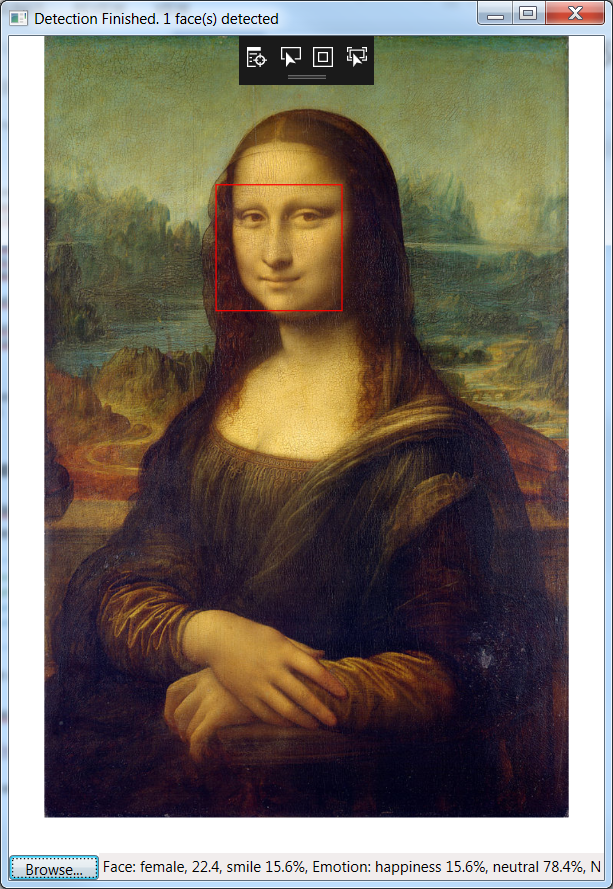
# Face API in C#



A Windows application that uses Microsoft’s Cloud based algorithms - Face API. The application detects faces in an image, draws a frame around each face and displays a description of the face on the status bar.

# Internet Access

You need internet access so make sure you are registered for

CSSWFI (via text registration) or

BPV-DIGITAL (pw 4CC355D1G1T4L)

Not ADC-230 because of the port configuration

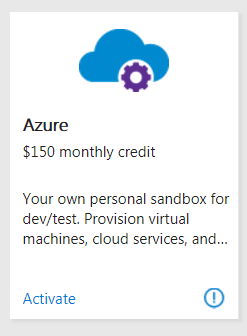
# Register For Azure

Skip this section if you have an Azure login already

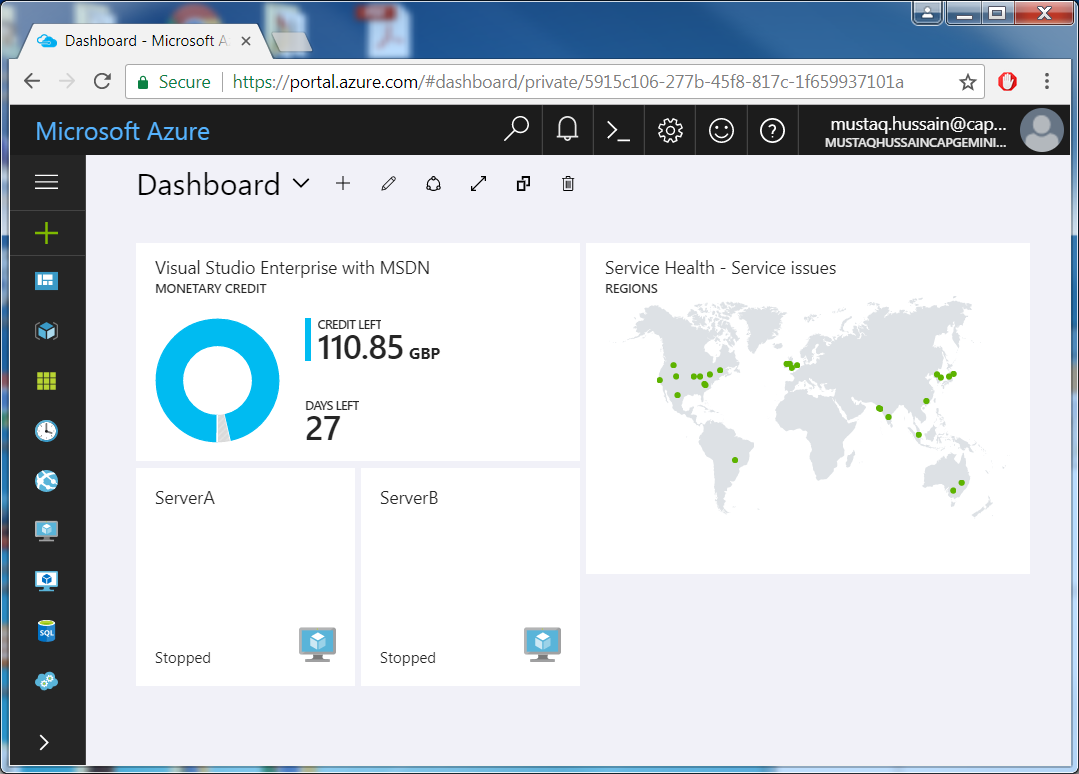
1. Login to – using your Capgemini email (which is tied in to your MSDN subscription)

http://my.visualstudio.com

1. Click under Azure Activate



1. Follow the registration process. Eventually you will get the Azure Dashboard



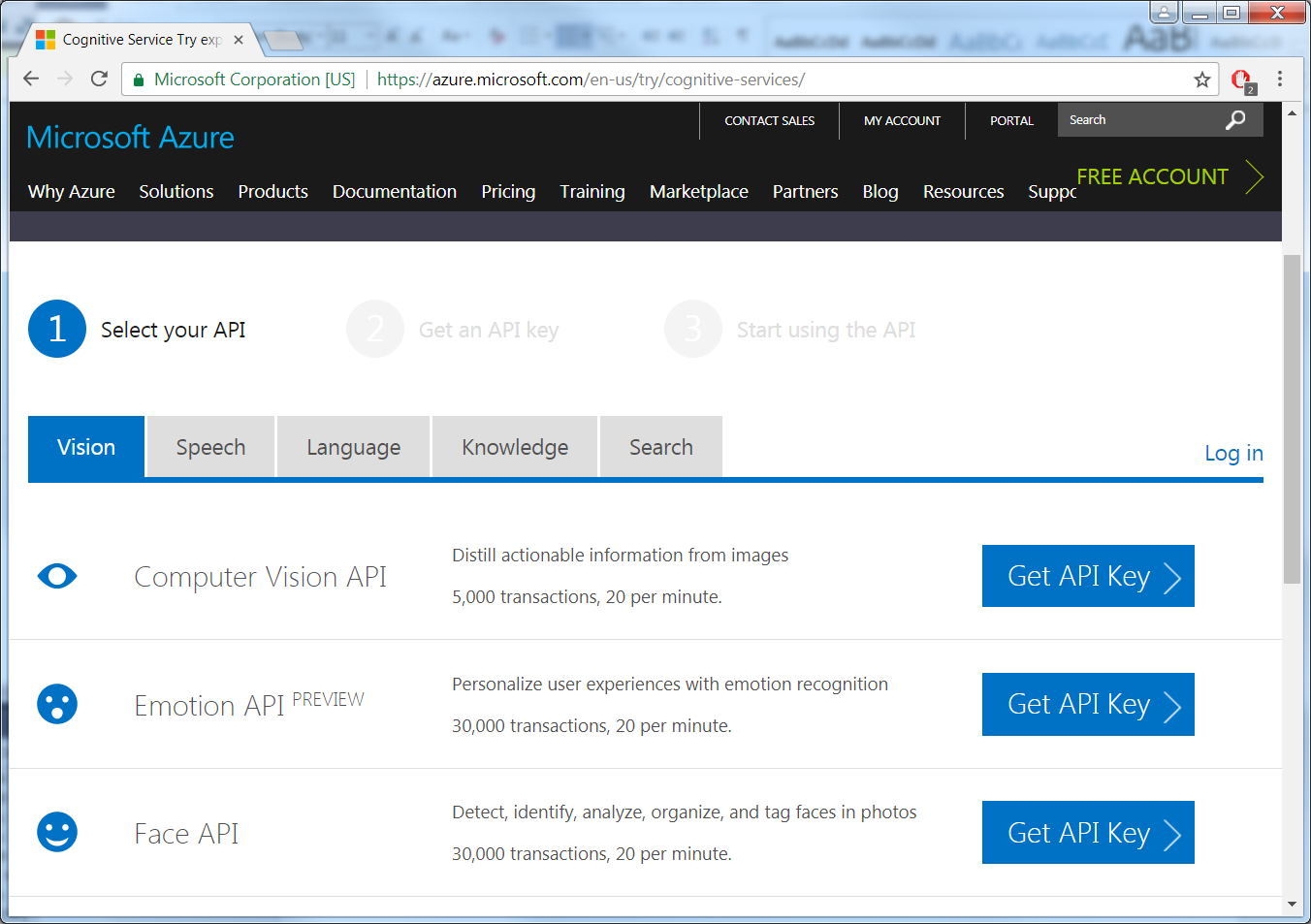
You have now successfully registered on Azure – you will have a monthly allowance of £115.

Note if you spend more than £115 in a month your Azure services will be suspended until the following month

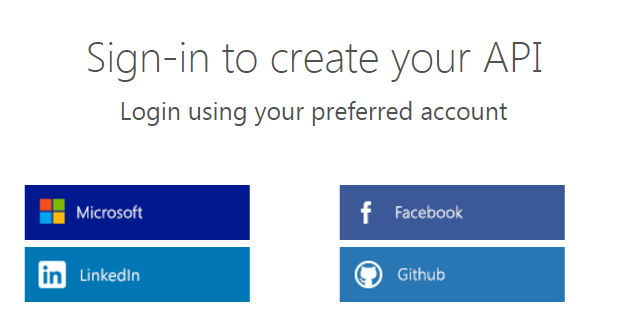
# Subscribe to Face API

1. Click on link:

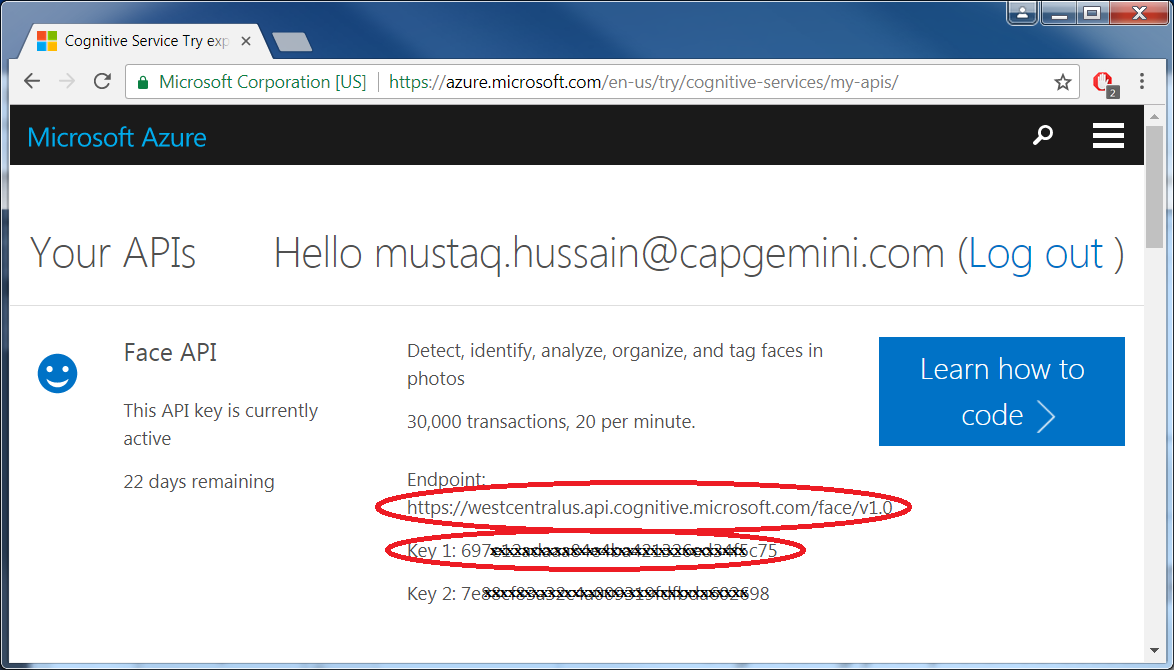
<https://azure.microsoft.com/en-us/try/cognitive-services/>



1. Click Face API’s ‘Get API Key’
2. Select Microsoft



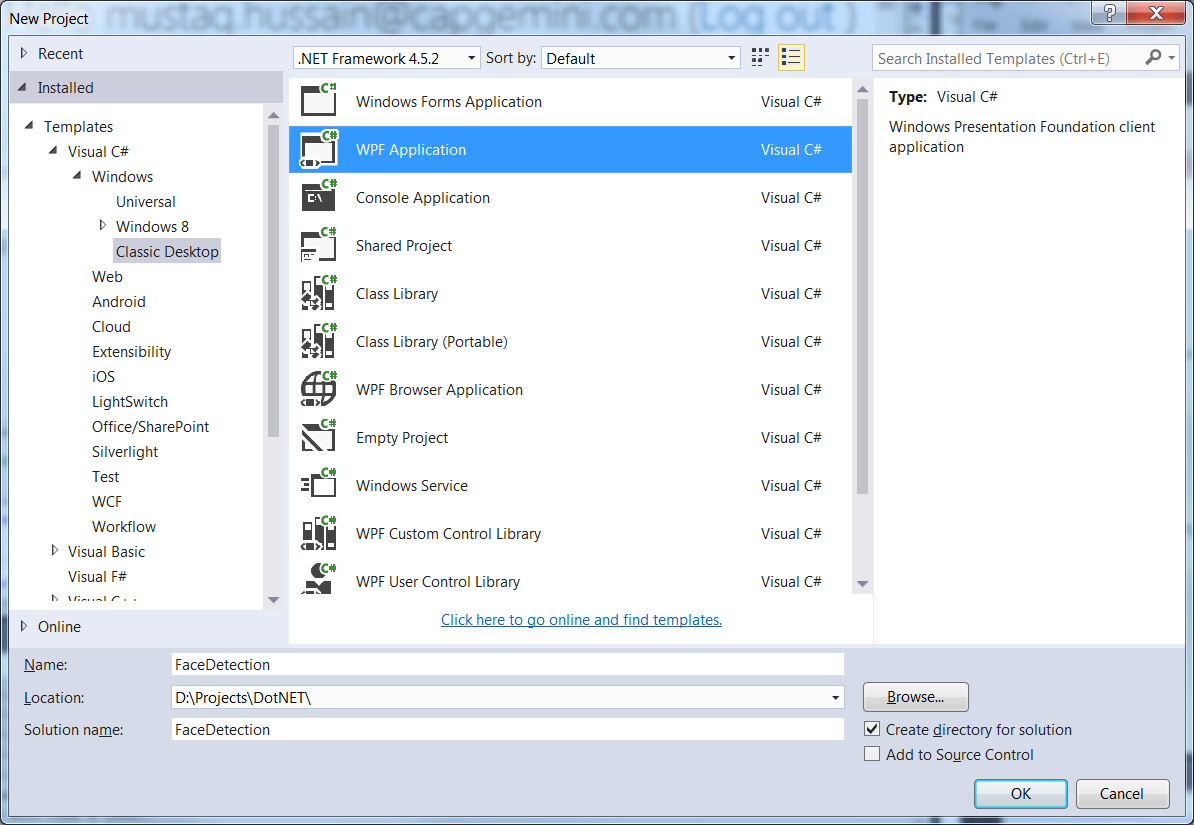
1. Login using your Capgemini login
2. You should now have one endpoint and 2 Keys (we shall be using the 1st key)



# Create a new Visual Studio Solution

1. Open Visual Studio 2015
2. File->New->Project
3. Installed->Templates->Visual C#->Windows->Classic Desktop->WPF Application
4. Select .NET Framework 4.5.2
5. Change
   1. Name : FaceDetection
   2. Location : D:\Projects\DotNET\
   3. Solution name: face Detection

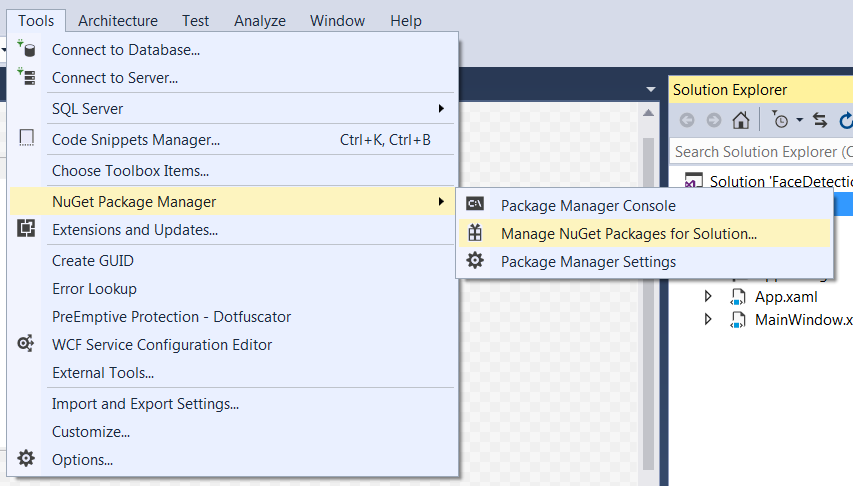
Create directory for solution ticked



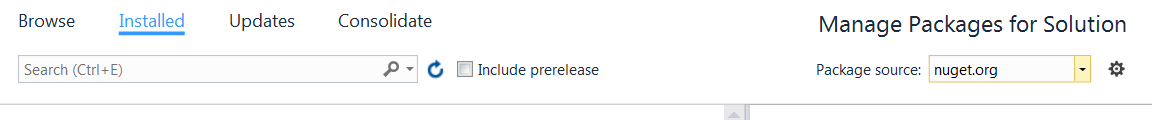
1. Press OK

# Install External Packages via NuGet

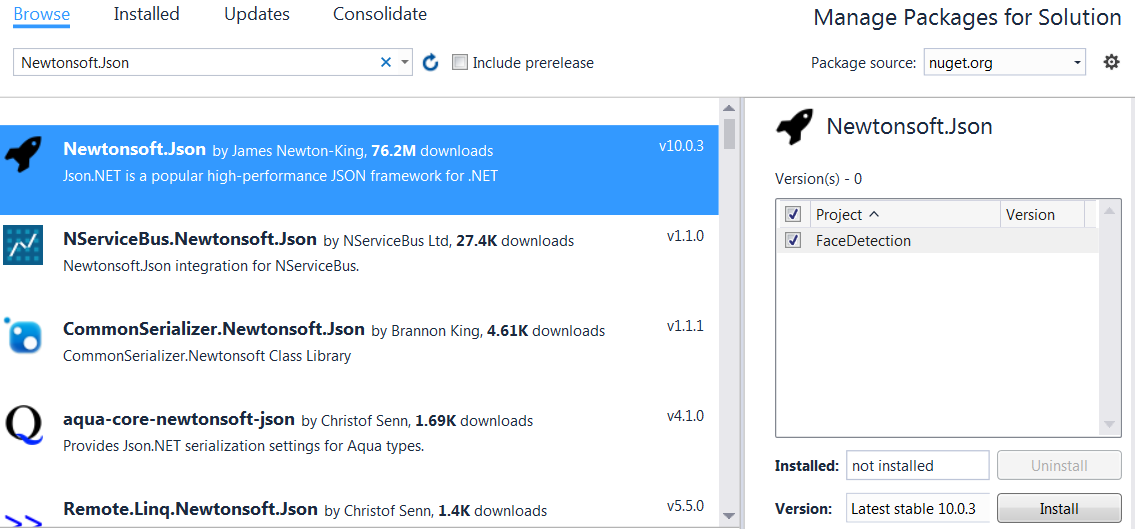
1. Select Tools->NuGet Package Manger->Manage NuGet Packages for Solution…



1. Select nuget.org under Package source



1. Select Browse and enter Newtonsoft.Json in the search box.
2. Select Newtonsoft.Json v10.0.3 and Project FaceDetection and Install



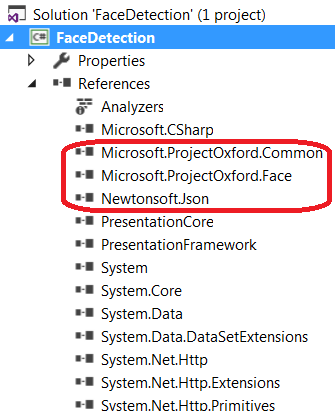
1. Click OK in Preview to complete installation

# Install Face APIs

Still in Manage Packages for Solution

1. In search box enter Microsoft.ProjectOxford.Face
2. Select Microsoft.ProjectOxford.Face v1.3.0 and Install then OK

Face APIs are now part of the project



# Copy & Paste Code

1. Select MainWindow.xaml and replace existing code with following

<Window x:Class="FaceTutorial.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="MainWindow" Height="700" Width="960">

<Grid x:Name="BackPanel">

<Image x:Name="FacePhoto" Stretch="Uniform" Margin="0,0,0,50" MouseMove="FacePhoto\_MouseMove" />

<DockPanel DockPanel.Dock="Bottom">

<Button x:Name="BrowseButton" Width="72" Height="20" VerticalAlignment="Bottom" HorizontalAlignment="Left"

Content="Browse..."

Click="BrowseButton\_Click" />

<StatusBar VerticalAlignment="Bottom">

<StatusBarItem>

<TextBlock Name="faceDescriptionStatusBar" />

</StatusBarItem>

</StatusBar>

</DockPanel>

</Grid>

</Window>

1. Select MainWindow.xaml.cs and replace existing code with following (goes from

**// MainWindow.xaml.cs START**

using System;

using System.Collections.Generic;

using System.IO;

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;

namespace FaceTutorial

{

public partial class MainWindow : Window

{

// Replace the first parameter with your valid subscription key.

//

// Replace or verify the region in the second parameter.

//

// You must use the same region in your REST API call as you used to obtain your subscription keys.

// For example, if you obtained your subscription keys from the westus region, replace

// "westcentralus" in the URI below with "westus".

//

// NOTE: Free trial subscription keys are generated in the westcentralus region, so if you are using

// a free trial subscription key, you should not need to change this region.

private readonly IFaceServiceClient faceServiceClient =

new FaceServiceClient("7ab0aa443a5d4b83a017a438418f21de", "https://westcentralus.api.cognitive.microsoft.com/face/v1.0");

Face[] faces; // The list of detected faces.

String[] faceDescriptions; // The list of descriptions for the detected faces.

double resizeFactor; // The resize factor for the displayed image.

public MainWindow()

{

InitializeComponent();

}

// Displays the image and calls Detect Faces.

private async void BrowseButton\_Click(object sender, RoutedEventArgs e)

{

// 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;

}

// Display the image file.

string filePath = openDlg.FileName;

Uri fileUri = new Uri(filePath);

BitmapImage bitmapSource = new BitmapImage();

bitmapSource.BeginInit();

bitmapSource.CacheOption = BitmapCacheOption.None;

bitmapSource.UriSource = fileUri;

bitmapSource.EndInit();

FacePhoto.Source = bitmapSource;

// Detect any faces in the image.

Title = "Detecting...";

faces = await UploadAndDetectFaces(filePath);

Title = String.Format("Detection Finished. {0} face(s) detected", faces.Length);

if (faces.Length > 0)

{

// Prepare to draw rectangles around the faces.

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.Length];

for (int i = 0; i < faces.Length; ++i)

{

Face face = faces[i];

// Draw a rectangle on the face.

drawingContext.DrawRectangle(

Brushes.Transparent,

new Pen(Brushes.Red, 2),

new Rect(

face.FaceRectangle.Left \* resizeFactor,

face.FaceRectangle.Top \* resizeFactor,

face.FaceRectangle.Width \* resizeFactor,

face.FaceRectangle.Height \* resizeFactor

)

);

// Store the face description.

faceDescriptions[i] = FaceDescription(face);

}

drawingContext.Close();

// Display the image with the rectangle around the face.

RenderTargetBitmap faceWithRectBitmap = new RenderTargetBitmap(

(int)(bitmapSource.PixelWidth \* resizeFactor),

(int)(bitmapSource.PixelHeight \* resizeFactor),

96,

96,

PixelFormats.Pbgra32);

faceWithRectBitmap.Render(visual);

FacePhoto.Source = faceWithRectBitmap;

// Set the status bar text.

faceDescriptionStatusBar.Text = "Place the mouse pointer over a face to see the face description.";

}

}

// Displays the face description when the mouse is over a face rectangle.

private void FacePhoto\_MouseMove(object sender, MouseEventArgs e)

{

// If the REST call has not completed, return from this method.

if (faces == null)

return;

// Find the mouse position relative to the image.

Point mouseXY = e.GetPosition(FacePhoto);

ImageSource imageSource = FacePhoto.Source;

BitmapSource bitmapSource = (BitmapSource)imageSource;

// Scale adjustment between the actual size and displayed size.

var scale = FacePhoto.ActualWidth / (bitmapSource.PixelWidth / resizeFactor);

// Check if this mouse position is over a face rectangle.

bool mouseOverFace = false;

for (int i = 0; i < faces.Length; ++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;

// Display the face description for this face if the mouse is over this face rectangle.

if (mouseXY.X >= left && mouseXY.X <= left + width && mouseXY.Y >= top && mouseXY.Y <= top + height)

{

faceDescriptionStatusBar.Text = faceDescriptions[i];

mouseOverFace = true;

break;

}

}

// If the mouse is not over a face rectangle.

if (!mouseOverFace)

faceDescriptionStatusBar.Text = "Place the mouse pointer over a face to see the face description.";

}

// Uploads the image file and calls Detect Faces.

private async Task<Face[]> UploadAndDetectFaces(string imageFilePath)

{

// The list of Face attributes to return.

IEnumerable<FaceAttributeType> faceAttributes =

new FaceAttributeType[] { FaceAttributeType.Gender, FaceAttributeType.Age, FaceAttributeType.Smile, FaceAttributeType.Emotion, FaceAttributeType.Glasses, FaceAttributeType.Hair };

// Call the Face API.

try

{

using (Stream imageFileStream = File.OpenRead(imageFilePath))

{

Face[] faces = await faceServiceClient.DetectAsync(imageFileStream, returnFaceId: true, returnFaceLandmarks: false, returnFaceAttributes: faceAttributes);

return faces;

}

}

// Catch and display Face API errors.

catch (FaceAPIException f)

{

MessageBox.Show(f.ErrorMessage, f.ErrorCode);

return new Face[0];

}

// Catch and display all other errors.

catch (Exception e)

{

MessageBox.Show(e.Message, "Error");

return new Face[0];

}

}

// Returns a string that describes the given face.

private string FaceDescription(Face face)

{

StringBuilder sb = new StringBuilder();

sb.Append("Face: ");

// Add the gender, age, and smile.

sb.Append(face.FaceAttributes.Gender);

sb.Append(", ");

sb.Append(face.FaceAttributes.Age);

sb.Append(", ");

sb.Append(String.Format("smile {0:F1}%, ", face.FaceAttributes.Smile \* 100));

// Add the emotions. Display all emotions over 10%.

sb.Append("Emotion: ");

EmotionScores emotionScores = face.FaceAttributes.Emotion;

if (emotionScores.Anger >= 0.1f) sb.Append(String.Format("anger {0:F1}%, ", emotionScores.Anger \* 100));

if (emotionScores.Contempt >= 0.1f) sb.Append(String.Format("contempt {0:F1}%, ", emotionScores.Contempt \* 100));

if (emotionScores.Disgust >= 0.1f) sb.Append(String.Format("disgust {0:F1}%, ", emotionScores.Disgust \* 100));

if (emotionScores.Fear >= 0.1f) sb.Append(String.Format("fear {0:F1}%, ", emotionScores.Fear \* 100));

if (emotionScores.Happiness >= 0.1f) sb.Append(String.Format("happiness {0:F1}%, ", emotionScores.Happiness \* 100));

if (emotionScores.Neutral >= 0.1f) sb.Append(String.Format("neutral {0:F1}%, ", emotionScores.Neutral \* 100));

if (emotionScores.Sadness >= 0.1f) sb.Append(String.Format("sadness {0:F1}%, ", emotionScores.Sadness \* 100));

if (emotionScores.Surprise >= 0.1f) sb.Append(String.Format("surprise {0:F1}%, ", emotionScores.Surprise \* 100));

// Add glasses.

sb.Append(face.FaceAttributes.Glasses);

sb.Append(", ");

// Add hair.

sb.Append("Hair: ");

// Display baldness confidence if over 1%.

if (face.FaceAttributes.Hair.Bald >= 0.01f)

sb.Append(String.Format("bald {0:F1}% ", face.FaceAttributes.Hair.Bald \* 100));

// Display all hair color attributes over 10%.

HairColor[] hairColors = face.FaceAttributes.Hair.HairColor;

foreach (HairColor hairColor in hairColors)

{

if (hairColor.Confidence >= 0.1f)

{

sb.Append(hairColor.Color.ToString());

sb.Append(String.Format(" {0:F1}% ", hairColor.Confidence \* 100));

}

}

// Return the built string.

return sb.ToString();

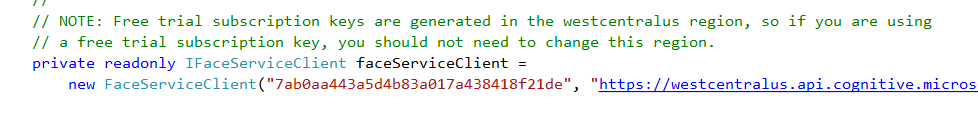
}

}

}

**// MainWindow.xaml.cs END**

1. Modify the key in MainWindow.xaml.cs at around position 30 to the key you received when you subscribed to the Face API (no need to change the endpoint URL)



1. You can now run the application – test against JPEG

Line 168 **UploadAndDetectFaces** – calls **faceServiceClient** which is the call to Face API

Try appending other **FaceAttributes** (**EyeMakeup**, **LipMakeup**) (navigate to **Face** Class) to description

<https://docs.microsoft.com/en-us/azure/cognitive-services/Face/Tutorials/FaceAPIinCSharpTutorial>

There is also a Java Android version (within above link) if you want to put it on an Android phone.

