The confused app

**{some coding required 😉}**

*Artificial intelligence, machine learning and quantum computing are able to generate random algorithms that will explicitly result in logistically reversed regressions. In order to decrease the randomization complexity, mixing LRRs with deep neural networks and making sure you have geo-redundant locally available back-up is the best way to ensure that your applications will remain lag-free, with a 99.9% SLA guaranteed uptime.*

**Is this clear? You look a bit confused.** How do I know it’s not clear? Because like any good teacher, I’m constantly monitoring and assessing the feedback from my students. It can be verbal feedback, but most times, the most honest feedback is the non-verbal one – gestures, expressions, sighs, and the *international headache I don’t get this stuff* pose

Long story short, teachers can see this **by analyzing students’ facial expression.** They know when it doesn’t make sense to you. By doing so they can evaluate and improve their teaching abilities and methodologies so that all students resonate with the teachings and with the content that is being taught.

The ultimate goal of education is to be available for everyone and to make complicated things look easy. Right?

But what happens when I get 200+ students? No matter how good a teacher is, there’s no way to constantly monitor and evaluate the feedback of 200+ faces, in real time. Most of them will be looking down on their phones anyways.

So we need to come up with a solution that will allow a teacher to monitor each student individually and aggregate all this data into a report that I will receive at the end of my lesson.

Luckily, we have AI for that. **Azure Cognitive Services** is a set of pre-defined and pre-coded machine learning models that is able to program a machine to make sense and react to the external, real world, like a human being would. Essentially, that is the definition of AI – to program a machine to monitor, ingest real world data, understand and interpret that data and react to it, similarly as a human being would do.

If you pay a visit to <https://azure.microsoft.com/en-us/services/cognitive-services/> you will see that there many cognitive services that you can use to make a machine think & react as a human. The best part about it? Those services were already programed and coded by Microsoft engineers, so if you want to use the, all you have to do is to pull them into your application.

Sounds fun? If yes, let’s get started!

The technologies we’re going to use are:

* Microsoft Azure Cognitive Services – Face & Emotion detection
* Microsoft Teams
* Visual Studio 2019

**Pre-requisites**

Azure - <https://azure.microsoft.com/en-us/free/>

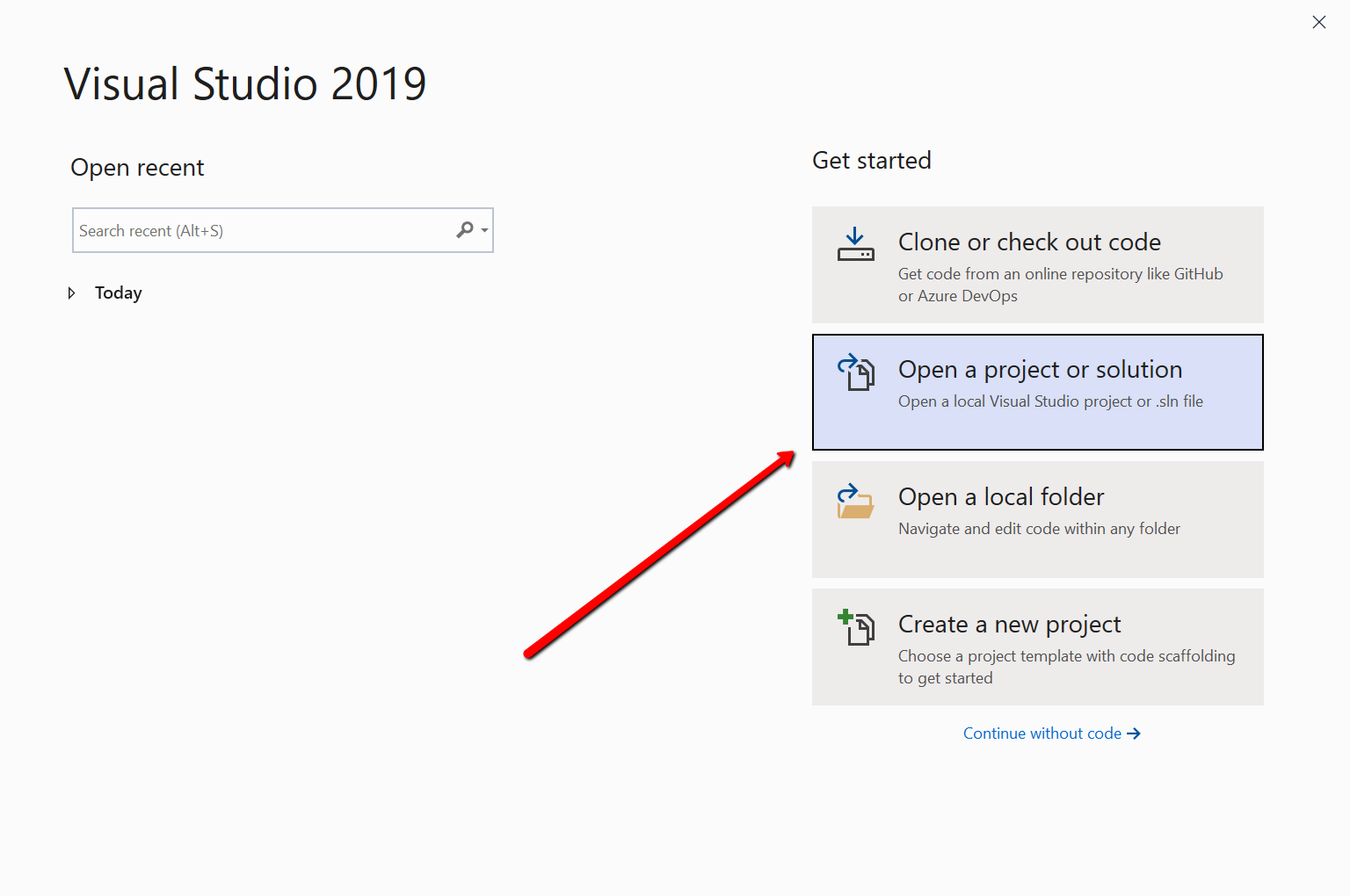
The AI tools and resources your application will be using are part of the cloud services available on Azure – Microsofts’ cloud computing platform. Any student is eligible to receive a free 12 months Azure subscription to use such AI services and many more. Access to the subscription is granted based on educational organization e-mail address – your student e-mail address. More details on https://azure.microsoft.com/en-us/free/free-account-students-faq/

Office 365 - <https://www.microsoft.com/en-us/microsoft-365/academic/compare-office-365-education-plans>

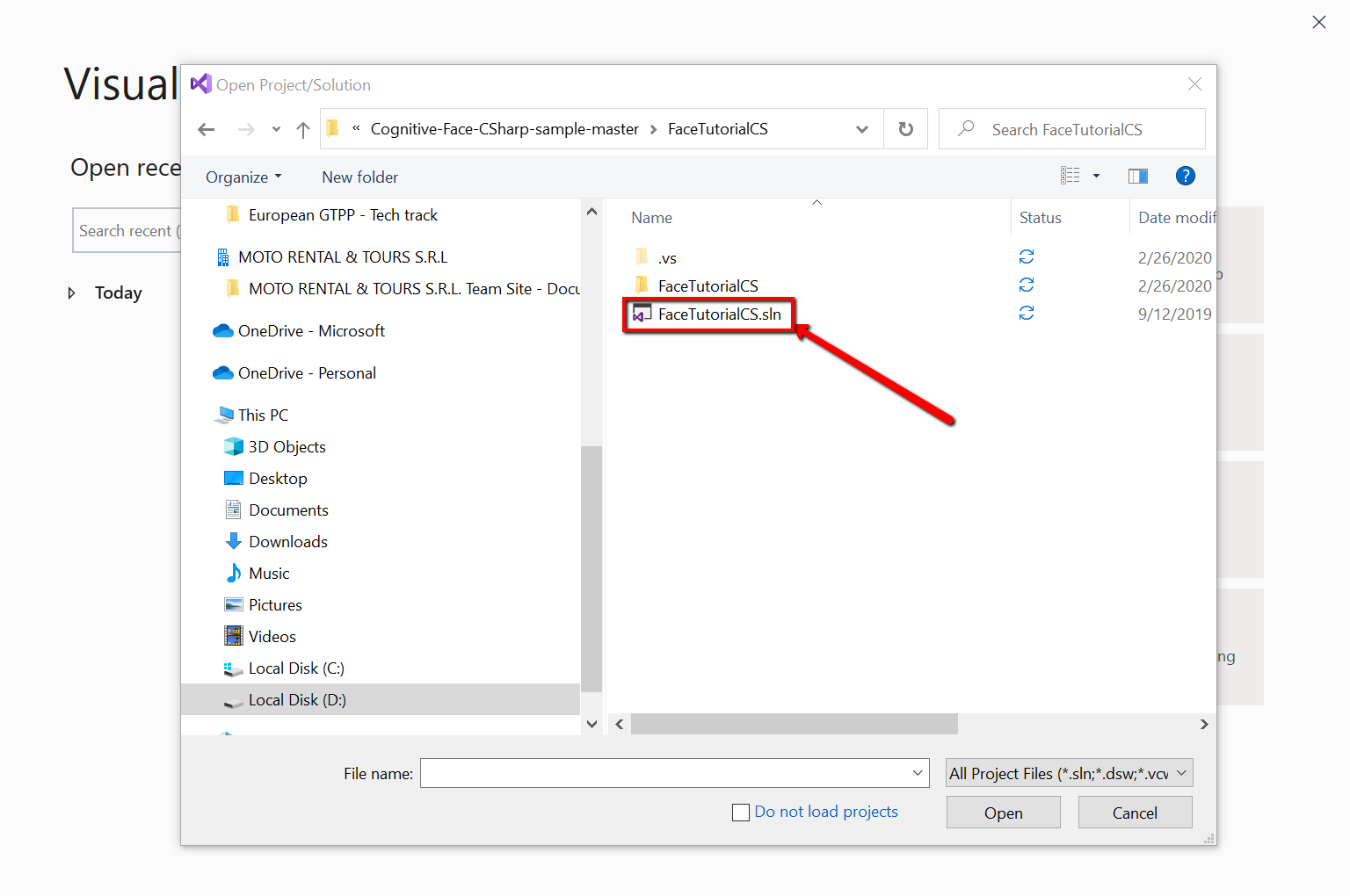
Microsoft Teams, PowerApps, OneDrive, Sway and a lot of other cool cloud products & services are available for free, both for teachers and students, by registering their school information on [Office 365 A1 School Sign-Up Form](https://signup.microsoft.com/signup?OfferId=6c1a80e7-cc3f-4e01-baaf-385fbda7cdfc,f31e46de-fe53-4a3e-ae1d-d6ffda791387&pc=93c908bd-d64f-4ac6-9522-03372c383511&DL=STANDARDWOFFPACK_FACULTY&culture=en-us&country=US&ali=1)

This tutorial based on <https://docs.microsoft.com/en-us/azure/cognitive-services/face/tutorials/faceapiincsharptutorial> so feel free to look at that as well and make sure you download the source code.

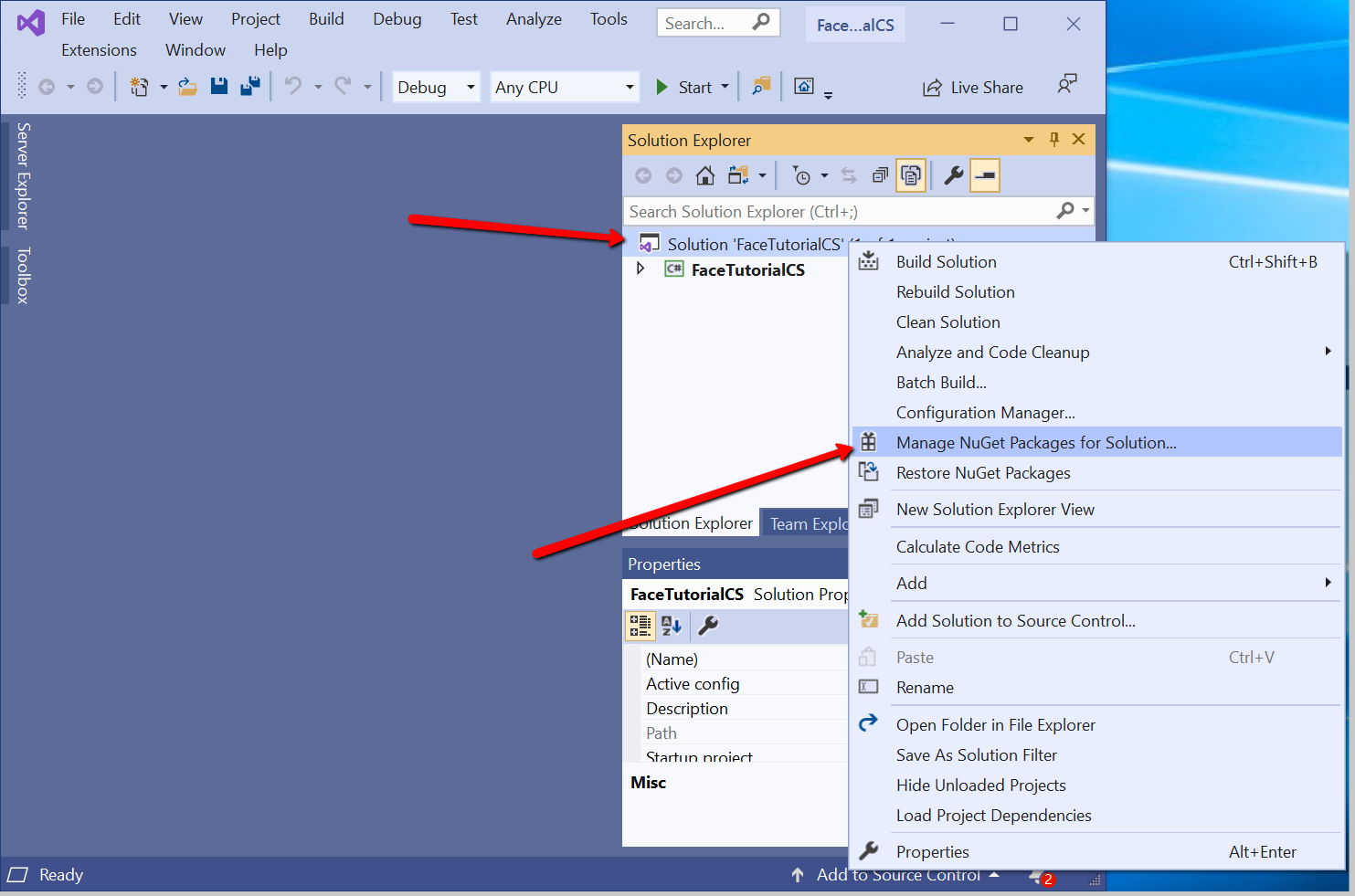
1. Open Visual Studio 2019 on your PC and click **Open a project or solution** (\*hint – the project files available on [GitHub](https://github.com/Azure-Samples/Cognitive-Face-CSharp-sample), make sure you download them to your PC)



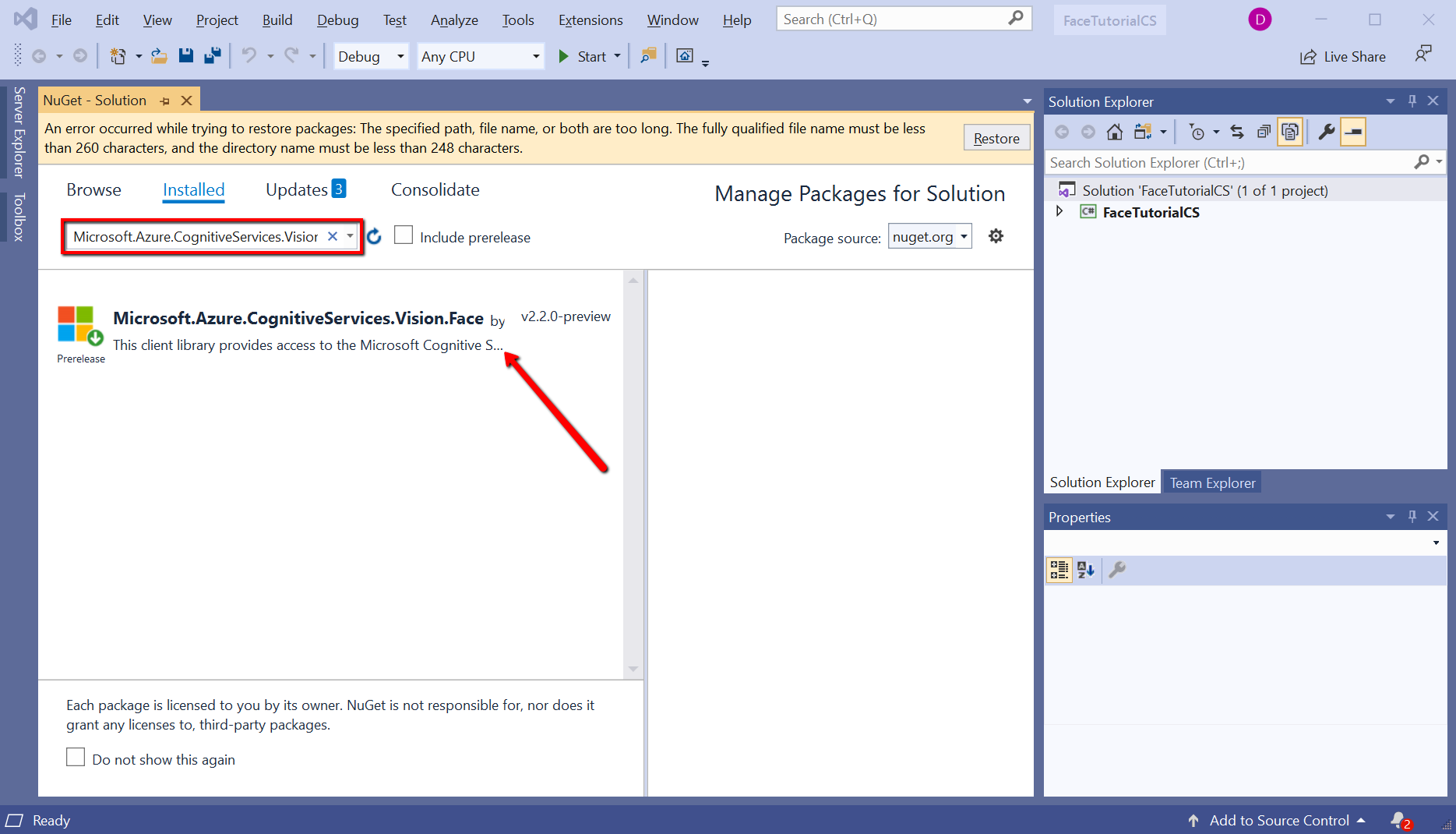
Browse to where you downloaded your project files and open the **FaceTutorial.sln** file



You’ve now opened the application in Visual Studio. The next thing is to include the packages we will be using to further customize and deploy the app. You can do so by **right clicking the Solution** -> **Manage NuGet Packages for Solution…**



Search and install the **Microsoft.Azure.CognitiveServices.Vision.Face** package

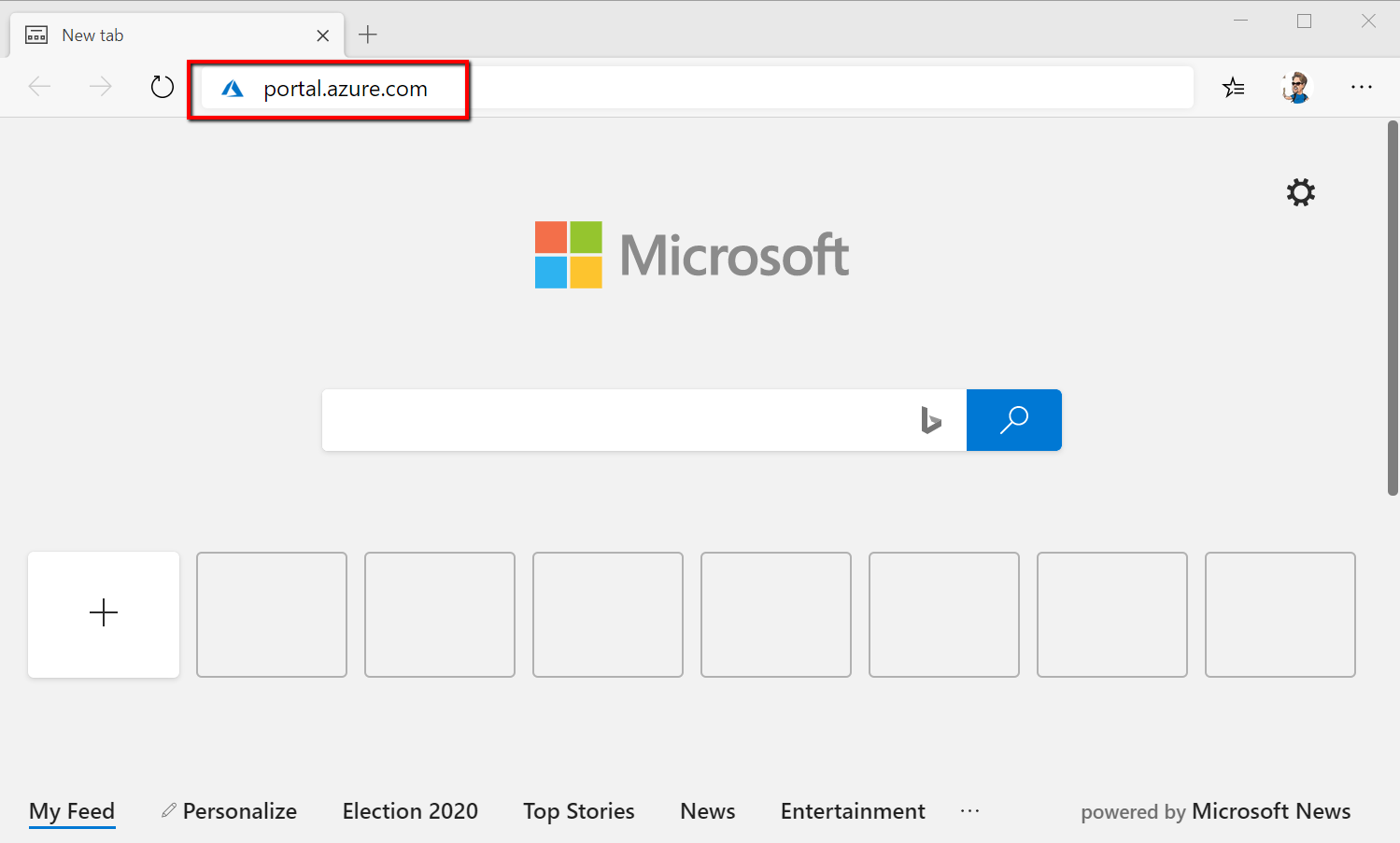


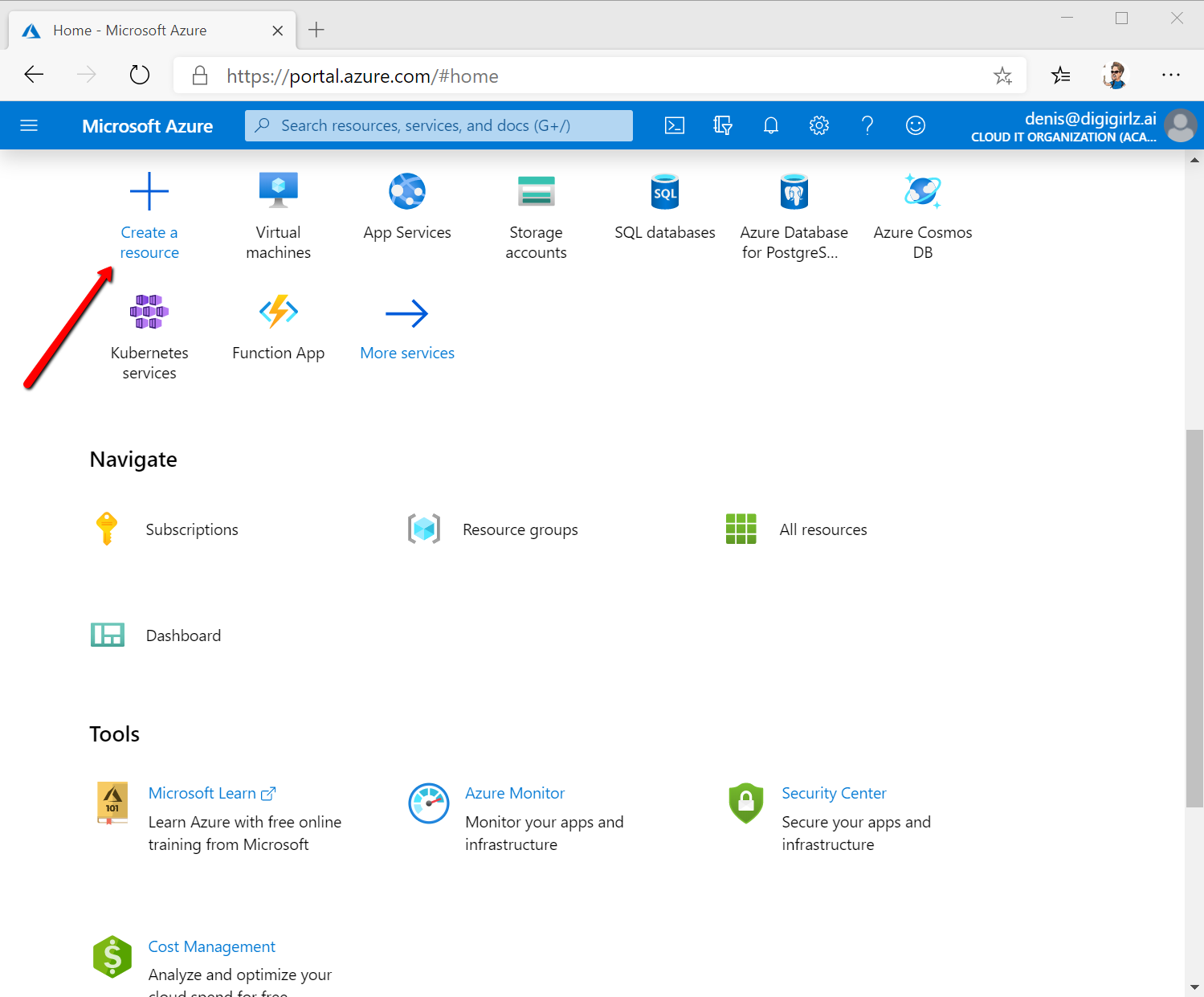
1. Create a Cognitive Services resource in Microsoft Azure

The application itself is just a framework – a set of instructions and procedures that can call out AI services from the cloud. Without the AI component, the app is not *intelligent* but it’s just an app that doesn’t do much on its own.

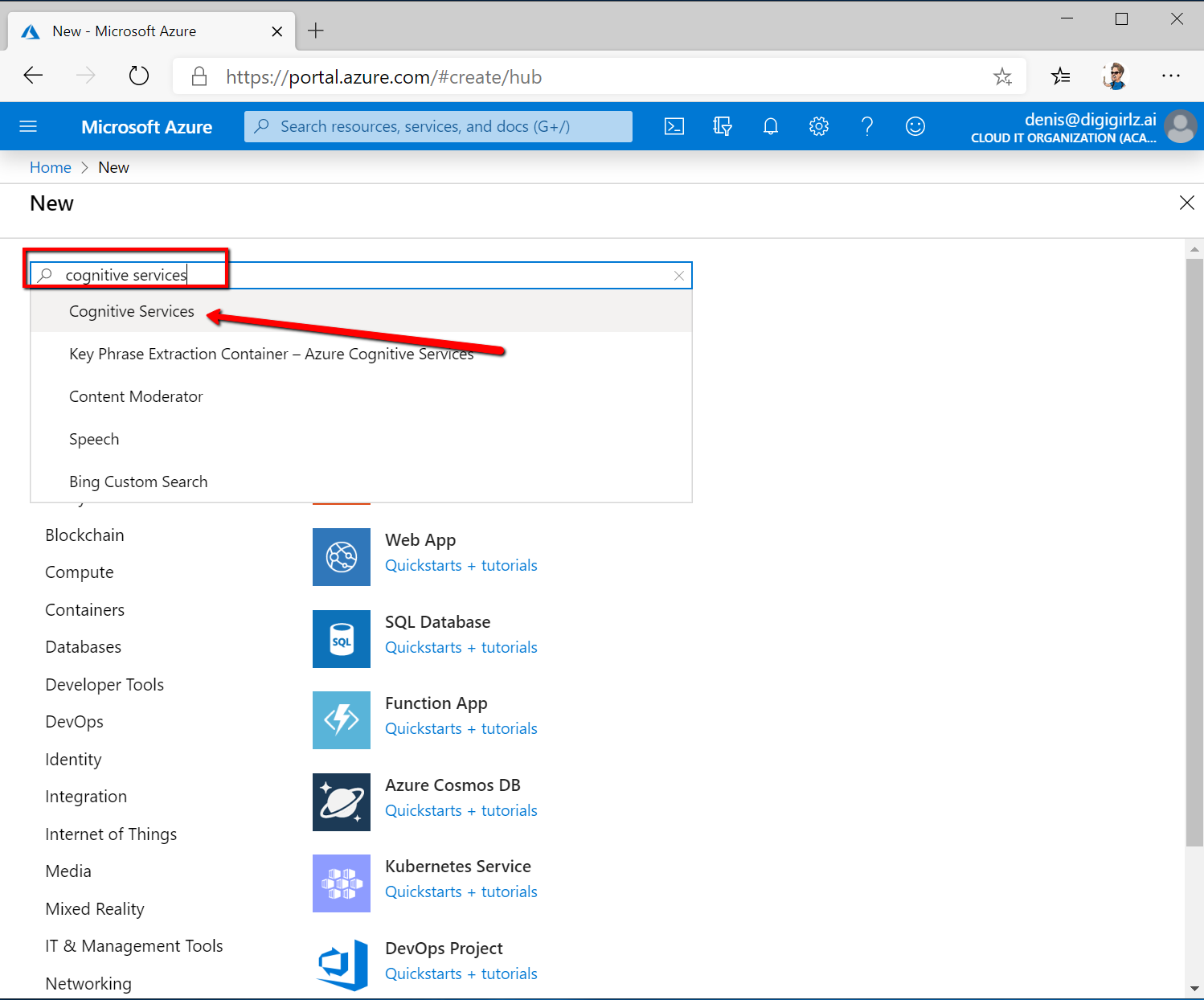
The next step will be to create **cognitive cloud service** that will be *called*  by the app to interpret the students image and check on his/her reaction.

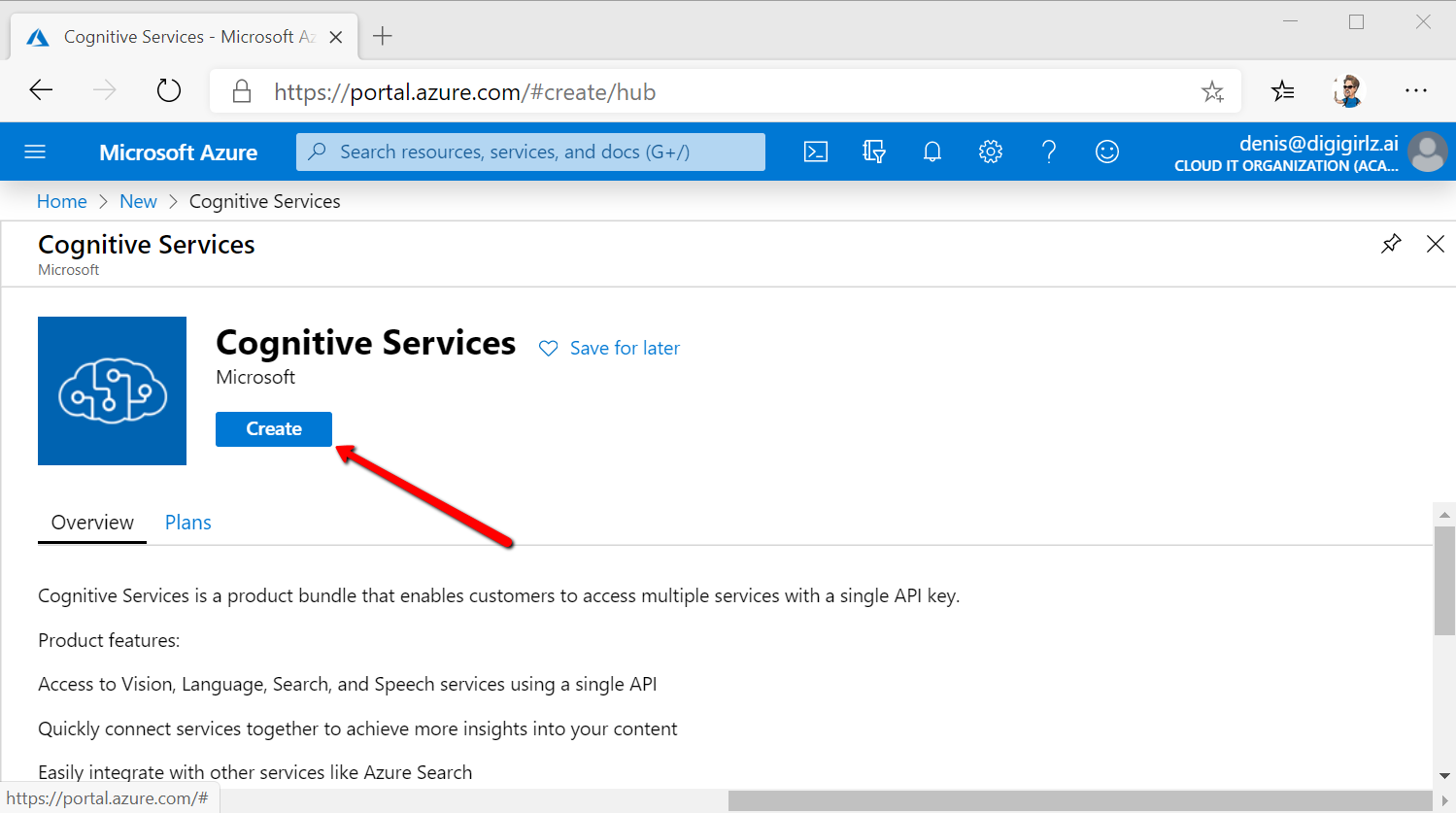
To do so we will navigate to [**http://portal.azure.com**](http://portal.azure.com)and login with your student account. Once you’re there tap **+ Create a resource**





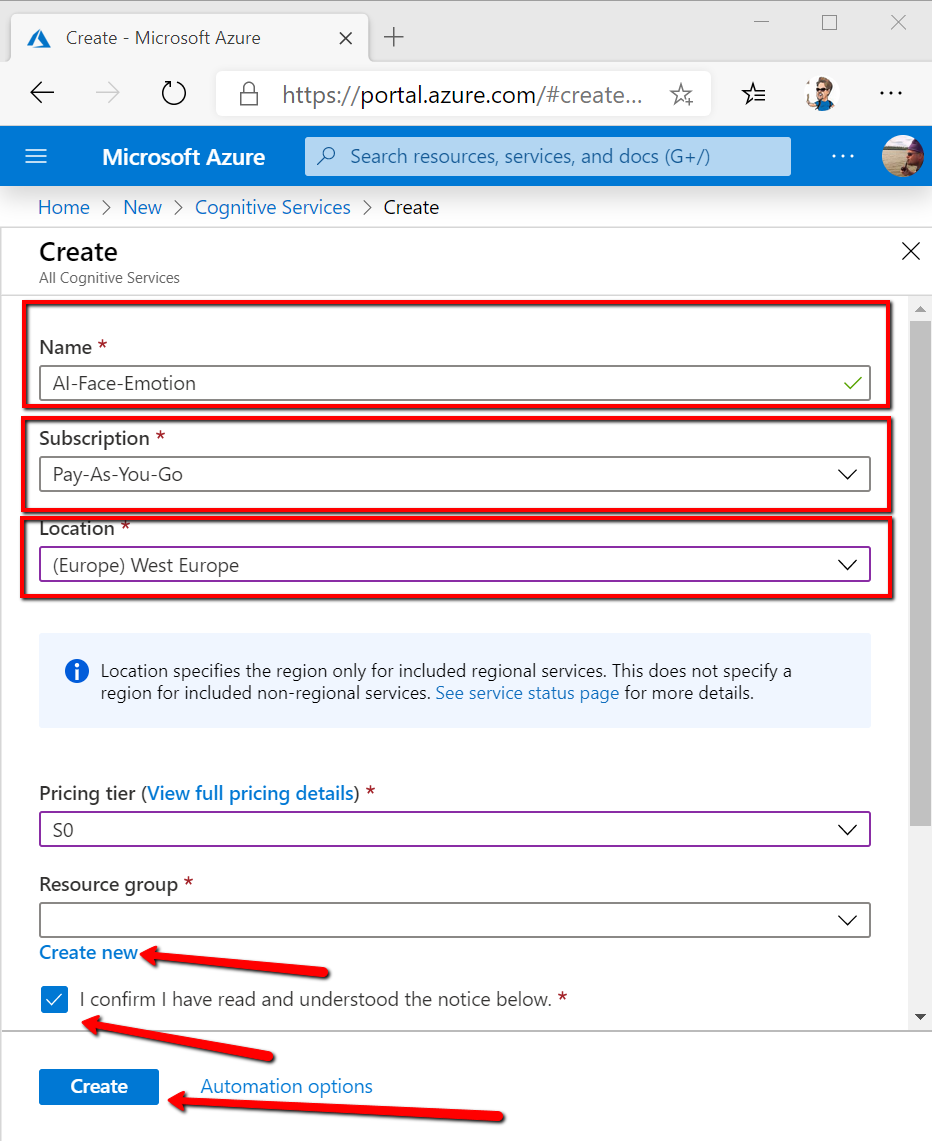
Search for **cognitive services** and click it. Click **Create**.





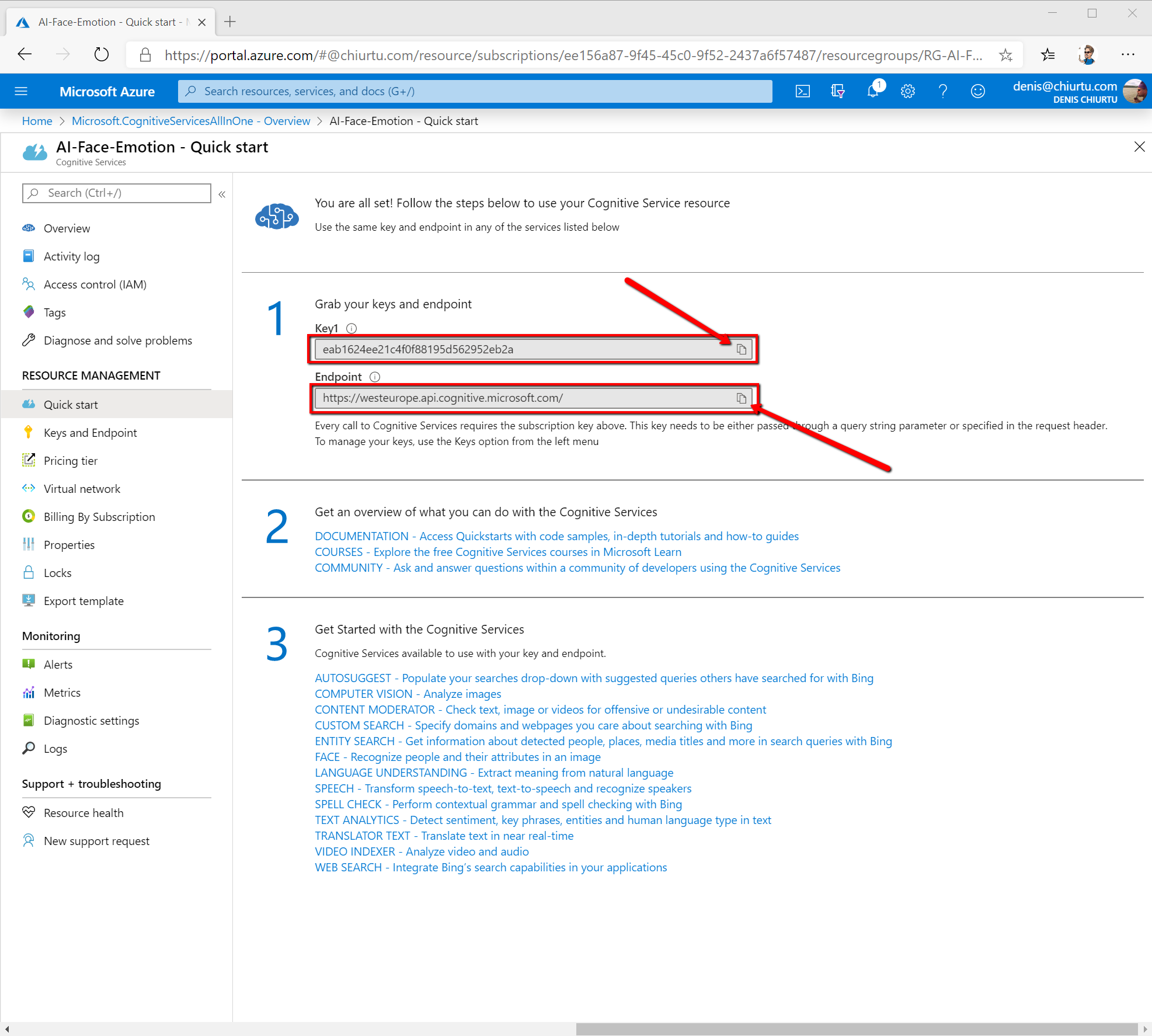
Define your resource by **naming it,** choosing your student subscription and selecting the closest geographical region as a location for the resource. You want to select location as geographically close to you as possible to have the best ping / reduce the network latency & number of hops. Price tier should be **S0** and you must **Create a new resource group** and name that as well.

Click **Create** when you’re done.

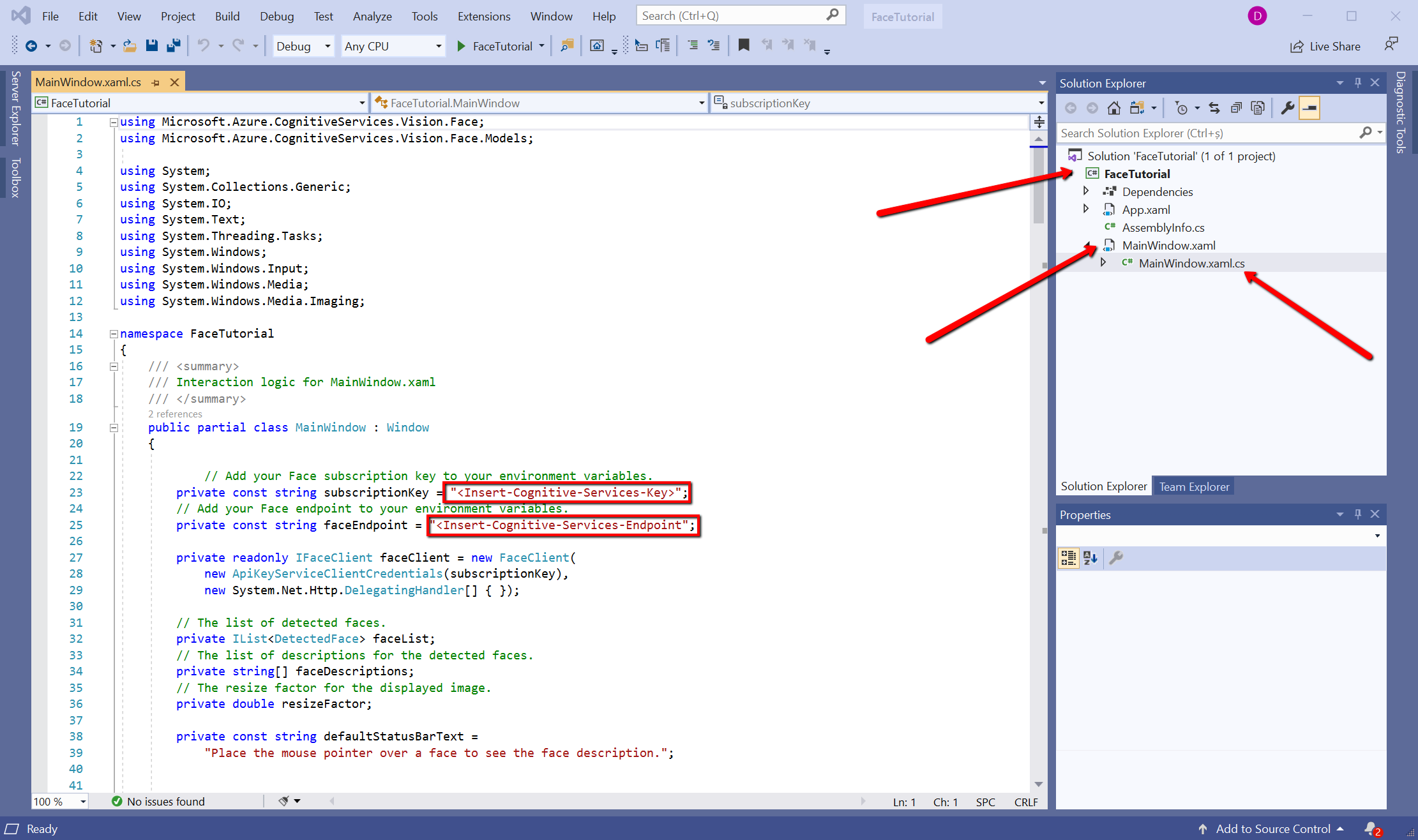


Wait for the deployment to finish and then click **Go to resource.**

From this dashboard, copy paste into Teams your **Key1** and **Endpoint.** You will be inputting these values into your Visual Studio application to call the AI service.

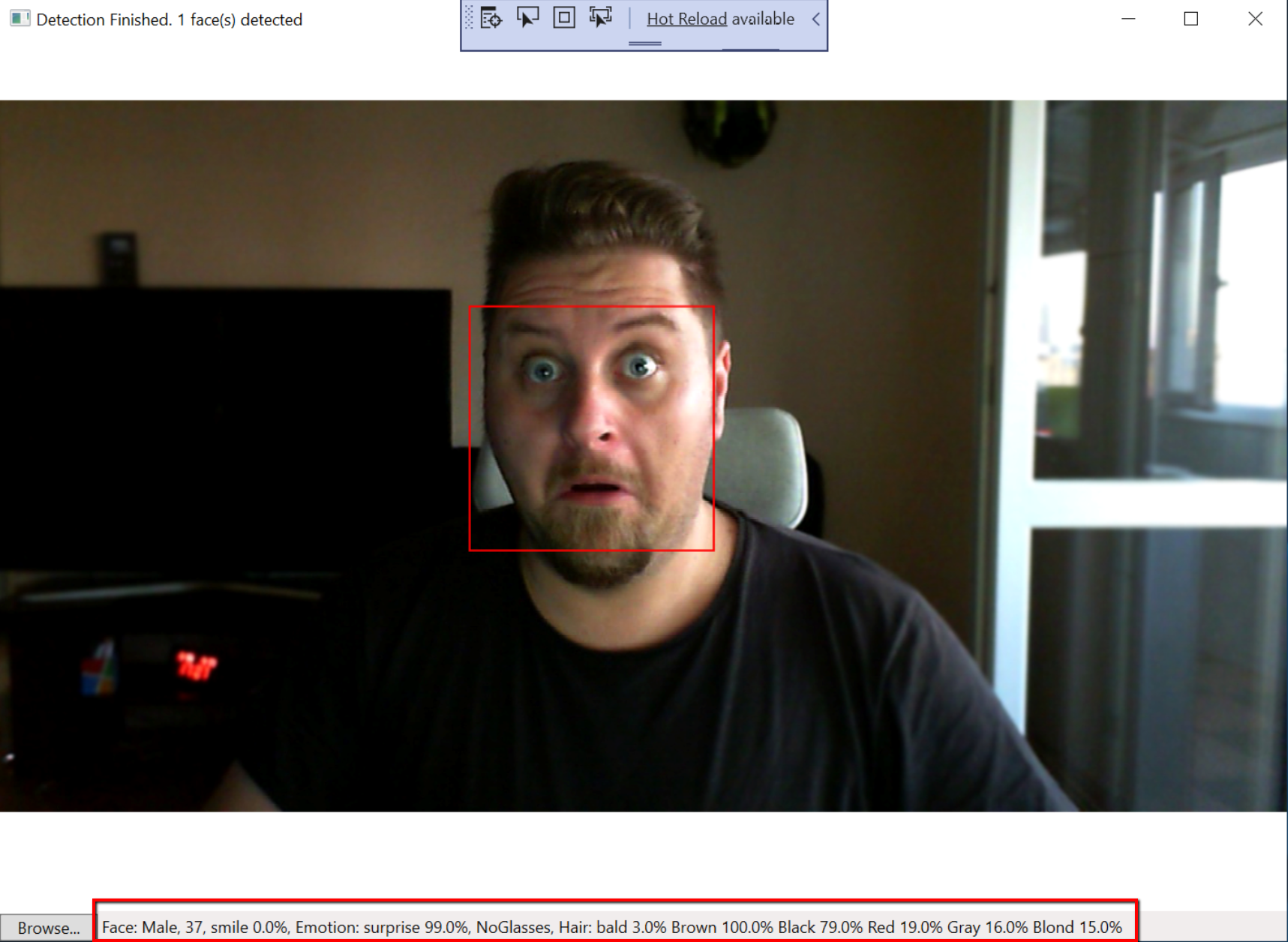


Revert back to Visual Studio and from the *Solution Explorer*  section on the right expand FaceTutoriaCS and then expand MainWindows.xaml. Click **MainWindow.xaml.cs** and replace *FACE\_SUBSCRIPTION\_KEY*  and *FACE\_ENDPOINT*  with the values you’ve just copied



Ctrl + S to save the file. Click **> FaceTutorial** to run the application.

Upon doing so, you will see a new window popping up on your desktop. Use the lower left **Browse** button to upload a face picture. For fun purposes, take selfie with your webcam, like I did, and see what emotions it’s able to decipher off your face and what other cool interesting data you can find out.



As you can see, the AI model picked up on a lot of features just by analyzing my image:

* It recognized a human face and marked it with a red rectangle
* It’s able to tell that I’m a male (no doubts here)
* It says I’m 37 years old… which isn’t exactly accurate, let’s put it that way :D
* The main **emotion you can read off my face is surprise – 99% -** I’m actually very surprised this app works!!!
* I’m not wearing any glasses
* I’m **3% bald**. It’s a start… someday I’ll probably get to 100%
* I’m of **brown** **skin** – which is a rather clumsy way to say that I’m of white/Caucasian ethnicity. There is no such thing as pure white skin, hence the word brown.
* And my hair is combination of **Black 79%, Red 19% (fun one), Gray 16% and Blond 15%**

Well done! You have built your own **AI Enabled application that is able to recognize people and their emotions.** It truly is impressive what you’ve managed to achieve in such short time. You can give yourself a well-deserved clap!

Now that you have a good understanding of how these technologies work, go ahead and look for **Intelligent Kiosk App** on your Windows PC to see other examples of how AI can be used and try those as well. Or spend your time improving this application you’ve just created. The best part about it? **It’s up to you** 😊 shape your world!