Work done during 3-1 to 16-1
Everything which you have done, small assignment done, learn should be kept in detail. Tools like Jira and it's screenshots and working all can be submitted. PDF only.

Date-3-1-2022

Declarative vs. Imperative Programming

- In declarative programming, the source code is written in a way that expresses the desired outcome of the code with little or no emphasis on the actual implementation.
- This leaves the framework to deal with parsing the declarative source code and handling the “heavy lifting” required to instantiate objects and set properties on the these objects, which are defined in the XAML elements.
- Imperative programming is the opposite of declarative programming.
- If declarative programming can be thought of as declaring what the desired outcome is, imperative programming can be viewed as writing lines of code that represent the instructions of how to achieve the desired outcome
- A great example of a declarative language and an imperative language working together is XAML and C#.
- Whether you are developing a smart client desktop application using WPF, a Windows Store app for use with tablets such as the Microsoft Surface Pro, or a mobile application using Windows Phone, any of these applications can be developed using a mixture of XAML and C#.

Working with C#

- Data Type
- Function
    - Call by Value
    - Call by Reference
    - Out Parameter
- Array
    - Array of functions
    - Jagged array
    - Params
    - Array class
- OOPs
    - Constructor
    - Destructor
    - Static clas
    - Structs

    - Enums
- Exception handling
- Delegates
- Collections
    - List
    - Hashset
    - Sorted set
    - Stack
    - Queue
    - Linked list
    - Dictionary
    - Sorted Dictionary
    - Sorted List

Tool

Postman

Introduction

- Postman is an Application Programming Interface (API) testing tool. API acts like an interface between a couple of applications and establishes a connection between them.
- Thus, an API is a collection of agreements, functions, and tools that an application can provide to its users for successful communication with another application. We require an API whenever we access an application like checking news over the phone, Facebook, and so on.
- It is a tool for testing the software of an API. It can be used to design, document, verify, create, and change APIs.
- Postman has the feature of sending and observing the Hypertext Transfer Protocol (HTTP) requests and responses. It has a graphical user interface (GUI) and can be used in platforms like Linux, Windows and Mac. It can build multiple HTTP requests − POST, PUT, GET, PATCH and translate them to code.

Need

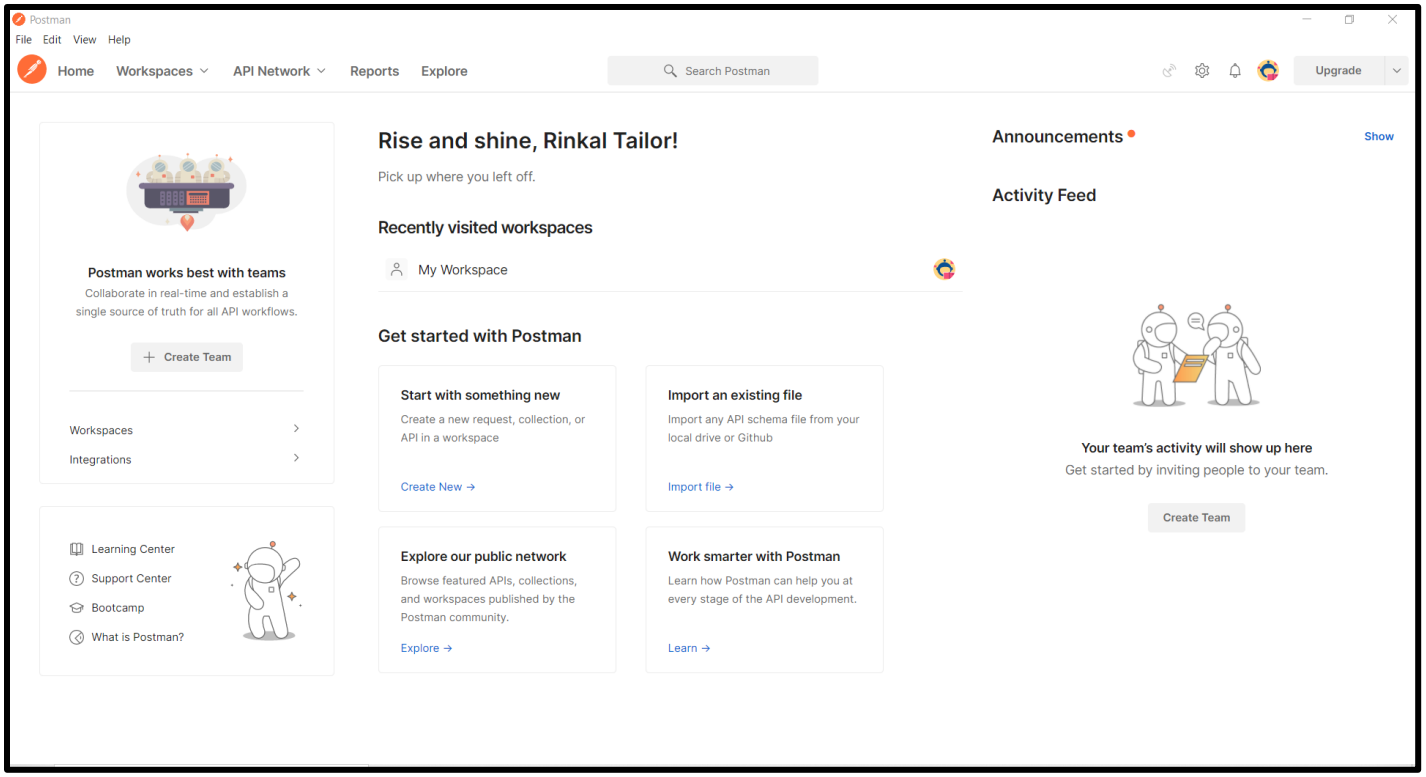
- Postman comes without any licensing cost and is suitable for use for the teams with any capacity.
- Postman can be used very easily by just downloading it.
- Postman can be accessed very easily by logging into your own account after installation on the device.

- Postman allows easy maintenance of test suites with the help of collections. Users can make a collection of API calls which can have varied requests and subfolders.
- Postman is capable of building multiple API calls like SOAP, REST, and HTTP.
- Postman can be used for test development by addition of checkpoints to HTTP response codes and other parameters.
- Postman can be integrated with the continuous integration and either continuous delivery or continuous deployment (CI/CD) pipeline.
- Postman can be integrated with Newman or Collection Runner which allows executing tests in much iteration. Thus we can avoid repeated tests.
- Postman has big community support.
- The Postman console allows debugging test steps.
- With Postman, we can create more than one environment. Thus, a single collection can be used with various configurations.
- Postman gives the option to import/export Environments and Collections, enabling easy sharing of files.

Sections

- Header
- Response
- Sidebar
- Builder

Postman App

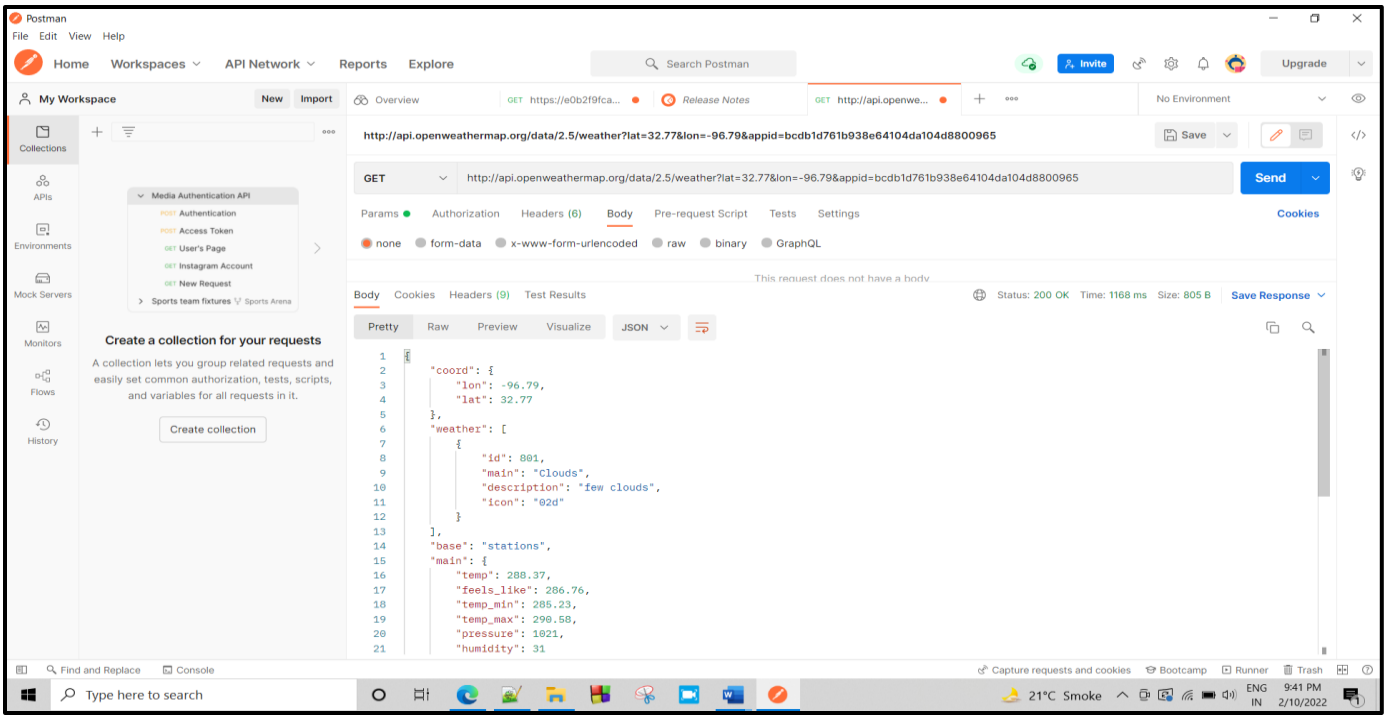


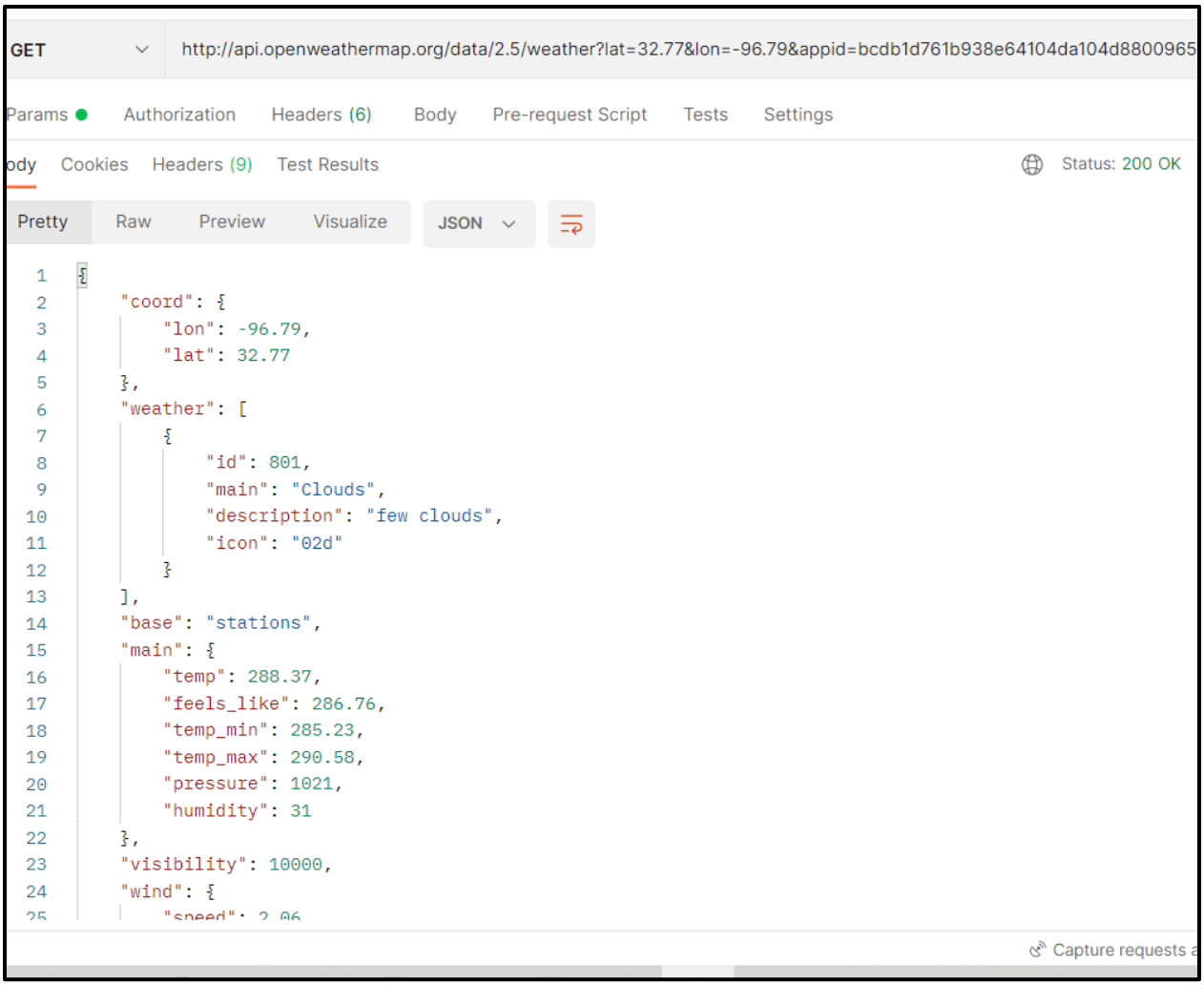
Getting Data from api url

Weather api

http://api.openweathermap.org/data/2.5/weather?lat=32.77&lon=96.79&appid=bcdb1d761b938e64104da104d8800965

Output


Postman
File Edit View Help
Home Workspaces API Network Reports Explore
Search Postman
Invite
Upgrade
My Workspace
New Import
Overview
Release Notes
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
Flows
History
Media Authentication API
Authentication
Access Token
User's Page
Instagram Account
New Request
Sports team fixtures
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create collection
http://api.openweathermap.org/data/2.5/weather?lat=32.77&lon=-96.79&appid=bcdb1d761b938e64104da104d8800965
Save
GET
http://api.openweathermap.org/data/2.5/weather?lat=32.77&lon=-96.79&appid=bcdb1d761b938e64104da104d8800965
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
This request does not have a body
Body
Cookies
Headers (9)
Test Results
Status: 200 OK
Time: 1168 ms
Size: 805 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"coord": {
"lon": -96.79,
"lat": 32.77
},
"weather": [
{
"id": 801,
"main": "Clouds",
"description": "few clouds",
"icon": "02d"
}
],
"base": "stations",
"main": {
"temp": 288.37,
"feels_like": 286.76,
"temp_min": 285.23,
"temp_max": 290.58,
"pressure": 1021,
"humidity": 31
Find and Replace
Console
Capture requests and cookies
Bootcamp
Runner
Trash
Type here to search
21°C Smoke
ENG
IN
9:41 PM
2/10/2022


GET
http://api.openweathermap.org/data/2.5/weather?lat=32.77&lon=-96.79&appid=bcdb1d761b938e64104da104d8800965
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
ody
Cookies
Headers (9)
Test Results
Status: 200 OK
Pretty
Raw
Preview
Visualize
JSON
{
"coord": {
"lon": -96.79,
"lat": 32.77
},
"weather": [
{
"id": 801,
"main": "Clouds",
"description": "few clouds",
"icon": "02d"
}
],
"base": "stations",
"main": {
"temp": 288.37,
"feels_like": 286.76,
"temp_min": 285.23,
"temp_max": 290.58,
"pressure": 1021,
"humidity": 31
},
"visibility": 10000,
"wind": {
Capture requests

Date 4-1-2022

WinRT and MVVM

- The Model is the layer that deals with data and raw content . It is often involved with obtaining and maintaining data from files or web services.
- The View is the presentation layer of controls and graphics, generally implemented in XAML .
- The View Model sits between the Model and View . In the general case, it is responsible for making the data or content from the Model more conducive to the View.

- **Data bindings**
- types of interactions
- The View can transfer user input to the View Model .
- The View Model can notify the View when updated data is available .
- The View can obtain and display updated data from the View Model.

Types of Databinding

1. One-way Data binding
2. Two-way Data binding

Data Binding notification

- When the binding source is a dependency property, the actual mechanism is internal to the Windows Runtime . Undoubtedly an event is involved . The Binding object installs a handler for an event that provides a notification when the Value property of the Slider changes, and the Binding object sets that changed value to the Text property of the TextBlock,
- The binding targets are still elements in the XAML file, but the binding sources are properties in the View Model class.

- In order for the binding to work properly, the binding source must implement some other kind of notification mechanism to signal to the Binding object when a property has changed.
- This notification does not happen automatically; it must be implemented through an event.
- The standard way for a View Model to serve as a binding source is by implementing the INotifyPropertyChanged interface defined in the System.ComponentModel namespace .
- When a class implements INotifyPropertyChanged, it fires a PropertyChanged event whenever one of its properties changes.

Collections

List view and grid view

ListView

Ways to add data in listView

- Add Items through XAML
- Add Items through code

Code:-Full-semester-external-Project-Work-Automated-Sofware/ItemInListView at main · rinkal651/Full-semester-external-Project-Work-Automated-Sofware (github.com)

MainWindow.xaml

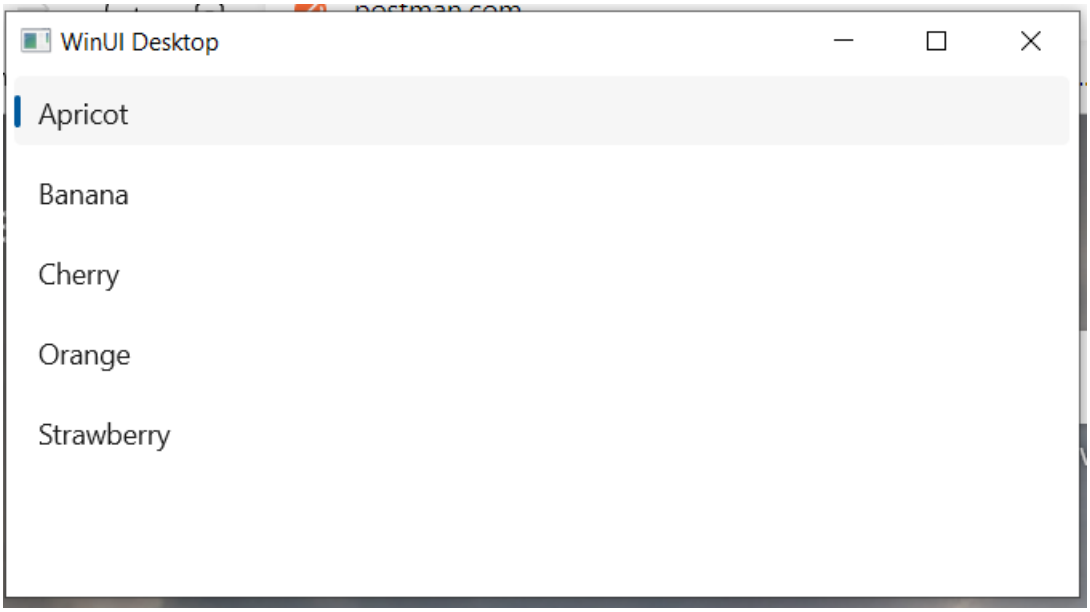
```
<Window
    x:Class="Report2_Project.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:Report2_Project"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    xmlns:muxc="using:Microsoft.UI.Xaml.Controls">

    <ListView x:Name="Fruits">
        <x:String>Apricot</x:String>
        <x:String>Banana</x:String>
        <x:String>Cherry</x:String>
        <x:String>Orange</x:String>
        <x:String>Strawberry</x:String>
    </ListView>

</Window>
```

MainWindow.xaml.cs

```
using Microsoft.UI.Xaml;
using Microsoft.UI.Xaml.Controls;
using Microsoft.UI.Xaml.Controls.Primitives;
using Microsoft.UI.Xaml.Data;
using Microsoft.UI.Xaml.Input;
using Microsoft.UI.Xaml.Media;
using Microsoft.UI.Xaml.Navigation;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.Foundation;
using Windows.Foundation.Collections;

// To learn more about WinUI, the WinUI project structure,
```

```
// and more about our project templates, see: http://aka.ms/winui-project-info.

namespace Report2_Project
{
    /// <summary>
    /// An empty window that can be used on its own or navigated to within a Frame.
    /// </summary>
    ///

    public sealed partial class MainWindow : Window
    {
        public MainWindow()
        {
            this.InitializeComponent();
            ListView Fruits = new ListView();
            Fruits.Items.Add("Apricot");
            Fruits.Items.Add("Banana");
            Fruits.Items.Add("Cherry");
            Fruits.Items.Add("Orange");
            Fruits.Items.Add("Strawberry");

            // Add the ListView to a parent container in the visual tree (that you
created in the corresponding XAML file).
            //  FruitsPanel.Children.Add(Fruits);
        }


    }


}
```

Output:



ORM

Entity Framework is an *object-relational mapping* (ORM) framework.
purpose

- keep all of the data access code for an app in one place, written in one language (C# in this case)
- organized according to one design philosophy (object-oriented design).
- EF allows the developer to maintain a single mental model for how an app's data is organized, rather than switching between an object model and a relational model.

DbContext

- responsibilities of a context class is to expose data from a database table in the form of a DbSet, a representation of a database entity.

Process- I have describe whole process in the github.
Github link:- Full-semester-external-Project-Work-Automated-Sofware/orm at main · rinkal651/Full-semester-external-Project-Work-Automated-Sofware (github.com)

Code:
Full-semester-external-Project-Work-Automated-Sofware/orm at main · rinkal651/Full-semester-external-Project-Work-Automated-Sofware (github.com)

MainWindow.xaml

```
<Window
    x:Class="table4.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:table4"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
     xmlns:model="using:table4.Model">

    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition/>
        </Grid.RowDefinitions>

        <RelativePanel HorizontalAlignment="Center"
                       VerticalAlignment="Center">
            <TextBlock Name="TitleTextBlock"
                       Text="Title"
                       FontSize="16"
                       RelativePanel.AlignLeftWithPanel="True"
                       RelativePanel.AlignTopWithPanel="True"/>

            <TextBox Name="TitleTextBox"
                     Width="250"
                     RelativePanel.AlignLeftWithPanel="True"
                     RelativePanel.Below="TitleTextBlock"/>
```

```
        <TextBlock Name="ContentTextBlock"
                   Text="Content"
                   FontSize="16"
                   Margin="0,10,0,0"
                   RelativePanel.AlignLeftWithPanel="True"
                   RelativePanel.Below="TitleTextBox"/>

        <TextBox Name="ContentTextBox"
                 Width="250"
                 RelativePanel.AlignLeftWithPanel="True"
                 RelativePanel.Below="ContentTextBlock"/>

        <Button Name="SaveButton"
                Content="Save"
                FontSize="16"
                Margin="0,10,0,0"
                MinWidth="100"
                RelativePanel.AlignHorizontalCenterWithPanel="True"
                RelativePanel.Below="ContentTextBox"
                Click="SaveButton_Click"/>
    </RelativePanel>

    <ListView Grid.Row="1"
              ItemsSource="{x:Bind Posts,Mode=OneWay}"
              Margin="5,10"
              BorderBrush="LightGray"
              BorderThickness="1"
              Header="Posts"
              Width="300"
              HorizontalAlignment="Center"
              VerticalAlignment="Stretch">

        <ListView.HeaderTemplate>
            <DataTemplate>
                <TextBlock Text="{Binding}"
                           FontSize="18"
                           HorizontalAlignment="Center"/>
            </DataTemplate>
        </ListView.HeaderTemplate>

        <ListView.ItemTemplate>
            <DataTemplate x:DataType="model:Post">
                <StackPanel Margin="0,5">
                    <TextBlock FontSize="20"
                               Text="{Binding Title, Mode=OneWay}"
                               TextWrapping="Wrap"/>

                    <TextBlock Text="{Binding Content,Mode=OneWay}"
                               TextWrapping="Wrap"
                               Margin="0,5,0,0"/>
                </StackPanel>
            </DataTemplate>
        </ListView.ItemTemplate>

    </ListView>

</Grid>
```

```
</Window>
```

MainWindow.xaml.cs

```
using Microsoft.UI.Xaml;
using Microsoft.UI.Xaml.Controls;
using Microsoft.UI.Xaml.Controls.Primitives;
using Microsoft.UI.Xaml.Data;
using Microsoft.UI.Xaml.Input;
using Microsoft.UI.Xaml.Media;
using Microsoft.UI.Xaml.Navigation;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.Foundation;
using Windows.Foundation.Collections;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Threading.Tasks;
using table4.Model;
using Microsoft.EntityFrameworkCore;
// To learn more about WinUI, the WinUI project structure,
// and more about our project templates, see: http://aka.ms/winui-project-info.

namespace table4
{
    /// <summary>
    /// An empty window that can be used on its own or navigated to within a Frame.
    /// </summary>
    public sealed partial class MainWindow : Window, INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler PropertyChanged;

        private void OnPropertyChanged(string propertyName) =>
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

        private ObservableCollection<Post> _posts;

        public ObservableCollection<Post> Posts
        {
            get { return _posts; }
            set
            {
                _posts = value;
                OnPropertyChanged(nameof(Posts));
            }
        }

        public MainWindow()
        {
            this.InitializeComponent();
            Posts = new ObservableCollection<Post>();
        }
        private async void Page_Loaded(object sender, RoutedEventArgs e)
```

```
{
    try
    {
        Posts.Clear();

        var posts = await GetAsync();
        posts.ForEach(p => Posts.Add(p));
    }
    catch (Exception)
    {
    }
}

private async void SaveButton_Click(object sender, RoutedEventArgs e)
{
    try
    {
        var post = new Post()
        {
            Title = TitleTextBox.Text,
            Content = ContentTextBox.Text,
        };

        if (await AddAsync(post))
        {
            Posts.Add(post);
        }
    }
    catch (Exception)
    {
    }
}

private async Task<bool> AddAsync(Post post)
{
    try
    {
        using (var db = new DataCont())
        {
            db.Posts.Add(post);
            return (await db.SaveChangesAsync()) > 0;
        }
    }
    catch (Exception)
    {
        throw;
    }
}

private async Task<List<Post>> GetAsync()
{
    try
    {
        using (var db = new DataCont())
        {
            return await db.Posts.ToListAsync();
        }
    }
```

```
            catch (Exception)
            {

                throw;
            }
        }
    }
}
```

DataCont.cs

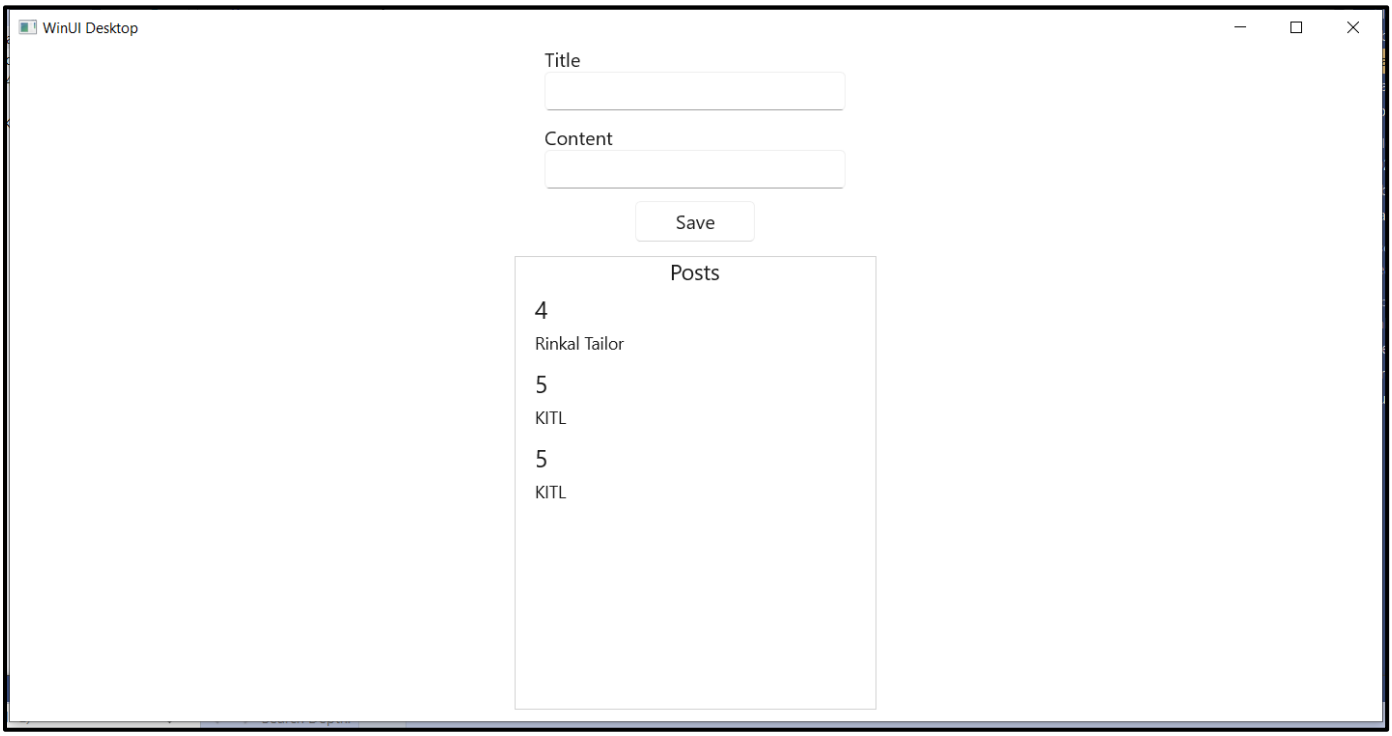
```
using System;
using System.ComponentModel.DataAnnotations;
using Microsoft.EntityFrameworkCore;
namespace table4.Model
{
    public class DataCont:DbContext
    {
        public DbSet<Post> Posts { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlite("Data Source=blogging.db");
        }
    }

    public class Post
    {
        public int PostId { get; set; }
        public string Title { get; set; }
        public string Content { get; set; }

        public override string ToString() => Title;
    }
}
```

App.xaml.cs

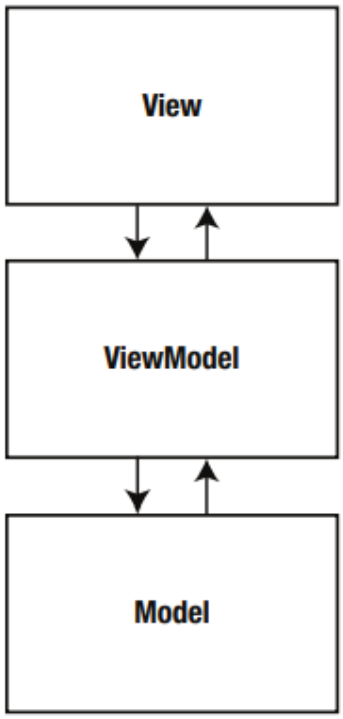
```
using Microsoft.EntityFrameworkCore;
using Microsoft.UI.Xaml;
using Microsoft.UI.Xaml.Controls;
using Microsoft.UI.Xaml.Controls.Primitives;
using Microsoft.UI.Xaml.Data;
using Microsoft.UI.Xaml.Input;
using Microsoft.UI.Xaml.Media;
using Microsoft.UI.Xaml.Navigation;
using Microsoft.UI.Xaml.Shapes;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.ApplicationModel;
using Windows.ApplicationModel.Activation;
using Windows.Foundation;
```

```
using Windows.Foundation.Collections;

// To learn more about WinUI, the WinUI project structure,
// and more about our project templates, see: http://aka.ms/winui-project-info.

namespace table4
{
    /// <summary>
    /// Provides application-specific behavior to supplement the default Application
class.
    /// </summary>
    public partial class App : Application
    {
        /// <summary>
        /// Initializes the singleton application object.  This is the first line of
authored code
        /// executed, and as such is the logical equivalent of main() or WinMain().
        /// </summary>
        public App()
        {
            this.InitializeComponent();
        }

        /// <summary>
        /// Invoked when the application is launched normally by the end user.  Other
entry points
        /// will be used such as when the application is launched to open a specific
file.
        /// </summary>
        /// <param name="args">Details about the launch request and process.</param>
        protected override void OnLaunched(Microsoft.UI.Xaml.LaunchActivatedEventArgs
args)
        {
            m_window = new MainWindow();
            m_window.Activate();
            using(var db=new table4.Model.DataCont())
            {
                db.Database.Migrate();
            }
        }

        private Window m_window;
    }
}
```

Output:


WinUI Desktop
Title
Content
Save
Posts
4
Rinkal Tailor
5
KITL
5
KITL

Date 5-1-2021
Mvvm



- The Model is the layer that deals with data and raw content . It is often involved with obtaining and maintaining data from files or web services.
- The View is the presentation layer of controls and graphics, generally implemented in XAML .
- The View Model sits between the Model and View . In the general case, it is responsible for making the data or content from the Model more conducive to the View.

ColorScrollWithViewModel
Code:
MainPage.xaml

```
<Page>
    <Page.Resources>
        <Style TargetType="TextBlock">
        <Setter Property="FontSize" Value="24" />
        <Setter Property="Margin" Value="24 0 0 0" />
        <Setter Property="VerticalAlignment" Value="Center" />
    </Style>

    <Style TargetType="TextBox">
```

```
        <Setter Property="Margin" Value="24 48 96 48" />
        <Setter Property="VerticalAlignment" Value="Center" />
    </Style>
</Page.Resources>
<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>

    <Grid Grid.Column="0">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="Auto" />
            <RowDefinition Height="Auto" />
        </Grid.RowDefinitions>

        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>

        <TextBlock Text="Red: "
 Grid.Row="0"
 Grid.Column="0" />

        <TextBox Text="{Binding Red, Mode=TwoWay}"
 Grid.Row="0"
 Grid.Column="1" />
        <TextBlock Text="Green: "
 Grid.Row="1"
 Grid.Column="0" />
        <TextBox Text="{Binding Green, Mode=TwoWay}"
 Grid.Row="1"
 Grid.Column="1" />
        <TextBlock Text="Blue: "
 Grid.Row="2"
 Grid.Column="0" />
        <TextBox Text="{Binding Blue, Mode=TwoWay}"
 Grid.Row="2"
 Grid.Column="1" />
    </Grid>
    <!-- Result -->
    <Rectangle Grid.Column="1">
        <Rectangle.Fill>
            <SolidColorBrush Color="{Binding Color}" />
        </Rectangle.Fill>
    </Rectangle>
</Grid>
</Page>
```

RgbViewModel.cs

```
       using System.ComponentModel;
using System.Runtime.CompilerServices;
using Windows.UI;
namespace ColorScrollWithDataContext
```

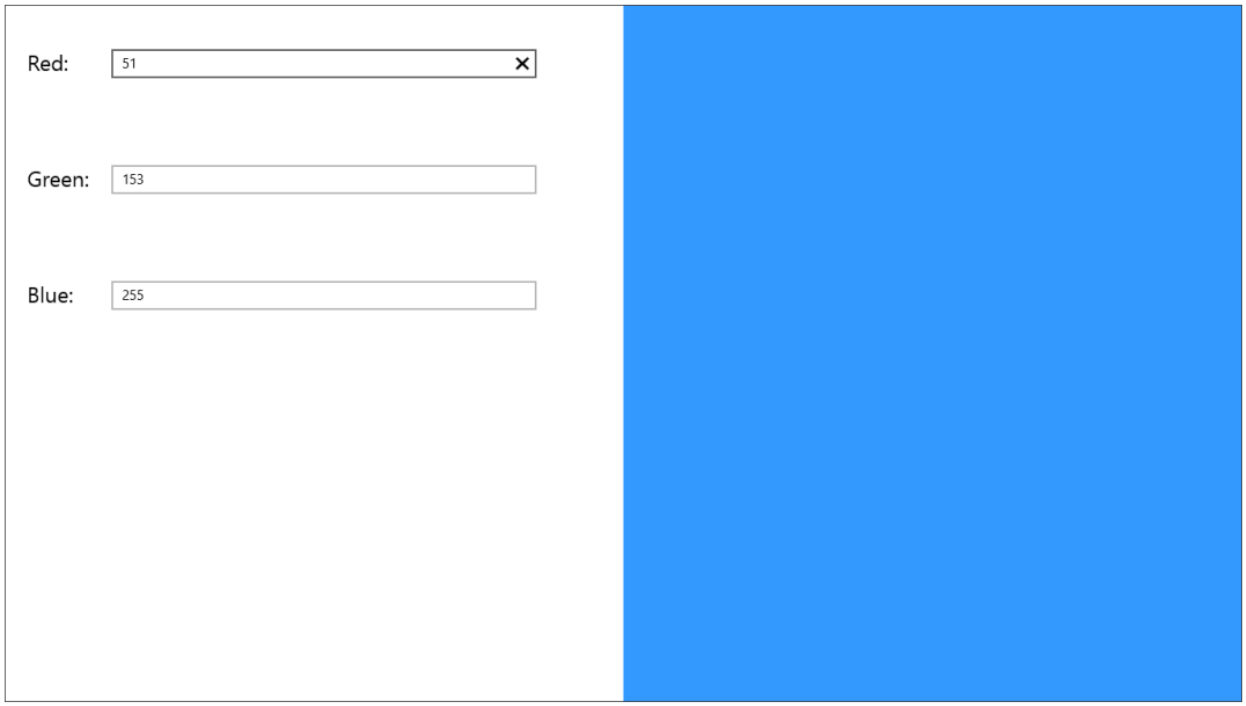
```
{
    public class RgbViewModel : INotifyPropertyChanged
    {
        double red, green, blue;
 Color color = Color.FromArgb(255, 0, 0, 0);
        public event PropertyChangedEventHandler PropertyChanged;
        public double Red
        {
            set
            {
                if (SetProperty<double>(ref red, value, "Red"))
                    Calculate();
            }
            get
            {
                return red;
            }
        }
        public double Green
        {
            set
            {
                if (SetProperty<double>(ref green, value))
                    Calculate();
            }
            get
            {
                return green;
            }
        }
        public double Blue
        {
            set
            {
                if (SetProperty<double>(ref blue, value))
                    Calculate();
            }
            get
            {
                return blue;
            }
        }
        public Color Color
        {
            set
            {
                if (SetProperty<Color>(ref color, value))
                {
                    this.Red = value.R;
                    this.Green = value.G;
                    this.Blue = value.B;
                }
            }
            get
            {
                return color;
            }
        }
```

```
        void Calculate()
        {
            this.Color = Color.FromArgb(255, (byte)this.Red, (byte)this.Green,
(byte)this.Blue);
        }
        protected bool SetProperty<T>(ref T storage, T value,
        [CallerMemberName] string propertyName = null)
        {
            if (object.Equals(storage, value))
                return false;
            storage = value;
            OnPropertyChanged(propertyName);
            return true;
        }
        protected void OnPropertyChanged(string propertyName)
        {
            if (PropertyChanged != null)
                PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}
```

MainPage.xaml.cs

```
public MainPage()
{
    this.InitializeComponent();
    this.DataContext = new RgbViewModel();
    // Initialize to highlight color
    (this.DataContext as RgbViewModel).Color =
    new UISettings().UIElementColor(UIElementType.Highlight);
}
```

Output:



Date 6-1-2021

Asynchronicity

Reason of using asynchronous

- It’s important that applications process events as quickly as possible and then return control back to the operating system to wait for more events.
- If an application doesn’t process an event quickly,
- it could become unresponsive and annoy the user .

Use

- file operation
- Message dialougbox
- Working with message dialoug box

Methods

- OnMessageDialogShowAsyncCompleted
- OnDispatcherRunAsyncCallback

Code:

Github Link-Full-semester-external-Project-Work-Automated-Sofware/MessageDialog at main · rinkal651/Full-semester-external-Project-Work-Automated-Sofware (github.com)

MainWindow.xaml

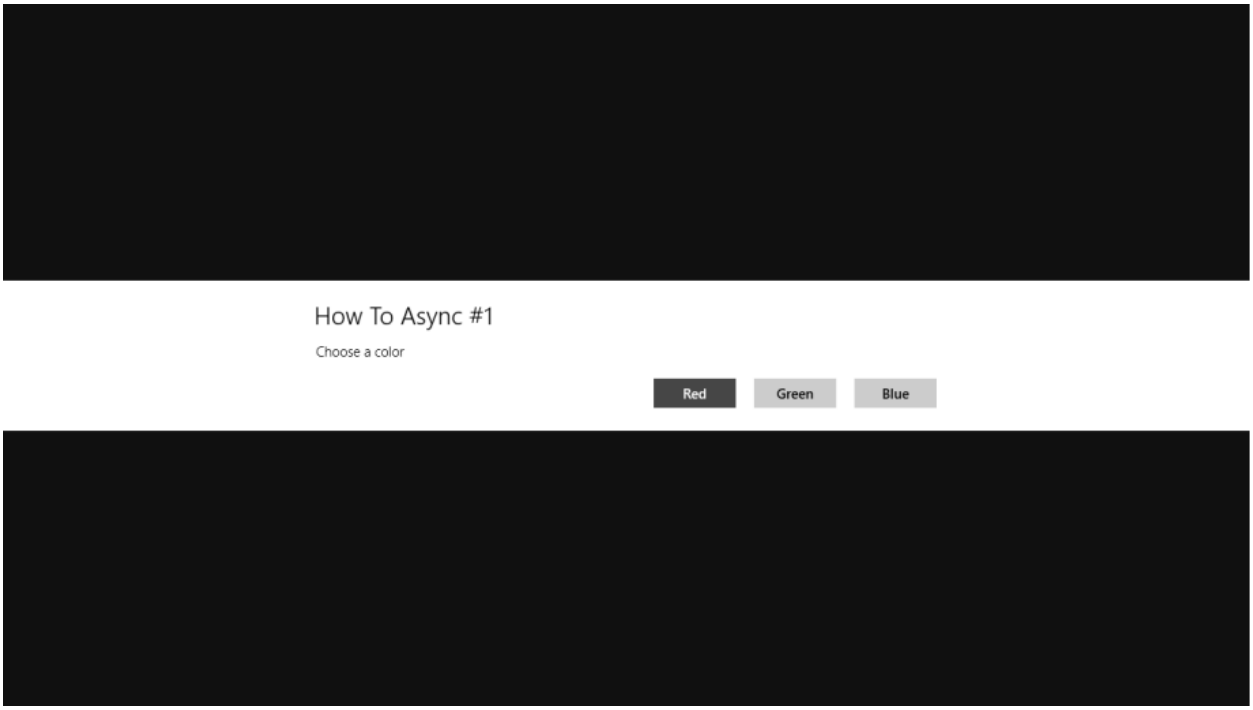
```
<Window
    x:Class="Report2_Project.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:Report2_Project"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    xmlns:muxc="using:Microsoft.UI.Xaml.Controls">

    <Grid Name="contentGrid"
 Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
        <Button Content="Show me a MessageDialog!"
 HorizontalAlignment="Center"
 VerticalAlignment="Center"
 Click="OnButtonClick" />

    </Grid>

</Window>
```

MainWindow.xaml.cs

```
using Microsoft.UI;
using Microsoft.UI.Xaml;
using Microsoft.UI.Xaml.Controls;
using Microsoft.UI.Xaml.Controls.Primitives;
using Microsoft.UI.Xaml.Data;
using Microsoft.UI.Xaml.Input;
using Microsoft.UI.Xaml.Media;
using Microsoft.UI.Xaml.Media.Imaging;
using Microsoft.UI.Xaml.Navigation;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.ApplicationModel.DataTransfer;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.Storage;
using Windows.UI.Popups;

// To learn more about WinUI, the WinUI project structure,
// and more about our project templates, see: http://aka.ms/winui-project-info.

namespace Report2_Project
{
    /// <summary>
```

```
    /// An empty window that can be used on its own or navigated to within a Frame.
    /// </summary>
    ///


    public sealed partial class MainWindow : Window
    {
        Color clr;
        public MainWindow()
        {
            this.InitializeComponent();

        }
        void OnButtonClick(object sender, RoutedEventArgs args)
        {
            MessageDialog msgdlg = new MessageDialog("Choose a color", "How To Async
#1");
            msgdlg.Commands.Add(new UICommand("Red", null, Colors.Red));
            msgdlg.Commands.Add(new UICommand("Green", null, Colors.Green));
            msgdlg.Commands.Add(new UICommand("Blue", null, Colors.Blue));
            // Show the MessageDialog with a Completed handler
            IAsyncOperation<IUICommand> asyncOp = msgdlg.ShowAsync();
            asyncOp.Completed = OnMessageDialogShowAsyncCompleted;
        }
        void OnMessageDialogShowAsyncCompleted(IAsyncOperation<IUICommand> asyncInfo,
        AsyncStatus asyncStatus)
        {
            // Get the Color value
            IUICommand command = asyncInfo.GetResults();
            clr = (Color)command.Id;
            // Use a Dispatcher to run in the UI thread
            IAsyncAction asyncAction =
this.Dispatcher.RunAsync(CoreDispatcherPriority.Normal,
            OnDispatcherRunAsyncCallback);
        }
        void OnDispatcherRunAsyncCallback()
        {
            // Set the background brush
            contentGrid.Background = new SolidColorBrush(clr);
        }


    }


}
```

Output



Callback as a lambda function

Code

Github Link- Full-semester-external-Project-Work-Automated-Sofware/LambdaFunctionDialoug at main · rinkal651/Full-semester-external-Project-Work-Automated-Sofware (github.com)

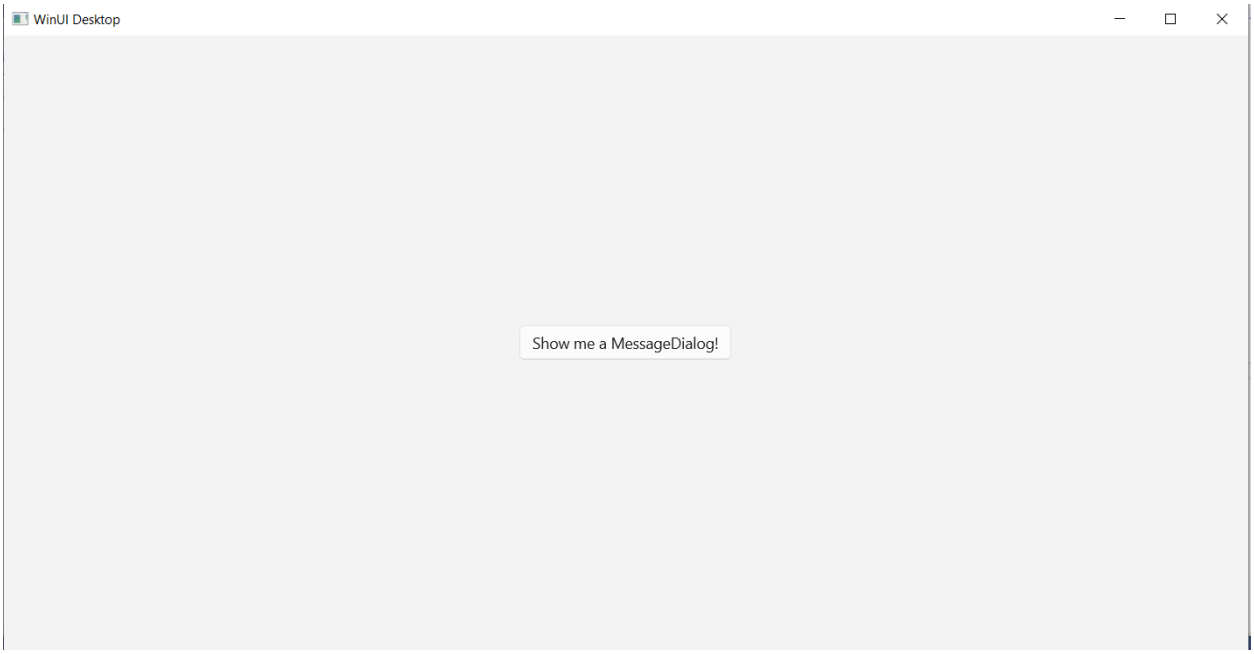
MainWindow.xaml

```
<Window
    x:Class="Report2_Project.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:Report2_Project"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    xmlns:muxc="using:Microsoft.UI.Xaml.Controls">

    <Grid Name="contentGrid"
 Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
        <Button Content="Show me a MessageDialog!"
 HorizontalAlignment="Center"
 VerticalAlignment="Center"
 Click="OnButtonClick" />

    </Grid>

</Window>
```

MainWindow.xaml.cs

```
using Microsoft.UI;
using Microsoft.UI.Xaml;
using Microsoft.UI.Xaml.Controls;
using Microsoft.UI.Xaml.Controls.Primitives;
using Microsoft.UI.Xaml.Data;
using Microsoft.UI.Xaml.Input;
using Microsoft.UI.Xaml.Media;
using Microsoft.UI.Xaml.Media.Imaging;
using Microsoft.UI.Xaml.Navigation;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.ApplicationModel.DataTransfer;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.Storage;
using Windows.UI.Core;
using Windows.UI;
using Windows.UI.Popups;

// To learn more about WinUI, the WinUI project structure,
```

```
// and more about our project templates, see: http://aka.ms/winui-project-info.

namespace Report2_Project
{
    /// <summary>
    /// An empty window that can be used on its own or navigated to within a Frame.
    /// </summary>
    ///


    public sealed partial class MainWindow : Window
    {
        Color clr;
        public MainWindow()
        {
            this.InitializeComponent();

        }
        void OnButtonClick(object sender, RoutedEventArgs args)
        {
            MessageDialog msgdlg = new MessageDialog("Choose a color", "How To Async
#2");
            msgdlg.Commands.Add(new UICommand("Red", null, Colors.Red));
            msgdlg.Commands.Add(new UICommand("Green", null, Colors.Green));
            msgdlg.Commands.Add(new UICommand("Blue", null, Colors.Blue));
            // Show the MessageDialog with a Completed handler
            IAsyncOperation<IUICommand> asyncOp = msgdlg.ShowAsync();
            asyncOp.Completed = (asyncInfo, asyncStatus) =>
            {
                // Get the Color value
                IUICommand command = asyncInfo.GetResults();
                Color clr = (Color)command.Id;
                // Use a Dispatcher to run in the UI thread
                IAsyncAction asyncAction =
this.Dispatcher.RunAsync(CoreDispatcherPriority.Normal,
                () =>
                {
                    // Set the background brush
                    contentGrid.Background = new SolidColorBrush(clr);
                });
            };
        }



    }


}
```



Characteristic of Asynch operation

- Cancellation
- Progress
- Error

Working with file
Open a file, Save as a file operations.

Code:
Github Link-Full-semester-external-Project-Work-Automated-Sofware/FileOperation at main · rinkal651/Full-semester-external-Project-Work-Automated-Sofware (github.com)
MainWindow.xaml

```
<Page
    x:Class="Report2_Program.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:Report2_Program"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
    <Page.Resources>
        <Style x:Key="buttonStyle" TargetType="ButtonBase">
            <Setter Property="HorizontalAlignment" Value="Center" />
            <Setter Property="Margin" Value="0 12" />
        </Style>
    </Page.Resources>
    <Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>

        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>

        <Button Content="Open..."
Grid.Row="0"
Grid.Column="0"
Style="{StaticResource buttonStyle}"
Click="OnFileOpenButtonClick" />

        <Button Content="Save As..."
Grid.Row="0"
Grid.Column="1"
Style="{StaticResource buttonStyle}"
Click="OnFileSaveAsButtonClick" />

        <ToggleButton Name="wrapButton"
Content="No Wrap"
Grid.Row="0"
Grid.Column="2"
Style="{StaticResource buttonStyle}"
Checked="OnWrapButtonChecked"
Unchecked="OnWrapButtonChecked" />
```

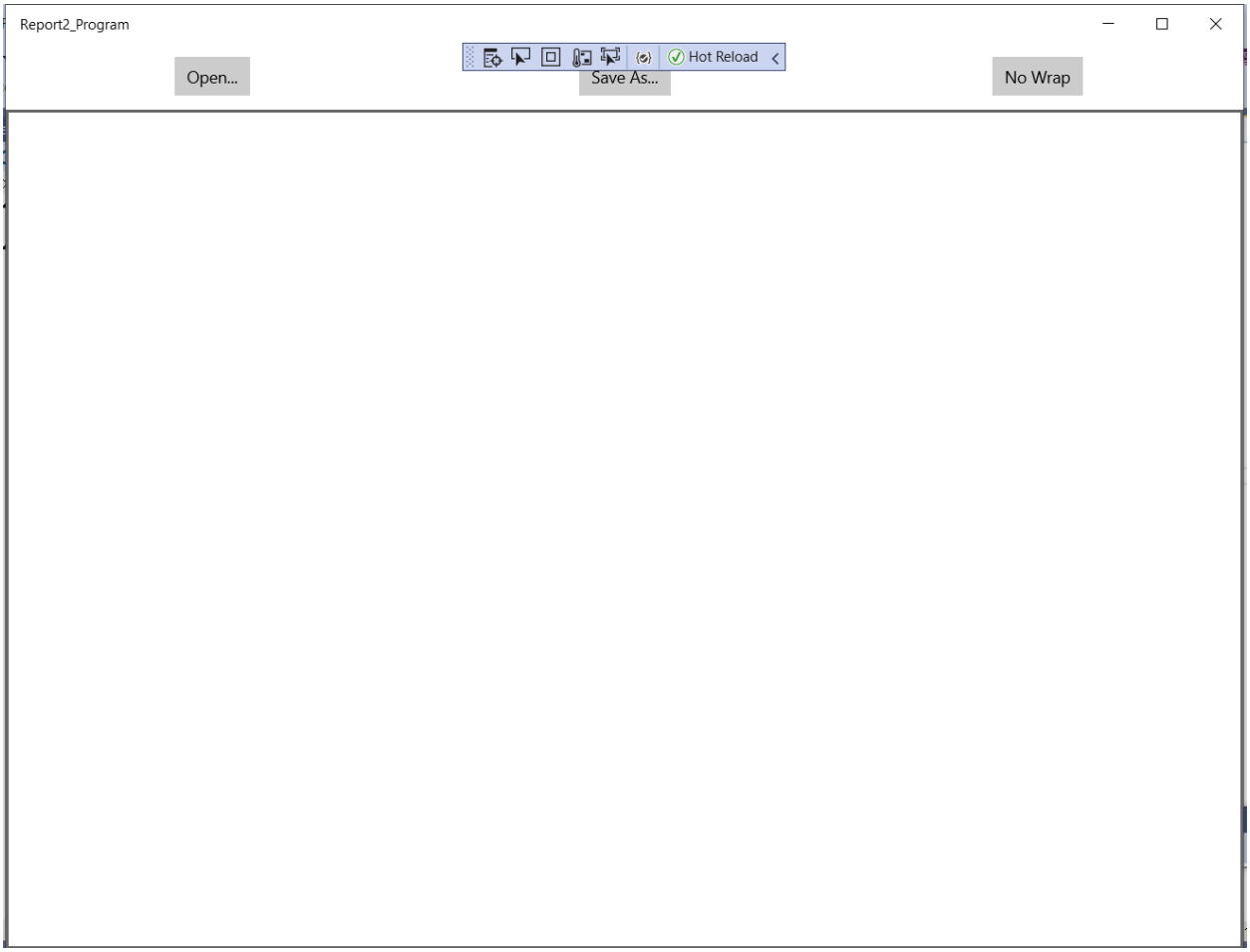
```
        <TextBox Name="txtbox"
 Grid.Row="1"
 Grid.Column="0"
 Grid.ColumnSpan="3"
 FontSize="24"
 AcceptsReturn="True" />
    </Grid>

</Page>
```

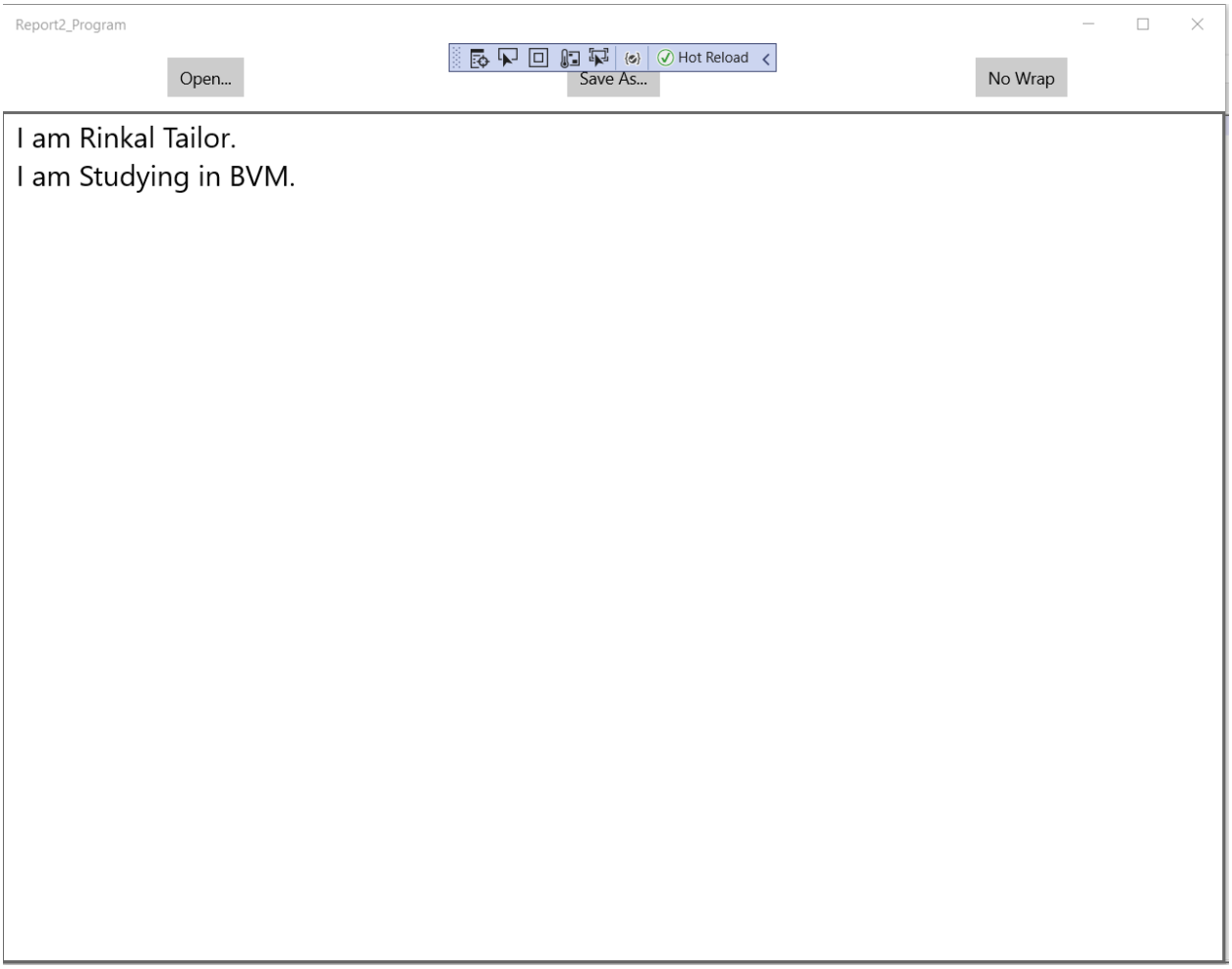
MainWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.UI;
using Microsoft.UI;
using Windows.UI.Core;
using Windows.UI.Popups;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;
using Windows.Storage.Pickers;
using Windows.Storage;
using Windows.Storage.Streams;
using System.Threading.Tasks;

// The Blank Page item template is documented at
https://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x409

namespace Report2_Program
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a Frame.
    /// </summary>

    public sealed partial class MainPage : Page
    {
        ApplicationDataContainer appData = ApplicationData.Current.LocalSettings;

        public MainPage()
        {
            this.InitializeComponent();
            Loaded += (sender, args) =>
            {
                if (appData.Values.ContainsKey("TextWrapping"))
                    txtbox.TextWrapping = (TextWrapping)appData.Values["TextWrapping"];
                wrapButton.IsChecked = txtbox.TextWrapping == TextWrapping.Wrap;
                wrapButton.Content = (bool)wrapButton.IsChecked ? "Wrap" : "No Wrap";
```

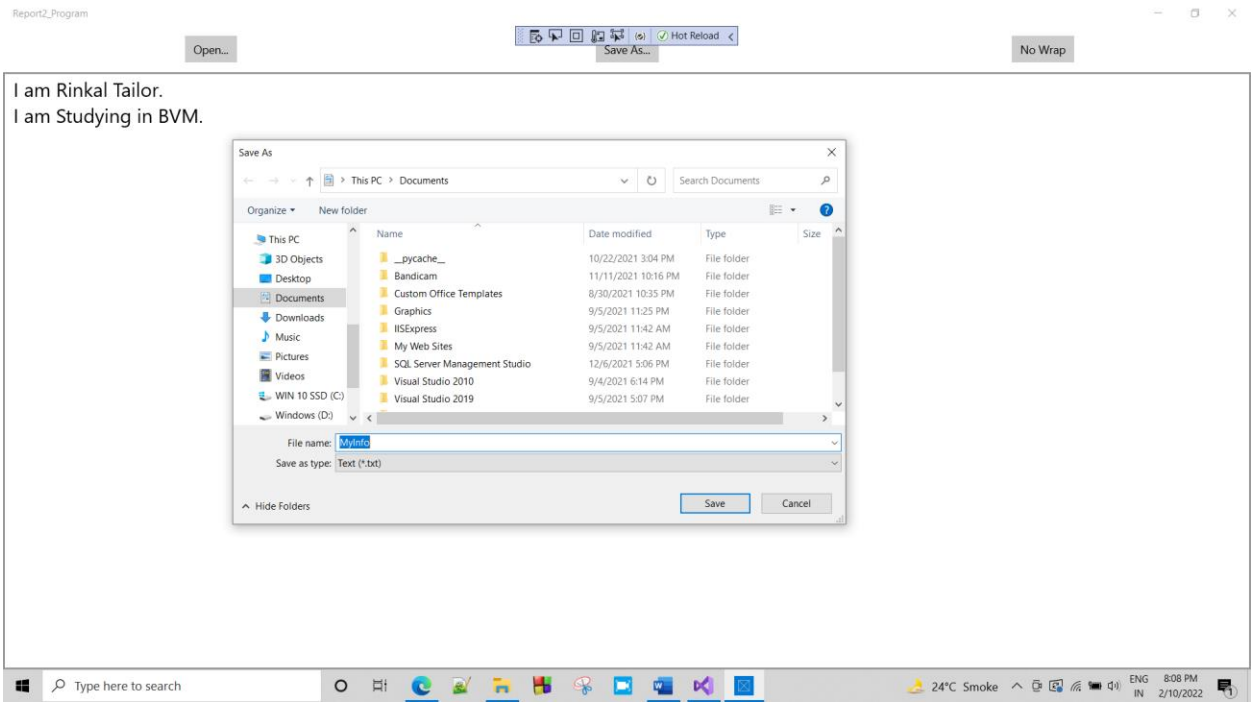
```
                txtbox.Focus(FocusState.Programmatic);
            };

        }

        async void OnFileOpenButtonClick(object sender, RoutedEventArgs args)
        {
            FileOpenPicker picker = new FileOpenPicker();
            picker.FileTypeFilter.Add(".txt");
            StorageFile storageFile = await picker.PickSingleFileAsync();
            if (storageFile == null)
                return;
            using (IRandomAccessStream stream = await storageFile.OpenReadAsync())
            {
                using (DataReader dataReader = new DataReader(stream))
                {
                    uint length = (uint)stream.Size;
                    await dataReader.LoadAsync(length);
                    txtbox.Text = dataReader.ReadString(length);
                }
            }

        }
        void OnWrapButtonChecked(object sender, RoutedEventArgs args)
        {
            txtbox.TextWrapping = (bool)wrapButton.IsChecked ? TextWrapping.Wrap :
            TextWrapping.NoWrap;
            wrapButton.Content = (bool)wrapButton.IsChecked ? "Wrap" : "No Wrap";
            appData.Values["TextWrapping"] = (int)txtbox.TextWrapping;
        }
        async void OnFileSaveAsButtonClick(object sender, RoutedEventArgs args)
        {
            await SaveFileAsync(txtbox.Text);
        }
        async Task SaveFileAsync(string text)
        {
            FileSavePicker picker = new FileSavePicker();
            picker.DefaultFileExtension = ".txt";
            picker.FileTypeChoices.Add("Text", new List<string> { ".txt" });
            StorageFile storageFile = await picker.PickSaveFileAsync();
            // If user presses Cancel, result is null
            if (storageFile == null)
                return;
            using (IRandomAccessStream stream = await
storageFile.OpenAsync(FileAccessMode.ReadWrite))
            {
                using (DataWriter dataWriter = new DataWriter(stream))
                {
                    dataWriter.WriteString(text);
                    await dataWriter.StoreAsync();
                }
            }
        }
    }
}
```

Output:

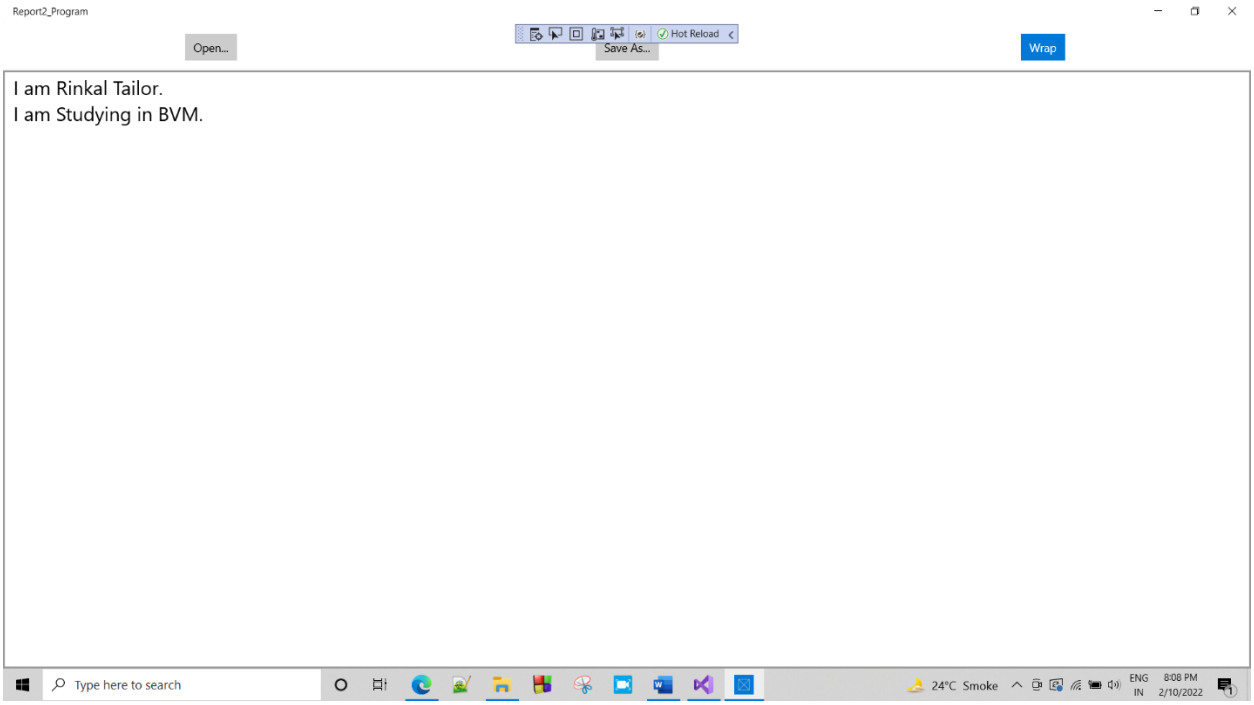




Save as operation in app



Wrap Mode in output



Date 7-1-2021

Popup dailog

Popup Dialoag with different actions like change color and font size change.

Code:

Github Link Full-semester-external-Project-Work-Automated-Sofware/PopUpMenuwithOperation at main · rinkal651/Full-semester-external-Project-Work-Automated-Sofware (github.com)

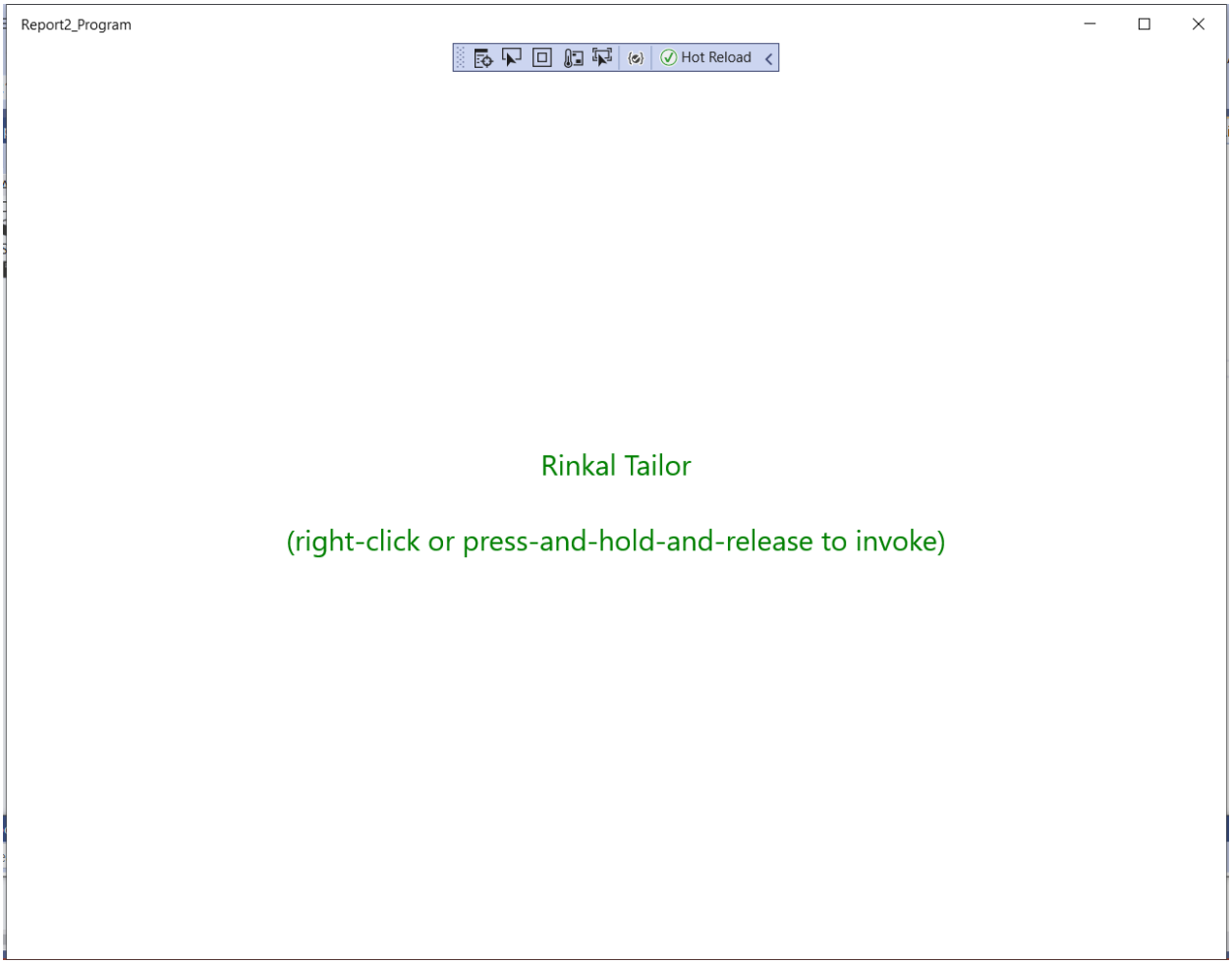
MainWindow.xaml

```
<Page
    x:Class="Report2_Program.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:Report2_Program"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
        <TextBlock Name="textBlock"
 FontSize="24"
 HorizontalAlignment="Center"
 VerticalAlignment="Center"
 TextAlignment="Center"
 RightTapped="OnTextBlockRightTapped">
Rinkal Tailor
 <LineBreak />
 <LineBreak />
 (right-click or press-and-hold-and-release to invoke)
        </TextBlock>
    </Grid>


</Page>
```

MainWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.UI;
using Microsoft.UI;
using Windows.UI.Core;
using Windows.UI.Popups;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
```

```
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;
using Windows.Storage.Pickers;
using Windows.Storage;
using Windows.Storage.Streams;
using System.Threading.Tasks;

// The Blank Page item template is documented at
https://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x409

namespace Report2_Program
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a Frame.
    /// </summary>

    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();


        }

        async void OnTextBlockRightTapped(object sender, RightTappedRoutedEventArgs args)
        {
            PopupMenu popupMenu = new PopupMenu();
            popupMenu.Commands.Add(new UICommand("Larger Font", OnFontSizeChanged, 1.2));
            popupMenu.Commands.Add(new UICommand("Smaller Font", OnFontSizeChanged, 1 /
1.2));
            popupMenu.Commands.Add(new UICommandSeparator());
            popupMenu.Commands.Add(new UICommand("Red", OnColorChanged, Colors.Red));
            popupMenu.Commands.Add(new UICommand("Green", OnColorChanged, Colors.Green));
            popupMenu.Commands.Add(new UICommand("Blue", OnColorChanged, Colors.Blue));
            await popupMenu.ShowAsync(args.GetPosition(this));
        }
        void OnFontSizeChanged(IUICommand command)
        {
            textBlock.FontSize *= (double)command.Id;
        }
        void OnColorChanged(IUICommand command)
        {
            textBlock.Foreground = new SolidColorBrush((Color)command.Id);
        }
    }
}
```


Report2_Program
Hot Reload
Rinkal Tailor
(right-click or press-and-hold-and-release to invoke)

Date-8-1-2021

Appbar for notepad
Create an unconventinal notepad.

Github Link- https://github.com/rinkal651/Full-semester-external-Project-Work-Automated-Sofware/blob/main/UnconventionalAppbar

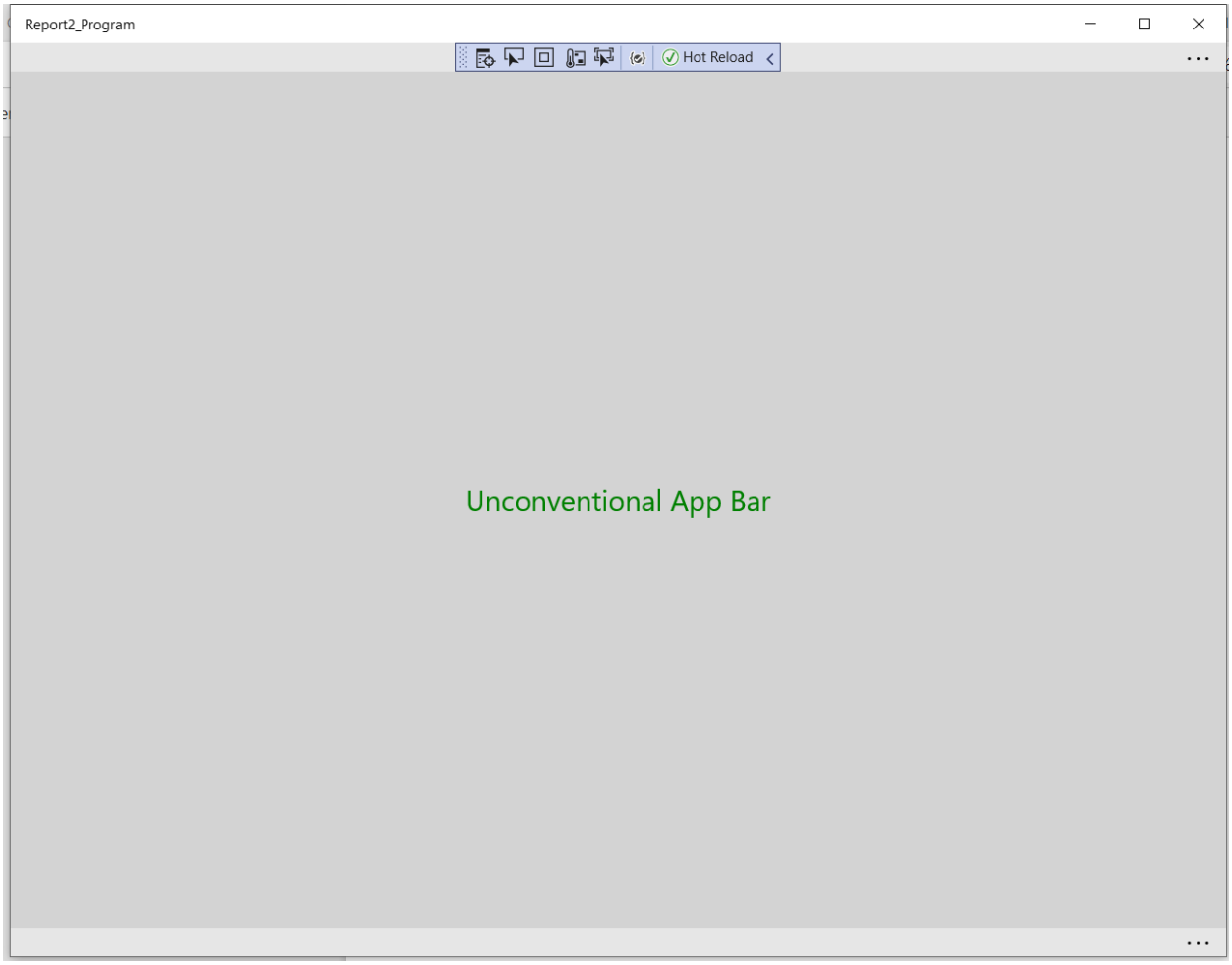
MainWindow.xaml

```
<Page
    x:Class="Report2_Program.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:Report2_Program"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
    <Grid Background="LightGray">
        <TextBlock Name="textBlock"
 Text="Unconventional App Bar  Rinkal Tailor KITL"
 HorizontalAlignment="Center"
 VerticalAlignment="Center"
 FontSize="{Binding ElementName=slider, Path=Value}" />
    </Grid>
    <Page.TopAppBar>
        <AppBar Name="topAppBar">
            <Slider Name="slider"
 Minimum="8"
 Maximum="196"
 Value="24" />
        </AppBar>
    </Page.TopAppBar>
    <Page.BottomAppBar>
        <AppBar Name="bottomAppBar">
            <StackPanel Orientation="Horizontal"
 HorizontalAlignment="Right">

                <Button Content="Red"
 Foreground="Red"
 Margin="24 12"
 Click="OnAppBarButtonClick" />

                <Button Content="Green"
 Foreground="Green"
 Margin="24 12"
 Click="OnAppBarButtonClick" />
                <Button Content="Blue"
 Foreground="Blue"
 Margin="24 12"
```

```
 Click="OnAppBarButtonClick" />
            </StackPanel>
        </AppBar>
    </Page.BottomAppBar>
</Page>
```

MainWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.UI;
using Microsoft.UI;
using Windows.UI.Core;
using Windows.UI.Popups;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;
using Windows.Storage.Pickers;
using Windows.Storage;
using Windows.Storage.Streams;
using System.Threading.Tasks;

// The Blank Page item template is documented at
https://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x409

namespace Report2_Program
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a Frame.
    /// </summary>

    public sealed partial class MainPage : Page
    {

        public MainPage()
        {
            this.InitializeComponent();
        }
        void OnAppBarButtonClick(object sender, RoutedEventArgs args)
        {

            textBlock.Foreground = (sender as Button).Foreground;
            topAppBar.IsOpen = false;
            bottomAppBar.IsOpen = false;
        }
    }
}
```


Report2_Program
Hot Reload
Unconventional App Bar

TextFromattingApp

Create a simple Text fromatting app which can do Under line, bold, italic and different font color.

Code:

Github Link-Full-semester-external-Project-Work-Automated-Sofware/TextFormattingApp at main · rinkal651/Full-semester-external-Project-Work-Automated-Sofware (github.com)

MainWindow.xaml

```
<Page
    x:Class="Report2_Program.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:Report2_Program"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
    <Grid Background="LightGray">
        <TextBlock Name="textBlock"
 FontFamily="Times New Roman"
 FontSize="96"
 HorizontalAlignment="Center"
 VerticalAlignment="Center">
 <Run> Rinkal Tailor From BVM</Run>
        </TextBlock>
    </Grid>

    <Page.BottomAppBar>
        <AppBar>
            <StackPanel Orientation="Horizontal">
                <CheckBox
 Checked="OnBoldAppBarCheckBoxChecked"
 Unchecked="OnBoldAppBarCheckBoxChecked" />
                <CheckBox
 Checked="OnItalicAppBarCheckBoxChecked"
 Unchecked="OnItalicAppBarCheckBoxChecked" />
                <CheckBox
 Checked="OnUnderlineAppBarCheckBoxChecked"
 Unchecked="OnUnderlineAppBarCheckBoxChecked" />
                <Polyline Points="0 12, 0 48"
 Stroke="{StaticResource ApplicationForegroundThemeBrush}"
 VerticalAlignment="Top" />

                <RadioButton Name="redRadioButton"

 Foreground="Red"
 AutomationProperties.Name="Red"
 Checked="OnFontColorAppBarRadioButtonChecked" />
                <RadioButton
 Foreground="Green"
 AutomationProperties.Name="Green"
 Checked="OnFontColorAppBarRadioButtonChecked" />
                <RadioButton
```

```
 Foreground="Blue"
 AutomationProperties.Name="Blue"
 Checked="OnFontColorAppBarRadioButtonChecked" />
            </StackPanel>
        </AppBar>
    </Page.BottomAppBar>
</Page>
```

MainWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.UI;
using Microsoft.UI;
using Windows.UI.Core;
using Windows.UI.Popups;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;
using Windows.Storage.Pickers;
using Windows.Storage;
using Windows.Storage.Streams;
using System.Threading.Tasks;
using Windows.UI.Text;
using Windows.UI.Xaml.Documents;

// The Blank Page item template is documented at
https://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x409

namespace Report2_Program
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a Frame.
    /// </summary>

    public sealed partial class MainPage : Page
    {

        public MainPage()
        {
            this.InitializeComponent();

        }
        void OnBoldAppBarCheckBoxChecked(object sender, RoutedEventArgs args)
        {
            CheckBox chkbox = sender as CheckBox;
            textBlock.FontWeight = (bool)chkbox.IsChecked ? FontWeights.Bold :
FontWeights.Normal;
        }
        void OnItalicAppBarCheckBoxChecked(object sender, RoutedEventArgs args)
```

```
        {
            CheckBox chkbox = sender as CheckBox;
            textBlock.FontStyle = (bool)chkbox.IsChecked ? FontStyle.Italic :
FontStyle.Normal;
        }
        void OnUnderlineAppBarCheckBoxChecked(object sender, RoutedEventArgs args)
        {
            CheckBox chkbox = sender as CheckBox;
            Inline inline = textBlock.Inlines[0];
            if ((bool)chkbox.IsChecked && !(inline is Underline))
            {
                Underline underline = new Underline();
                textBlock.Inlines[0] = underline;
                underline.Inlines.Add(inline);
            }
            else if (!(bool)chkbox.IsChecked && inline is Underline)
            {
                Underline underline = inline as Underline;
                Run run = underline.Inlines[0] as Run;
                underline.Inlines.Clear();
                textBlock.Inlines[0] = run;
            }
        }
        void OnFontColorAppBarRadioButtonChecked(object sender, RoutedEventArgs args)
        {
            textBlock.Foreground = (sender as RadioButton).Foreground;
        }

    }
}
```

Output:

Date 9-1-2021-10-1-2021

Wheather api learn
With weather api link fetch the data.

WeatherMapProxy.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;
using System.Runtime.Serialization.Json;
using System.Runtime.Serialization;
using System.IO;

namespace Weather_API
{
   public class WeatherMapProxy
    {
        public async static Task<Root> GetWeather(double lat, double lan)
        {
            var http=new HttpClient();
            var response = await
http.GetAsync("http://api.openweathermap.org/data/2.5/weather?lat=32.77&lon=-
96.79&appid=bcdb1d761b938e64104da104d8800965");
            var result = await response.Content.ReadAsStringAsync();
            var serializer = new DataContractJsonSerializer(typeof(Root));
            var ms = new MemoryStream(Encoding.UTF8.GetBytes(result));
            var data = (Root) serializer.ReadObject(ms);
            return data;
        }
    }
    [DataContract]
    public class Coord
    {
        [DataMember]
        public double lon { get; set; }
        [DataMember]
        public double lat { get; set; }
    }

    [DataContract]
    public class Weather
    {
        [DataMember]
        public int id { get; set; }
        [DataMember]
        public string main { get; set; }
        [DataMember]
        public string description { get; set; }
        [DataMember]
        public string icon { get; set; }
    }
```

```
[DataContract]
public class Main
{
    [DataMember]
    public double temp { get; set; }
    [DataMember]
    public double feels_like { get; set; }
    [DataMember]
    public double temp_min { get; set; }
    [DataMember]
    public double temp_max { get; set; }
    [DataMember]
    public double pressure { get; set; }
    [DataMember]
    public double humidity { get; set; }
}

[DataContract]
public class Wind
{
    [DataMember]
    public double speed { get; set; }
    [DataMember]
    public double deg { get; set; }
}

[DataContract]
public class Clouds
{
    [DataMember]
    public double all { get; set; }
}

[DataContract]
public class Sys
{
    [DataMember]
    public int type { get; set; }
    [DataMember]
    public int id { get; set; }
    [DataMember]
    public string country { get; set; }
    [DataMember]
    public int sunrise { get; set; }
    [DataMember]
    public int sunset { get; set; }
}

[DataContract]
public class Root
{
    [DataMember]
    public Coord coord { get; set; }
    [DataMember]
    public List<Weather> weather { get; set; }
    [DataMember]
    public string @base { get; set; }
```

```
        [DataMember]
        public Main main { get; set; }
        [DataMember]
        public double visibility { get; set; }
        [DataMember]
        public Wind wind { get; set; }
        [DataMember]
        public Clouds clouds { get; set; }
        [DataMember]
        public double dt { get; set; }
        [DataMember]
        public Sys sys { get; set; }
        [DataMember]
        public int timezone { get; set; }
        [DataMember]
        public int id { get; set; }
        [DataMember]
        public string name { get; set; }
        [DataMember]
        public int cod { get; set; }
    }
}
```

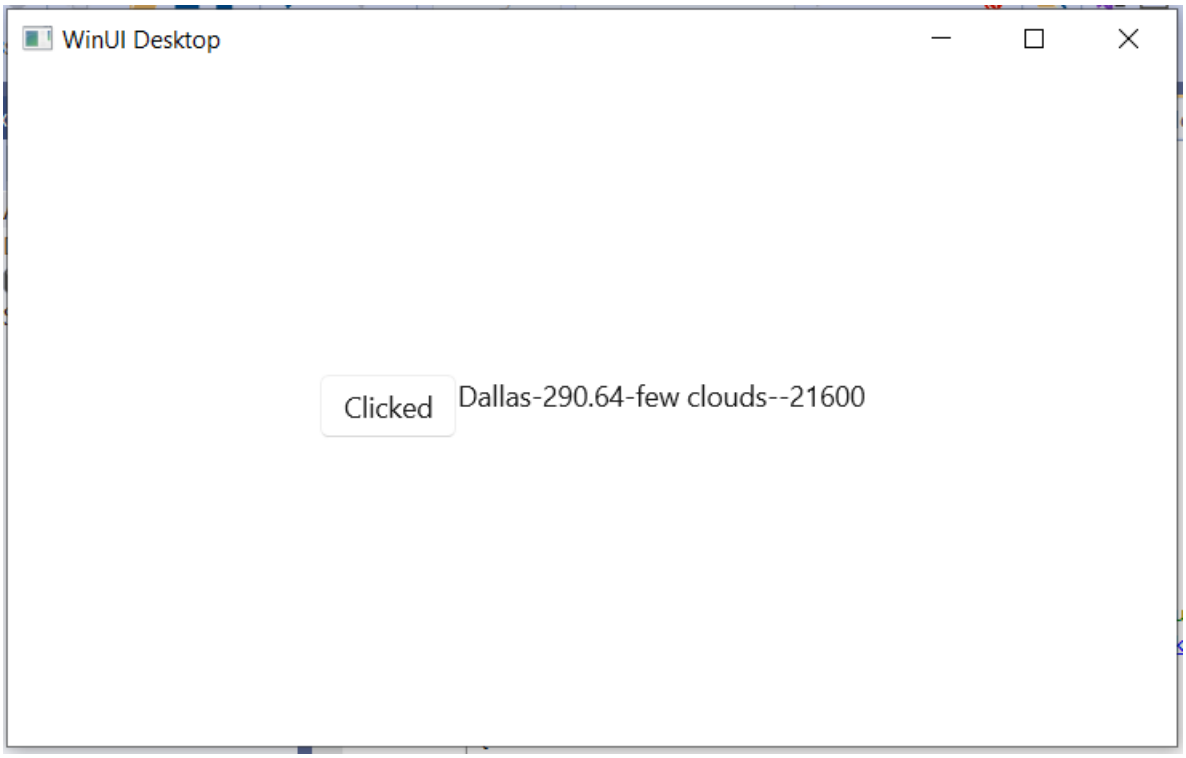
MainWindow.xaml

```
<Window
    x:Class="Weather_API.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:Weather_API"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">

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

MainWindow.xaml.cs

```
using Microsoft.UI.Xaml;
using Microsoft.UI.Xaml.Controls;
using Microsoft.UI.Xaml.Controls.Primitives;
using Microsoft.UI.Xaml.Data;
using Microsoft.UI.Xaml.Input;
using Microsoft.UI.Xaml.Media;
using Microsoft.UI.Xaml.Navigation;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.Foundation;
```

```
using Windows.Foundation.Collections;

// To learn more about WinUI, the WinUI project structure,
// and more about our project templates, see: http://aka.ms/winui-project-info.

namespace Weather_API
{
    /// <summary>
    /// An empty window that can be used on its own or navigated to within a Frame.
    /// </summary>
    public sealed partial class MainWindow : Window
    {
        public MainWindow()
        {
            this.InitializeComponent();
        }

        private async void myButton_Click(object sender, RoutedEventArgs e)
        {
            Root my = await WeatherMapProxy.GetWeather(20.0, -30.0);
            tr.Text = my.name + "-" + my.main.temp + "-" + my.weather[0].description+"-"+
my.timezone;
            myButton.Content = "Clicked";
        }
    }
}
```



Date 11-1-2021

Shopify api learn

On shopify website I have created account and by development personal api I have fetched the product details from the website.

Console app

```
using System;
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Net;
using System.Net.Http.Headers;
using System.Threading.Tasks;
using System.Runtime.Serialization.Json;
using System.Runtime.Serialization;
using System.IO;
using System.Text;

namespace ConsoleApp3
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
            Main1();
        }


        public static async void Main1()
        {
            Console.WriteLine("start");
            string ShopPath = "/admin/api/2022-01/";
            string shopifyGetProductFmt = "products.json";
            string URL = string.Format("https://{0}.myshopify.com{1}", "rinkal-tailor",
ShopPath);
            var clientHandler = new HttpClientHandler
            {
                Credentials = new NetworkCredential("7f4121cfa5915793e3f2aef339dd87f7",
"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"),
                PreAuthenticate = true
            };
            HttpClient client = new HttpClient(clientHandler);
            client.BaseAddress = new Uri(URL);

            // Add an Accept header for JSON format.

            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));

            // need an access token...
```

```
            client.DefaultRequestHeaders.Add("7f4121cfa5915793e3f2aef339dd87f7",
"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");

            string shopifyGetProduct = String.Format(shopifyGetProductFmt);
            var getResponse = await
client.GetAsync(shopifyGetProduct).Result.Content.ReadAsStringAsync();  // Blocking call!
Program will wait here until a response is received or a timeout occurs.

//  var serializer = new DataContractJsonSerializer(typeof(Data));

// var ms = new MemoryStream(Encoding.UTF8.GetBytes(getResponse));

//var data = (Data)serializer.ReadObject(ms);


            Console.WriteLine("getResponse 5 : {0}", getResponse);

        }

    }

    // Root myDeserializedClass = JsonConvert.DeserializeObject<Root>(myJsonResponse);
    [DataContract]
    public class Variant
    {
        [DataMember]
        public object id { get; set; }
        [DataMember]
        public object product_id { get; set; }
        [DataMember]
        public string title { get; set; }
        [DataMember]
        public string price { get; set; }
        [DataMember]
        public string sku { get; set; }
        [DataMember]
        public int position { get; set; }
        [DataMember]
        public string inventory_policy { get; set; }
        [DataMember]
        public string compare_at_price { get; set; }
        [DataMember]
        public string fulfillment_service { get; set; }
        [DataMember]
        public string inventory_management { get; set; }
        [DataMember]
        public string option1 { get; set; }
        [DataMember]
        public string option2 { get; set; }
        [DataMember]
        public object option3 { get; set; }
    /*    [DataMember]
        public DateTime created_at { get; set; }
        [DataMember]
        public DateTime updated_at { get; set; }  */
        [DataMember]
        public bool taxable { get; set; }
        [DataMember]
```

```
        public string barcode { get; set; }
        [DataMember]
        public int grams { get; set; }
        [DataMember]
        public object image_id { get; set; }
        [DataMember]
        public double weight { get; set; }
        [DataMember]
        public string weight_unit { get; set; }
        [DataMember]
        public object inventory_item_id { get; set; }
        [DataMember]
        public int inventory_quantity { get; set; }
        [DataMember]
        public int old_inventory_quantity { get; set; }
        [DataMember]
        public bool requires_shipping { get; set; }
        [DataMember]
        public string admin_graphql_api_id { get; set; }
    }

    [DataContract]
    public class Option
    {
        [DataMember]
        public object id { get; set; }
        [DataMember]
        public object product_id { get; set; }
        [DataMember]
        public string name { get; set; }
        [DataMember]
        public int position { get; set; }
        [DataMember]
        public List<string> values { get; set; }
    }
    [DataContract]
    public class Image
    {
        [DataMember]
        public long id { get; set; }
        [DataMember]
        public long product_id { get; set; }
        [DataMember]
        public int position { get; set; }
    /*      [DataMember]
        public DateTime created_at { get; set; }
        [DataMember]
        public DateTime updated_at { get; set; } */
        [DataMember]
        public object alt { get; set; }
        [DataMember]
        public int width { get; set; }
        [DataMember]
        public int height { get; set; }
        [DataMember]
        public string src { get; set; }
        [DataMember]
        public List<object> variant_ids { get; set; }
```

```
        [DataMember]
        public string admin_graphql_api_id { get; set; }
    }

    [DataContract]
    public class Image2
    {
        [DataMember]
        public long id { get; set; }
        [DataMember]
        public long product_id { get; set; }
        [DataMember]
        public int position { get; set; }
/*      [DataMember]
        public DateTime created_at { get; set; }
        [DataMember]
        public DateTime updated_at { get; set; } */
        [DataMember]
        public object alt { get; set; }
        [DataMember]
        public int width { get; set; }
        [DataMember]
        public int height { get; set; }
        [DataMember]
        public string src { get; set; }
        [DataMember]
        public List<object> variant_ids { get; set; }
        [DataMember]
        public string admin_graphql_api_id { get; set; }
    }

    [DataContract]
    public class Product
    {
        [DataMember]
        public long id { get; set; }
        [DataMember]
        public string title { get; set; }
        [DataMember]
        public string body_html { get; set; }
        [DataMember]
        public string vendor { get; set; }
        [DataMember]
        public string product_type { get; set; }
      /*  [DataMember]
        public DateTime created_at { get; set; }*/
        [DataMember]
        public string handle { get; set; }
        /*  [DataMember]
         public DateTime updated_at { get; set; }
        [DataMember]
         public object published_at { get; set; }*/
        [DataMember]
        public string template_suffix { get; set; }
        [DataMember]
        public string status { get; set; }
        [DataMember]
        public string published_scope { get; set; }
```

```
        [DataMember]
        public string tags { get; set; }
        [DataMember]
        public string admin_graphql_api_id { get; set; }
        [DataMember]
        public List<Variant> variants { get; set; }
        [DataMember]
        public List<Option> options { get; set; }
        [DataMember]
        public List<Image> images { get; set; }
        [DataMember]
        public Image image { get; set; }
    }

    [DataContract]
    public class Root
    {
        [DataMember]
        public List<Product> products { get; set; }
    }


}
```

Output:

Date 12-1-2021

Animation

Create a simple animation with button size.

Code:

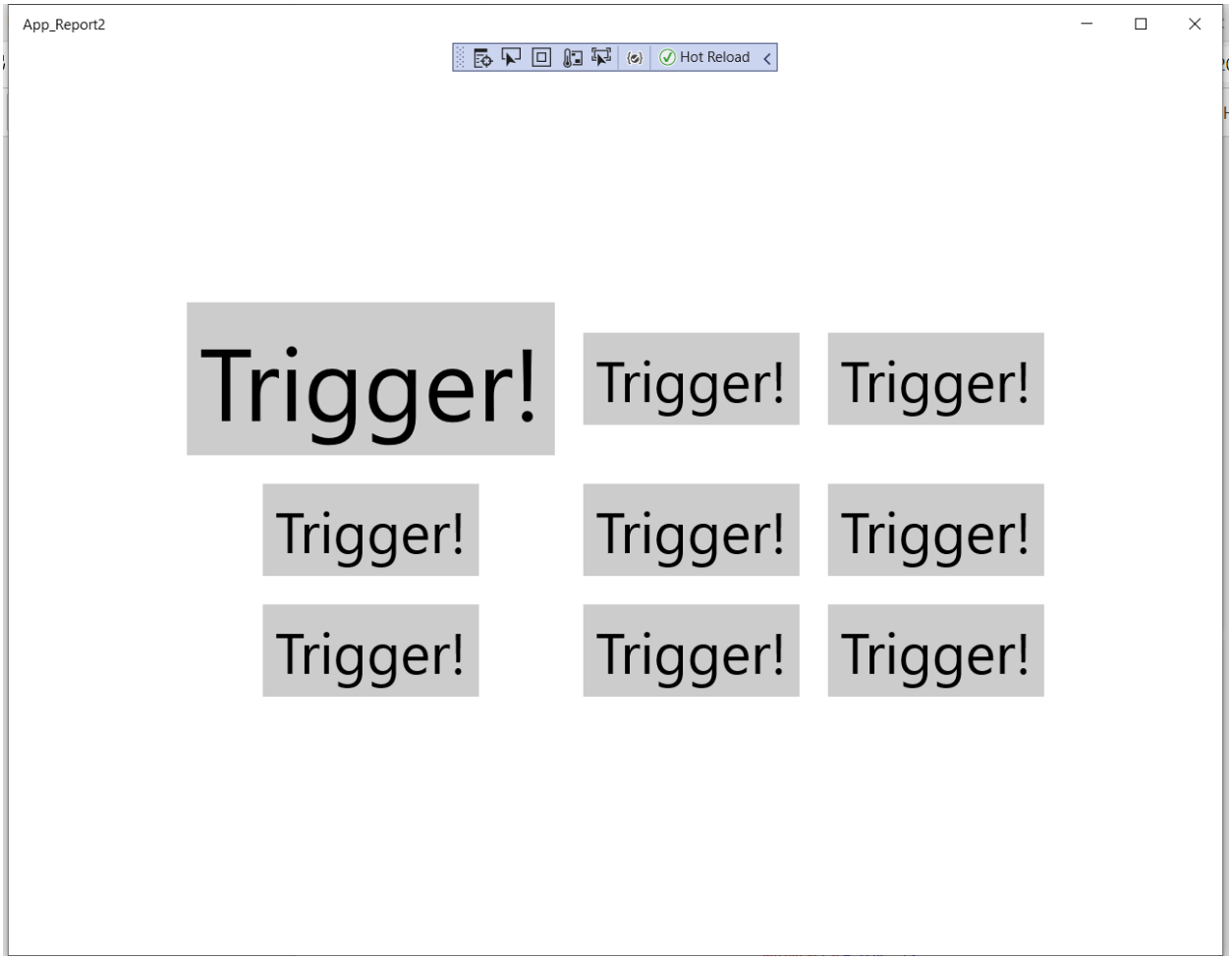
Github Link-Full-semester-external-Project-Work-Automated-Sofware/Animation at main · rinkal651/Full-semester-external-Project-Work-Automated-Sofware (github.com)

MainWindow.xaml

```
<Page
    x:Class="App_Report2.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:App_Report2"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
    <Page.Resources>
        <Style TargetType="Button">
            <Setter Property="Content" Value="Trigger!" />
            <Setter Property="FontSize" Value="48" />
            <Setter Property="HorizontalAlignment" Value="Center" />
            <Setter Property="VerticalAlignment" Value="Center" />
            <Setter Property="Margin" Value="12" />
        </Style>
    </Page.Resources>
    <Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
        <Grid HorizontalAlignment="Center"
 VerticalAlignment="Center">
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto" />
                <RowDefinition Height="Auto" />
                <RowDefinition Height="Auto" />
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="Auto" />
                <ColumnDefinition Width="Auto" />
                <ColumnDefinition Width="Auto" />
            </Grid.ColumnDefinitions>
            <Button Grid.Row="0" Grid.Column="0" Click="OnButtonClick" />
            <Button Grid.Row="0" Grid.Column="1" Click="OnButtonClick" />
            <Button Grid.Row="0" Grid.Column="2" Click="OnButtonClick" />
            <Button Grid.Row="1" Grid.Column="0" Click="OnButtonClick" />
            <Button Grid.Row="1" Grid.Column="1" Click="OnButtonClick" />
            <Button Grid.Row="1" Grid.Column="2" Click="OnButtonClick" />
            <Button Grid.Row="2" Grid.Column="0" Click="OnButtonClick" />
            <Button Grid.Row="2" Grid.Column="1" Click="OnButtonClick" />
            <Button Grid.Row="2" Grid.Column="2" Click="OnButtonClick" />
        </Grid>
    </Grid>
</Page>
```

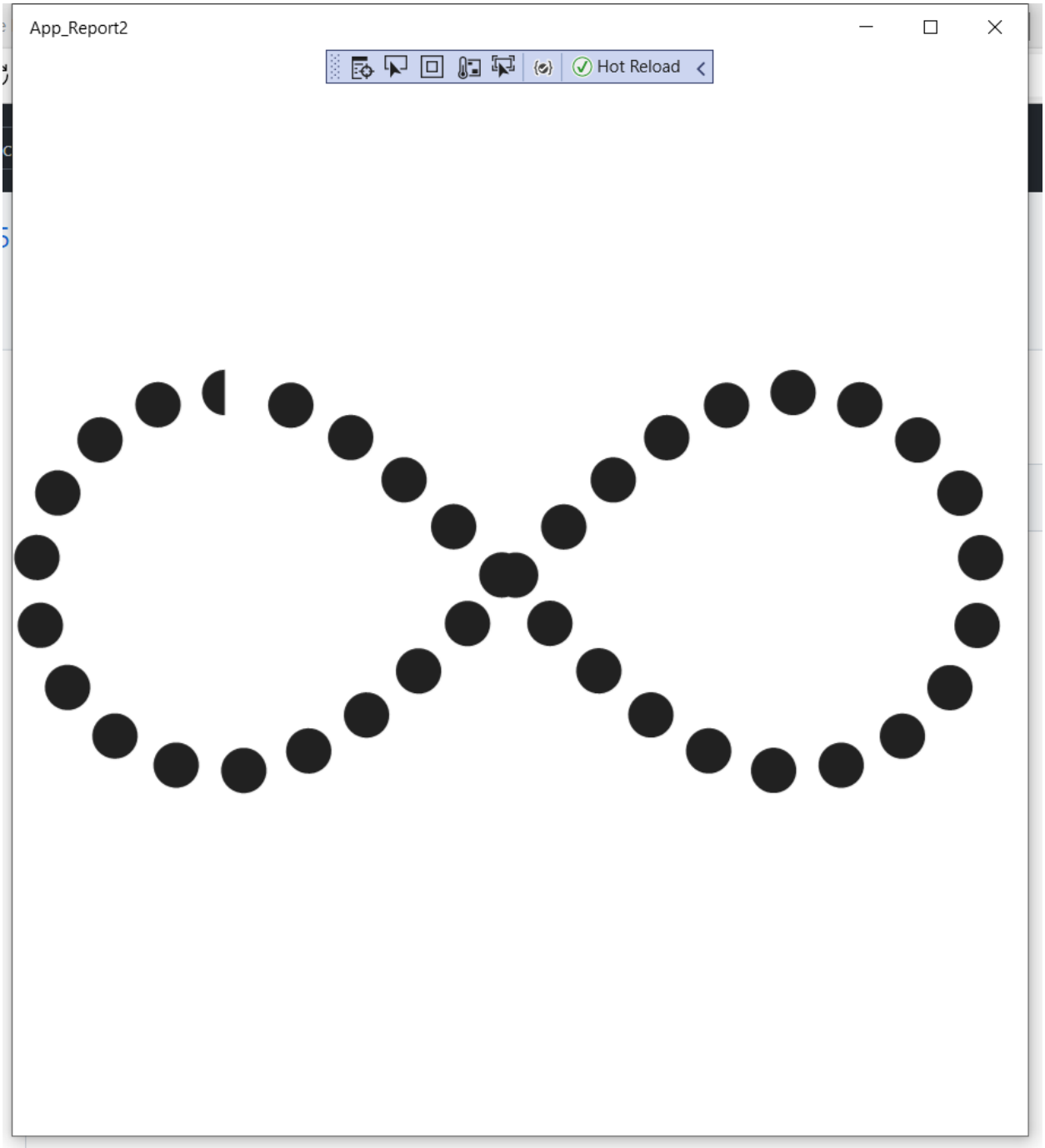
MainWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Media.Animation;
using Windows.UI.Xaml.Navigation;

// The Blank Page item template is documented at
https://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x409

namespace App_Report2
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a Frame.
    /// </summary>
    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();
        }
        void OnButtonClick(object sender, RoutedEventArgs args)
        {
            DoubleAnimation anima = new DoubleAnimation
            {
                EnableDependentAnimation = true,
                To = 96,
                Duration = new Duration(new TimeSpan(0, 0, 1)),
                AutoReverse = true,
                RepeatBehavior = new RepeatBehavior(3)
            };
            Storyboard.SetTarget(anima, sender as Button);
            Storyboard.SetTargetProperty(anima, "FontSize");
            Storyboard storyboard = new Storyboard();
            storyboard.Children.Add(anima);
            storyboard.Begin();
        }
    }
}
```


App_Report2
Hot Reload
Trigger!
Trigger!
Trigger!
Trigger!
Trigger!
Trigger!
Trigger!
Trigger!
Trigger!

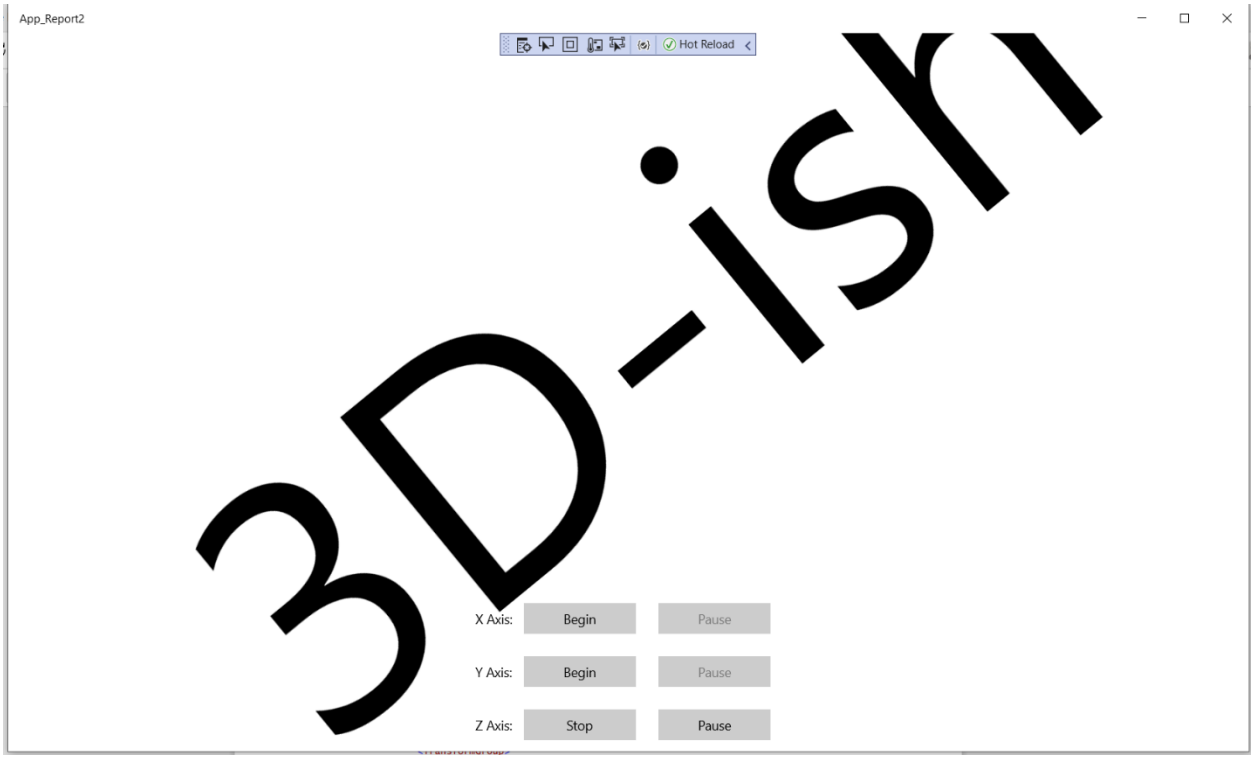
Animated Dash Offset
Create a simple design pattern with animated dash offset.

Code:
Github Link-
[Full-semester-external-Project-Work-Automated-Sofware/AnimateDashOffset at main · rinkal651/Full-semester-external-Project-Work-Automated-Sofware (github.com)](#)



3d Rotate
Create a 3D rotation animation with start and stop button.

Code:
Github Link-
[Full-semester-external-Project-Work-Automated-Sofware/3dRotate at main · rinkal651/Full-semester-external-Project-Work-Automated-Sofware (github.com)](https://github.com/rinkal651/Full-semester-external-Project-Work-Automated-Sofware/tree/main/3dRotate)



Date 13-1-2021
Speedometer Progressbar
Vary the speedometer speed based on progressbar.

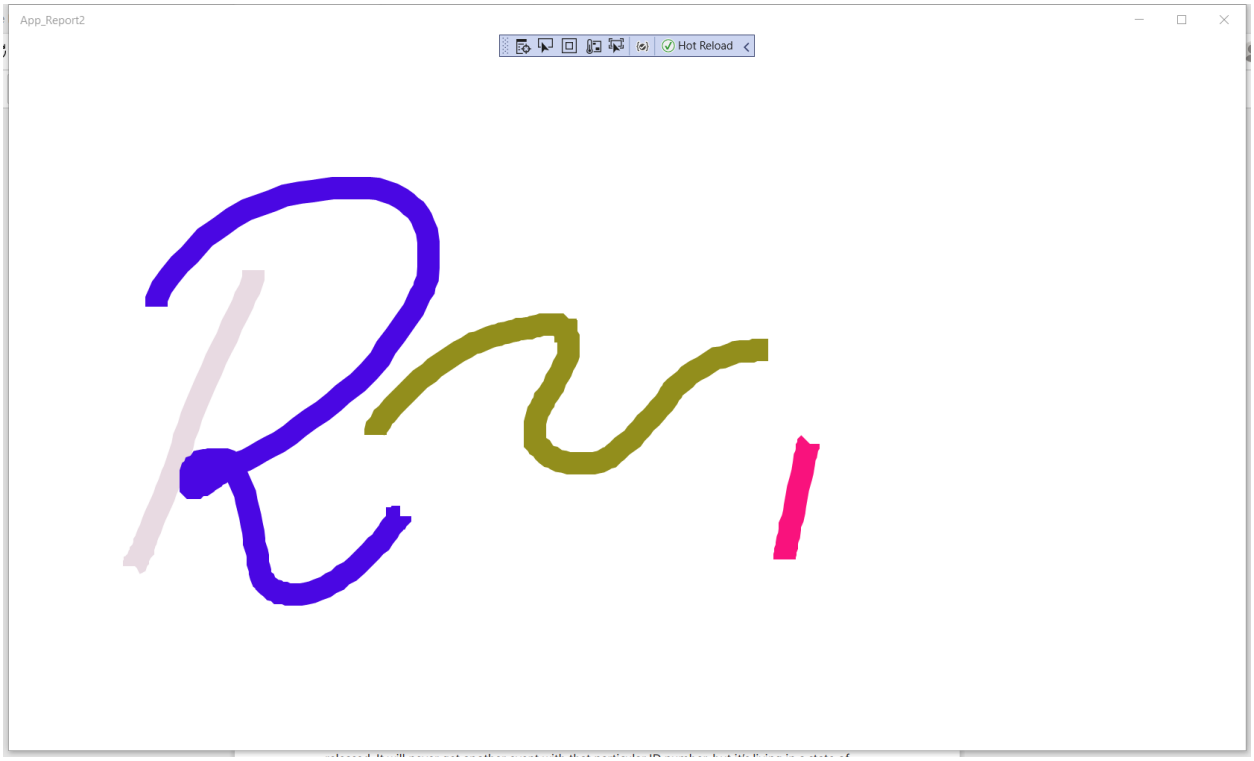
Code:

Github Link-[Full-semester-external-Project-Work-Automated-Sofware/SpeedometerProgressbar at main · rinkal651/Full-semester-external-Project-Work-Automated-Sofware (github.com)]

Paint Tool
Create a simple paint tool by which User can draw with mouse.

Code:

Github Link-[Full-semester-external-Project-Work-Automated-Sofware/Paint at main · rinkal651/Full-semester-external-Project-Work-Automated-Sofware (github.com)]




App_Report2
Hot Reload
PAINT

Date:17-1-2021

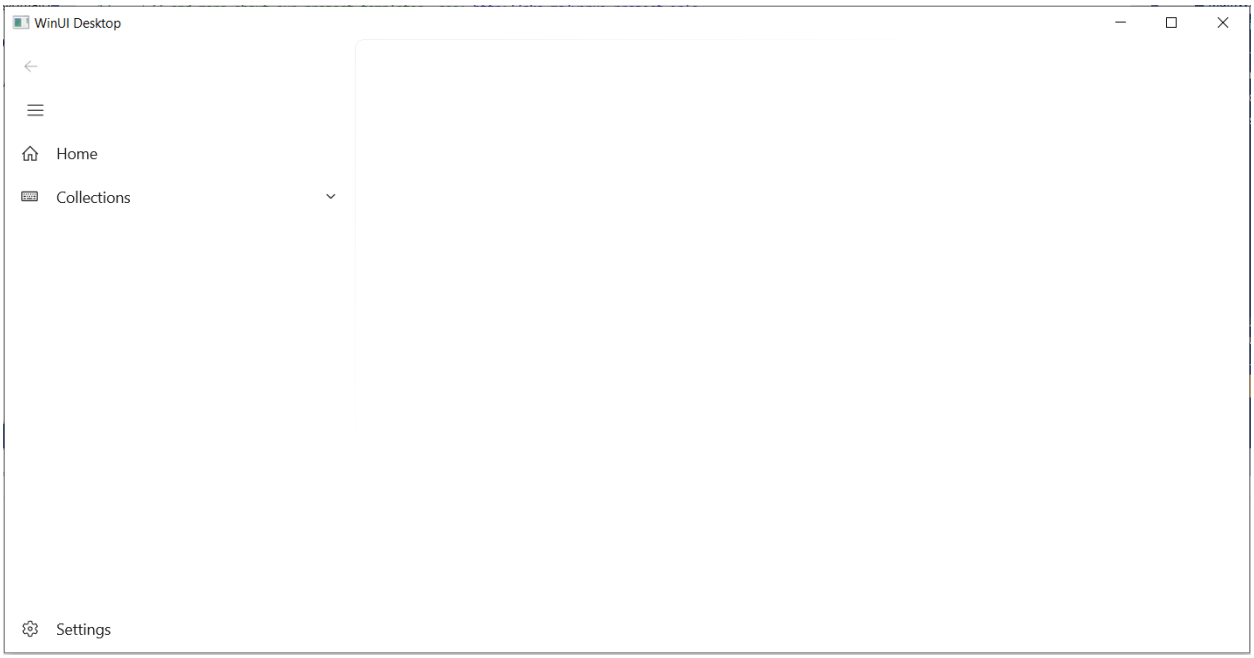
Working with Navigation

Create navigation with navigation control.

Code:

Github Link-Full-semester-external-Project-Work-Automated-Sofware/NavigationWithMarkupItems at main · rinkal651/Full-semester-external-Project-Work-Automated-Sofware (github.com)

Output-



Actions On navigation with binding

Performing expanding and collapsed method on Navigation control.

Code-

Github Link-Full-semester-external-Project-Work-Automated-Sofware/ActionsOnNavigationWithBinding at main · rinkal651/Full-semester-external-Project-Work-Automated-Sofware (github.com)

Output-