="69A1848E-7439-4DAC-B253-A4E6A9488332">
                <File Id="F_MathLib" Name="MathLib.dll"
                      Source="$(var.mathLib.TargetDir)MathLib.dll" />
            </Component>

            <Component Id="Cmp_HelpersLib" Guid="4D3A97C6-8EF7-4AE8-8682-0DFE50E58EEC">
                <File Id="F_HelpersLib" Name="HelpersLib.dll"
                      Source="$(var.helpersLib.TargetDir)HelpersLib.dll" />
            </Component>
        </DirectoryRef>

        <DirectoryRef Id="ApplicationProgramsFolder">
            <Component Id="Cmp_StartMenuShortcut" Guid="{7F00CCBF-DDFE-428C-BF71-E540C7B13F79}">
                <Shortcut Id="StartMenuShortcut"
                          Name="Task13 Calculator"
                          Description="Launch Task13 Calculator"
                          Target="[INSTALLFOLDER]Task13App.exe"
                          WorkingDirectory="INSTALLFOLDER" />
                <RemoveFolder Id="RemoveProgramsFolder" On="uninstall" />
                <RegistryValue Root="HKCU"
                               Key="Software\Task13Calculator"
                               Name="installed" Type="integer" Value="1" KeyPath="yes" />
            </Component>
        </DirectoryRef>

        <DirectoryRef Id="DesktopFolder">
            <Component Id="Cmp_DesktopShortcut" Guid="EC4E8152-FA8D-45C4-98D6-946A83D9BF1C">
                <Shortcut Id="DesktopShortcut"
                          Name="Task13 Calculator"
                          Target="[INSTALLFOLDER]Task13App.exe"
                          WorkingDirectory="INSTALLFOLDER" />
                <RemoveFolder Id="RemoveDesktopShortcut" On="uninstall" />
                <RegistryValue Root="HKCU"
                               Key="Software\Task13Calculator"
                               Name="desktop" Type="integer" Value="1" KeyPath="yes" />
            </Component>
        </DirectoryRef>

        <Feature Id="ProductFeature" Title="Task13 Calculator" Level="1">
            <ComponentRef Id="Cmp_MainExe" />
            <ComponentRef Id="Cmp_MathLib" />
            <ComponentRef Id="Cmp_HelpersLib" />
            <ComponentRef Id="Cmp_StartMenuShortcut" />
            <ComponentRef Id="Cmp_DesktopShortcut" />
        </Feature>
    </Product>
</Wix>
```

- Product.wxs XML with EXE and DLLs defined.

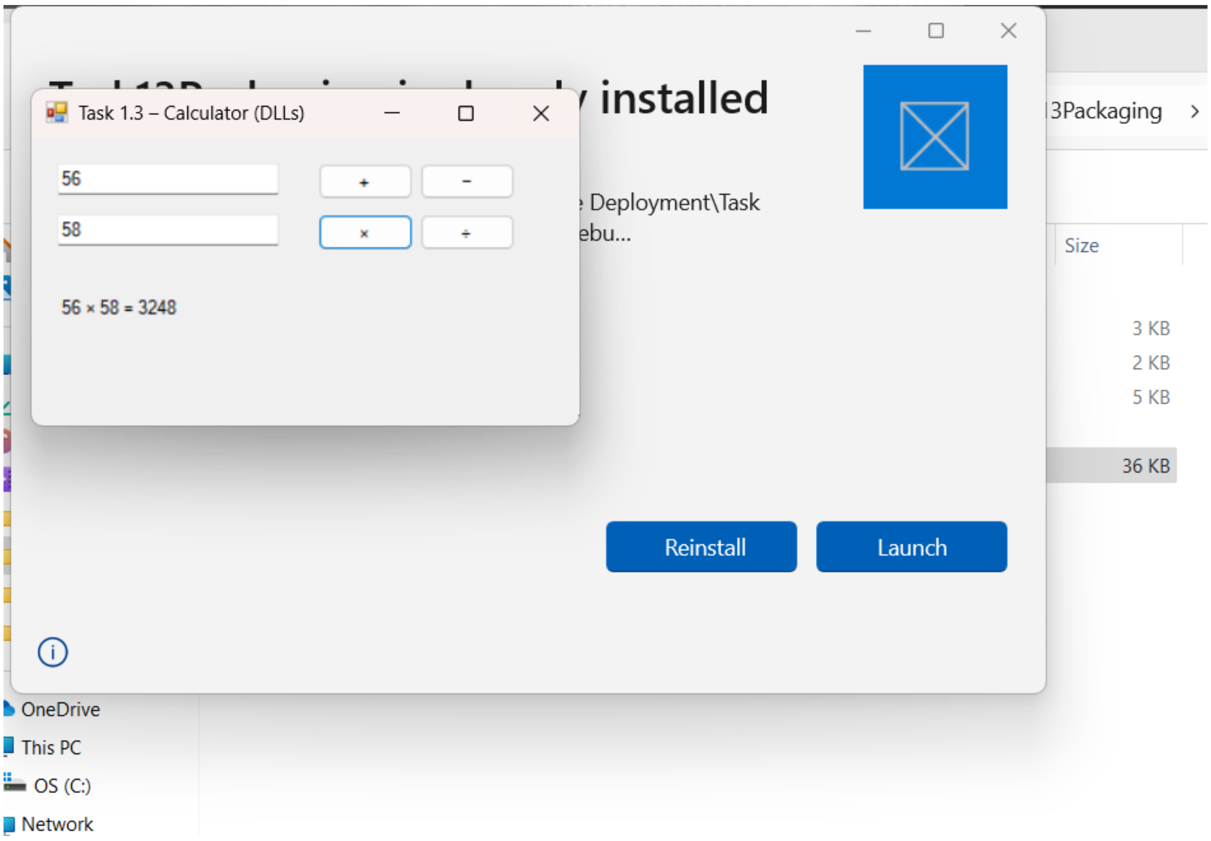


Install Task13Packaging?

Publisher: avishki
Version: 1.0.0.0
Source: C:\Users\avish\Desktop\Year 3 Sem 2\software Deployment\Task 1\task1.3\Task13App\Task13Packaging\bin\AnyCPU\Debu...

Capabilities:
• Uses all system resources
• Access your Internet connection

- Installing the program



- Calculator running via Start and Desktop shortcuts.

Issues and resolutions

- **Class library cannot start:** Set Task13App as startup project.
- **Missing Form1_Load:** Removed event hook or added empty handler.

- **WiX file not found:** Rebuilt all projects in Release and confirmed correct casing in $(var.<project>.TargetDir).
- **Shortcuts missing:** Added <ComponentRef> for shortcuts and rebuilt.
- **Certificate trust issue:** Used Task13Packaging to generate a self-signed certificate and imported it into Local Machine → Trusted Root.

Task 1.4 High Distinction: deploying to the Microsoft store

For the High Distinction task, I chose to explain in detail how an application like my Task13 Calculator could be deployed to the Microsoft Store. I didn't actually publish the app, but I followed the full process and documented the steps as if I was preparing to release it.

1. Set up a developer account – I would start by creating a Microsoft Partner Center developer account. This is required to publish apps, and it also lets me reserve the app name so no one else can take it.
2. Add a packaging project – Inside Visual Studio 2022, I added a Windows Application Packaging Project to my solution and referenced my main app (Task13App). This ensures the app is wrapped correctly for Store submission.
3. Associate with the Store – Using Publish → Associate App with the Store, I connected my packaging project to the reserved name in Partner Center. This automatically fills in the identity and publisher information.
4. Create Store packages – I then used Publish → Create App Packages and selected "Microsoft Store". Visual Studio produced an .msixupload file which is the file format the Store expects.
5. Run certification tests – Before submission, I would run the Windows App Certification Kit (WACK). This ensures the app meets Microsoft's quality and security checks.
6. Submit via Partner Center – Finally, I would upload the .msixupload to Partner Center, fill in details like description, screenshots, category, pricing, and age rating, and then submit it for review. Once approved, it would appear on the Microsoft Store.

This process gave me a strong understanding of how apps go from being local projects to official Store listings. The biggest learning was that sideloading is fine for testing, but Store publishing requires passing compliance checks and having Microsoft sign the package.

Quick Checklist (summary)

If I were to publish Task13App to the Microsoft Store, the steps would be:

- Create a Microsoft Partner Center developer account.
- Add a Windows Application Packaging Project in Visual Studio.

- Reference my app (Task13App) inside the packaging project.
- Use Publish → Associate App with the Store.
- Generate a Store package (.msixupload) with Publish → Create App Packages.
- Run the Windows App Certification Kit (WACK).
- Log into Partner Center, upload the package, complete app details, and submit for review.

Conclusion

Through this portfolio, I was able to progressively learn and apply different levels of software deployment using the WiX Toolset and Visual Studio. In **Task 1.1**, I successfully deployed a simple sample desktop app to confirm that my environment was set up correctly. In **Task 1.2**, I extended this by deploying my own customised WinForms application. Moving further, in **Task 1.3** I added complexity by introducing multiple DLLs (MathLib and HelpersLib) and packaging them together with the main executable, including working shortcuts. Finally, in **Task 1.4**, I researched and documented the complete process of preparing and deploying an application to the Microsoft Store, understanding how packaging, certificates, compliance checks, and Partner Center submissions fit together.

Overall, these tasks gave me practical, step-by-step experience with both local deployment and Store publishing workflows, showing how desktop applications move from development to real-world distribution.