Paul McKillop
HND Calculator

HND Calculator Application

Development Handbook

Contents

Application overview

This application will provide a simple calculator for arithmetic operations:

- Addition
- Subtraction
- Multiplication
- Division

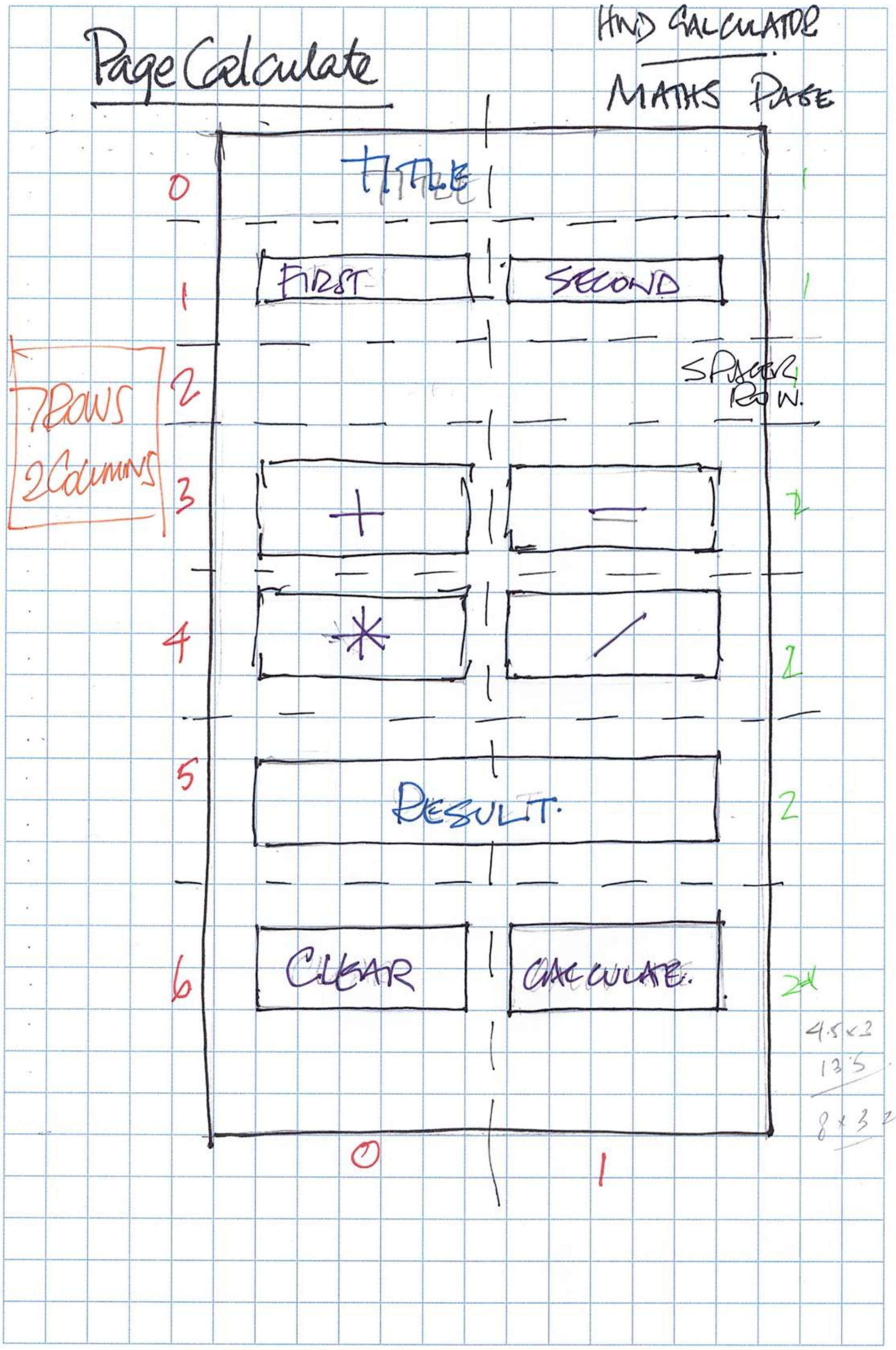
The application will be developed using The **Windows Presentation Framework** (**WPF**). This provides the tooling to create interfaces using **eXtensible Application Markup Language** (**XAML**). The application logic will be implemented in Object Oriented Programming using C# as its language.

The application will be built using **Microsoft Visual Studio Community 2022**.

Future expansion is considered. The intent would be to provide application features that allow the user to convert a small range of value types from the Imperial to Metric measurement systems, and also in the opposite direction. This could include the ability to copy a value between the two calculation screens.

Design Sketches

Page Calculate



Free Plain Graph Paper from http://incompetech.com/graphpaper/plain/

Figure 1 PageCalculate Sketch

Design Thoughts

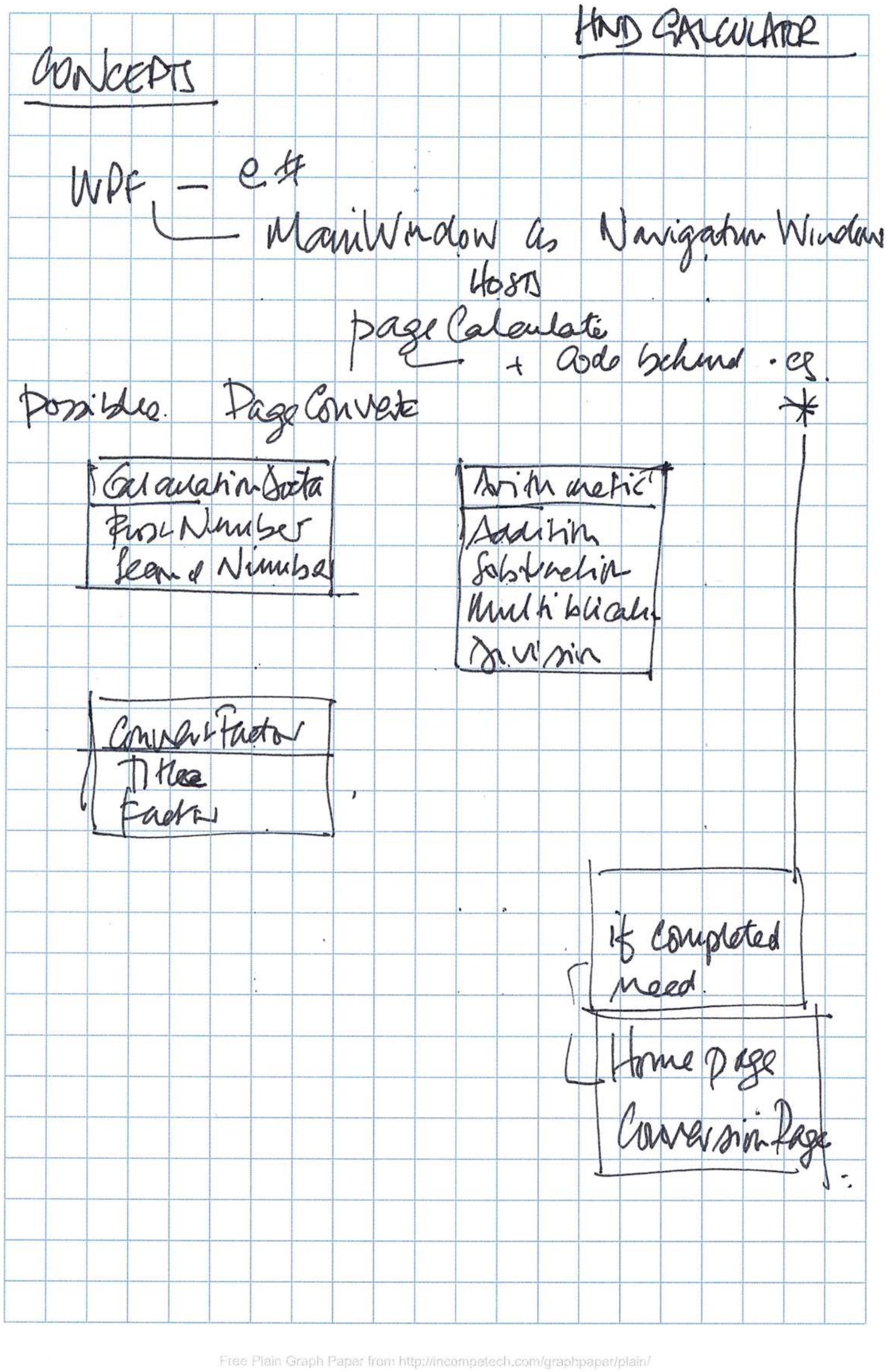


Figure 2 Design thoughts

Expansion Concepts

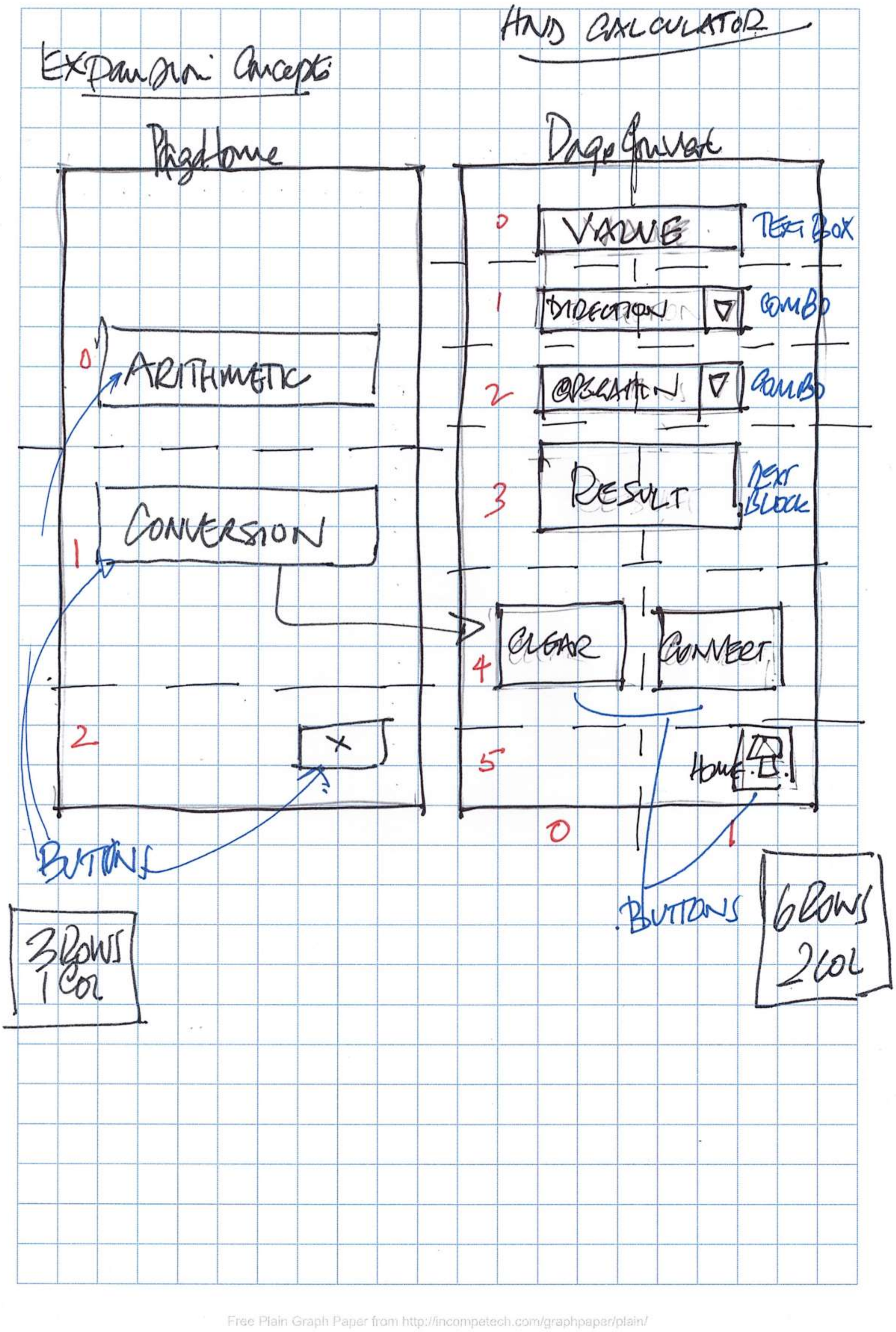


Figure 3 Expansion Concepts

UML Diagram

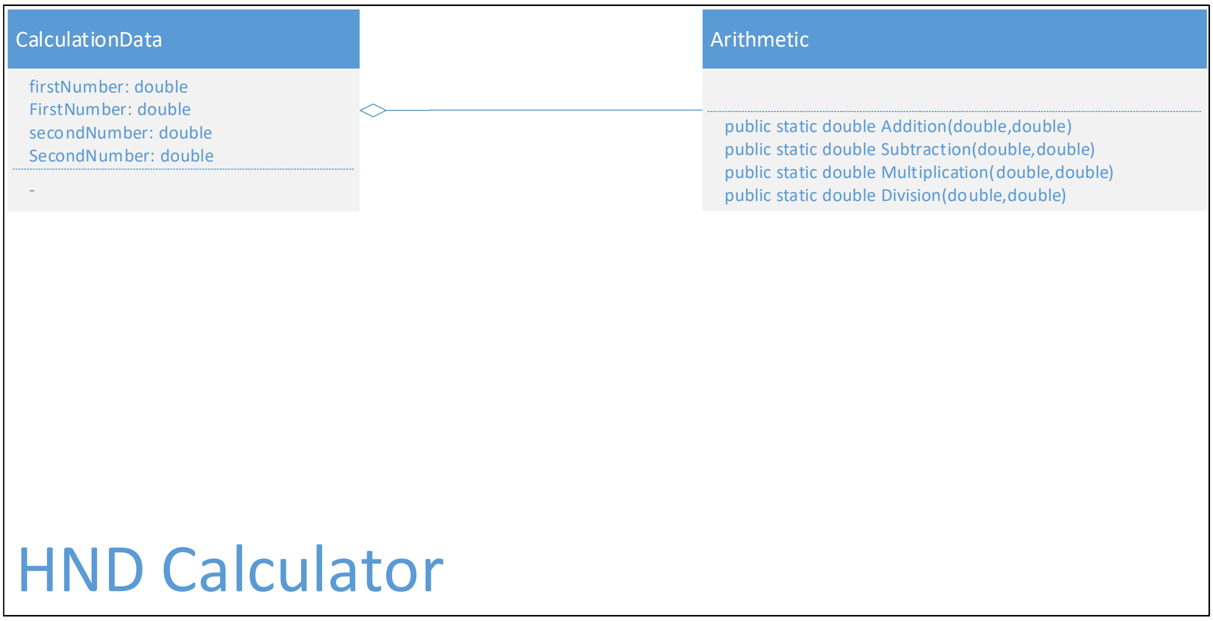

CalculationData
firstNumber: double
FirstNumber: double
secondNumber: double
SecondNumber: double
-
Arithmetic
public static double Addition(double,double)
public static double Subtraction(double,double)
public static double Multiplication(double,double)
public static double Division(double,double)
HND Calculator

Figure 4 UML Diagram

Flowchart Diagram

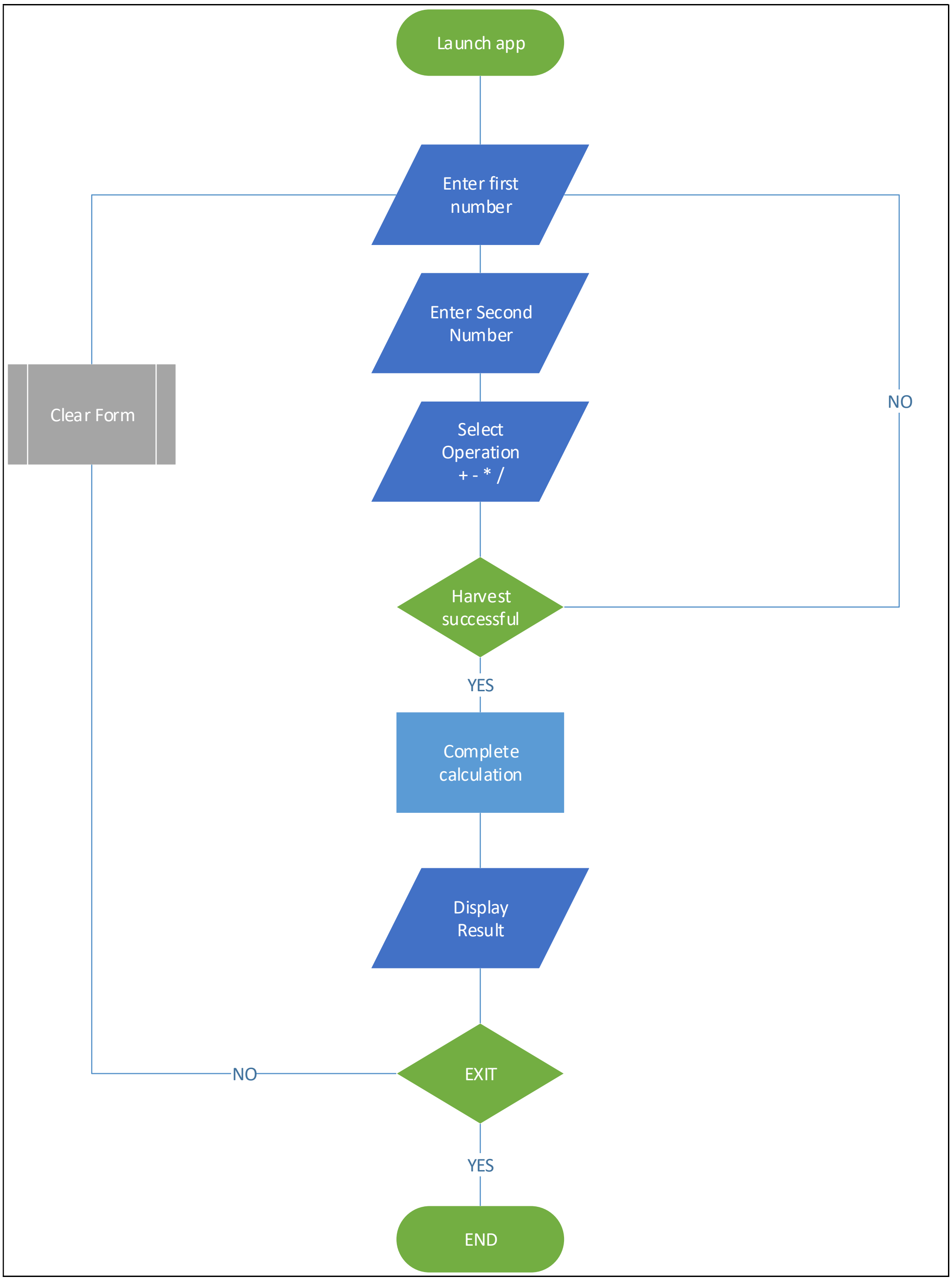


Figure 5 Flowchart application algorithm

Entity list

- CLASS **Arithmetic**
- CLASS **CalculationData**
- NavigationWindow: **MainWindow**: Navigation window to host pages
- PAGE: **PageAbout**: Program information page
- PAGE: **PageCalculate**: Process interface page

Interface Designs

Visual studio designs of pages

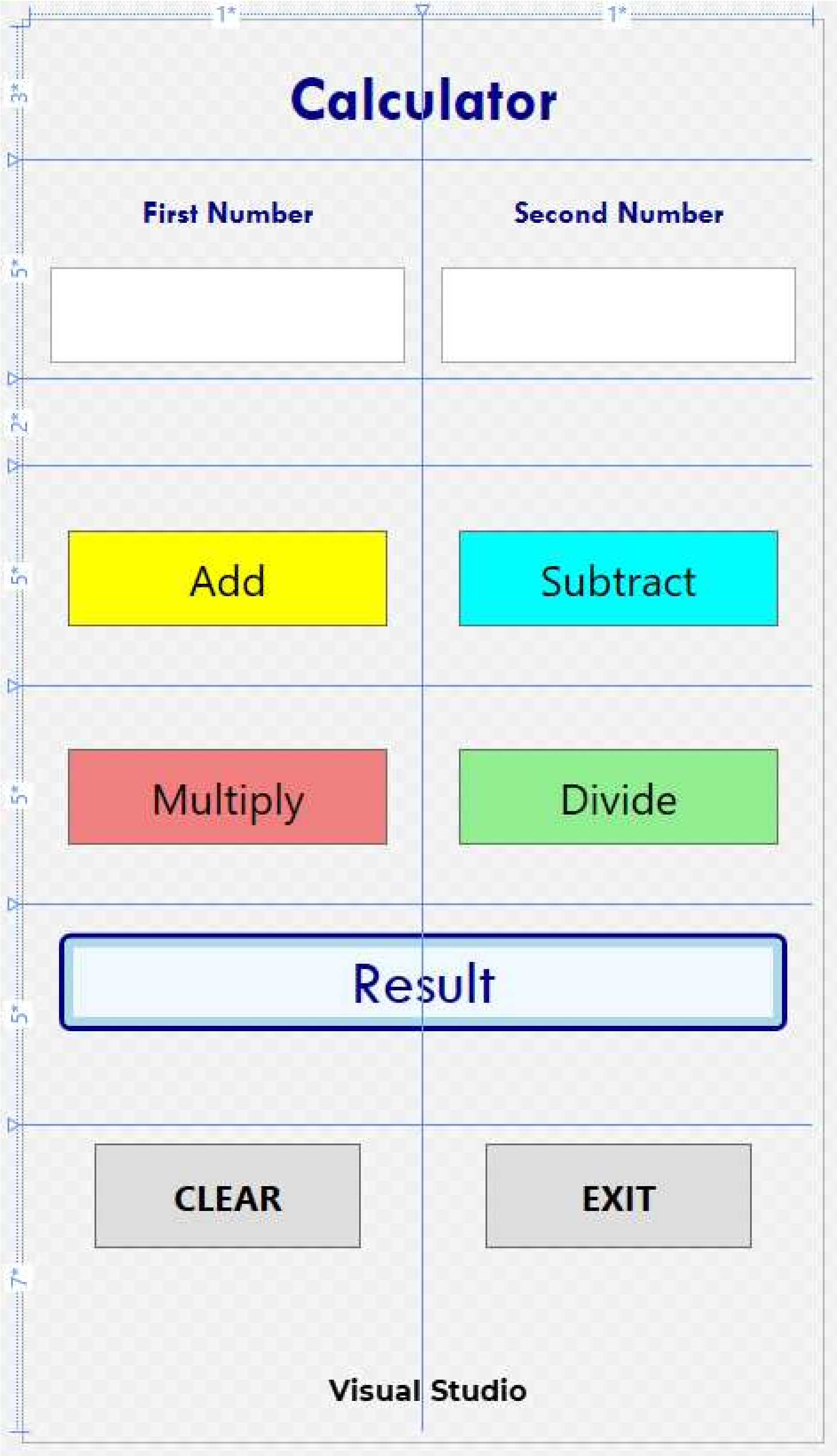
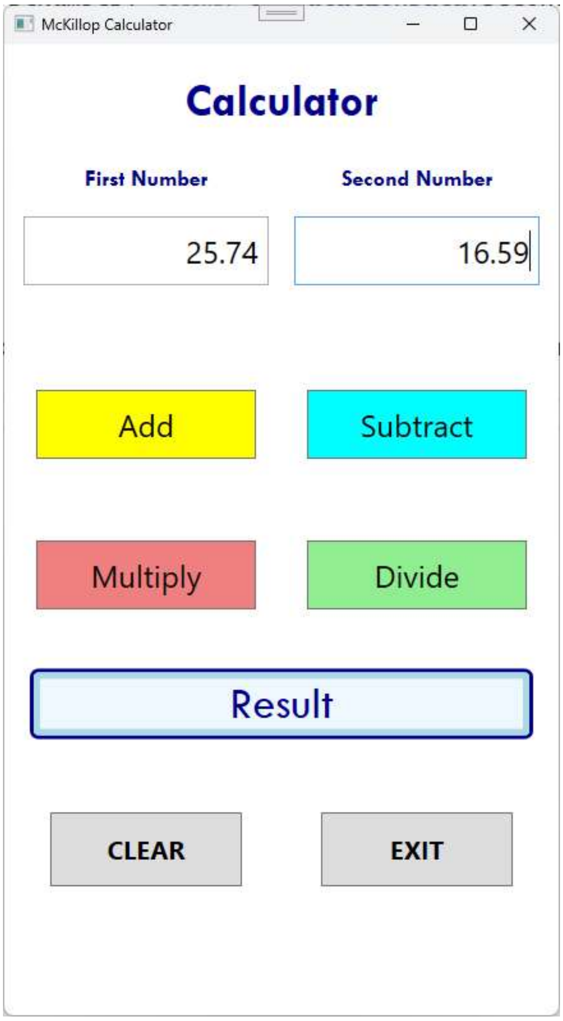


Figure 6 PageCalculate design

Test Plan

Item Number	Item	Test	Method	Expected outcome
001	Test app launch	Launch app	Launch in Visual Studio	Launch with PageCalculate
002	Control layout	Control positions and colours	Launch in Visual Studio	Al controls in correct positions in form
003	Clear	Clear controls with button	Click CLEAR button	First and second numbers cleared
004	Clear to focus	Focus goes to FirstNumber	Click CLEAR button	Cursor focus in First Number control
005	Exit app	Make sure app shuts down	Click EXIT button	Application closes
006	ETC.	ETC	ETC	ETC

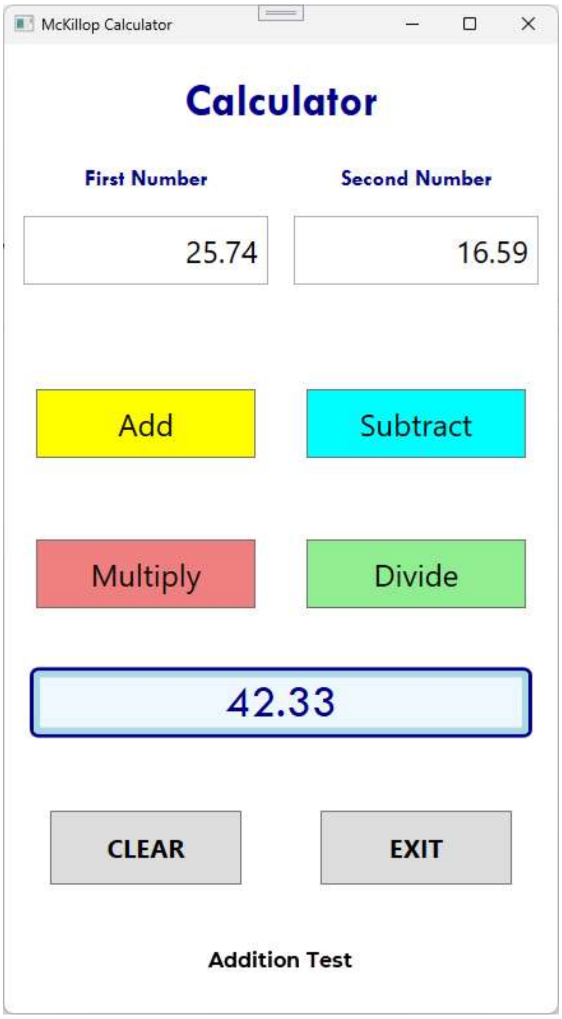
Evidence of running application



First Number: 25.74

Second Number: 16.59

Figure 7 Data Entry



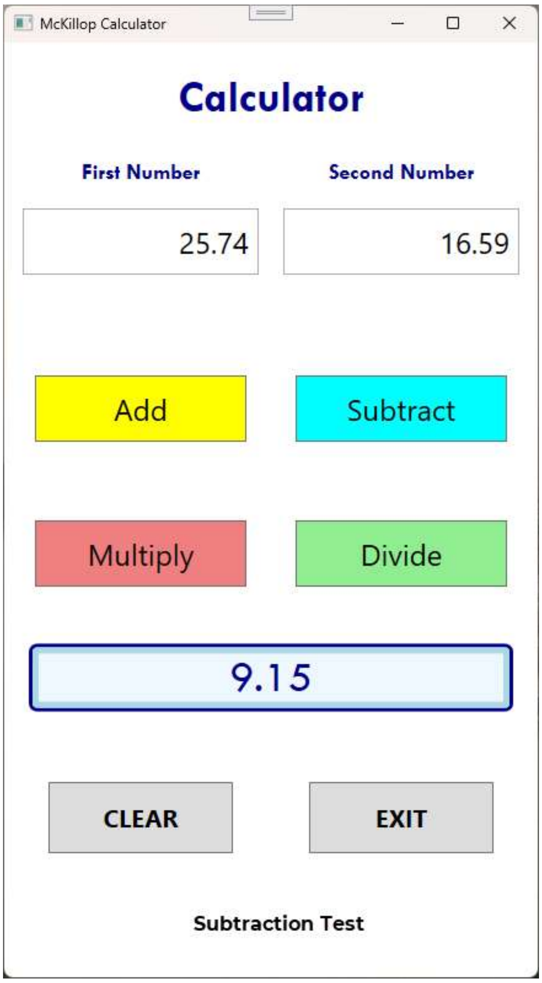
First Number: 25.74

Second Number: 16.59

42.33

Addition Test

Figure 8 Addition



McKillop Calculator

Calculator

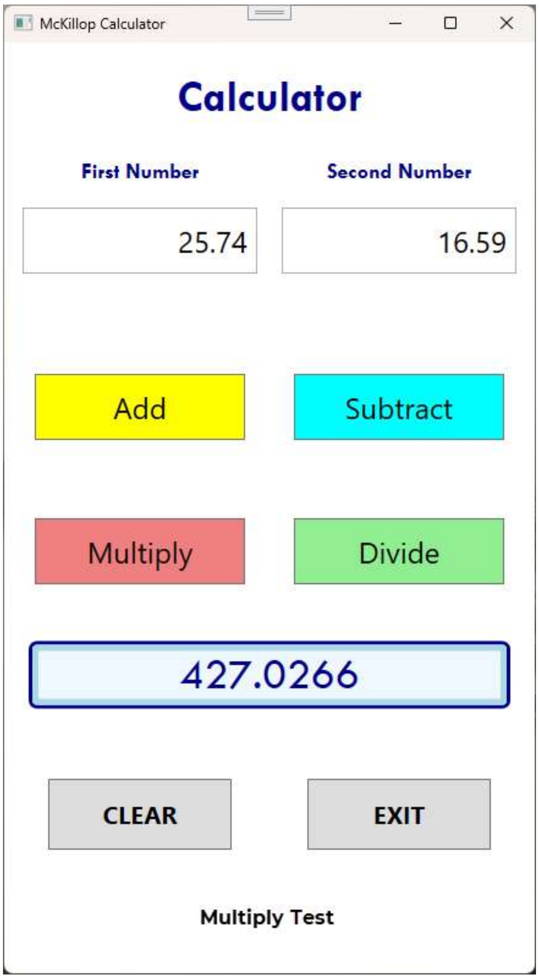
First Number: 25.74

Second Number: 16.59

9.15

Subtraction Test

Figure 9 Subtraction



25.74

16.59

427.0266

Multiply Test

Figure 10 Multiplication

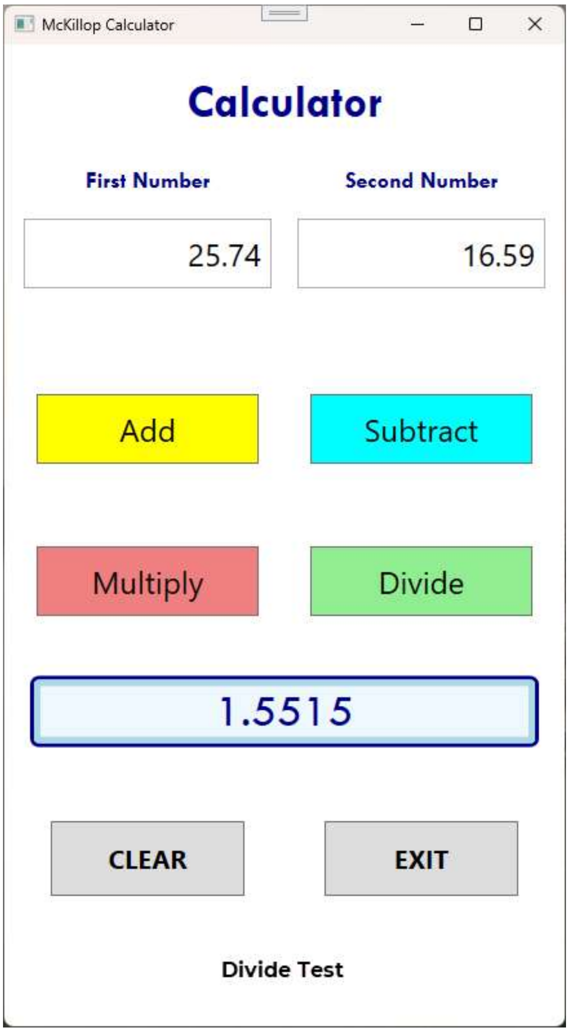


Figure 11 Division

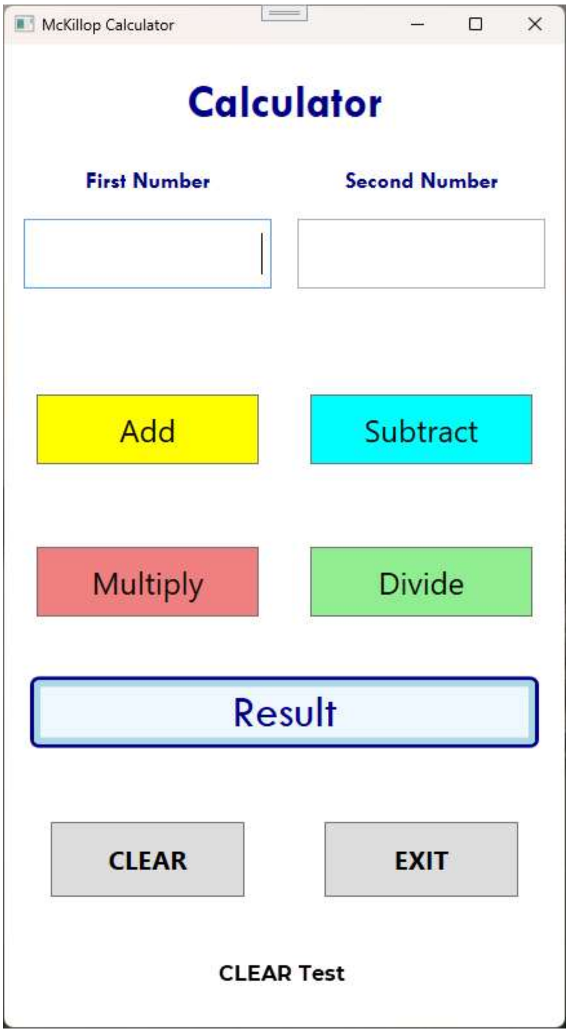


Figure 12 Clear controls

Test Log

Item Number	Item	Test	Method	Expected outcome	Date	Actual Outcome	Action taken
001	Test app launch	Launch app	Launch in Visual Studio	Launch with PageCalculate	28/10/2022	Lunch function OK	NA
002	Control layout	Control positions and colours	Launch in Visual Studio	Al controls in correct positions in form	28/10/2022	Layout as planned	NA
003	Clear	Clear controls with button	Click CLEAR button	First and second numbers cleared	28/10/2022	Controls are cleared	NA
004	Clear to focus	Focus goes to FirstNumber	Click CLEAR button	Cursor focus in First Number control	28/10/2022	Focus not set	Method fixed and works
005	Exit app	Make sure app shuts down	Click EXIT button	Application closes	28/10/2022	Application Closes	NA
006	ETC.	ETC	ETC	ETC			

APPENDIX A

XAML Code

Interface code

MainWindow

```
<NavigationWindow x:Class="McKillopCalculator.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:McKillopCalculator"
        mc:Ignorable="d"
        Title="McKillop Calculator"
                  Height="800" Width="450"
                  Source="PageCalculate.xaml"
                  WindowStartupLocation="CenterScreen">
</NavigationWindow>
```

PageCalculate

```
<Page x:Class="McKillopCalculator.PageCalculate"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:McKillopCalculator"
      mc:Ignorable="d"
      d:DesignHeight="800" d:DesignWidth="450"
      Title="Calculate" ShowsNavigationUI="False">

    <Grid Margin="5">
        <!-- Define rows and columns -->
        <Grid.RowDefinitions>
            <RowDefinition Height="3*"/>
            <RowDefinition Height="5*"/>
            <RowDefinition Height="2*"/>
            <RowDefinition Height="5*"/>
            <RowDefinition Height="5*"/>
            <RowDefinition Height="5*"/>
            <RowDefinition Height="7*"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition/>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>

    <!--Title row index 0-->
        <StackPanel
            Grid.ColumnSpan="2"
            HorizontalAlignment="Center">
            <TextBlock
                Style="{StaticResource PageHeaderStyle}">
                Calculator
            </TextBlock>
        </StackPanel>
    <!-- Data entry of numbers Row Index 1-->
        <!-- First number data entry -->
        <StackPanel
            Grid.Row="1"
            Orientation="Vertical">
            <TextBlock
                Style="{StaticResource ControlLabelStyle}"
                TextAlignment="Center">
                First Number
            </TextBlock>
            <TextBox
                Style="{StaticResource ValueTextBox}"
                x:Name="FirstNumberTextBox">

            </TextBox>

        </StackPanel>

    <!-- Second number data entry -->
        <StackPanel
            Grid.Row="1"
            Grid.Column="1"
            Orientation="Vertical">
            <TextBlock
                Style="{StaticResource ControlLabelStyle}"
                TextAlignment="Center">
                Second Number
            </TextBlock>
            <TextBox
                Style="{StaticResource ValueTextBox}"
                x:Name="SecondNumberTextBox">

            </TextBox>

        </StackPanel>

        <!-- Spacer/Spare Row Index 2 -->

    <!-- Buttons Addition Subtraction Row Index 3 -->
        <Button
            Grid.Row="3"
            Style="{StaticResource BigButton}"
            Background="Yellow"
            x:Name="AddButton"
            Click="AddButton_Click">
            Add
        </Button>

        <Button
            Grid.Row="3"
            Grid.Column="1"
            Style="{StaticResource BigButton}"
            Background="Aqua"
            x:Name="SubtractButton"
            Click="SubtractButton_Click">
            Subtract
        </Button>

        <!-- Buttons Multiplication Division Row Index 4 -->

        <Button
            Grid.Row="4"
            Style="{StaticResource BigButton}"
            Background="LightCoral"
            x:Name="MultiplyButton"
            Click="MultiplyButton_Click">
            Multiply
        </Button>

        <Button
            Grid.Row="4"
            Grid.Column="1"
            Style="{StaticResource BigButton}"
            Background="LightGreen"
            x:Name="DivideButton"
            Click="DivideButton_Click">
            Divide
        </Button>

        <!-- Diplay result Row Index 5 -->
        <StackPanel
            Grid.Row="5"
            Grid.ColumnSpan="2"
            Margin="10">
            <Border
                Margin="5"
                Padding="5"
                Background="LightBlue"
                BorderBrush="DarkBlue"
                BorderThickness="3"
                CornerRadius="5">
                <TextBlock
                    Background="AliceBlue"
                VerticalAlignment="Center"
                HorizontalAlignment="Stretch"
                TextAlignment="Center"
                    Foreground="DarkBlue"
                FontFamily="TW Cen MT"
                FontSize="36"
                x:Name="ResultTextBlock">Result</TextBlock>
            </Border>
        </StackPanel>
    <!-- Buttons Clear and Calculate Row Index 6-->
        <StackPanel
            Grid.Row="6"
            Grid.ColumnSpan="2">
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition/>
                    <ColumnDefinition/>
                </Grid.ColumnDefinitions>

                <Button
                    Style="{StaticResource FormButton}"
                    x:Name="ClearButton"
                    Click="ClearButton_Click">
                    CLEAR
                </Button>

                <Button
                    Grid.Column="1"
                    Style="{StaticResource FormButton}"
                    x:Name="ExitButton"
                    Click="ExitButton_Click">
                    EXIT
                </Button>
            </Grid>
        </StackPanel>


    </Grid>




</Page>
```

APPENDIX B

C# Code

CLASS Arithmetic

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

/*
 * Title:   Arithmetic
 * Author:  Paul McKillop
 * Date:    October 2022
 * Purpose: Arithmetic methods - Method only class
 */

namespace McKillopCalculator
{
    public class Arithmetic
    {
        // -- Calculation rules for subtraction and division:
        // --- first - second
        // --- first / second

        public static double Addition(double first, double second)
        {
            return first + second;
        }

        public static double Subtraction(double first, double second)
        {
            return first - second;
        }

        public static double Multiplication(double first, double second)
        {
            return first * second;
        }

        public static double Division(double first, double second)
        {
            return first / second;
        }

    }
}
```

CLASS CalculationData

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

/*
 * Title:   CalculationData
 * Author:  Paul McKillop
 * Date:    October 2022
 * Purpose: Hold data for calculations
 */

namespace McKillopCalculator
{
    public class CalculationData
    {
             // -- Data member for first number
             private double firstNumber;

             public double FirstNumber
             {
                   get { return firstNumber; }
                   set { firstNumber = value; }
             }

             private double secondNumber;

             public double SecondNumber
             {
                   get { return secondNumber; }
                   set { secondNumber = value; }
             }


      }
}
```

LOGIC MainWindow

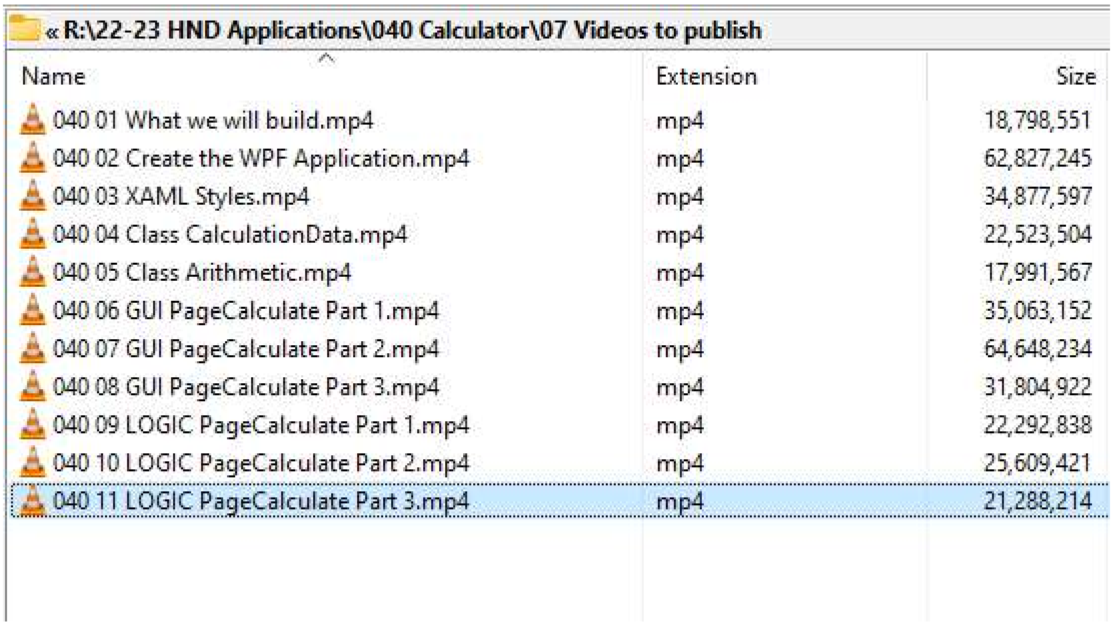
```
using System.Windows.Navigation;

namespace McKillopCalculator
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : NavigationWindow
    {
        public MainWindow()
        {
            InitializeComponent();
        }
    }
}
```

LOGIC PageCalculate

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace McKillopCalculator
{
    /// <summary>
    /// Interaction logic for PageCalculate.xaml
    /// </summary>
    public partial class PageCalculate : Page
    {
        public PageCalculate()
        {
            InitializeComponent();
            ClearControls();
        }

        private void AddButton_Click(object sender, RoutedEventArgs e)
        {
            var calculationData = new CalculationData();
            calculationData = HarvestData();

            var result = Arithmetic.Addition(calculationData.FirstNumber,
calculationData.SecondNumber);
            ResultTextBlock.Text = result.ToString("#.####");
        }

        private void SubtractButton_Click(object sender, RoutedEventArgs e)
        {
            var calculationData = new CalculationData();
            calculationData = HarvestData();

            var result = Arithmetic.Subtraction(calculationData.FirstNumber,
calculationData.SecondNumber);

            ResultTextBlock.Text = result.ToString("#.####");
        }

        private void MultiplyButton_Click(object sender, RoutedEventArgs e)
        {
            var calculationData = new CalculationData();
            calculationData = HarvestData();

            var result = Arithmetic.Multiplication(calculationData.FirstNumber,
calculationData.SecondNumber);

            ResultTextBlock.Text = result.ToString("#.####");
        }

        private void DivideButton_Click(object sender, RoutedEventArgs e)
        {
            var calculationData = new CalculationData();
            calculationData = HarvestData();

            var result = Arithmetic.Division(calculationData.FirstNumber, 
calculationData.SecondNumber);

            ResultTextBlock.Text = result.ToString("#.####");
        }

        private void ClearButton_Click(object sender, RoutedEventArgs e)
        {
            ClearControls();
        }

        private void ExitButton_Click(object sender, RoutedEventArgs e)
        {
            Application.Current.Shutdown();
        }


        // -- Utility
        private void ClearControls()
        {
            // Control
            FirstNumberTextBox.Text = "";
            SecondNumberTextBox.Text = "";
            ResultTextBlock.Text = "Result";
            // -- Focus
            FirstNumberTextBox.Focus();
        }

        private CalculationData HarvestData()
        {
            //-- Object to hold data while being gathered
            var tempData = new CalculationData();

            //-- Variables to hold text box string values and assign with
            //-- current values
            var firstNumberText = FirstNumberTextBox.Text;
            var secondNumberText = SecondNumberTextBox.Text;

            //-- Number variables to hold values for CalculationData object
            double firstNumber;
            double secondNumber;

            //-- Check there is data
            //-- This method has a weakness! If the user doesn't enter
            //-- text that can be converted to a number.
            if (!string.IsNullOrEmpty(firstNumberText))
            {
                try
                {
                    //-- Convert to a number
                    firstNumber = double.Parse(firstNumberText);
                    tempData.FirstNumber = firstNumber;
                }
                catch (FormatException e)
                {
                    //-- Tell the user what happened as feedback
                    MessageBox.Show(e.Message);
                }
            }
            else //-- Error
            {
                MessageBox.Show("You must enter a value for First Number");
                //-- drop out of the method
                return tempData;
            }

            //-- Let's try a better method for the second number. This is 
sophisticated code!
            if (Double.TryParse(secondNumberText, out secondNumber))
            {
                //-- value is good to go so assign to object
                tempData.SecondNumber = secondNumber;
            }
            else //-- Error
            {
                //-- Tell them the bad news
                MessageBox.Show($"Could not convert {secondNumberText} to a 
number");
                //-- Break out of the method
                return tempData;
            }

            return tempData;

        }
    }
}
```

Development Video List

Reference list



« R:\22-23 HND Applications\040 Calculator\07 Videos to publish

Name	Extension	Size
040 01 What we will build.mp4	mp4	18,798,551
040 02 Create the WPF Application.mp4	mp4	62,827,245
040 03 XAML Styles.mp4	mp4	34,877,597
040 04 Class CalculationData.mp4	mp4	22,523,504
040 05 Class Arithmetic.mp4	mp4	17,991,567
040 06 GUI PageCalculate Part 1.mp4	mp4	35,063,152
040 07 GUI PageCalculate Part 2.mp4	mp4	64,648,234
040 08 GUI PageCalculate Part 3.mp4	mp4	31,804,922
040 09 LOGIC PageCalculate Part 1.mp4	mp4	22,292,838
040 10 LOGIC PageCalculate Part 2.mp4	mp4	25,609,421
040 11 LOGIC PageCalculate Part 3.mp4	mp4	21,288,214

References and links

The complete WPF tutorial. (2022). Retrieved 28 October 2022, from https://wpf-tutorial.com/

Windows Presentation Foundation for .NET 5 documentation. Windows Presentation Foundation for .NET 5 documentation | Microsoft Learn. Retrieved October 28, 2022, from https://learn.microsoft.com/en-us/dotnet/desktop/wpf/?view=netdesktop-6.0

GitHub Repository

https://github.com/pgmckillop/HND-Calculator