Lesson 9: Locations and Maps

Table of Contents

Using the Geolocation API

Estimated current latitude and longitude data is provided by Windows 8 geolocation API (Application Programming Interface). The location data is derived from the most accurate hardware and software capabilities present. This might be Wi-Fi triangulation, IP address information, or a Global Position System (GPS) chip if available.

Latitude and longitude measurements provide a geographic coordinate system by which any location on Earth can be specified. Latitude provides the surface position on a north-south basis, while longitude provides the surface position on an east-west basis. The equator provides the zero reference mark for latitude. Positive values are north of the equator and negative values are south. The line that passes vertically through the Royal Observatory in Greenwich, England—called the Prime Meridian—was established as the zero degree mark for longitude. The longitude is measured as the angle east or west from the Prime Meridian, ranging from a +180° eastward and a −180° westward.

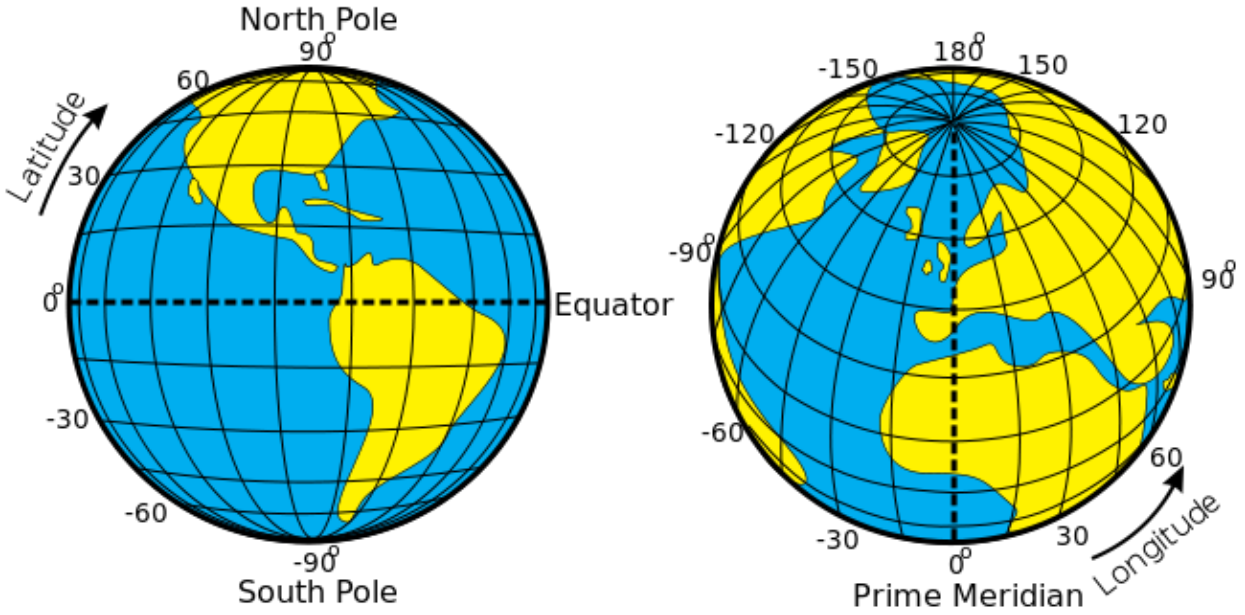

Figure 1 *– Geoposition on the surface of the earth is established by latitude and longitude measurements. Latitude is 90 to -90 degrees from the North Pole to the South Pole with the equator being zero. Longitude is measured up to a positive 180 degrees eastward from the Prime Meridian (which runs through the Royal Observatory in Greenwich, England) to a negative 180 degrees westward.*

Your first project uses the device's geolocation API to display the current latitude and longitude, along with the accuracy of the geoposition calculation method being used by the device. It then issues a URL call to Google Maps to display the mapped location in a WebView control.

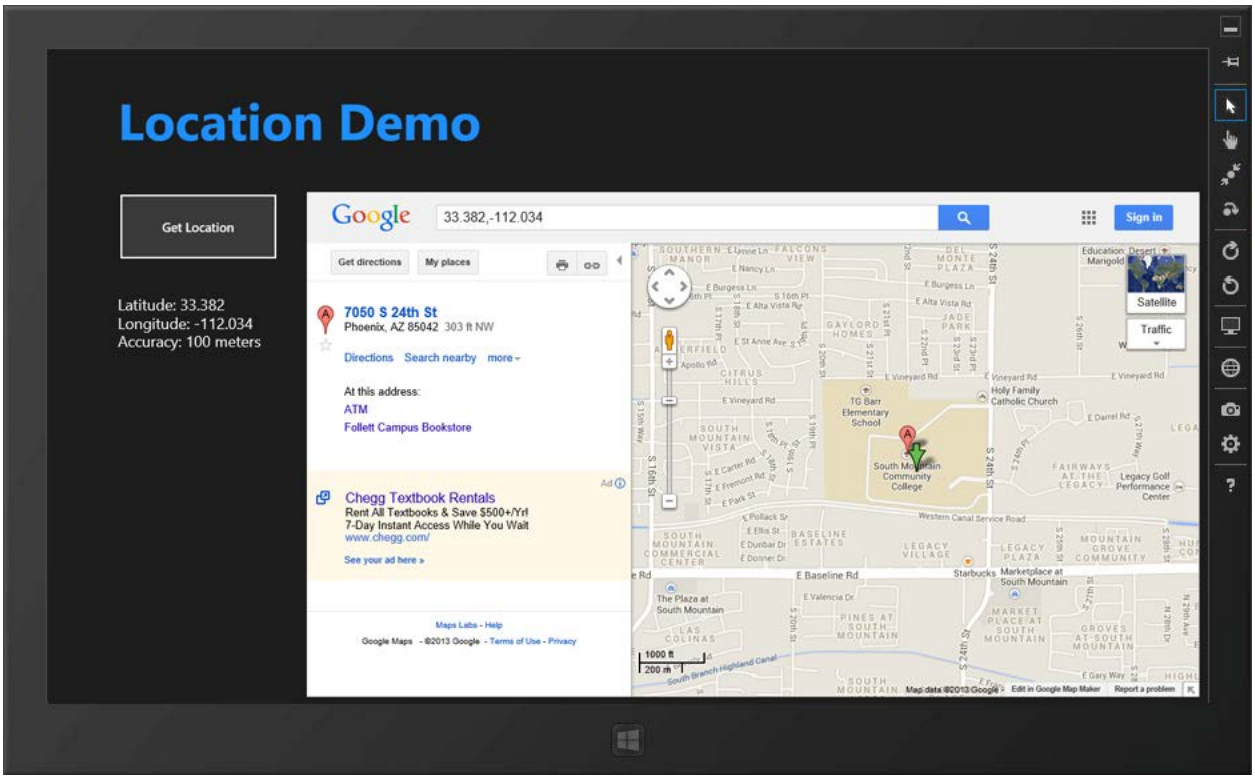

Figure 2 *– The Location Demo displays the current latitude, longitude, and accuracy of the location calculation. The coordinates are mapped in a WebView control with a call to Google Maps.*

To use the geolocation API, complete the following steps:

Step 1: Create a new project using the Blank Page template.

Step 2: In the *package.appxmanifest*, establish approval for the use of the geolocation API by checking the Locations box in the Capabilities tab.

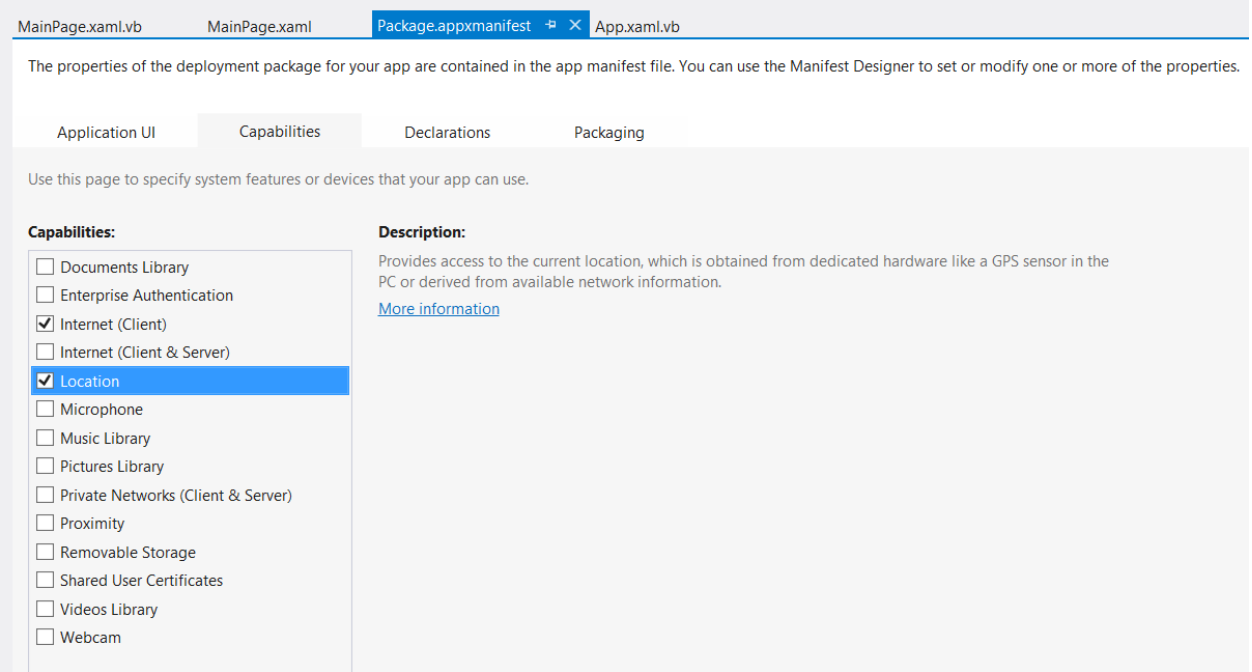

Figure 3 *– To use the geolocation API, the Location services must be enabled in the package.appxmanifest capabilities.*

Step 3: Create the user interface by adding a TextBlock title; a Button; a TextBlock to display the latitude, longitude, and accuracy; and WebView to display a Google Maps web page. Name the display TextBlock "txtLocationInfo" and name the WebView control "wvMap", as you will refer to these in the code-behind for MainPage.xaml.cs or MainPage.xaml.vb. Set the button up to respond to a Click event.

XAML Code for MainPage.xaml

```
<Page
    x:Class="_09A_Location_Demo_CS.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:_09A_Location_Demo_CS"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">

    <Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
        <TextBlock HorizontalAlignment="Left" Height="86" Margin="85,50,0,0"
              TextWrapping="Wrap" Text="Location Demo"
              VerticalAlignment="Top" Width="1214" Foreground="DodgerBlue" FontSize="60"
              FontWeight="Bold"/>
        <Button x:Name="btnGetLocation" Content="Get Location" HorizontalAlignment="Left"
              Height="81" Margin="85,165,0,0" VerticalAlignment="Top" Width="189"
              Click="GetLocation" />
        <TextBlock x:Name ="txtLocationInfo" HorizontalAlignment="Left" Height="154"
              Margin="85,288,0,0" TextWrapping="Wrap" Text="Status"
              VerticalAlignment="Top" Width="247" FontFamily="Global User Interface"
              FontSize="18"/>
        <WebView x:Name="wvMap" HorizontalAlignment="Left" Height="593"
              Margin="307,165,0,0" VerticalAlignment="Top" Width="1049"/>
    </Grid>
</Page>
```

Step 4: Create the code-behind for the MainPage.xaml.cs or MainPage.xaml.vb to display the latitude and longitude information as well as the accuracy of the calculation, and to then send a URL request to Google to map the location in the WebView. Be sure to add directives for using Windows.Devices.GeoLocation to access the geolocation API and Windows.UI.Popups to display a MessageDialog if an error occurs.

C# Code for MainPage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
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.Navigation;

using Windows.Devices.Geolocation;
using Windows.UI.Popups;


// The Blank Page item template is documented at http://go.microsoft.com/fwlink/?LinkId=234238

namespace _09A_Location_Demo_CS
{

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

        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
        }

        private async void GetLocation(object sender, RoutedEventArgs e)
        {
            Geolocator myLocation = new Geolocator();
            try
            {
                myLocation.DesiredAccuracy = PositionAccuracy.High;
                Geoposition geoPos = await myLocation.GetGeopositionAsync();
                double lat = geoPos.Coordinate.Latitude;
                double lng = geoPos.Coordinate.Longitude;
                txtLocationInfo.Text = "Latitude: " + lat.ToString() + "\nLongitude: " +
                    lng.ToString() + "\nAccuracy: " + geoPos.Coordinate.Accuracy + " meters";
                string mapUri =  "https://maps.google.com/maps?q=" +
                        lat.ToString() + "," + lng.ToString() + "&z=15";
                // &z=15 is the zoom level (higher the number the greater the zoom)
                wvMap.Source = new Uri(mapUri);
            }
            catch (Exception ex)
            {
```

```
                ShowMessage(ex.Message, "Error");
            }
        }

        private async void ShowMessage(string prompt, string title)
        {
            MessageDialog md = new MessageDialog(prompt, title);
            await md.ShowAsync();
        }
    }
}
```

VB Code for MainPage.xaml.vb

```
Imports Windows.Devices.Geolocation
Imports Windows.Devices.Geolocation.Geoposition
Imports Windows.UI.Popups


Public NotInheritable Class MainPage
    Inherits Page


    Protected Overrides Sub OnNavigatedTo(e As Navigation.NavigationEventArgs)

    End Sub

    Private Async Sub GetLocation(sender As Object, e As RoutedEventArgs) _
                                                    Handles btnGetLocation.Click
        Dim myLocation As Geolocator = New Geolocator()
        Try
            myLocation.DesiredAccuracy = PositionAccuracy.High
            Dim geoPos As Geoposition = Await myLocation.GetGeopositionAsync()
            Dim lat As Double = geoPos.Coordinate.Latitude
            Dim lng As Double = geoPos.Coordinate.Longitude
            txtLocationInfo.Text = "Latitude: " & lat.ToString() & _
                vbCrLf & "Longitude: " & lng.ToString() & _
                vbCrLf & "Accuracy: " & geoPos.Coordinate.Accuracy & " meters"
                Dim mapUri As String = "https://maps.google.com/maps?q=" & _
                        lat.ToString() & "," & lng.ToString() & "&z=15"
               ' &z=15 is the zoom level (higher the number the greater the zoom)
            wvMap.Source = New Uri(mapUri)
        Catch ex As Exception
            ShowMessage(ex.Message, "Error")
        End Try
    End Sub

    Private Async Sub ShowMessage(ByVal prompt As String, ByVal title As String)
        Dim md As MessageDialog = New MessageDialog(prompt, title)
        Await md.ShowAsync()
    End Sub
End Class
```

Step 5: Test and debug the app.

Using Bing Maps API to Display a Location

Microsoft offers a Bing Maps API for C# and VB developers. A Bing Maps account is necessary to obtain a usage key. The cost of the license depends on the project. Review the "Bing Maps Platform" on Microsoft's Bing website for information and to get started. A 90-day trial is available as well as an educational license. At the time of this writing, licenses for Windows Store apps were also free.

To use Bing Maps API to display a location, complete the following steps:

Step 1: Create a Bing Maps developer account and obtain a key for the project. The key will be a long alphanumeric string such as: "Anz2Wc8C6cTNYy0-bm1ussbKLYf-yhQj1uzzXproUTLHnyUblyfxgLGHqzi8L_py8".

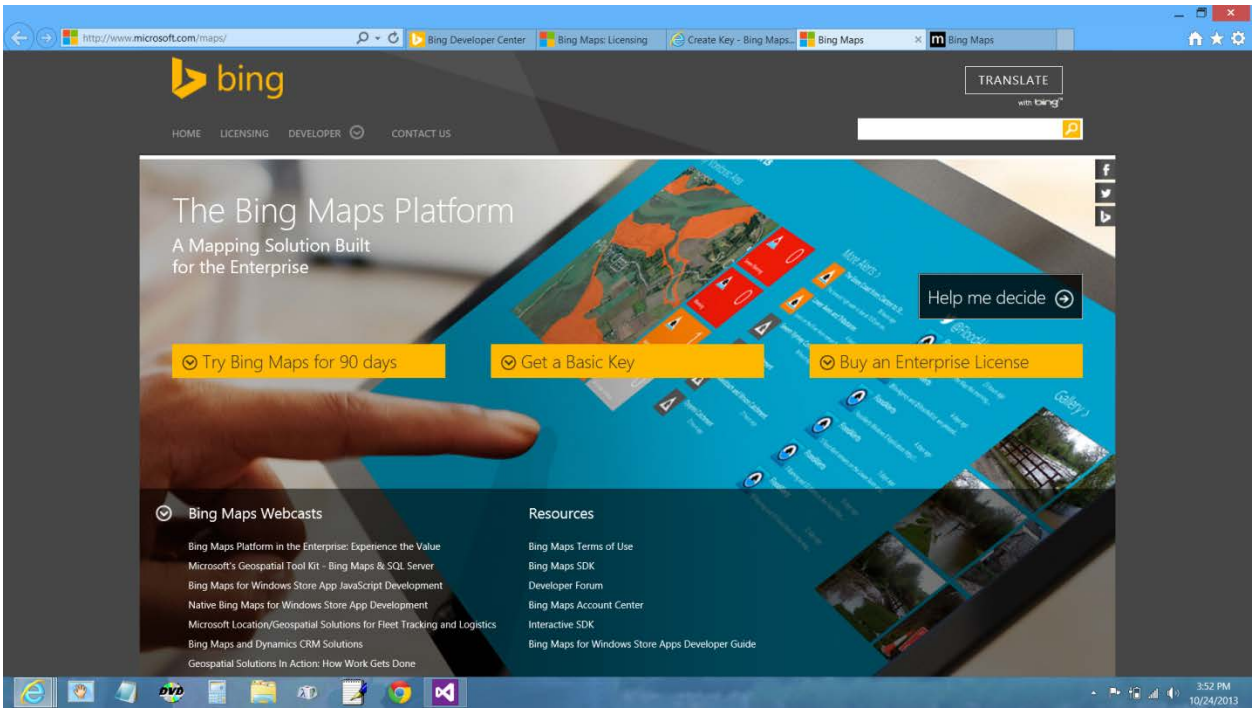

Figure 4 *– A license and a key to access the Bing Maps platform site is available at the Bing website.*

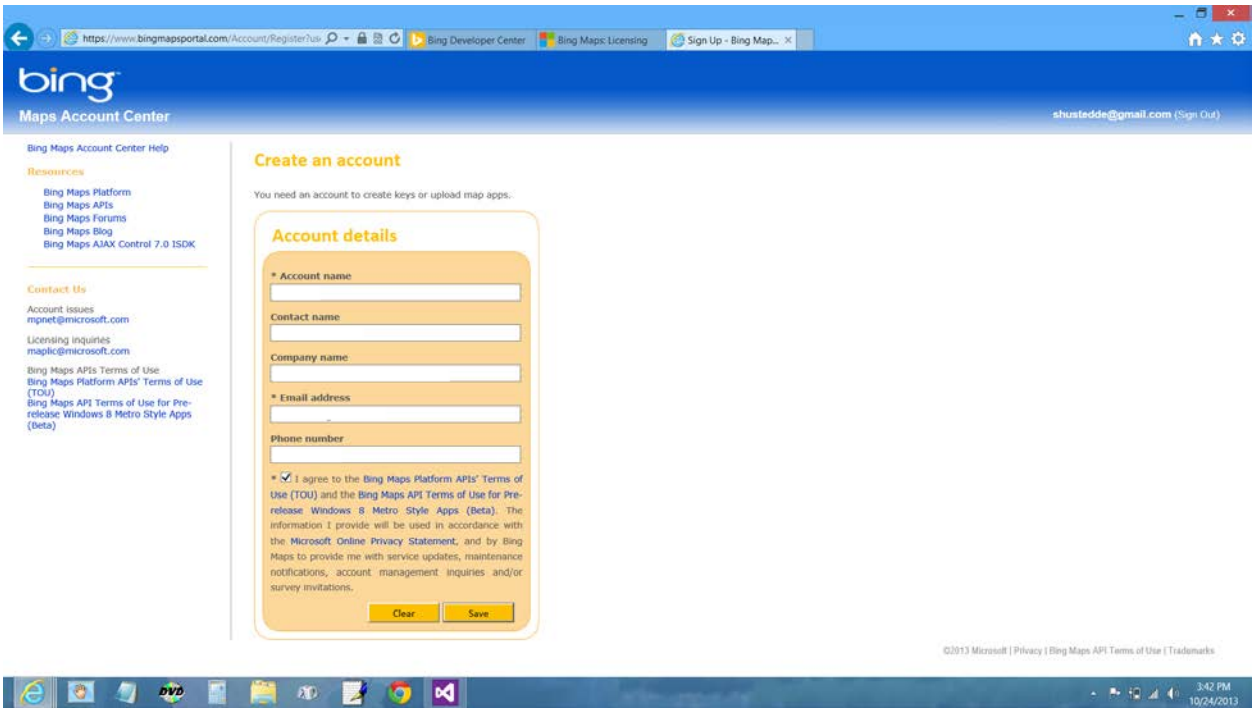

Figure 5 *– The first step is to create an account. A unique account name must be provided along with an e-mail address and other contact information.*

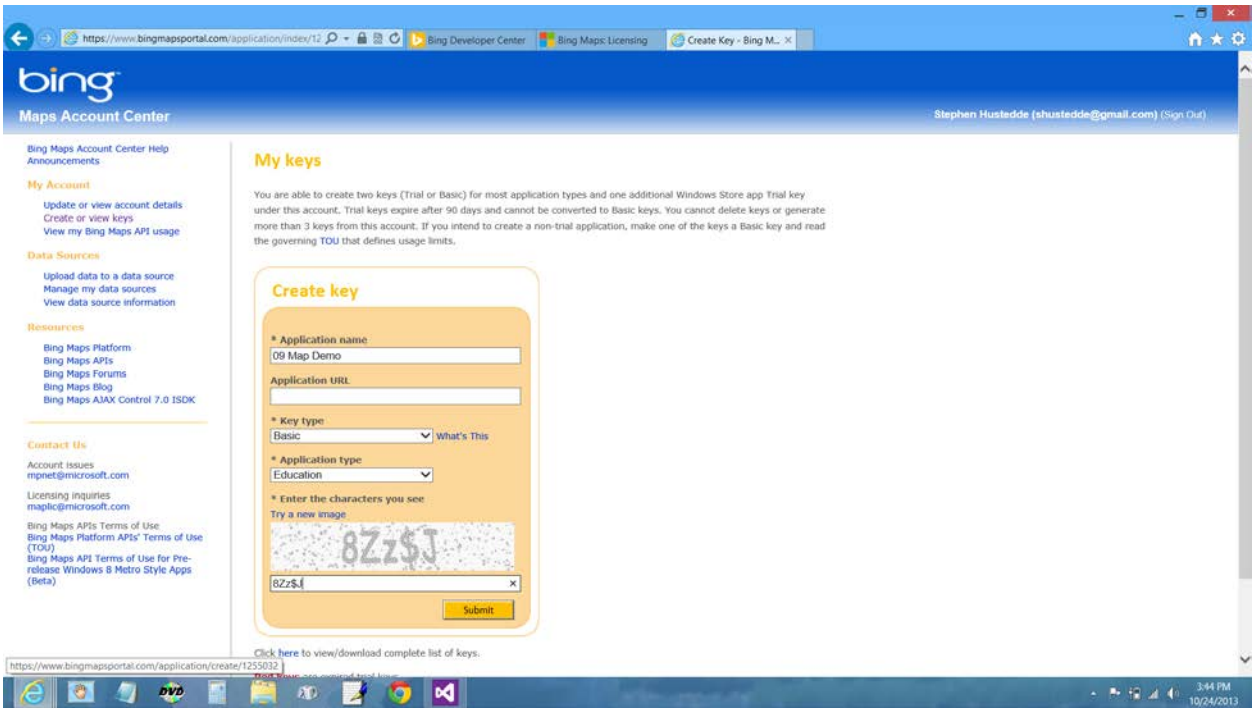

Figure 6 *– A key is then created for the project. Copy and paste the key into your project's code as a comment in either the XAML code or the C#/VB code-behind for the MainPage. It will be needed in the XAML code as the Credentials property for the Map control.*

Step 2: Download and install the Bing Maps SDK for your OS and Visual Studio version.

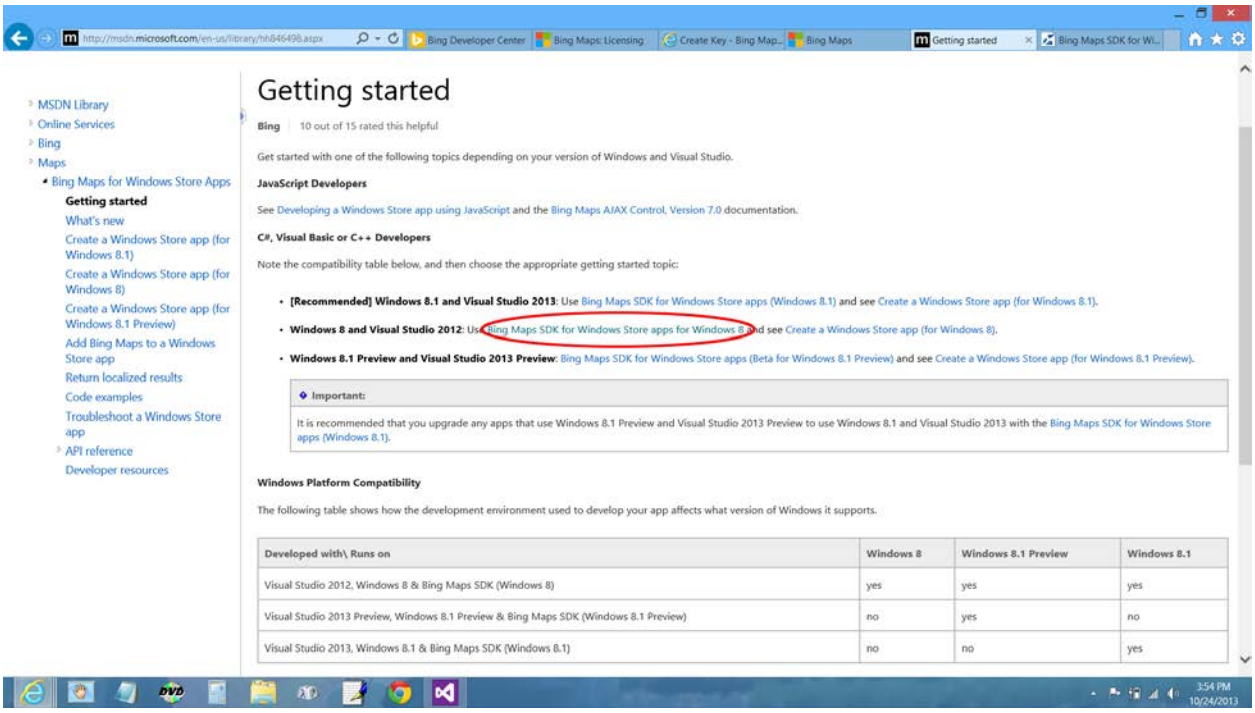

Figure 7 *– The Getting Started page provides the link to the appropriate SDK. Click the link for your operating system and version of Visual Studio.*

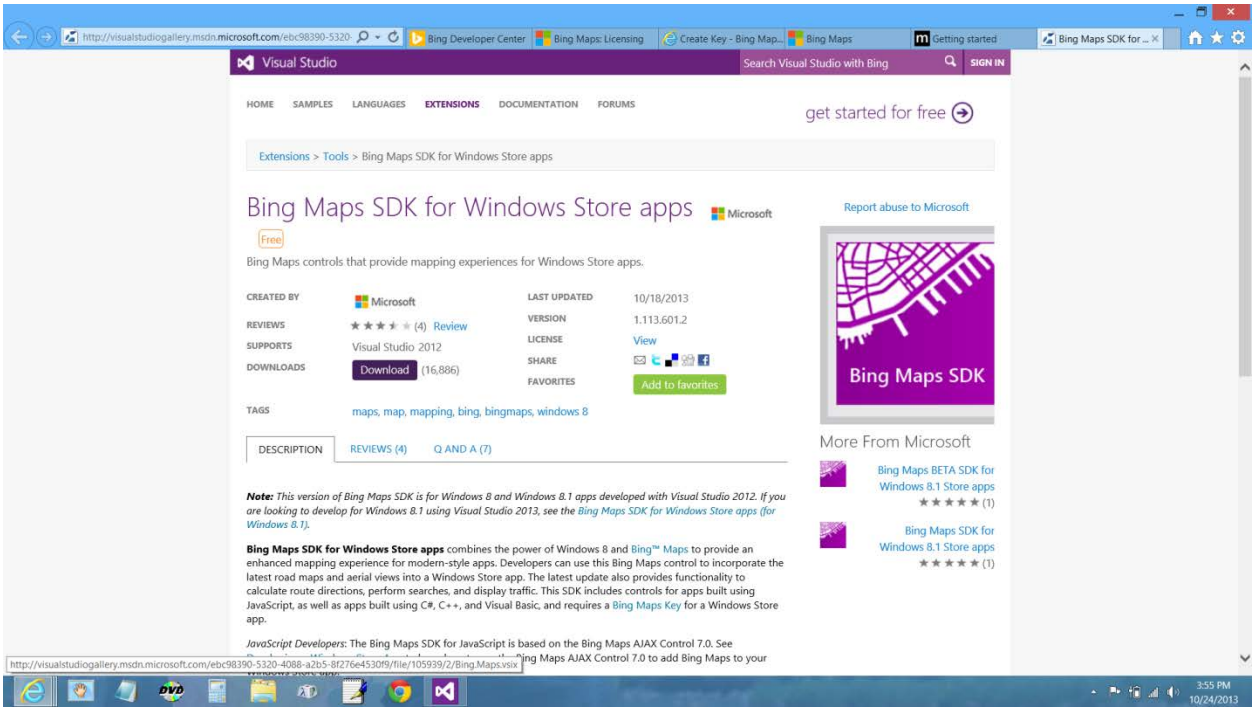

Figure 8 *– The Bing Maps SDK for Windows Store Apps must be downloaded and installed to the developer computer.*

Step 3: Create a new project using the Blank Page template.

Step 4: Access the Configuration Manager from the Build menu in Visual Studio. Choose Debug from the Configuration and choose a Platform appropriate to your testing device.

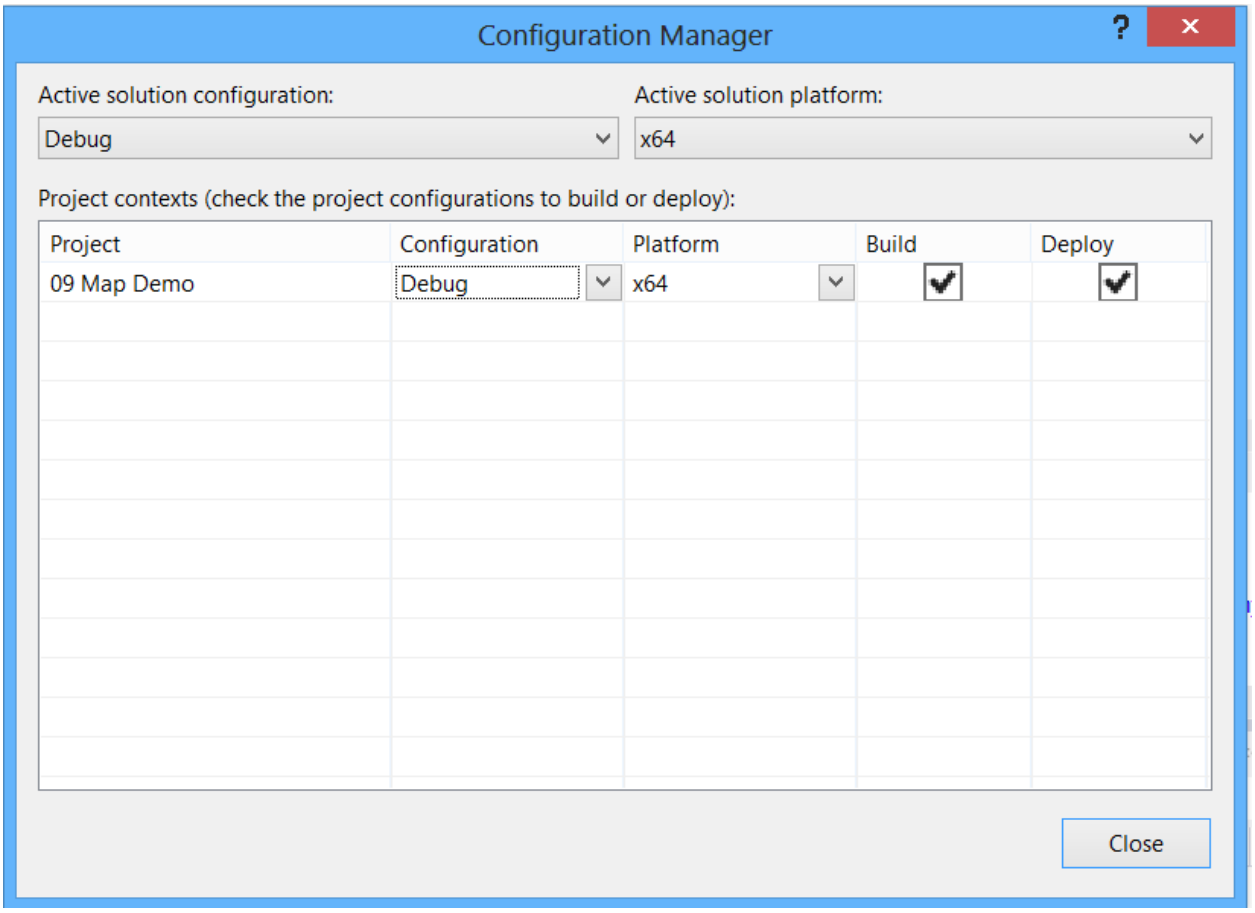

Figure 9 *– Choose a configuration from the Build menu > Configuration Manager.*

Step 5: Add a reference to the Bing Maps API by choosing Add Reference from the Project menu. In the resulting dialog, expand the Windows area on the left and select Extensions. Check the boxes for “Bing Maps for C#, C++, or Visual Basic” and “Microsoft Visual C++ Runtime Package”, then click the “Okay” button.

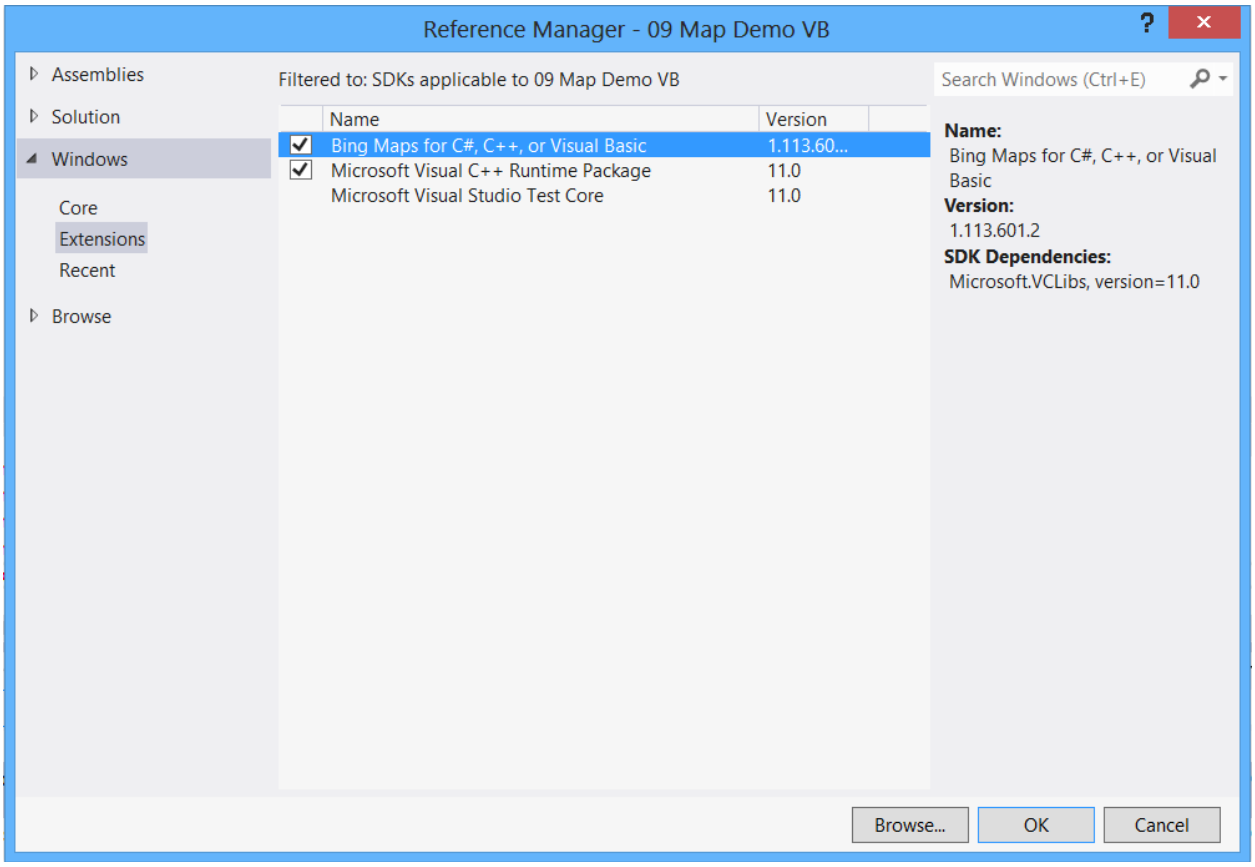

Figure 10 *– A reference to the Bing Maps API is added via the Reference Manager. (Choose "Add Reference…" from the Project menu in Visual Studio.)*

Step 6: Establish the Locations capability in Package.appxmanifest.

Step 7: Build the interface in the MainPage.xaml file. Add a directive for the Bing.Maps in the <Page> header of the XAML code with a namespace such as "bingmap". In the Grid, add a Map control using the namespace (shown in the example as <bingmap:Map>). Paste your Bing Maps key as the value for the Map's Credentials property, and provide a name for the Map control so it can be referenced in the code-behind.

TIP: One thing to be aware of is that the Design panel of the XAML interface builder cannot visually display the interface when the platform is set to x64 or ARM in the Configuration Manager. You may wish to return the setting to "any platform" while you build the interface. You will also need to temporarily comment out the Map control, because it cannot be displayed and the resulting error restricts the entire display in the Designer panel. Once everything else is built, uncomment the Map control and reset the Configuration Manager to use the x64 platform.

```
<Page
    x:Class="_09_Map_Demo_VB.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:_09_Map_Demo_VB"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:bingmap="using:Bing.Maps"
```

```
    mc:Ignorable="d">

    <Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
        <bingmap:Map
              Credentials="Anz2Wc8C6cTNYy0-bm1ussbKLYf-yhQjIuzzXproUTLHnyUblyfxgLGHqzi8L_py8"
              x:Name="myMap"></bingmap:Map>
    </Grid>
</Page>
```

Step 8: Code the C# or VB code-behind to map the current location. A directive for the Bing.Maps API must also be added in addition to directives added in the previous project, which were to use the Geolocation API and the Windows.UI.Popups to show a MessageDialog.

When the page intilizes (OnNavigatedTo procedure), it calls a method named ShowCurentLocationOnMap. In that method, a GeoLocator object is created with the highest degree of accuracy possible on the device. The GeoPositionAsync() method is called to get the current location. The latitude and longitude of the current location are then used as parameters in a call to Location( ) method of the Bing.Maps API, which is used to establish the center of the myMap Map control. The zoom level is also established for the Map control (from a range of 1 to 20—with 1 being the lowest zoom and 20 being the highest zoom), and a pushpin marker is displayed with a label of “A”.

C# Code for MainPage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
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.Navigation;

using Windows.Devices.Geolocation;
using Windows.UI.Popups;
using Bing.Maps;

namespace _09_Map_Demo
{
    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();
        }

        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            ShowCurrentLocationOnMap();
        }

        private async void ShowCurrentLocationOnMap()
        {
```

```
            try
            {
                //NOTE: Set Location Services in package.appxmanifest
                Geolocator myLocation = new Geolocator();
                myLocation.DesiredAccuracy = PositionAccuracy.High;
                Geoposition geoPos = await myLocation.GetGeopositionAsync();
                double lat = geoPos.Coordinate.Latitude;
                double lng = geoPos.Coordinate.Longitude;
                //Display map data
                myMap.Center = new Bing.Maps.Location(lat, lng);
                myMap.ZoomLevel = 16;
                myMap.MapType = MapType.Aerial;
                Pushpin pushpin = new Pushpin();
                pushpin.Text = "A";
                MapLayer.SetPosition(pushpin, new Location(lat, lng));
                myMap.Children.Add(pushpin);
            }
            catch (Exception ex)
            {
                ;
            }
        }

        private async void ShowMessage(string prompt, string title)
        {
            MessageDialog md = new MessageDialog(prompt, title);
            await md.ShowAsync();
        }
    }
}
```

VB Code for MainPage.xaml.vb

```
Imports Windows.Devices.Geolocation
Imports Windows.UI.Popups
Imports Bing.Maps


Public NotInheritable Class MainPage
    Inherits Page

    Protected Overrides Sub OnNavigatedTo(e As Navigation.NavigationEventArgs)
        MapCurrentLocation()
    End Sub

    Private Async Sub MapCurrentLocation()
        'NOTE: Set Location Service sin package.appxmanifest
        Dim myLocation As Geolocator = New Geolocator()
        myLocation.DesiredAccuracy = PositionAccuracy.High
        Dim geoPos As Geoposition = Await myLocation.GetGeopositionAsync()
        Dim lat As Double = geoPos.Coordinate.Latitude
        Dim lng As Double = geoPos.Coordinate.Longitude

        'Display the data on the map control
        myMap.Center = New Bing.Maps.Location(lat, lng)
        myMap.ZoomLevel = 16
        myMap.MapType = MapType.Aerial
        Dim pushpin As Pushpin = New Pushpin()
        pushpin.Text = "A"
        MapLayer.SetPosition(pushpin, New Location(lat, lng))
        myMap.Children.Add(pushpin)
```

```
    End Sub

    Private Async Sub ShowMessage(ByVal prompt As String, ByVal title As String)
        Dim md As MessageDialog = New MessageDialog(prompt, title)
        Await md.ShowAsync()
    End Sub

End Class
```

Mapping a Specific Location or Landmark

The GeocodeRequestOptions object of the Bing.Maps.Search API can be used to determine the longitude and latitude of a street address or place name in conjunction with a SearchManager Object. The GeocodeAsync method of the SearchManager is deployed with the GeocodeRequestOptions object as its lone parameter.

C# Code snippet

```
Bing.Maps.Search.GeocodeRequestOptions requestOptions = new
       Bing.Maps.Search.GeocodeRequestOptions("7050 S. 24th Street, Phoenix, AZ, 85029");
//Bing.Maps.Search.GeocodeRequestOptions requestOptions = new
       Bing.Maps.Search.GeocodeRequestOptions("Arizona State Capitol");
Bing.Maps.Search.SearchManager searchManager = myMap.SearchManager;
Bing.Maps.Search.LocationDataResponse response = await
searchManager.GeocodeAsync(requestOptions);
```

VB Code snippet

```
Bing.Maps.Search.GeocodeRequestOptions requestOptions = new _
       Bing.Maps.Search.GeocodeRequestOptions("7050 S. 24th Street, Phoenix, AZ, 85029")
‘ Bing.Maps.Search.GeocodeRequestOptions requestOptions = new
       Bing.Maps.Search.GeocodeRequestOptions("Arizona State Capitol")
Bing.Maps.Search.SearchManager searchManager = myMap.SearchManager
Bing.Maps.Search.LocationDataResponse response = await
searchManager.GeocodeAsync(requestOptions)
```

Instead of hard coding an address or location place name, textboxes might be employed to get an address or place name from the user, as demonstrated in the following example.

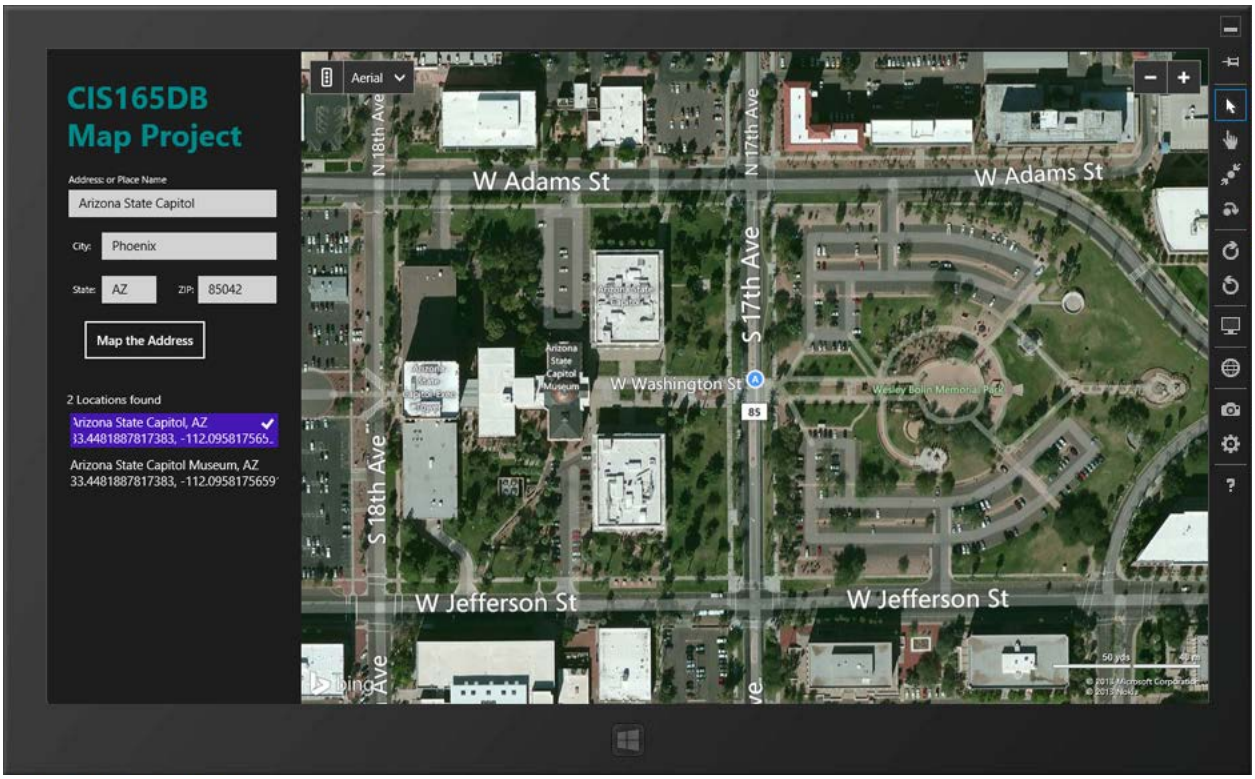

Figure 11 *– Locations to map can be resolved from user entries in TextBoxes. Both full address and landmark place names can be searched. If multiple locations are found, the first returned location is mapped, but addresses for the others are displayed in a ListView. The user can click an item in the ListView to show the map for that chosen location.*

To map a specific location or landmark, complete the following steps:

Step 1: Create a new project using the Blank Page template. This project assumes that you have previously downloaded and installed the Bing Maps SDK and obtained a Bing Map key.

Step 2: Choose a target processor (such as x64) in the Configuration Manager (Build menu). See Figure 9 in the "Using Bing Maps API to Display a Location" section.

Step 3: Add a reference to the Bing Maps API, checking the boxes for "Bing Maps for C#, C++, or Visual Basic" and "Microsoft Visual C++ Runtime Package." See Figure 10 in the "Using Bing Maps API to Display a Location" section.

Step 4: Create the XAML interface. At its foundational level, the interface consists of two columns. The left column contains TextBlock and TextBox controls to display the app's title and get input from the user in terms of an address or landmark to display on the map. The ListView is also used to output multiple locations that are returned as the result of a search. For instance, a search for "450 Camelback Road" in Phoenix, AZ is incomplete and will return both "450 W. Camelback Road" and "450 E. Camelback Road". If you specify just a city, such as "Miami", Bing Maps will map to the center of town. In the case of Miami, it returns not only Miami, Florida, but also Miami, Arizona, and five other locations. Add a namespace directive to the Page's code with a name (such as bingmap) that can be utilized to establish the Map in the right column (Column 1). Event handlers are established for clicking the "Map the Address" button and for a change of selection in the ListView. Use your own unique Bing Map key for the Credentials property of the map control.

XAML Code for Mainpage.xaml

```
<Page
    x:Class="_09D_Searchable_Map.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:_09D_Searchable_Map"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:bingmap="using:Bing.Maps"
    mc:Ignorable="d">



    <Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="300"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>

        <Rectangle Grid.Column="1" Fill="DarkCyan" />
        <TextBlock HorizontalAlignment="Left" Height="96" Margin="25,35,0,0"
               TextWrapping="Wrap" Text="CIS165DB Map Project" VerticalAlignment="Top"
               Width="230" Foreground="DarkCyan" FontSize="36" FontWeight="Bold"/>
        <TextBlock HorizontalAlignment="Left" Height="18" Margin="27,144,0,0"
               TextWrapping="Wrap" Text="Address: or Place Name" VerticalAlignment="Top"
               Width="205"/>
        <TextBox x:Name="txtAddress" HorizontalAlignment="Left" Height="28"
               Margin="27,162,0,0" TextWrapping="Wrap" Text="7050 S. 24th Street"
               VerticalAlignment="Top" Width="244"/>
        <TextBlock HorizontalAlignment="Left" Height="17" Margin="32,222,0,0"
               TextWrapping="Wrap" Text="City:" VerticalAlignment="Top" Width="47"/>
        <TextBox x:Name="txtCity" HorizontalAlignment="Left" Height="28"
               Margin="66,212,0,0" TextWrapping="Wrap" Text="Phoenix"
               VerticalAlignment="Top" Width="205"/>
        <TextBlock HorizontalAlignment="Left" Height="17" Margin="32,273,0,0"
               TextWrapping="Wrap" Text="State:" VerticalAlignment="Top" Width="47"/>
        <TextBox x:Name="txtState" HorizontalAlignment="Left" Height="28"
               Margin="66,263,0,0" TextWrapping="Wrap" Text="AZ" VerticalAlignment="Top"
               Width="3"/>
        <TextBlock HorizontalAlignment="Left" Height="13" Margin="156,273,0,0"
               TextWrapping="Wrap" Text="ZIP:" VerticalAlignment="Top" Width="23"/>
        <TextBox x:Name="txtZip" HorizontalAlignment="Left" Height="28"
               Margin="179,263,0,0" TextWrapping="Wrap" Text="85042"
               VerticalAlignment="Top" Width="92"/>
        <Button Content="Map the Address" HorizontalAlignment="Left" Margin="43,313,0,0"
               VerticalAlignment="Top" Width="147" Height="50" Click="MapAddress"/>
        <TextBlock x:Name="txtReturns" Margin="25,400,17,69" FontSize="14" Text=""/>
        <ListView x:Name="lvOptions" Margin="25,420,17,69" Height="330" Width="270"
               IsItemClickEnabled="False" SelectionChanged="ShowNewAddress"
               ScrollViewer.VerticalScrollBarVisibility="Auto"/>

        <bingmap:Map Grid.Column="1" Credentials="Anz2Wc8C6cTNYy0-bm1ussbKLYf-
               yhQjIuzzXproUTLHnyUblyfxgLGHqzi8L_py8" x:Name="myMap" />
        <!--Use your assigned unique Bing Maps key for the Credentials above-->
    </Grid>
</Page>
```

Step 5: Code the C# or VB code-behind for the XAML page. The MapAddress procedure handles the button click event. It builds a search string from the data entered into the various TextBox controls. Then, it creates a GeocodeRequestOptions object from the Bing.Maps.Search API, using the assembled search string as its specified search parameter. The maximum number of returned results were increased from the default value of five to twenty, the greatest number allowable. A SearchManager object is created from which the GeocodeAsync method can be employed, with the GeocodeRequestOptions object passed as its parameter. The returned result object contains a LocationData[] array. The items are listed in the ListView by means of a 'for loop', and first item, element 0, is mapped to the Bing Map control, with a pushpin displayed, and the map is zoomed into level 17.

When the selection is changed in the ListView control, the latitude and longitude are read from the returnedLocaitons list and the coordinates are mapped to the Bing Map control.

C# Code for Mainpage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
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.Navigation;
//Add the following directives
using Windows.Devices.Geolocation;
using Windows.UI.Popups;
using Bing.Maps;


namespace _09D_Searchable_Map
{
    public sealed partial class MainPage : Page
    {
        //class-level variables
        double lat, lng;
        struct geoLoc  //used to track the returned locations
        {
            public double lat;
            public double lng;
            public string latLng;
            public string address;
        }
        IList<geoLoc> returnedLocations = new List<geoLoc>();

        public MainPage()
        {
            this.InitializeComponent();
        }

        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
        }
```

```
private async void MapAddress(object sender, RoutedEventArgs e)
{   // Use Bing.Mpas.Search to find the user-provided address / landmark
    int locCount = 0;
    try
    {
        string myAddress = txtAddress.Text + "," + txtCity.Text + "," +
                 txtState.Text + "," + txtZip.Text;
        Bing.Maps.Search.GeocodeRequestOptions requestOptions = new
                Bing.Maps.Search.GeocodeRequestOptions(myAddress);
        requestOptions.MaxResults = 20;  // 20 is the largest value allowed
        Bing.Maps.Search.SearchManager searchMgr = myMap.SearchManager;
        Bing.Maps.Search.LocationDataResponse response = await
                searchMgr.GeocodeAsync(requestOptions);
        locCount = response.LocationData.Count;
        returnedLocations.Clear();
        lvOptions.Items.Clear();
        geoLoc xyz;
        if (locCount > 0)
        {
            for (int i = 0; i < locCount; i++)
            {  //build the ListView display and populate returnedLocations list
                xyz.lat = response.LocationData[i].Location.Latitude;
                xyz.lng = response.LocationData[i].Location.Longitude;
                xyz.latLng = xyz.lat.ToString() + ", " + xyz.lng.ToString();
                Bing.Maps.Search.ReverseGeocodeRequestOptions zyx = new
                          Bing.Maps.Search.ReverseGeocodeRequestOptions(new
                          Location(xyz.lat, xyz.lng));
                xyz.address = response.LocationData[i].Address.FormattedAddress;
                returnedLocations.Add(xyz);
                //lvOptions.Items.Add(xyz);
                lvOptions.Items.Add(xyz.address + "\n" + xyz.latLng);
            }
            lvOptions.SelectedIndex = 0;
            txtReturns.Text = locCount.ToString() + " Locations found"; //max is 20
            // Map the first Address
            lat = response.LocationData[0].Location.Latitude;
            lng = response.LocationData[0].Location.Longitude;
            myMap.Children.Clear();
            myMap.Center = new Bing.Maps.Location(lat, lng);
            myMap.ZoomLevel = 17;
            myMap.MapType = MapType.Aerial;
            //Add a pushpin to the target
            Pushpin pp = new Pushpin();
            pp.Text = "A";
            MapLayer.SetPosition(pp, new Location(lat, lng));
            myMap.Children.Add(pp);
        }
        else
        {
            ShowMessage("Could not find " + myAddress,"Address/Place Not found");
        }
    }
    catch (Exception ex)
    {
        ShowMessage(ex.Message + "\n Locations returned: " + locCount.ToString()
                  + "\n" + lat.ToString() + ", " + lng.ToString(), "Error –
                   Unable to display the map");
    }
}
```

```
        private async void ShowMessage(string prompt, string title)
        {
            MessageDialog md = new MessageDialog(prompt, title);
            await md.ShowAsync();
        }

        private void ShowNewAddress(object sender, SelectionChangedEventArgs e)
        {
            ListView myLV = (ListView)sender;
            int xyz = myLV.SelectedIndex;
            if (xyz > -1)
            {
                double lat = returnedLocations[xyz].lat;
                double lng = returnedLocations[xyz].lng;
                myMap.Children.Clear();
                myMap.Center = new Bing.Maps.Location(lat, lng);
                myMap.MapType = MapType.Aerial;
                Pushpin pp = new Pushpin();
                pp.Text = "A";
                MapLayer.SetPosition(pp, new Location(lat, lng));
                myMap.Children.Add(pp);
            }
        }
    }
}
```

VB Code for MainPage.xaml.vb

```
Imports Windows.Devices.Geolocation
Imports Windows.UI.Popups
Imports Bing.Maps

Public NotInheritable Class MainPage
    Inherits Page
    'Class Level Variables:
    Dim lat, lng As Double
    Structure geoLoc  'used to track the returned locations
        Dim lat As Double
        Dim lng As Double
        Dim latLng As String
        Dim address As String
    End Structure
    Dim returnedLocations As IList(Of geoLoc) = New List(Of geoLoc)()


    Protected Overrides Sub OnNavigatedTo(e As Navigation.NavigationEventArgs)

    End Sub

    Private Async Sub MapAddress(sender As Object, e As RoutedEventArgs)
        Try
            Dim myAddress As String = txtAddress.Text & "," & txtCity.Text & "," & _
                    txtState.Text & "," & txtZip.Text
            Dim requestOptions As Bing.Maps.Search.GeocodeRequestOptions = New  _
                    Bing.Maps.Search.GeocodeRequestOptions(myAddress)
            requestOptions.MaxResults = 20   ' 20 is the largest value allowed
            Dim searchMgr As Bing.Maps.Search.SearchManager = myMap.SearchManager

            Dim response As Bing.Maps.Search.LocationDataResponse = Await _
                searchMgr.GeocodeAsync(requestOptions)
            Dim locCount As Integer = response.LocationData.Count
```

```
            returnedLocations.Clear()
            lvOptions.Items.Clear()
            Dim xyz As geoLoc
            If (locCount > 0) Then
                'build the ListView display and populate returnedLocations list
                For i = 0 To locCount - 1
                    xyz.lat = response.LocationData(i).Location.Latitude
                    xyz.lng = response.LocationData(i).Location.Longitude
                    xyz.latLng = xyz.lat.ToString() & ", " & xyz.lng.ToString()
                    Dim zyx As Bing.Maps.Search.ReverseGeocodeRequestOptions = New  _
                        Bing.Maps.Search.ReverseGeocodeRequestOptions(New  _
                                  Location(xyz.lat, xyz.lng))
                    xyz.address = response.LocationData(i).Address.FormattedAddress
                    returnedLocations.Add(xyz)
                    lvOptions.Items.Add(xyz.address & vbCrLf & xyz.latLng)
                Next
                lvOptions.SelectedIndex = 0
                txtReturns.Text = locCount.ToString() & " Locations found"   'max is 20
                ' Map the first Address
                lat = response.LocationData(0).Location.Latitude
                lng = response.LocationData(0).Location.Longitude

                myMap.Children.Clear()
                myMap.Center = New Bing.Maps.Location(lat, lng)
                myMap.ZoomLevel = 17
                myMap.MapType = MapType.Aerial
                'Add a pushpin to the target
                Dim pp As Pushpin = New Pushpin()
                pp.Text = "A"
                MapLayer.SetPosition(pp, New Location(lat, lng))
                myMap.Children.Add(pp)
            Else
                ShowMessage("Could not find " + myAddress, "Address/Place Not found")
            End If
        Catch ex As Exception
            ShowMessage(ex.Message, "Error")
        End Try
    End Sub

    Private Async Sub ShowMessage(ByVal prompt As String, ByVal title As String)
        Dim md As MessageDialog = New MessageDialog(prompt, title)
        Await md.ShowAsync()
    End Sub

    Private Sub ShowNewAddress(sender As Object, e As SelectionChangedEventArgs) _
                                          Handles lvOptions.SelectionChanged
        Dim myLV As ListView = sender
        Dim xyz As Integer = myLV.SelectedIndex
        If (xyz > -1) Then
            Dim lat As Double = returnedLocations(xyz).lat
            Dim lng As Double = returnedLocations(xyz).lng
            myMap.Children.Clear()
            myMap.Center = New Bing.Maps.Location(lat, lng)
            myMap.MapType = MapType.Aerial
            Dim pp As Pushpin = New Pushpin()
            pp.Text = "A"
            MapLayer.SetPosition(pp, New Location(lat, lng))
            myMap.Children.Add(pp)
        End If
    End Sub
End Class
```

Mapping a Driving Route between Two Points

The Directions API of Bing Maps is used to plot two points on the map and draw the driving path between the points. The Directions API of Bing Maps also provides step-by-step directions to navigate from the first address/landmark to the second. In this example, the user will enter two points and will then click a button to display the driving path on the map and the step-by-step directions in a TextBox. The Directions API can also determine the total distance and an estimated time of travel.

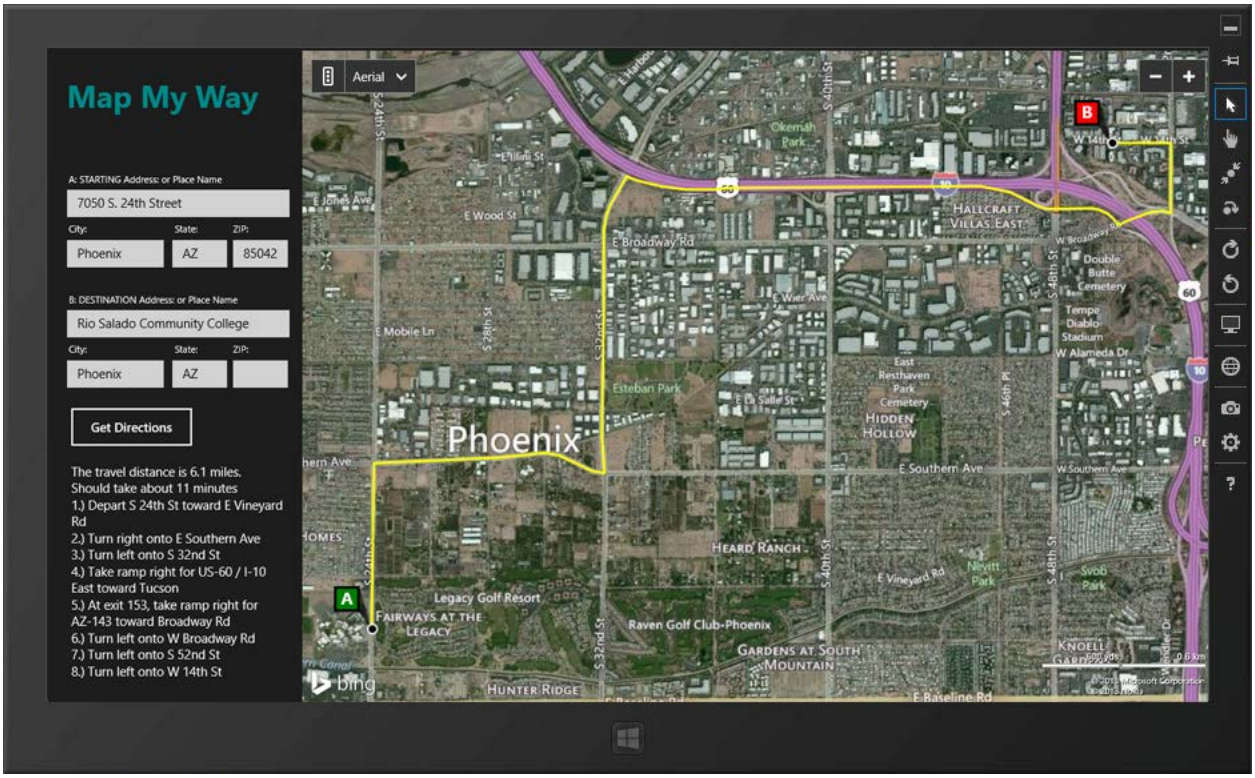

Figure 12 *– Objects and methods of the Directions API are employed to map two points on the map, draw a driving route between them, and provide written driving directions.*

To map a driving route between two points, complete the following steps:

Step 1: Create a new project using the Blank Page template. This project assumes that you have previously downloaded and installed the Bing Maps SDK and obtained a Bing Map key.

Step 2: Choose a target processor (such as x64) in the Configuration Manager (Build menu). See Figure 9 in the "Using Bing Maps API to Display a Location" section.

Step 3: Add a reference to the Bing Maps API, checking the boxes for "Bing Maps for C#, C++, or Visual Basic" and "Microsoft Visual C++ Runtime Package." See Figure 10 in the "Using Bing Maps API to Display a Location" section.

Step 4: Create the XAML interface. At its foundational level, the interface consists of two columns. The left column contains TextBlock and TextBox controls to display the app's title

and get input from the user in terms of two addresses or landmarks to display on the map. A button is used to call the code-behind process to map the points and the driving route between them in the map control. In addition, you also generate step-by-step driving directions and total distance and estimated travel time, which are outputted to a TextBox control.

XAML Code for MainPage.xaml

```
<Page
    x:Class="_09E_Mapping_Directions__VB_.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:_09E_Mapping_Directions__VB_"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:bingmap="using:Bing.Maps"
    mc:Ignorable="d">



    <Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="300"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>

        <Rectangle Grid.Column="1" Fill="DarkCyan" />
        <TextBlock HorizontalAlignment="Left" Height="96" Margin="25,35,0,0"
               TextWrapping="Wrap" Text="Map My Way" VerticalAlignment="Top"
               Width="230" Foreground="DarkCyan" FontSize="36" FontWeight="Bold"/>
        <TextBlock HorizontalAlignment="Left" Height="18" Margin="27,145,0,0"
               TextWrapping="Wrap" Text="A: STARTING Address: or Place Name"
               VerticalAlignment="Top" Width="205"/>
        <TextBox x:Name="txtStartAddress" HorizontalAlignment="Left" Height="28"
               Margin="25,163,0,0" TextWrapping="Wrap" Text="7050 S. 24th Street"
               VerticalAlignment="Top" Width="260"/>
        <TextBlock HorizontalAlignment="Left" Height="17" Margin="27,203,0,0"
               TextWrapping="Wrap" Text="City:" VerticalAlignment="Top" Width="47"/>
        <TextBox x:Name="txtStartCity" HorizontalAlignment="Left" Height="28"
               Margin="25,222,0,0" TextWrapping="Wrap" Text="Phoenix"
               VerticalAlignment="Top" Width="113"/>
        <TextBlock HorizontalAlignment="Left" Height="17" Margin="151,203,0,0"
               TextWrapping="Wrap" Text="State:" VerticalAlignment="Top" Width="47"/>
        <TextBox x:Name="txtStartState" HorizontalAlignment="Left" Height="28"
               Margin="148,222,0,0" TextWrapping="Wrap" Text="AZ" VerticalAlignment="Top"
               Width="29"/>
        <TextBlock HorizontalAlignment="Left" Height="13" Margin="219,203,0,0"
               TextWrapping="Wrap" Text="ZIP:" VerticalAlignment="Top" Width="23"/>
        <TextBox x:Name="txtStartZip" HorizontalAlignment="Left" Height="28"
               Margin="219,222,0,0" TextWrapping="Wrap" Text="85042"
               VerticalAlignment="Top" Width="53"/>
        <TextBlock HorizontalAlignment="Left" Height="18" Margin="27,286,0,0"
              TextWrapping="Wrap" Text="B: DESTINATION Address: or Place Name"
              VerticalAlignment="Top" Width="205"/>
        <TextBox x:Name="txtDestAddress" HorizontalAlignment="Left" Height="28"
             Margin="25,304,0,0" TextWrapping="Wrap" Text="Rio Salado Community College"
             VerticalAlignment="Top" Width="260"/>
        <TextBlock HorizontalAlignment="Left" Height="17" Margin="27,344,0,0"
```

```
            TextWrapping="Wrap" Text="City:" VerticalAlignment="Top" Width="47"/>
        <TextBox x:Name="txtDestCity" HorizontalAlignment="Left" Height="28"
            Margin="25,363,0,0" TextWrapping="Wrap" Text="Phoenix"
            VerticalAlignment="Top" Width="113"/>
        <TextBlock HorizontalAlignment="Left" Height="17" Margin="151,344,0,0"
            TextWrapping="Wrap" Text="State:" VerticalAlignment="Top" Width="47"/>
        <TextBox x:Name="txtDestState" HorizontalAlignment="Left" Height="28"
            Margin="148,363,0,0" TextWrapping="Wrap" Text="AZ" VerticalAlignment="Top"
            Width="29"/>
        <TextBlock HorizontalAlignment="Left" Height="13" Margin="219,344,0,0"
            TextWrapping="Wrap" Text="ZIP:" VerticalAlignment="Top" Width="23"/>
        <TextBox x:Name="txtDestZip" HorizontalAlignment="Left" Height="28"
            Margin="219,363,0,0" TextWrapping="Wrap" Text=""
            VerticalAlignment="Top" Width="53"/>
        <Button Content="Get Directions" HorizontalAlignment="Left" Margin="27,416,0,0"
            VerticalAlignment="Top" Width="147" Height="50" Click="ShowDirections"/>
        <TextBox x:Name="txtItinerary" HorizontalAlignment="Left" Height="260"
            Margin="20,480,0,0" TextWrapping="Wrap" Text="" Foreground="White"
            Background="{StaticResource ApplicationPageBackgroundThemeBrush}"
            VerticalAlignment="Top" ScrollViewer.VerticalScrollBarVisibility="Auto"
            Width="270" BorderThickness="0"/>
        <bingmap:Map Grid.Column="1" Credentials=" Anz2Wc8C6cTNYy0-bm1ussbKLYf-
            yhQjIuzzXproUTLHnyUblyfxgLGHqzi8L_py8" x:Name="myMap" />
    </Grid>
</Page>
```

Step 5: Write the code. In the code-behind, the starting and destination addresses are gathered from the input textboxes. Then, they are geocoded using the Search Manager and LocationDataResponse objects from the Search API of Bing.Maps to find the latitude and longitude of each address/landmark. With that information, the Directions API of Bing.Maps is employed to use a DirectionsManager object to map the points and plot the route. The path is displayed in yellow by modifying the RenderOptions.ActiveRoutePolylineOptions.LineColor of the DirectionsManager object to an ARGB value of 255,255,255,0.

The TravelDistance and TravelDuration properties of the DirectionsManager object are used to display the estimated distance and travel time. The step-by-step itinerary is determined from the results of a CalculateDirectionsAsync() call. These results are parsed out as ItineraryItems of the RouteLeg array in a 'for loop' and added to the TextBox output display at the bottom of the left column.

C# Code for MainPage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
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.Navigation;
```

```
using Windows.Devices.Geolocation;
using Windows.UI.Popups;
using Bing.Maps;

namespace _09E_Mapping_Directions__CS_
{

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


        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
        }

        private async void ShowDirections(object sender, RoutedEventArgs e)
        {
            try
            {
                txtItinerary.Text = "";
                //Geocode the starting address
                string myStartAddress = txtStartAddress.Text + "," + txtStartCity.Text
                       + "," + txtStartState.Text + "," + txtStartZip.Text;
                Bing.Maps.Search.GeocodeRequestOptions startRequestOptions = new
                          Bing.Maps.Search.GeocodeRequestOptions(myStartAddress);
                Bing.Maps.Search.SearchManager startSearchMgr = myMap.SearchManager;
                Bing.Maps.Search.LocationDataResponse startResponse = await
                          startSearchMgr.GeocodeAsync(startRequestOptions);
                double startLat = startResponse.LocationData[0].Location.Latitude;
                double startLng = startResponse.LocationData[0].Location.Longitude;

                // Geocode the destination Address
                string myDestinationAddress = txtDestAddress.Text + "," +
                     txtDestCity.Text + "," + txtDestState.Text + "," + txtDestZip.Text;
                Bing.Maps.Search.GeocodeRequestOptions destinationRequestOptions = new
                        Bing.Maps.Search.GeocodeRequestOptions(myDestinationAddress);
                Bing.Maps.Search.SearchManager destinationSearchMgr =
                        myMap.SearchManager;
                Bing.Maps.Search.LocationDataResponse destinationResponse = await
                        destinationSearchMgr.GeocodeAsync(destinationRequestOptions);
                double destinationLat =
                        destinationResponse.LocationData[0].Location.Latitude;
                double destinationLng =
                        destinationResponse.LocationData[0].Location.Longitude;
                myMap.MapType = MapType.Aerial;

                //Calculate and show the driving route
                Bing.Maps.Directions.Waypoint beginWaypoint = new
                     Bing.Maps.Directions.Waypoint(new
                     Bing.Maps.Location(startLat, startLng));
                Bing.Maps.Directions.Waypoint destinationWaypoint = new
                     Bing.Maps.Directions.Waypoint(new
                     Bing.Maps.Location(destinationLat, destinationLng));
                Bing.Maps.Directions.WaypointCollection waypoints = new
```

```
                    Bing.Maps.Directions.WaypointCollection();
                waypoints.Add(beginWaypoint);
                waypoints.Add(destinationWaypoint);
                Bing.Maps.Directions.DirectionsManager directionsManager =
                    myMap.DirectionsManager;
                // Set route color to yellow
                directionsManager.RenderOptions.ActiveRoutePolylineOptions.LineColor =
                     Windows.UI.Color.FromArgb(255, 255, 255, 0);
                directionsManager.Waypoints = waypoints;
                //Calculate route directions
                Bing.Maps.Directions.RouteResponse myRoutes = await
                    directionsManager.CalculateDirectionsAsync();
                //Display the route on the map
                directionsManager.ShowRoutePath(myRoutes.Routes[0]);

                //Display Itinerary
                int i;
                double dist = myRoutes.Routes[0].TravelDistance;
                string dur = (myRoutes.Routes[0].TravelDuration / 60).ToString("N0");
                string routeInfo = "YOUR ITINERARY \n";
                routeInfo = "The travel distance is " + dist.ToString("N1") + " " +
                   myRoutes.Routes[0].DistanceUnit.ToString().ToLower() +
                   "s. \nShould take about " + dur + " minutes";
                //Get step by step directions

                for (i=0; i< myRoutes.Routes[0].RouteLegs[0].ItineraryItems.Count; i++)
                {
                    routeInfo += "\n" + (i + 1).ToString() + ".) " +
                      myRoutes.Routes[0].RouteLegs[0].ItineraryItems[i].Instruction.Text;
                }
                txtItinerary.Text = routeInfo;
            }
            catch (Exception ex)
            {
                ShowMessage("Unable to find points or route.\n" + ex.Message,
                        "Modify Addresses and Try Again");
            }
        }

        private async void ShowMessage(string prompt, string title)
        {
            MessageDialog md = new MessageDialog(prompt, title);
            await md.ShowAsync();
        }
    }
}
```

VB Code for MainPage.xaml.vb

```
Imports Windows.Devices.Geolocation
Imports Windows.UI.Popups
Imports Bing.Maps

Public NotInheritable Class MainPage
    Inherits Page


    Protected Overrides Sub OnNavigatedTo(e As Navigation.NavigationEventArgs)
```

```
    End Sub

    Private Async Sub ShowDirections(sender As Object, e As RoutedEventArgs)
        Try
            txtItinerary.Text = ""
            'Geocode the starting address
            Dim myStartAddress As String = txtStartAddress.Text & "," & txtStartCity.Text _
                  & "," & txtStartState.Text & "," & txtStartZip.Text
            Dim startRequestOptions As Bing.Maps.Search.GeocodeRequestOptions = New  _
                    Bing.Maps.Search.GeocodeRequestOptions(mystartAddress)
            Dim startSearchMgr As Bing.Maps.Search.SearchManager = myMap.SearchManager
            Dim startResponse As Bing.Maps.Search.LocationDataResponse = Await _
                startSearchMgr.GeocodeAsync(startRequestOptions)
            Dim startLat As Double = startResponse.LocationData(0).Location.Latitude
            Dim startLng As Double = startResponse.LocationData(0).Location.Longitude

            ' Geocode the destination Address
            Dim myDestinationAddress As String = txtDestAddress.Text & "," & _
                    txtDestCity.Text & "," & txtDestState.Text & "," & txtDestZip.Text
            Dim destinationRequestOptions As Bing.Maps.Search.GeocodeRequestOptions = New  _
                    Bing.Maps.Search.GeocodeRequestOptions(myDestinationAddress)
            Dim destinationSearchMgr As Bing.Maps.Search.SearchManager = myMap.SearchManager
            Dim destinationResponse As Bing.Maps.Search.LocationDataResponse = Await _
                destinationSearchMgr.GeocodeAsync(destinationRequestOptions)
            Dim destinationLat As Double =
                   destinationResponse.LocationData(0).Location.Latitude
            Dim destinationLng As Double =
                   destinationResponse.LocationData(0).Location.Longitude
            myMap.MapType = MapType.Aerial

            'Calculate and show the driving route
            Dim beginWaypoint As Bing.Maps.Directions.Waypoint = New  _
                  Bing.Maps.Directions.Waypoint(New  _
                  Bing.Maps.Location(startLat, startLng))
            Dim destinationWaypoint As Bing.Maps.Directions.Waypoint = New  _
                  Bing.Maps.Directions.Waypoint(New  _
                  Bing.Maps.Location(destinationLat, destinationLng))
            Dim waypoints As Bing.Maps.Directions.WaypointCollection = New  _
                  Bing.Maps.Directions.WaypointCollection()
            waypoints.Add(beginWaypoint)
            waypoints.Add(destinationWaypoint)
            Dim directionsManager As Bing.Maps.Directions.DirectionsManager = _
                  myMap.DirectionsManager
            ' Set route color to yellow
            directionsManager.RenderOptions.ActiveRoutePolylineOptions.LineColor = _
                 Windows.UI.Color.FromArgb(255, 255, 255, 0)
            directionsManager.Waypoints = waypoints
            'Calculate route directions
            Dim myRoutes As Bing.Maps.Directions.RouteResponse = _
                Await directionsManager.CalculateDirectionsAsync()
            'Display the route on the map
            directionsManager.ShowRoutePath(myRoutes.Routes(0))

            'Display Itinerary
            Dim i As Integer
            Dim dist As Double = myRoutes.Routes(0).TravelDistance
            Dim dur As String = (myRoutes.Routes(0).TravelDuration / 60).ToString("N0")
            Dim routeInfo As String = "YOUR ITINERARY" & vbCrLf
            routeInfo = "The travel distance is " & dist.ToString("N1") & " " & _
                   myRoutes.Routes(0).DistanceUnit.ToString().ToLower() & _
                   "s. " & vbCrLf & "Should take about " & dur & " minutes"
```

```
            'Get step by step directions
            For i = 0 To myRoutes.Routes(0).RouteLegs(0).ItineraryItems.Count - 1
                routeInfo += vbCrLf & (i + 1).ToString() + ".) " & _
                      myRoutes.Routes(0).RouteLegs(0).ItineraryItems(i).Instruction.Text
            Next
            txtItinerary.Text = routeInfo

        Catch ex As Exception
            ShowMessage("Unable to find points or route.", _
                        "Modify Addresses andTry Again")
        End Try
    End Sub

    Private Async Sub ShowMessage(ByVal prompt As String, ByVal title As String)
        Dim md As MessageDialog = New MessageDialog(prompt, title)
        Await md.ShowAsync()
    End Sub

End Class
```

Step 6: Test thoroughly. Modify and debug as needed.