**Problem Statement:**

Photos are helpful for identifying users. User can have many photos at a time. When there are many photos it will be hard to organize and find who was there in the photos. Addis Photo Imaging Inc has stored many user photos as part of their business. They want to notify the user in real time if they found and identify a user from the existing digitally stored photos. To be able to do so, they want to present the result in the web application so that user can access the result remotely. They also want to present certain simple analysis like age range and gender on the number of people found. To solve their problem, I will use Azure Face API to detect, identify, train and store user photos, present those photos in a web application that for subscribe the real-time message notification service, charting and image control. The service and the web application will be hosted on Azure portal.

**Overview of the Technology:**

The project the demonstrate the use and benefits of Azure Face API to solve the problem stated above. Face API is one branch of an Artificial Intelligence and Machine – Learning that concentrate on analyzing photo faces and present very useful face attributes. Face attributes could determine Age, Gender, Smile, Emotion and more so on and so forth. Such results are useful to determine

1. Detect, Identify, Verify, Train different user photos.
2. The number of people based on Gender, Age range

It also uses [SignalR](https://www.asp.net/signalr) based REST service called Real-Time Message Broker(RTMB) for real time notification and [Plotly](https://plot.ly/javascript/#basic-charts) JS framework for charting.

**High Level Steps:**

1. Download the source code from GitHub repository.
2. Download data source.
3. Build Azure Face API client on WPF application.
4. Build Real-time Message Broker(RTMB) service on Web API.
5. Build Azure Face API Chart and Photo viewer on web application.
6. Deploy/Host the Web API (5) and web application (6) on Azure Portal. They are already hosted.
7. Run Azure Face API client on WPF application (4) apply the data source.

**Data source:**

<https://www.dropbox.com/sh/tlizujsmex4c2aa/AADmug45j7kAk7XCrjA5JTRXa?dl=0>

**Hardware Used:**

Windows 7 or 10, 64bit processor laptop

**Limitation:**

Azure Face API will not recognize every given photo. Thus, there is a high chance face won’t be returned even if a face shown in the photo. This is the limitation of Azure Face API Artificial Intelligence recognition capability.

**Software Used:**

Visual Studio 2017 latest version (<https://www.visualstudio.com/downloads/>)

**Slides**

<https://github.com/wondechala/AzureFaceAPISampleSlides>

**Report**

<https://github.com/wondechala/AzureFaceAPISampleReport>

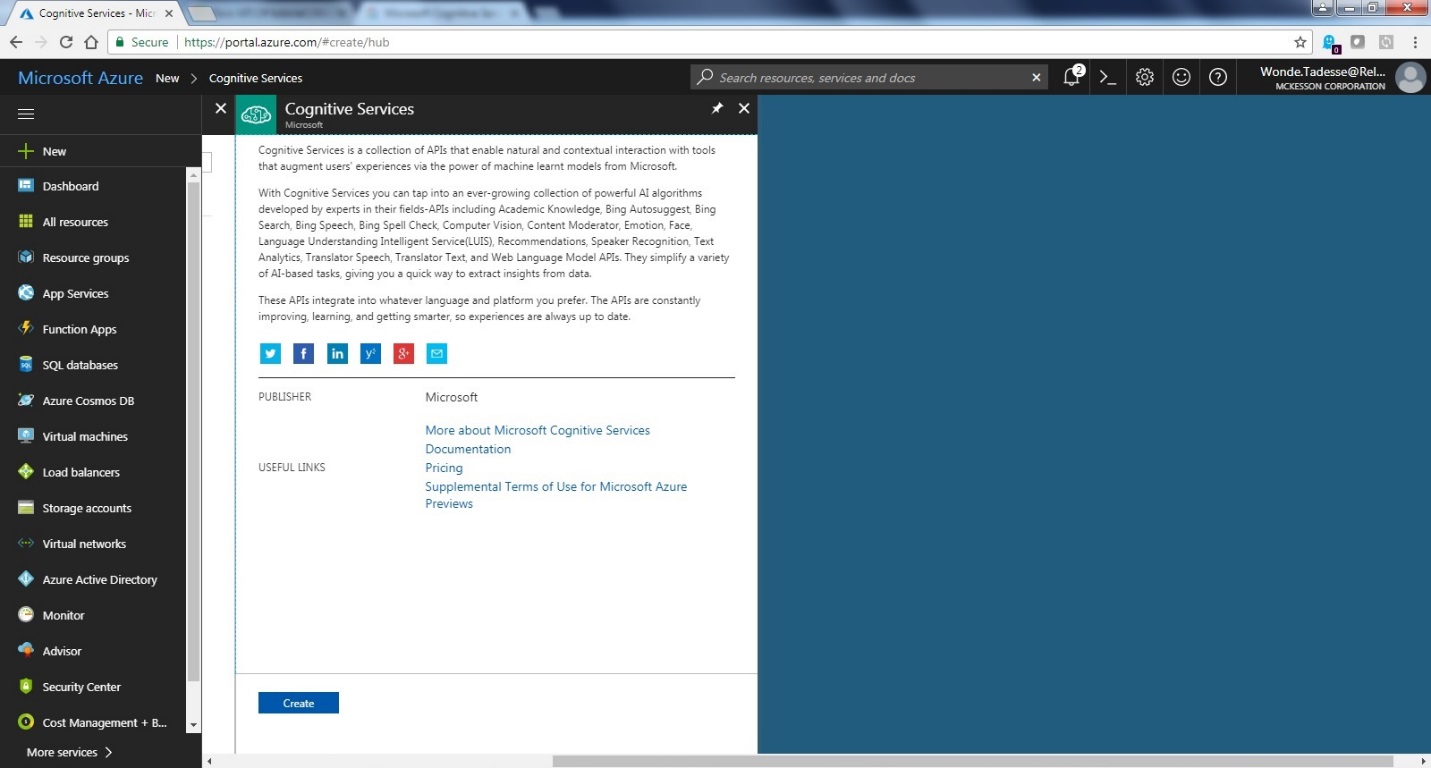
**GitHub Repository (Source Code)**

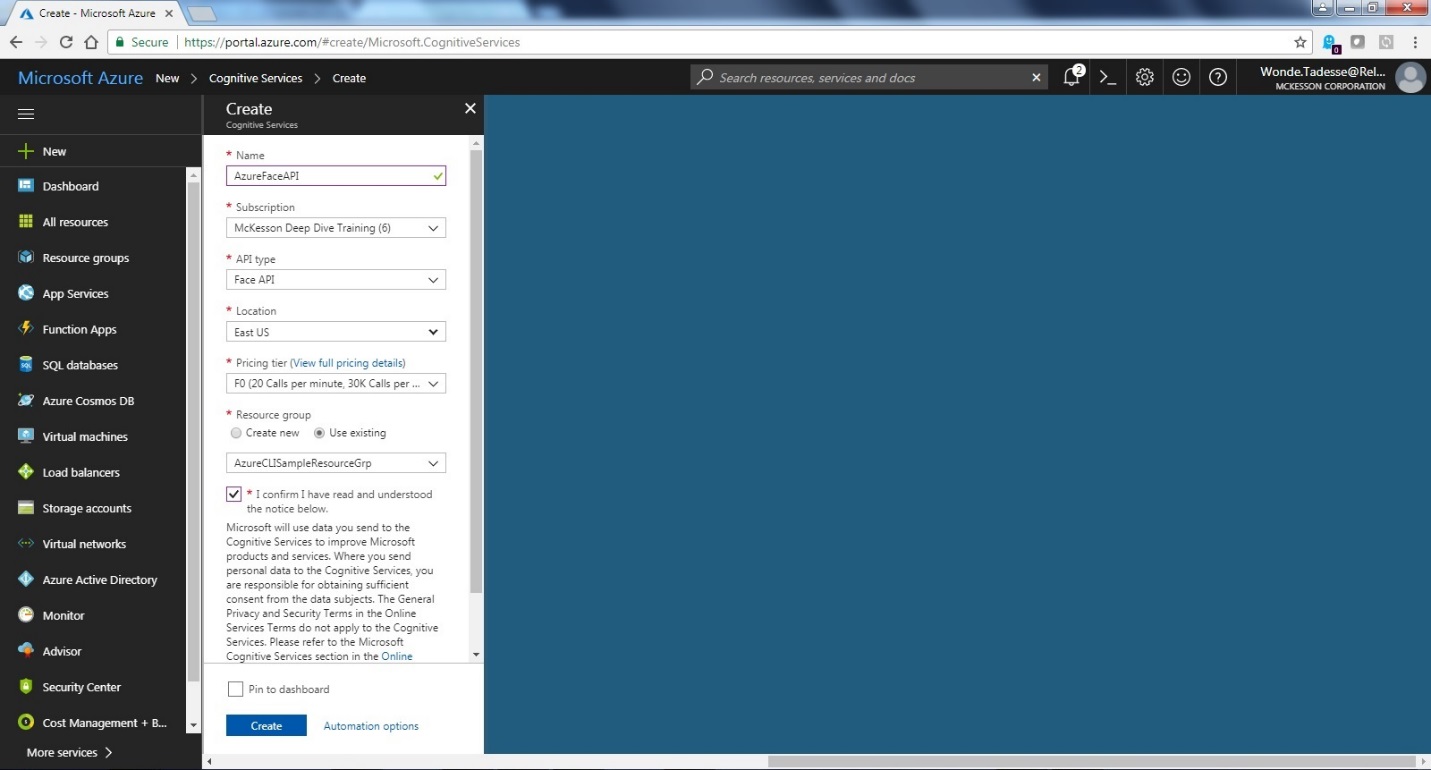
<https://github.com/wondechala/AzureFaceAPISample>

Solution Steps:

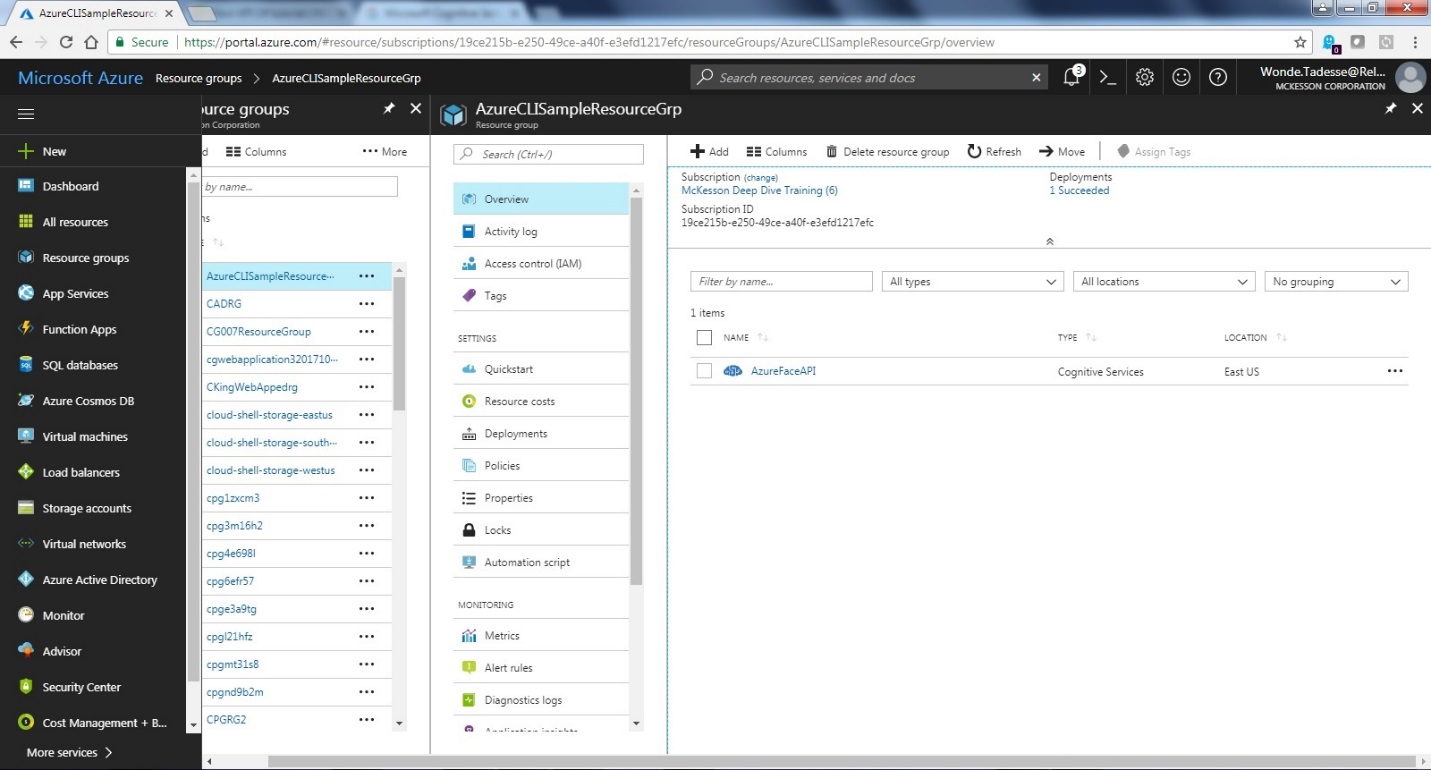
1. First let’s create Azure Face API on Azure. Follow the following steps.

a.

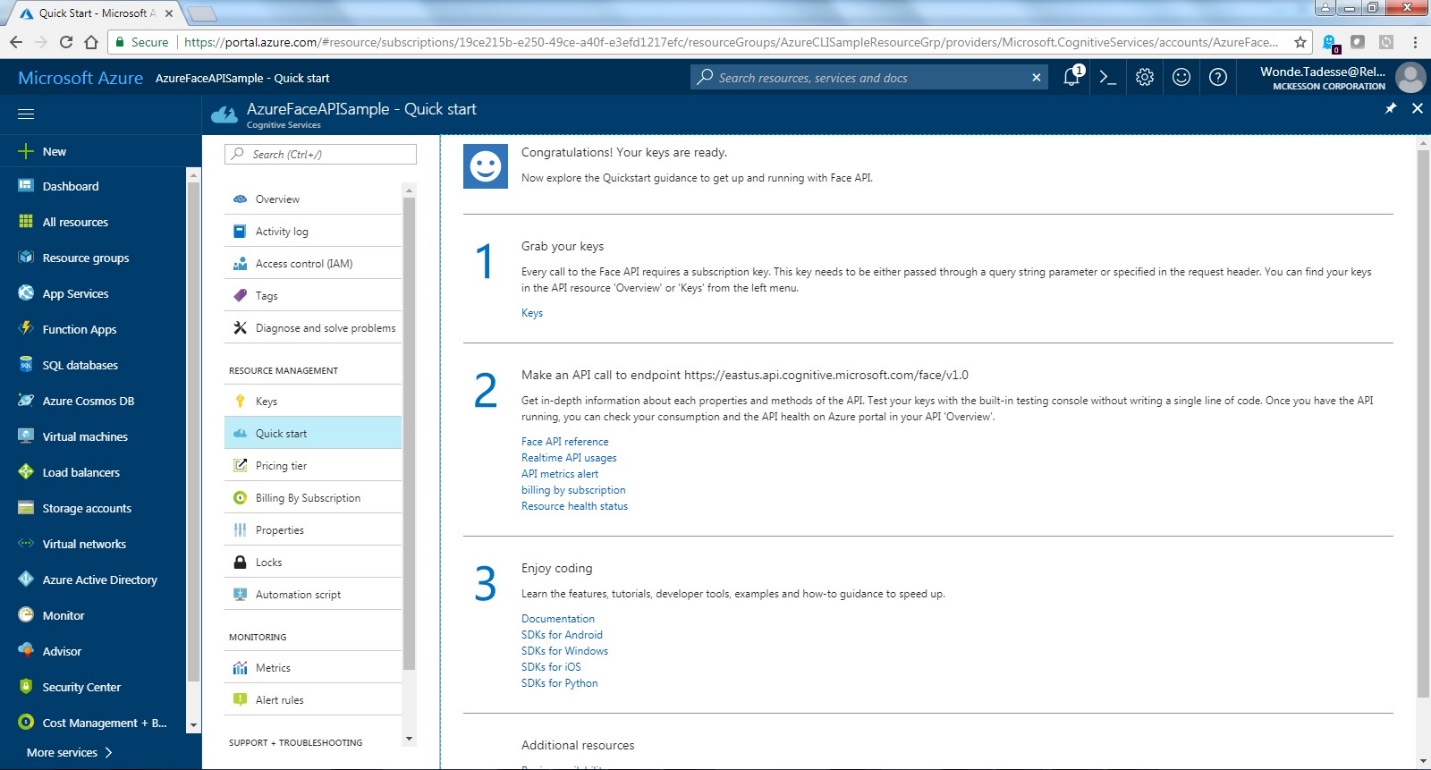


b. 

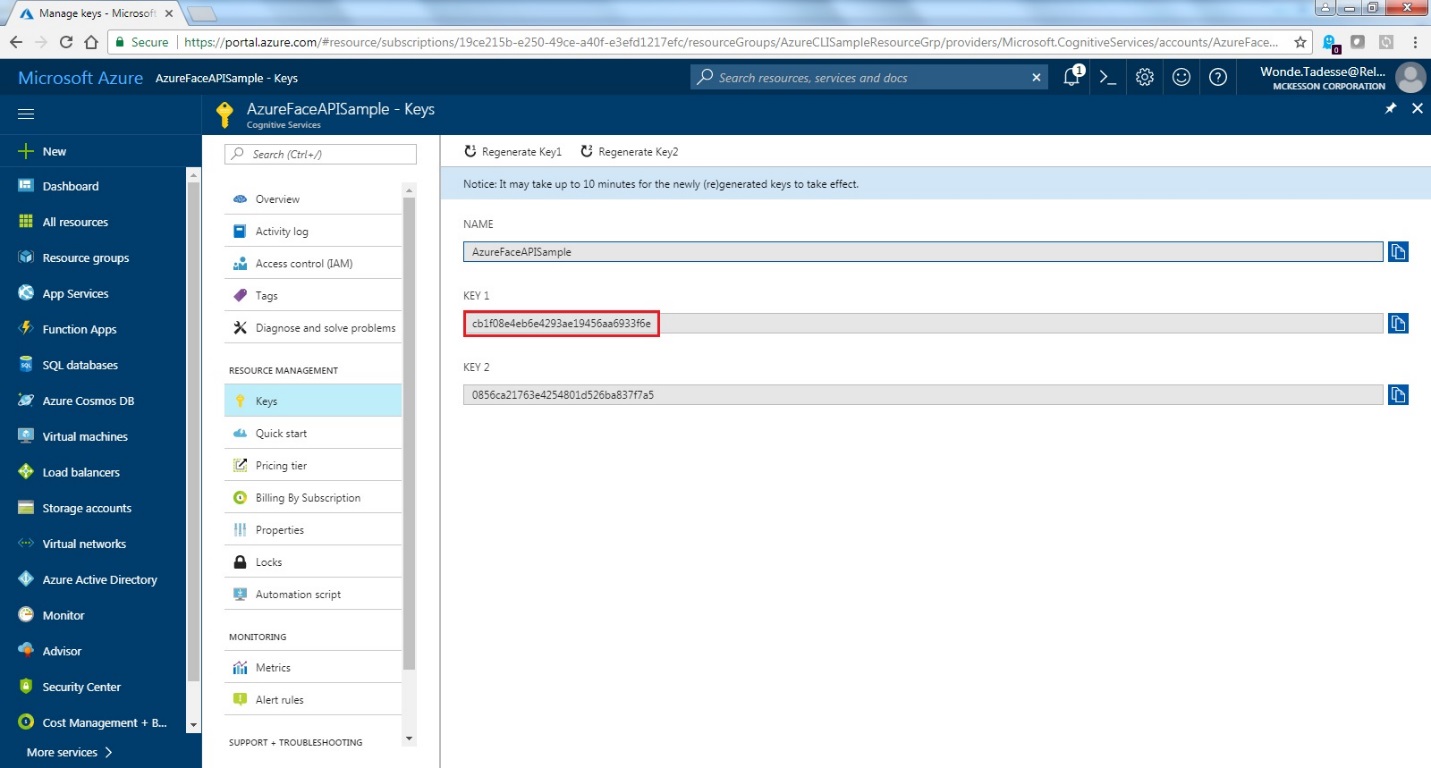
c.

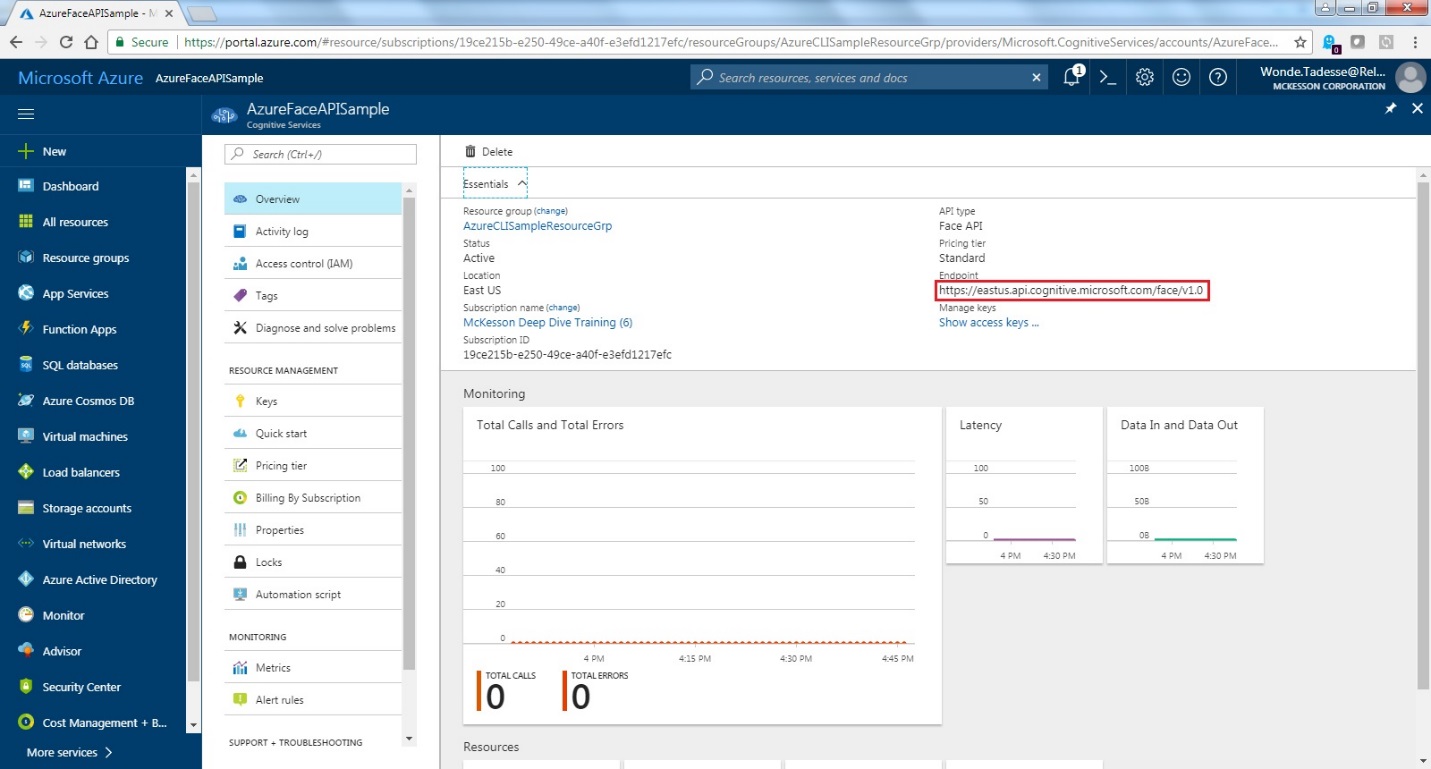


d.



e.

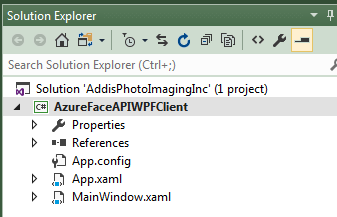


f. 

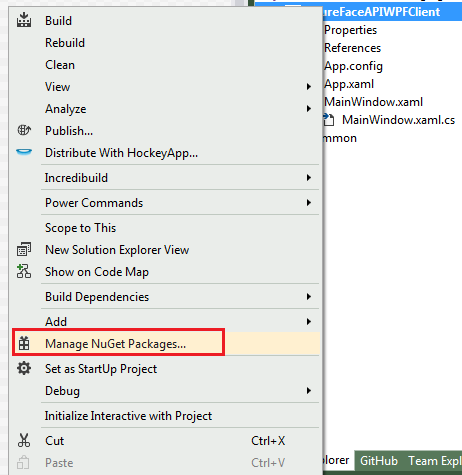
1. Download Visual Studio 2017 latest version. <https://www.visualstudio.com/downloads/>). If you don’t have, Community Version is preferable.
2. Create a WPF project named ‘AzureFaceAPIWPFClient’ and the Solution name called ‘AddisPhotoImagingInc’



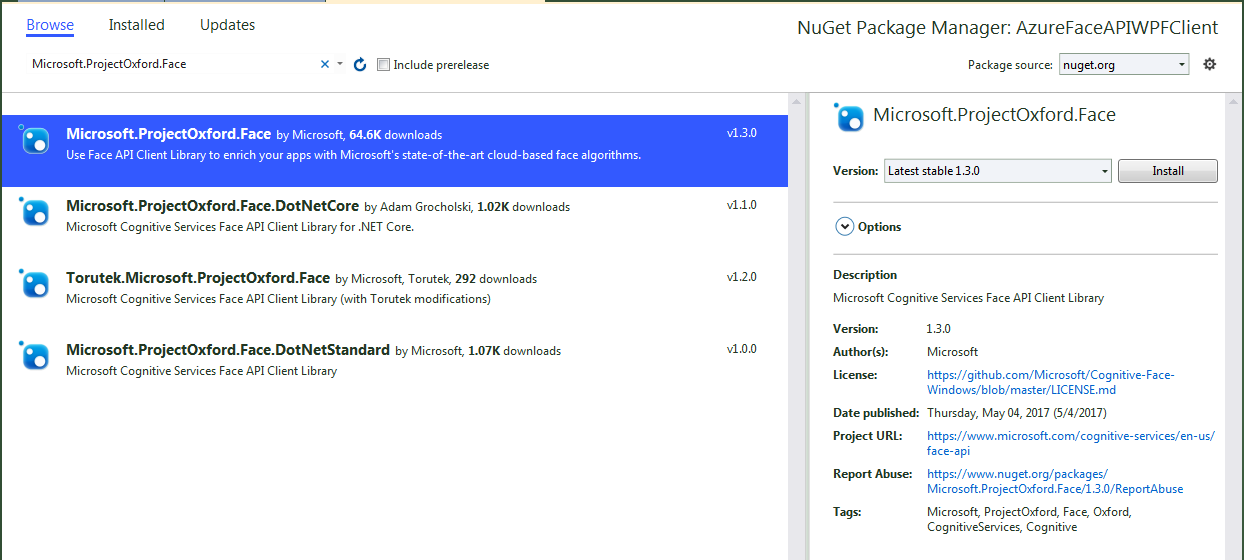
1. After press Ok, you should be able to see project on ‘Solution Explorer’



1. Go to Nuget Manager and the following package to ‘AzureFaceAPIWPFClient’ project. To do so, right click the project and select ‘Manage Nuget Packages…’



1. A Nuget Manager dialog will appear as follows, put ‘Microsoft.ProjectOxford.Face’



1. Apply the same step 12-13 for nuget package ‘Newtonsoft.Json’ and ‘System.Net.Http.Formatting’
2. While you add Nuget package, you might have noticed a file name called ‘packages.config’ file on ‘AzureFaceAPIWPFClient’ project. After successfully add those packages mentioned on the previous steps, you should be able to the following content on the file.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <packages>    <package id="Microsoft.Bcl" version="1.1.10" targetFramework="net462" />    <package id="Microsoft.Bcl.Build" version="1.0.14" targetFramework="net462" />    <package id="Microsoft.Net.Http" version="2.2.29" targetFramework="net462" />    <package id="Microsoft.ProjectOxford.Common" version="1.0.324" targetFramework="net462" />    <package id="Microsoft.ProjectOxford.Face" version="1.3.0" targetFramework="net462" />    <package id="Newtonsoft.Json" version="10.0.3" targetFramework="net462" />    <package id="System.Net.Http" version="2.0.20126.16343" targetFramework="net462" />    <package id="System.Net.Http.Formatting.Extension" version="5.2.3.0" targetFramework="net462" />  </packages> |

1. Double clicking ‘MainWindow.xaml’ file will open the design mode of the file.



1. Update the ‘XAML’ section with the following code

|  |
| --- |
| <Window x:Class="AzureFaceAPIWPFClient.MainWindow"          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"          Title="Azure Face API WPF App" Height="600" Width="800">      <Grid x:Name="BackPanel" Background="#007FFF">          <Border BorderThickness="2" BorderBrush="#FF000000">              <Image x:Name="FacePhoto"  Margin="0,0,155,50" MouseMove="FacePhoto\_MouseMove" />          </Border>          <DockPanel DockPanel.Dock="Right">              <Button x:Name="DetectFacesButton"                      Width="148" Height="50"                      VerticalAlignment="Top" HorizontalAlignment="Right"                      Content="Detect Faces"                      Click="DetectFacesButton\_Click" Margin="0,8,5,0">                  <Button.RenderTransform>                      <TransformGroup>                          <ScaleTransform/>                          <SkewTransform/>                          <RotateTransform/>                          <TranslateTransform/>                      </TransformGroup>                  </Button.RenderTransform>              </Button>          </DockPanel>          <DockPanel DockPanel.Dock="Right">              <Button x:Name="TrainFacesButton" Width="146" Height="50" VerticalAlignment="Top" HorizontalAlignment="Right"                      Content="Train Faces"                      Click="TrainFacesButton\_Click" RenderTransformOrigin="0.5,0.5" Margin="0,-20,5,0" >                  <Button.RenderTransform>                      <TransformGroup>                          <ScaleTransform/>                          <SkewTransform/>                          <RotateTransform/>                          <TranslateTransform Y="360"/>                      </TransformGroup>                  </Button.RenderTransform>              </Button>          </DockPanel>          <DockPanel DockPanel.Dock="Right">              <Button x:Name="GendeChartButton" Width="146" Height="50"                      VerticalAlignment="Top" HorizontalAlignment="Right"                      Content="Gender Chart Statistics"                      Click="GenderChartButton\_Click" RenderTransformOrigin="0.5,0.5" Margin="0,-16,5,0" >                  <Button.RenderTransform>                      <TransformGroup>                          <ScaleTransform/>                          <SkewTransform/>                          <RotateTransform/>                          <TranslateTransform Y="120"/>                      </TransformGroup>                  </Button.RenderTransform>              </Button>          </DockPanel>          <DockPanel DockPanel.Dock="Right">              <Button x:Name="AgeGroupChartButton" Width="148" Height="50"                      VerticalAlignment="Top" HorizontalAlignment="Right"                      Content="Age Group Chart Statistics"                      Click="AgeGroupChartButton\_Click" RenderTransformOrigin="0.5,0.5" Margin="0,-20,5,0" >                  <Button.RenderTransform>                      <TransformGroup>                          <ScaleTransform/>                          <SkewTransform/>                          <RotateTransform/>                          <TranslateTransform Y="240"/>                      </TransformGroup>                  </Button.RenderTransform>              </Button>          </DockPanel>          <DockPanel DockPanel.Dock="Right">              <Button x:Name="IdentifyFaceButton" Width="148" Height="50"                      VerticalAlignment="Top" HorizontalAlignment="Right"                      Content="Identify Face"                      Click="IdentifyFace\_Click" RenderTransformOrigin="0.5,0.5" Margin="0,-36,5,0" >                  <Button.RenderTransform>                      <TransformGroup>                          <ScaleTransform/>                          <SkewTransform/>                          <RotateTransform/>                          <TranslateTransform Y="480"/>                      </TransformGroup>                  </Button.RenderTransform>              </Button>          </DockPanel>          <DockPanel DockPanel.Dock="Bottom" Margin="5,5,5,5">              <StatusBar VerticalAlignment="Bottom" Width="949">                  <StatusBarItem Height="56" Width="947" RenderTransformOrigin="0.5,0.5">                      <TextBlock Name="faceDescriptionStatusBar" />                  </StatusBarItem>              </StatusBar>          </DockPanel>      </Grid>  </Window> |

1. Once you update, you should able to see the form as shown below



1. Add class called ‘AzureFaceAPIProcessor.cs’, with the following content.

|  |
| --- |
| using Microsoft.ProjectOxford.Face;  using Microsoft.ProjectOxford.Face.Contract;  using System;  using System.Collections.Generic;  using System.Configuration;  using System.IO;  using System.Linq;  using System.Text;  using System.Threading.Tasks;    using AzureFaceAPISample.Entities;    namespace AzureFaceAPIWPFClient  {      /// <summary>      /// Azure Face API processor      /// </summary>      public class AzureFaceAPIProcessor      {          #region Private Variable            private IFaceServiceClient faceServiceClient;            #endregion            #region Constructor            /// <summary>          /// Azure Face API processor          /// </summary>          /// <param name="subscriptionKey">SubscriptionKey value</param>          /// <param name="url">Url value</param>          public AzureFaceAPIProcessor(string subscriptionKey, string url)          {              faceServiceClient = new FaceServiceClient(subscriptionKey, url);          }            #endregion            #region Public Methods            /// <summary>          /// Detect Faces          /// </summary>          /// <param name="stream">Stream value</param>          /// <returns>List of face object</returns>          public async Task<List<Face>> DetectFaces(Stream stream)          {              if (stream == null)              {                  throw new ArgumentNullException("stream");              }              IEnumerable<FaceAttributeType> faceAttributes =                  new FaceAttributeType[]                  {                      FaceAttributeType.Accessories,                      FaceAttributeType.Blur,                      FaceAttributeType.HeadPose,                      FaceAttributeType.Exposure,                      FaceAttributeType.FacialHair,                      FaceAttributeType.Glasses,                      FaceAttributeType.Gender,                      FaceAttributeType.Age,                      FaceAttributeType.Smile,                      FaceAttributeType.Emotion,                      FaceAttributeType.Makeup,                      FaceAttributeType.Hair,                      FaceAttributeType.Occlusion,                      FaceAttributeType.Noise                  };                try              {                  using (stream)                  {                      Face[] faces = await faceServiceClient.DetectAsync(stream, returnFaceId: true, returnFaceLandmarks: false, returnFaceAttributes: faceAttributes);                      return faces.ToList();                  }              }              catch (FaceAPIException faceAPIException)              {                  throw faceAPIException;              }              catch (Exception exception)              {                  throw exception;              }          }            /// <summary>          /// Train Faces          /// </summary>          /// <param name="trainFace">TrainFace object</param>          /// <returns>true/false</returns>          public async Task<bool> TrainFaces(TrainFace trainFace)          {              if (trainFace == null)              {                  throw new ArgumentNullException("trainFace");              }              PersonGroup personGroup = null;              try              {                  personGroup = await faceServiceClient.GetPersonGroupAsync(trainFace.PersonGroupID);              }              catch (Exception)              {                  personGroup = null;              }              if (personGroup == null)              {                  try                  {                      await faceServiceClient.CreatePersonGroupAsync(trainFace.PersonGroupID,                          trainFace.PersonGroupName);                      await faceServiceClient.GetPersonGroupAsync(trainFace.PersonGroupID);                  }                  catch (Exception exception)                  {                      throw exception;                  }              }              try              {                  CreatePersonResult person = await faceServiceClient.CreatePersonAsync(                      trainFace.PersonGroupID,                      trainFace.PersonName                  );                    foreach (string imagePath in Directory.GetFiles(trainFace.ImageDirectoryPath, trainFace.ImageFilterType))                  {                      FaceRectangle faceRectangle = null;                      using (Stream stream = File.OpenRead(imagePath))                      {                          Face[] faces = await faceServiceClient.DetectAsync(stream);                          if (faces != null && faces.Count() == 1)                          {                              faceRectangle = faces[0].FaceRectangle;                          }                      }                      if (faceRectangle != null)                      {                          using (Stream stream = File.OpenRead(imagePath))                          {                              // Detect faces in the image and add to Anna                              await faceServiceClient.AddPersonFaceAsync(                                  trainFace.PersonGroupID, person.PersonId, stream, null, faceRectangle);                          }                      }                  }                    await faceServiceClient.TrainPersonGroupAsync(trainFace.PersonGroupID);                    TrainingStatus trainingStatus = null;                  while (true)                  {                      trainingStatus = await faceServiceClient.                          GetPersonGroupTrainingStatusAsync(trainFace.PersonGroupID);                        if (trainingStatus.Status != Status.Running)                      {                          break;                      }                      await Task.Delay(2000);                  }              }              catch (FaceAPIException faceAPIException)              {                  throw faceAPIException;              }              catch (Exception exception)              {                  throw exception;              }              return true;          }            /// <summary>          /// Get Gender Chart          /// </summary>          /// <param name="faces">List of Face object</param>          /// <returns>GenderChart object</returns>          public GenderChart GetGenderChart(List<Face> faces)          {              if (faces == null || (faces != null && faces.Count() == 0))                  throw new ArgumentNullException("faces");              GenderChart genderChart = new GenderChart();              var noOfMales = faces.ToList().FindAll(f => f.FaceAttributes.Gender.ToLower().Equals("male")).ToList().Count;              var noOfFemales = faces.ToList().FindAll(f => f.FaceAttributes.Gender.ToLower().Equals("female")).ToList().Count;              genderChart.NoOfMale = noOfMales.ToString();              genderChart.NoOfFemale = noOfFemales.ToString();              return genderChart;          }            /// <summary>          /// AgeGroup Chart          /// </summary>          /// <param name="faces">List of Face object</param>          /// <returns></returns>          public AgeGroupChart AgeGroupChart(List<Face> faces)          {              if (faces == null || (faces != null && faces.Count() == 0))                  throw new ArgumentNullException("faces");              AgeGroupChart ageGroupChart = new AgeGroupChart();              var noOfAdolescence = faces.ToList().FindAll(f => f.FaceAttributes.Age < 18).ToList().Count;              var noOfYoungAdult = faces.ToList().FindAll(f => f.FaceAttributes.Age >= 18 && f.FaceAttributes.Age <= 35).ToList().Count;              var noOfMiddleAgeAdult = faces.ToList().FindAll(f => f.FaceAttributes.Age > 35 && f.FaceAttributes.Age <= 55).ToList().Count;              var noOfOldAdult = faces.ToList().FindAll(f => f.FaceAttributes.Age > 55).ToList().Count;              ageGroupChart.NoOfAdolescence = noOfAdolescence.ToString();              ageGroupChart.NoOfYoungAdult = noOfYoungAdult.ToString();              ageGroupChart.NoOfMiddleAgedAdult = noOfMiddleAgeAdult.ToString();              ageGroupChart.NoOfOldAdult = noOfOldAdult.ToString();              return ageGroupChart;          }            /// <summary>          /// Identify Face          /// </summary>          /// <param name="personGroupId">PersonGroupID value</param>          /// <param name="stream">Stream value</param>          /// <returns></returns>          public async Task<IdentifyFace> IdentifyFace(string personGroupId, Stream stream)          {              IdentifyFace identifyFace = new IdentifyFace();              if (string.IsNullOrWhiteSpace(personGroupId))              {                  throw new ArgumentNullException("personGroupId");              }              if (stream == null)              {                  throw new ArgumentNullException("stream");              }              try              {                  using (stream)                  {                      var faces = await DetectFaces(stream);                      var faceIds = faces.Select(face => face.FaceId).ToArray();                        var results = await faceServiceClient.IdentifyAsync(personGroupId, faceIds);                      foreach (var identifyResult in results)                      {                          if (identifyResult.Candidates.Length != 0)                          {                              var candidateId = identifyResult.Candidates[0].PersonId;                              var person = await faceServiceClient.GetPersonAsync(personGroupId, candidateId);                              identifyFace.IsFound = true;                              identifyFace.PersonName = person.Name;                              break;                          }                      }                  }              }              catch (FaceAPIException faceAPIException)              {                  throw faceAPIException;              }              catch (Exception exception)              {                  throw exception;              }              return identifyFace;          }            #endregion      }  } |

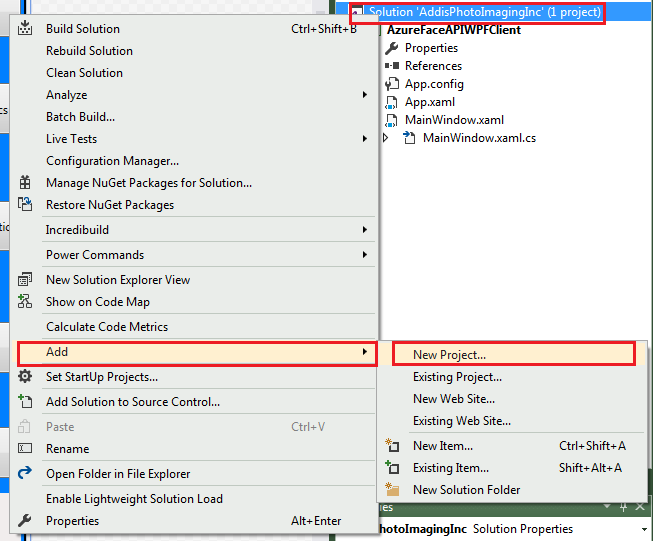
1. Open MainWindows.xaml.cs file put the following code and replace the existing code with the following code.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.IO;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows;  using System.Windows.Input;  using System.Windows.Media;  using System.Windows.Media.Imaging;  using Microsoft.ProjectOxford.Common.Contract;  using Microsoft.ProjectOxford.Face;  using Microsoft.ProjectOxford.Face.Contract;  using System.Net.Http;  using Newtonsoft.Json;  using System.Configuration;  using Microsoft.Win32;  using System.Windows.Forms;  using System.Web;      using AzureFaceAPISample.Entities;  using Common;    namespace AzureFaceAPIWPFClient  {      public partial class MainWindow : Window      {          #region Properties            List<Face> faces;          string[] faceDescriptions;          double resizeFactor;          BitmapImage bitmapSource = new BitmapImage();          AzureFaceAPIProcessor AzureFaceAPIProcessor = null;            #endregion            #region Constructor            public MainWindow()          {              InitializeComponent();              AzureFaceAPIProcessor = new AzureFaceAPIProcessor(ConfigurationManager.AppSettings["AzureFaceAPISubscritionKey"], ConfigurationManager.AppSettings["AzureFaceAPISubscritionURL"]);          }            #endregion            #region WPF Control Events            private void FacePhoto\_MouseMove(object sender, System.Windows.Input.MouseEventArgs e)          {              if (faces == null)                  return;                Point mouseXY = e.GetPosition(FacePhoto);                ImageSource imageSource = FacePhoto.Source;              BitmapSource bitmapSource = (BitmapSource)imageSource;                var scale = FacePhoto.ActualWidth / (bitmapSource.PixelWidth / resizeFactor);                bool mouseOverFace = false;                for (int i = 0; i < faces.Count; ++i)              {                  FaceRectangle fr = faces[i].FaceRectangle;                  double left = fr.Left \* scale;                  double top = fr.Top \* scale;                  double width = fr.Width \* scale;                  double height = fr.Height \* scale;                    if (mouseXY.X >= left && mouseXY.X <= left + width && mouseXY.Y >= top && mouseXY.Y <= top + height)                  {                      faceDescriptionStatusBar.Text = faceDescriptions[i];                      mouseOverFace = true;                      break;                  }              }                if (!mouseOverFace)                  faceDescriptionStatusBar.Text = "Place the mouse pointer over a face to see the face description.";          }            private async void TrainFacesButton\_Click(object sender, RoutedEventArgs e)          {              try              {                  var openDlg = new FolderBrowserDialog();                    var result = openDlg.ShowDialog();                    if (result != System.Windows.Forms.DialogResult.OK)                  {                      return;                  }                    Uri fileUri = new Uri(openDlg.SelectedPath);                  TrainFace trainFace = new TrainFace()                  {                      ImageDirectoryPath = openDlg.SelectedPath,                      PersonGroupID = ConfigurationManager.AppSettings["PersonGroupID"],                      PersonGroupName = ConfigurationManager.AppSettings["PersonGroupName"],                      PersonName = ConfigurationManager.AppSettings["PersonName"],                  };                    bool trainCompleted = await AzureFaceAPIProcessor.TrainFaces(trainFace);                    if (trainCompleted)                  {                      faceDescriptionStatusBar.Text = "Face Training is completed !";                  }                  openDlg.Dispose();                }              catch (Exception exception)              {                  System.Windows.MessageBox.Show(exception.Message, Title, MessageBoxButton.OK, MessageBoxImage.Error);              }          }            private async void GenderChartButton\_Click(object sender, RoutedEventArgs e)          {              try              {                  GenderChart genderChart = AzureFaceAPIProcessor.GetGenderChart(faces);                  await PublishSignalREvent(genderChart, EventName.ON\_CHART\_PRODUCED);              }              catch (Exception exception)              {                  await PublishExceptionEvent(exception);              }          }            private async void AgeGroupChartButton\_Click(object sender, RoutedEventArgs e)          {              try              {                  AgeGroupChart ageGroupChart = AzureFaceAPIProcessor.AgeGroupChart(faces);                  await PublishSignalREvent(ageGroupChart, EventName.ON\_CHART\_PRODUCED);              }              catch (Exception exception)              {                  await PublishExceptionEvent(exception);              }          }            private async void DetectFacesButton\_Click(object sender, RoutedEventArgs e)          {              try              {                  string filePath = BrowseAndLoadImage();                  if (string.IsNullOrWhiteSpace(filePath))                  {                      return;                  }                  Stream stream = File.OpenRead(filePath);                  faceDescriptionStatusBar.Text = "Detecting...";                  faces = await AzureFaceAPIProcessor.DetectFaces(stream);                  faceDescriptionStatusBar.Text = String.Format("Detection Finished. {0} face(s) detected", faces.Count);                    if (faces.Count > 0)                  {                      DrawingVisual visual = new DrawingVisual();                      DrawingContext drawingContext = visual.RenderOpen();                      drawingContext.DrawImage(bitmapSource,                          new Rect(0, 0, bitmapSource.Width, bitmapSource.Height));                      double dpi = bitmapSource.DpiX;                      resizeFactor = 96 / dpi;                      faceDescriptions = new string[faces.Count];                        for (int index = 0; index < faces.Count; ++index)                      {                          Face face = faces[index];                          string json = JsonConvert.SerializeObject(face);                          drawingContext.DrawRectangle(                              Brushes.Transparent,                              new Pen(Brushes.GreenYellow, 3),                              new Rect(                                  face.FaceRectangle.Left \* resizeFactor,                                  face.FaceRectangle.Top \* resizeFactor,                                  face.FaceRectangle.Width \* resizeFactor,                                  face.FaceRectangle.Height \* resizeFactor));                            faceDescriptions[index] = FaceDescription(face);                      }                        drawingContext.Close();                        RenderTargetBitmap faceWithRectBitmap = new RenderTargetBitmap(                          (int)(bitmapSource.PixelWidth \* resizeFactor),                          (int)(bitmapSource.PixelHeight \* resizeFactor), 96, 96,                          PixelFormats.Pbgra32);                        faceWithRectBitmap.Render(visual);                      FacePhoto.Source = faceWithRectBitmap;                        faceDescriptionStatusBar.Text = "Hover mouse over a face to see the face description.";                  }              }              catch (Exception exception)              {                  await PublishExceptionEvent(exception);              }          }            private async void IdentifyFace\_Click(object sender, RoutedEventArgs e)          {              try              {                  string filePath = BrowseAndLoadImage();                  if (string.IsNullOrWhiteSpace(filePath))                  {                      return;                  }                  Stream stream = File.OpenRead(filePath);                  IdentifyFace identifyFace = await AzureFaceAPIProcessor.IdentifyFace(ConfigurationManager.AppSettings["PersonGroupID"], stream);                  if (identifyFace.IsFound)                  {                      stream = File.OpenRead(filePath);                      BinaryReader binaryReader = new BinaryReader(stream);                      byte[] bytes = binaryReader.ReadBytes((Int32)stream.Length);                      string base64ImageString = Convert.ToBase64String(bytes, 0, bytes.Length);                      int startIndex = 0;                      int amount = 1500; // Publish 1.5K of base64Image string                      do                      {                          if (base64ImageString.Length - startIndex < amount)                          {                              identifyFace.Base64ImageString = base64ImageString.Substring(startIndex).Replace("+", "\_").Replace(" ", string.Empty);                              identifyFace.IsCompleted = true;                          }                          else                          {                              identifyFace.Base64ImageString = base64ImageString.Substring(startIndex, amount).Replace("+", "\_").Replace(" ", string.Empty);                              startIndex = startIndex + amount;                          }                          await PublishSignalREvent(identifyFace, EventName.ON\_PHOTO\_DETECTED);                      } while (identifyFace.Base64ImageString.Length == amount);                    }              }              catch (Exception exception)              {                  await PublishExceptionEvent(exception);              }          }            #endregion            #region Private Methods            private string BrowseAndLoadImage()          {              try              {                  // Get the image file to scan from the user.                  var openDlg = new Microsoft.Win32.OpenFileDialog();                    openDlg.Filter = "JPEG Image(\*.jpg)|\*.jpg";                  bool? result = openDlg.ShowDialog(this);                    // Return if canceled.                  if (!(bool)result)                  {                      return string.Empty;                  }                    bitmapSource = new BitmapImage();                  bitmapSource.BeginInit();                  bitmapSource.CacheOption = BitmapCacheOption.None;                  bitmapSource.UriSource = new Uri(openDlg.FileName);                  bitmapSource.EndInit();                    FacePhoto.Source = bitmapSource;                  faceDescriptionStatusBar.Text = "Loading image completed !";                  return openDlg.FileName;              }              catch (Exception exception)              {                  throw exception;              }          }            private string FaceDescription(Face face)          {              StringBuilder stringBuilder = new StringBuilder();                stringBuilder.Append("Face: ");                stringBuilder.Append(face.FaceAttributes.Gender);              stringBuilder.Append(", ");              stringBuilder.Append(face.FaceAttributes.Age);              stringBuilder.Append(", ");              stringBuilder.Append(String.Format("Smile {0:F1}%, ", face.FaceAttributes.Smile \* 100));                stringBuilder.Append("Emotion: ");              EmotionScores emotionScores = face.FaceAttributes.Emotion;              if (emotionScores.Anger >= 0.1f) stringBuilder.Append(string.Format("Anger {0:F1}%, ", emotionScores.Anger \* 100));              if (emotionScores.Contempt >= 0.1f) stringBuilder.Append(string.Format("Contempt {0:F1}%, ", emotionScores.Contempt \* 100));              if (emotionScores.Disgust >= 0.1f) stringBuilder.Append(string.Format("Disgust {0:F1}%, ", emotionScores.Disgust \* 100));              if (emotionScores.Fear >= 0.1f) stringBuilder.Append(string.Format("Fear {0:F1}%, ", emotionScores.Fear \* 100));              if (emotionScores.Happiness >= 0.1f) stringBuilder.Append(string.Format("Happiness {0:F1}%, ", emotionScores.Happiness \* 100));              if (emotionScores.Neutral >= 0.1f) stringBuilder.Append(string.Format("Neutral {0:F1}%, ", emotionScores.Neutral \* 100));              if (emotionScores.Sadness >= 0.1f) stringBuilder.Append(string.Format("Sadness {0:F1}%, ", emotionScores.Sadness \* 100));              if (emotionScores.Surprise >= 0.1f) stringBuilder.Append(string.Format("Surprise {0:F1}%, ", emotionScores.Surprise \* 100));                stringBuilder.Append(face.FaceAttributes.Glasses);              stringBuilder.Append(", ");                stringBuilder.Append("Hair: ");                if (face.FaceAttributes.Hair.Bald >= 0.01f)                  stringBuilder.Append(String.Format("Bald {0:F1}% ", face.FaceAttributes.Hair.Bald \* 100));                HairColor[] hairColors = face.FaceAttributes.Hair.HairColor;              foreach (HairColor hairColor in hairColors)              {                  if (hairColor.Confidence >= 0.1f)                  {                      stringBuilder.Append(hairColor.Color.ToString());                      stringBuilder.Append(String.Format(" {0:F1}% ", hairColor.Confidence \* 100));                  }              }              return stringBuilder.ToString();          }            private async Task<bool> PublishSignalREvent<T>(T type, EventName EventName) where T :              class          {              HttpClient httpClient = new HttpClient();              try              {                    string JSONify = JsonConvert.SerializeObject(type);                  string url = string.Concat(ConfigurationManager.AppSettings["RESTfulSignalRServiceURL"], "?message=", JSONify, "&eventName=", EventName.EnumDescription());                  var responseEntity = await httpClient.PostAsJsonAsync(url, JSONify);                  if (!responseEntity.IsSuccessStatusCode)                  {                      throw new Exception(string.Format("Http response Status Code : {0}, ReasonPhrase : {1}", (double)responseEntity.StatusCode, responseEntity.ReasonPhrase));                  }              }              catch (Exception exception)              {                  throw exception;              }              return true;          }            private async Task PublishExceptionEvent(Exception exception)          {              ErrorResponse errorResponse = new ErrorResponse()              {                  ErrorMessage = string.Concat("Error occurred while processing Azure Face API. Message : ", exception.Message)              };              try              {                  await PublishSignalREvent(errorResponse, EventName.ON\_EXCEPTION);              }              catch (Exception publishException)              {                  System.Windows.MessageBox.Show(publishException.Message, Title, MessageBoxButton.OK, MessageBoxImage.Error);              }          }            #endregion        }  } |

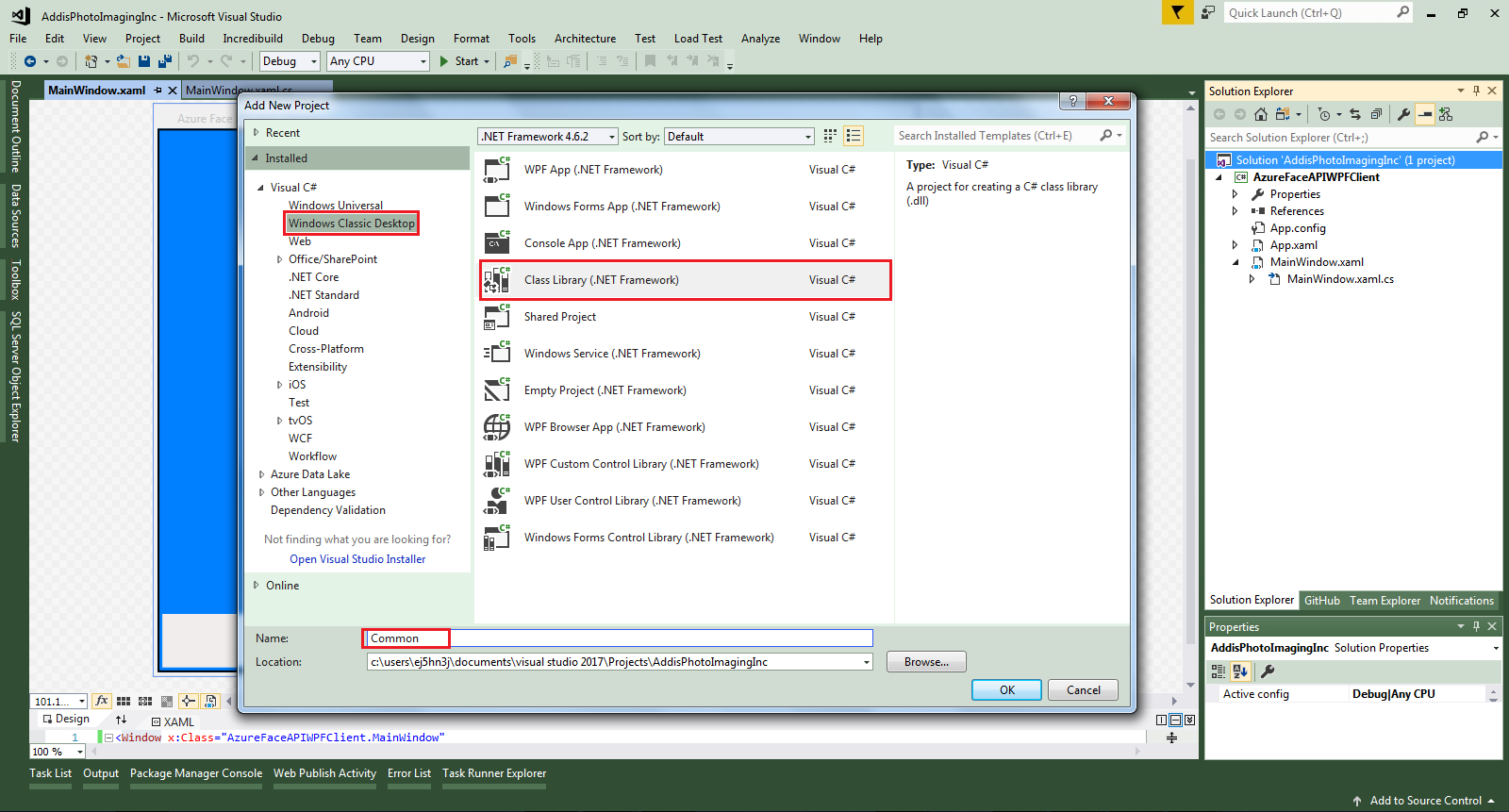
1. Open App.Config file and replace the content with the following code. Modify the yellow highlighted section. See Step 1.e and 1.f

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <configuration>    <appSettings>      <add key="AzureFaceAPISubscritionKey" value="cb1f08e4eb6e4293ae19456aa6933f6e" />      <add key="AzureFaceAPISubscritionURL" value="https://eastus.api.cognitive.microsoft.com/face/v1.0" />      <add key="RESTfulSignalRServiceURL" value="http://realtimemessagebrokerapi.azurewebsites.net/messagebroadcast/BroadCast" />      <add key="PersonGroupID" value="wondegroup" />      <add key="PersonGroupName" value="Wonde Photos" />      <add key="PersonName" value="Wonde" />    </appSettings>    <startup>      <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.6.2" />    </startup>    <runtime>      <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">        <dependentAssembly>          <assemblyIdentity name="Newtonsoft.Json" publicKeyToken="30ad4fe6b2a6aeed" culture="neutral" />          <bindingRedirect oldVersion="0.0.0.0-10.0.0.0" newVersion="10.0.0.0" />        </dependentAssembly>      </assemblyBinding>    </runtime>  </configuration> |

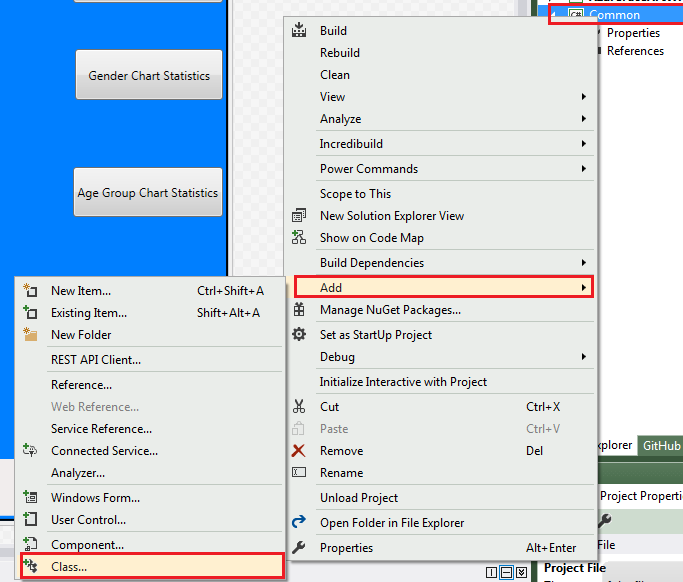
1. Right click under Solution ‘AddisPhotoImagincInc’ and add another project called ‘Common’ on the solution. See the picture below.



1. A dialog form will show which is similar like the image shown on step 2.



1. Add the following classes on the ‘Common’ project.
   1. Delete the default create class called ‘Class1.cs’. To delete, right click the ‘Class1.cs’ file and select Delete.
   2. To add new class, see the following picture.



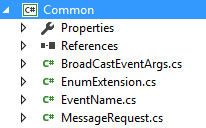
* 1. Add New Item dialog will appear and put ‘EnumName’ and press add.



* 1. Apply the following content on ‘EventName.cs’

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.ComponentModel;    namespace Common  {      /// <summary>      /// SignalR EventName enumeration.      /// Note : The Enum Description should match the defined SignalR method      /// </summary>      public enum EventName      {          /// <summary>          /// Unknown enum          /// </summary>          UNKNOWN = 0,            /// <summary>          /// On Message Listened(onMessageListened) enum value. For general purpose          /// </summary>          [Description("onMessageListened")]          ON\_MESSAGE\_LISTENED = 1,            /// <summary>          /// On Chart Produced(onChartProduced) enum value.          /// </summary>          [Description("onChartProduced")]          ON\_CHART\_PRODUCED = 2,            /// <summary>          /// On Photo Detected(onPhotoDetected) enum value.          /// </summary>          [Description("onPhotoDetected")]          ON\_PHOTO\_DETECTED = 3,            /// <summary>          /// On Exception(onException) enum value.          /// </summary>          [Description("onException")]          ON\_EXCEPTION = 4      }  } |

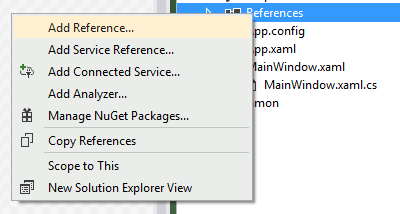
* 1. Please add similar way the following classes on the ‘Common’ project.



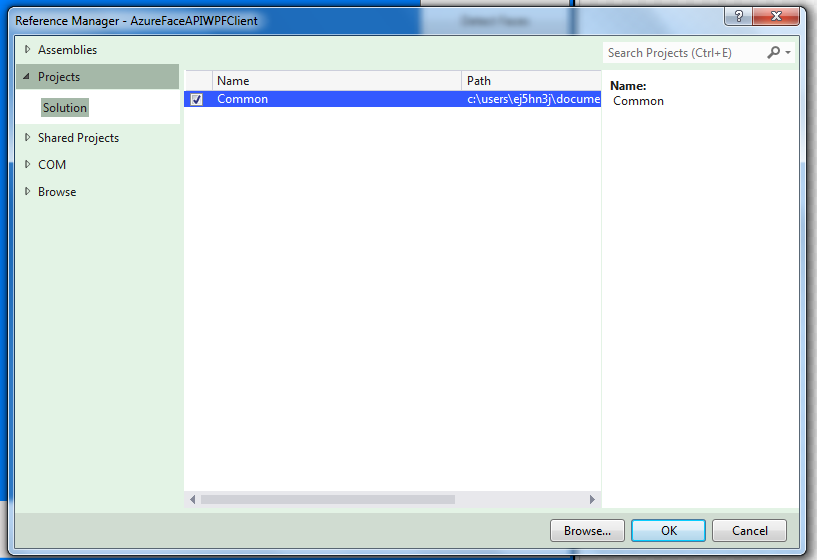
Note: The content of each class is located on GitHub.

1. Add reference of ‘Common’ project in to ‘AzureFaceAPIClient’ project.

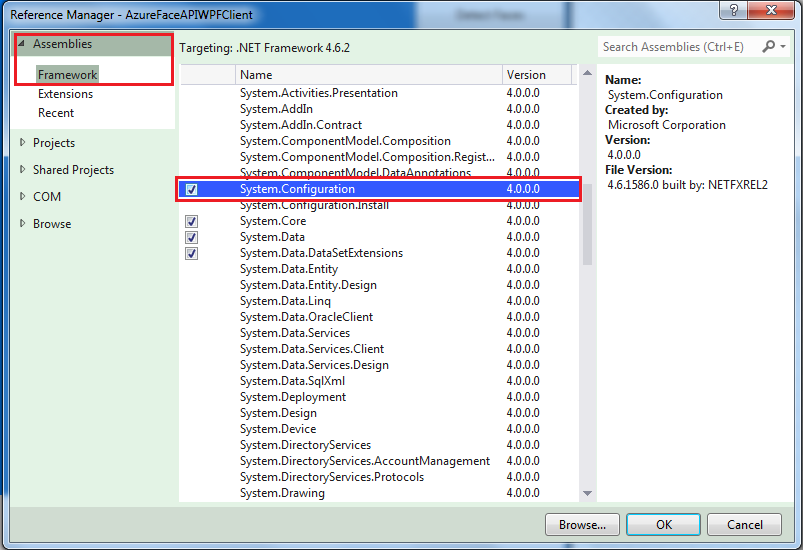
To add reference, right click under reference section and select ‘Add Reference…’ see the following picture.



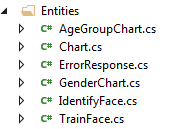
And Project, ‘Common’ check box and press Ok button.



1. On ‘AzureFaceAPIWPFClient’ project, Repeat Step 11 but select the ‘Assemblies’, ‘Framework’ and add ‘System.Configuration’



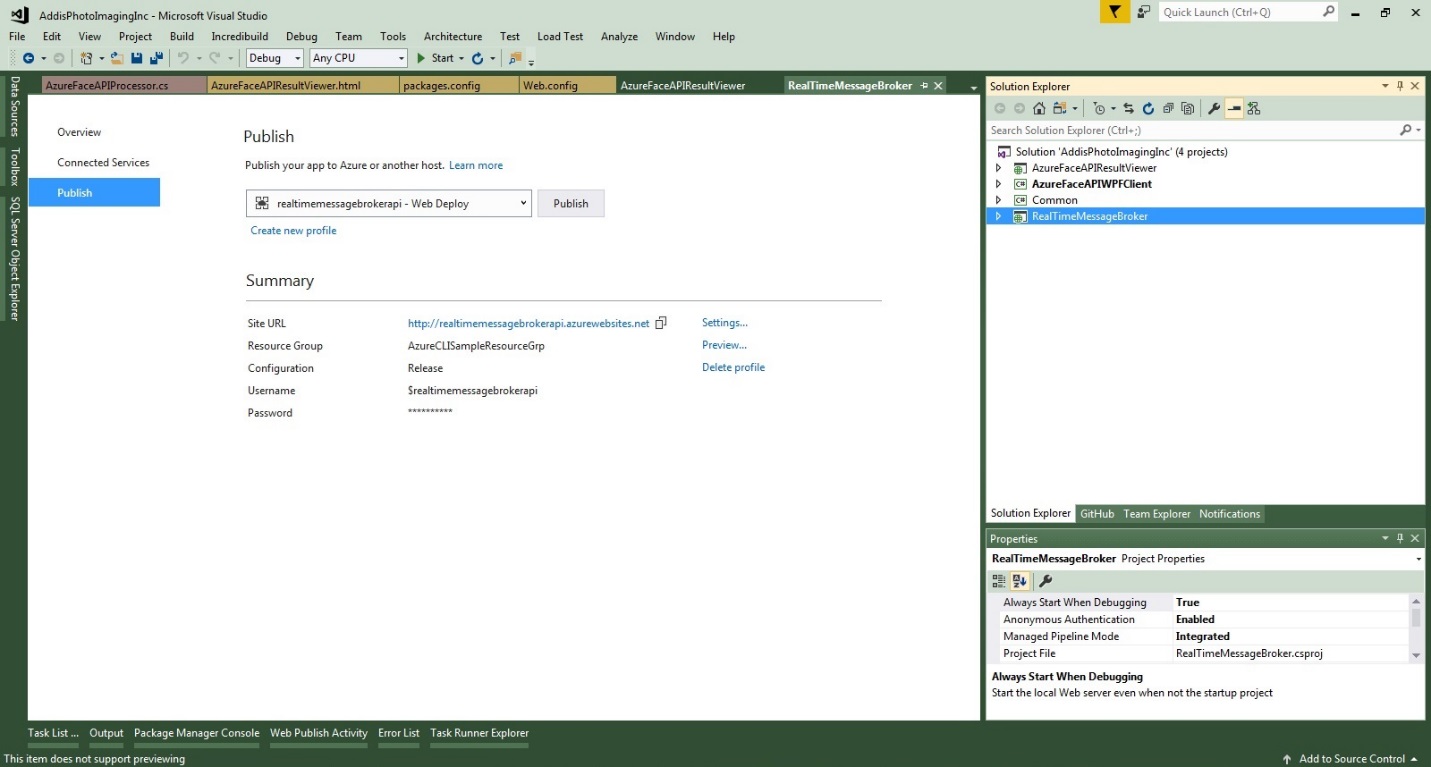
1. On ‘AzureFaceAPIWPFClient’ project add the following classes/enums under a folder name ‘Entities’



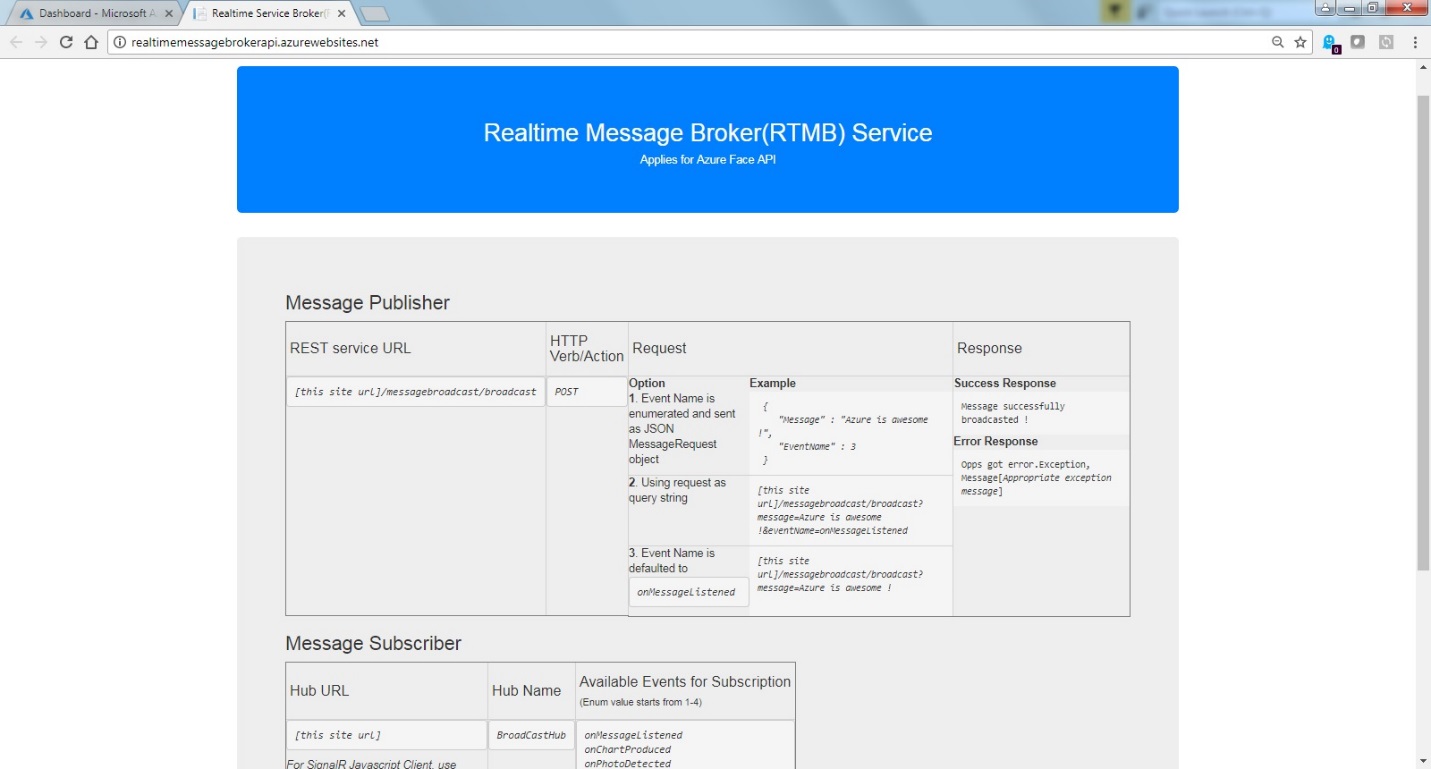
Note: The content of each class is located on GitHub.

1. Download from GitHub source code location and add the following projects into your solutions, RealTimeMessageBroker project.

Note: The project is already published on Azure portal.



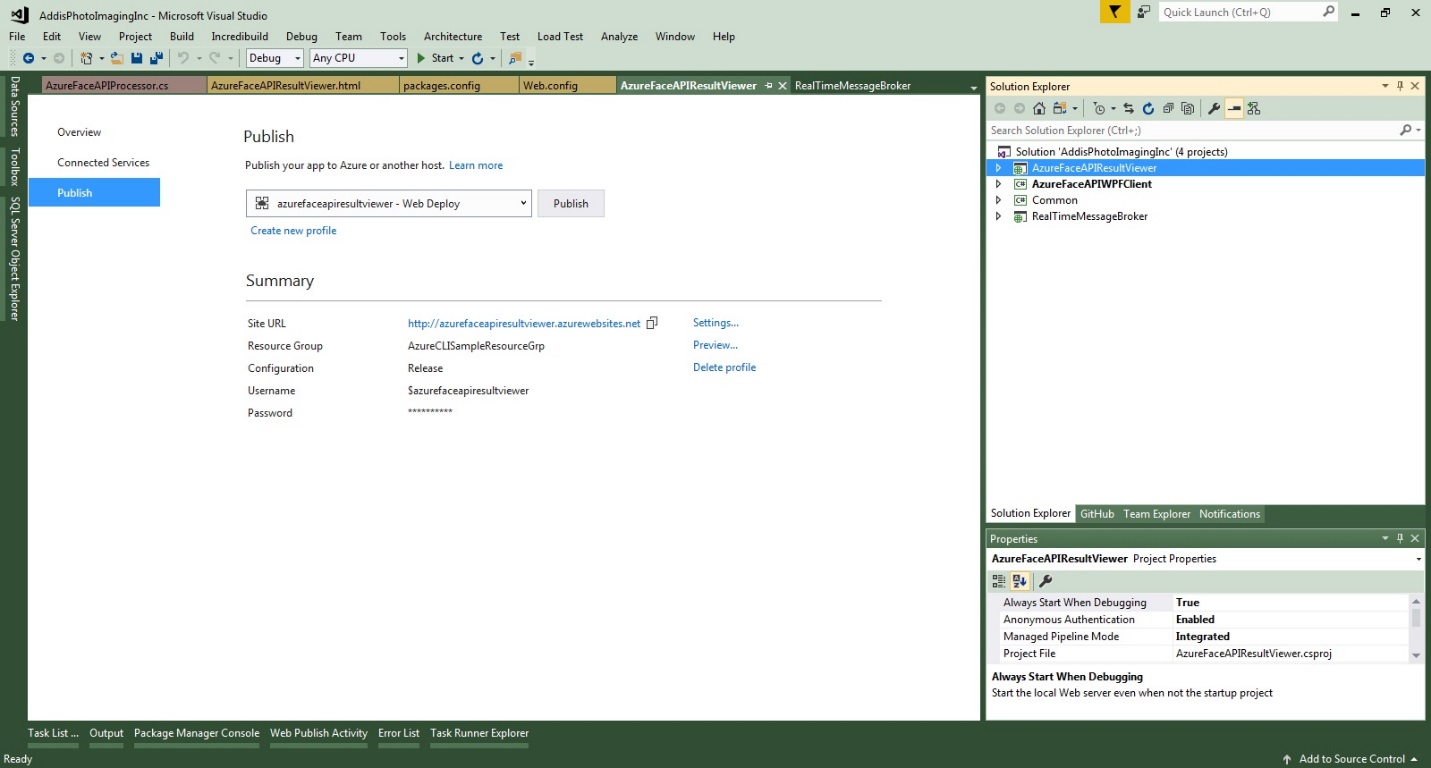
URL - <http://realtimemessagebrokerapi.azurewebsites.net/>



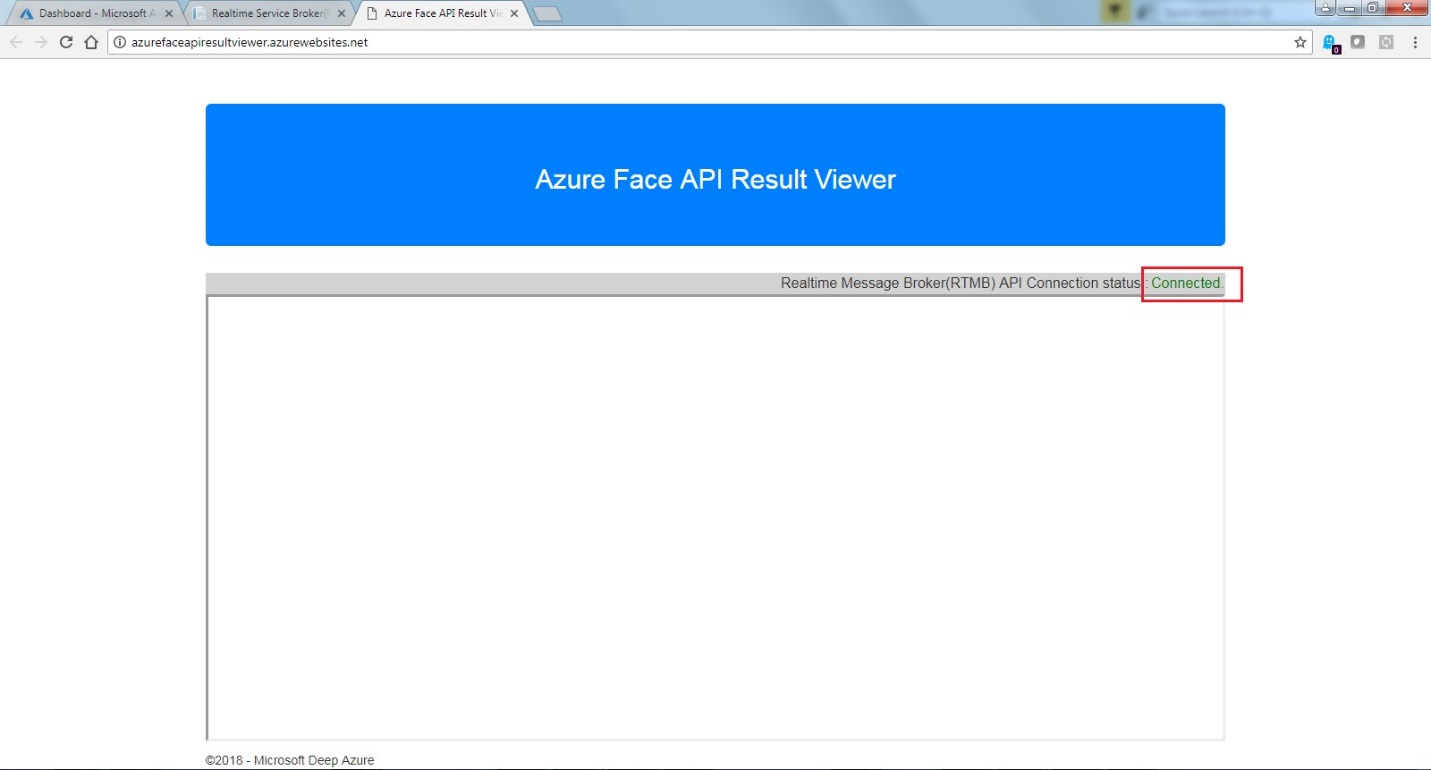
Please see the GitHub file to get and understand the code. It’s a bit long to put in the document.

1. Download from GitHub source code location and add the following project into your solution. AzureFaceAPIResultViewer project.

Note: The project is already published on Azure portal.



URL - <http://azurefaceapiresultviewer.azurewebsites.net/>

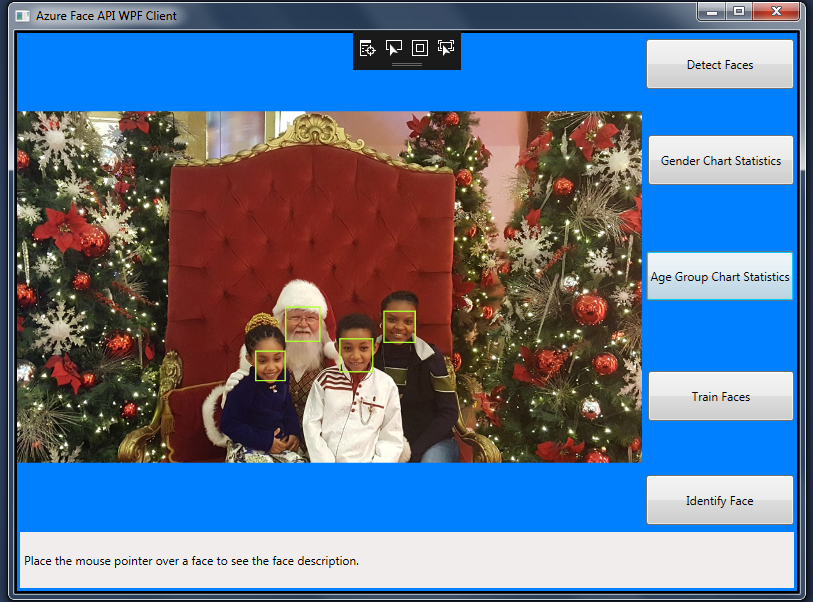


AzureFaceAPIResultViewer.html page is shown of the code can is shown below.

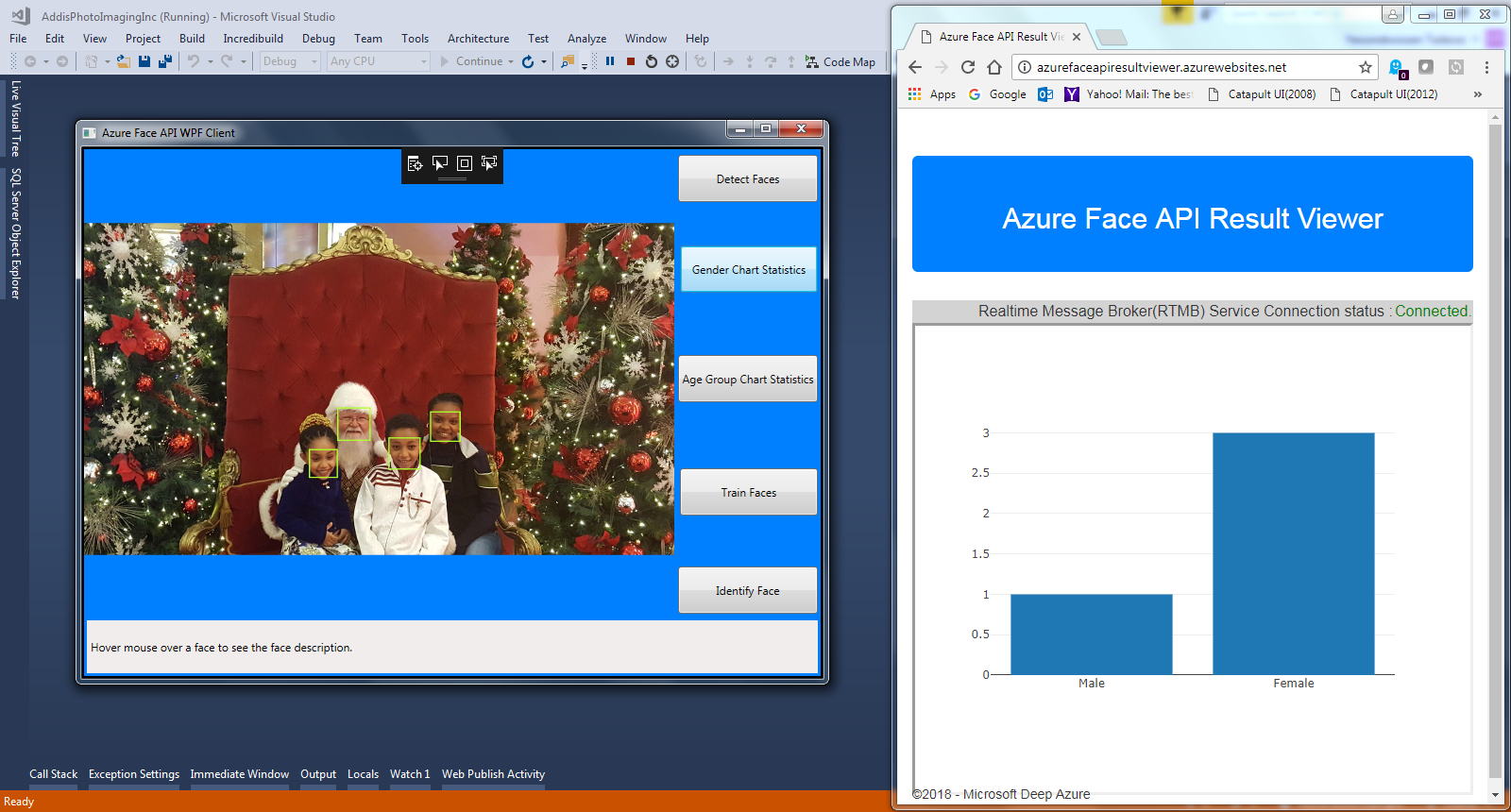
|  |
| --- |
| <!DOCTYPE html>  <html xmlns="http://www.w3.org/1999/xhtml">  <head>      <title>Azure Face API Result Viewer</title>      <link rel="stylesheet" type="text/css" href="Content/Site.css">      <link rel="stylesheet" type="text/css" href="Content/bootstrap.css">      <link rel="stylesheet" type="text/css" href="Content/bootstrap.min.css">      <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>      <script src="Scripts/jquery-1.6.4.min.js"></script>      <script src="Scripts/jquery.signalR-1.0.1.min.js"></script>      <script src="/signalR"></script>      <script type="text/javascript">          $(document).ready(function () {              $("#chart").attr("hidden", "hidden");              $("#photoDiv").attr("hidden", "hidden");              $("#exceptionContainer").attr("hidden", "hidden");              var connection = $.hubConnection('http://realtimemessagebrokerapi.azurewebsites.net/');              connection.start().done(function () {                  $("#status").text("").prepend("Connected.").css("color", "green");              });                var messageBroadCastHub = connection.createHubProxy('BroadCastHub');                // Subscribe onChartProduced SignalR event              processOnChartProducedEvent(messageBroadCastHub);                // Subscribe onPhotoDetected SignalR event              processOnPhotoDetectedEvent(messageBroadCastHub);                // Subscribe onException SignalR event              processOnExceptionEvent(messageBroadCastHub);                var today = new Date();              var year = today.getFullYear();              $('#copyright').html('&copy;').append(year).append(' - Microsoft Deep Azure');          });            // Subscribe onChartProduced SignalR event          function processOnChartProducedEvent(messageBroadCastHub) {              var eventName = 'onChartProduced';              messageBroadCastHub.on(eventName, function (message) {                  var jsonObject = JSON.parse(message);                    if (jsonObject.NoOfMale != undefined &&                      jsonObject.NoOfFemale != undefined) {                      plotGender(jsonObject);                  }                  else if (jsonObject.NoOfAdolescence != undefined &&                      jsonObject.NoOfYoungAdult != undefined &&                      jsonObject.NoOfMiddleAgedAdult != undefined &&                      jsonObject.NoOfOldAdult != undefined) {                      plotAgeGroup(jsonObject);                  }              });          }            // Subscribe onPhotoDetected SignalR event          function processOnPhotoDetectedEvent(messageBroadCastHub) {              var eventName = 'onPhotoDetected';              messageBroadCastHub.on(eventName, function (message) {                  var jsonObject = JSON.parse(message);                  showIdentifiedFace(jsonObject);              });          }            // Subscribe onException SignalR event          function processOnExceptionEvent(messageBroadCastHub) {              var eventName = 'onException';              messageBroadCastHub.on(eventName, function (message) {                  var jsonObject = JSON.parse(message);                  $("#exceptionContainer").removeAttr("hidden");                  $("#exceptionContainer").text("").prepend(jsonObject.ErrorMessage);                  $("#photoDiv").attr("hidden", "hidden");                  $("#chart").attr("hidden", "hidden");              });          }            // Plot Gender chart          function plotGender(genderChart) {              $("#chart").removeAttr("hidden");              $("#exceptionContainer").attr("hidden", "hidden");              $("#photoDiv").attr("hidden", "hidden");              var data = [                  {                      x: ["Male", "Female"],                      y: [genderChart.NoOfMale, genderChart.NoOfFemale],                      type: genderChart.Type                  }              ];              Plotly.newPlot('chart', data);          }            //Plot Age Group chart          function plotAgeGroup(ageGroupChart) {              $("#chart").removeAttr("hidden");              $("#exceptionContainer").attr("hidden", "hidden");              $("#photoDiv").attr("hidden", "hidden");              var data = [                  {                      labels: ["Adolescence", "Young Adult", "Middle Aged Adult", "Old Adult"],                      values: [ageGroupChart.NoOfAdolescence, ageGroupChart.NoOfYoungAdult, ageGroupChart.NoOfMiddleAgedAdult, ageGroupChart.NoOfOldAdult],                      type: ageGroupChart.Type                  }              ];              Plotly.newPlot('chart', data);          }            // Show identified face          function showIdentifiedFace(identifyFace) {              $("#chart").attr("hidden", "hidden");              $("#photoDiv").attr("hidden", "hidden");              $("#exceptionContainer").attr("hidden", "hidden");              if (identifyFace.IsFound != undefined &&                  identifyFace.IsFound == true) {                  var imgValue = identifyFace.Base64ImageString;                  $("#imageContainer").append(imgValue.replace(new RegExp('\_', 'g'), "+")); // replace any "\_" with "+"                    if (identifyFace.IsCompleted == true) {                      $("#photoDiv").removeAttr("hidden");                      var imageSource = "data:image/png;base64, " + $("#imageContainer").val();                      $("#photo").attr("src", imageSource);                      $("#photoName").text("").append(identifyFace.PersonName).css("font-weight", "Bold").append(" is identified !");                      $("#imageContainer").text("");                  }              }          }      </script>  </head>  <body class="container body-content" style="background-color:white">      <div class="jumbotron" style="background-color:#007FFF">          <h2 style="color:white;text-align:center">Azure Face API Result Viewer</h2>      </div>      <div style="font-size:medium;background-color:lightgray;">          <table style="width: auto;margin-right: 0px;margin-left: auto;">              <tr>                  <td style="vertical-align:top;">Realtime Message Broker(RTMB) API Connection status : </td>                  <td>                      <div id="status" style="color:red">Not Connected</div>                  </td>              </tr>          </table>      </div>      <div style="border:inset;width:100%;height:500px">          <textarea id="imageContainer" hidden="hidden"></textarea>          <div id="exceptionContainer" style="color:red;font-size:medium"></div>          <div id="photoDiv">              <table>                  <tr>                      <td style="vertical-align:top">                          <img id="photo"/>                      </td>                      <td style="vertical-align:top">                          <div id="photoName" style="color:green"></div>                      </td>                  </tr>              </table>          </div>          <div id="chart"></div>      </div>      <div id="copyright" style="text-align:center;position:fixed;bottom:0;" />  </body>  </html> |

Please see the GitHub file to get and understand the code. It’s a bit long to put in the document.

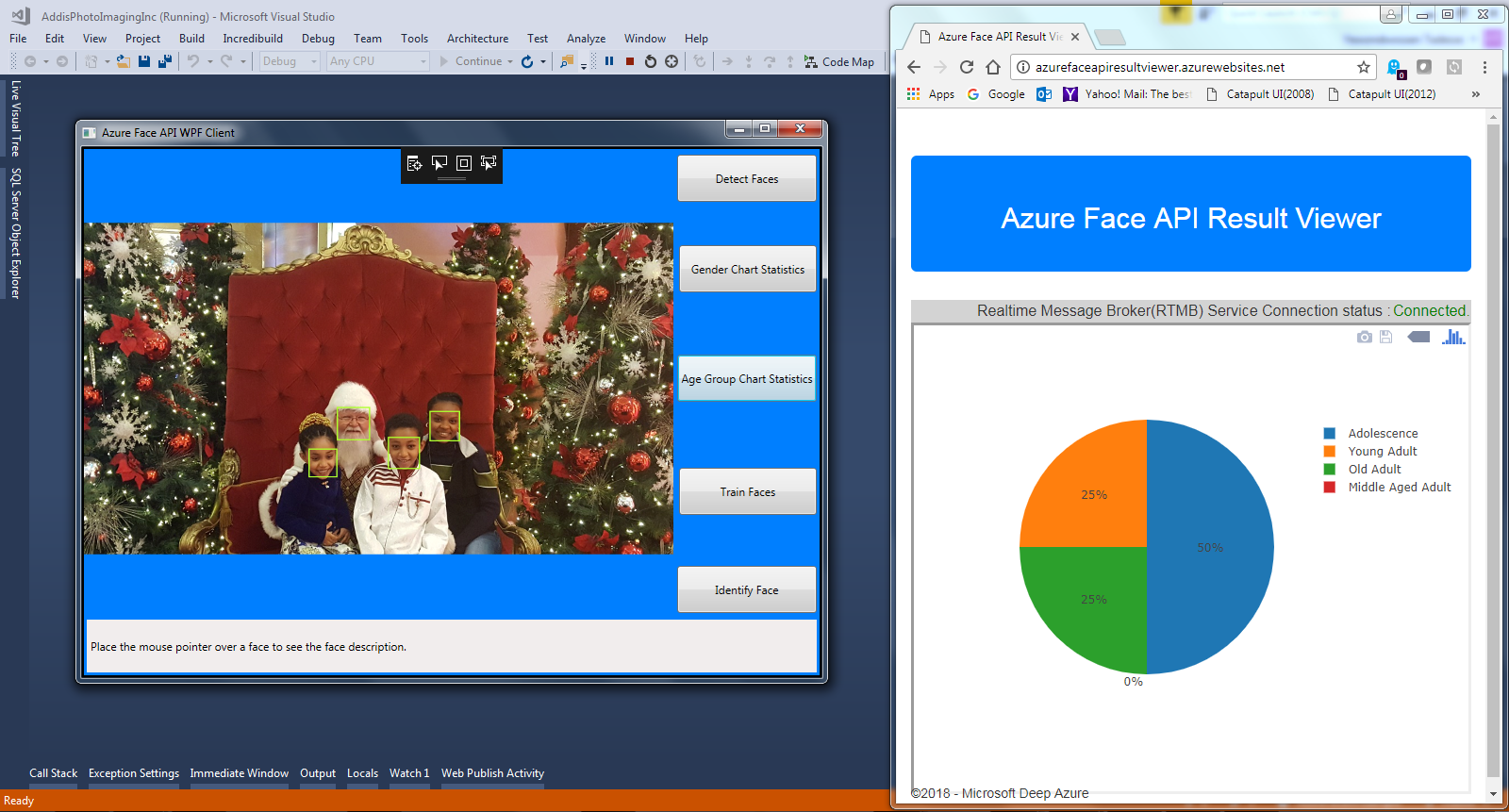
1. After build and running the application.
   1. After ‘Detect Faces’ button pressed and photo is selected



* 1. After photo is detected and ‘Gender Chart Statistics’ button pressed



* 1. After photo is detected and ‘Age Group Chart Statistics’ button pressed



* 1. After photos are trained and another photo is identified.

