

Developing and Deploying Intelligent Chat Bots

Objectives

Objectives

The aim of this lab is to build a bot called SentiBot that integrates Text Analytics Cognitive Services API. Specifically, we will use the Sentiment Analysis feature of Text Analytics Cognitive Services API to evaluate the sentiment of any message sent to the bot.

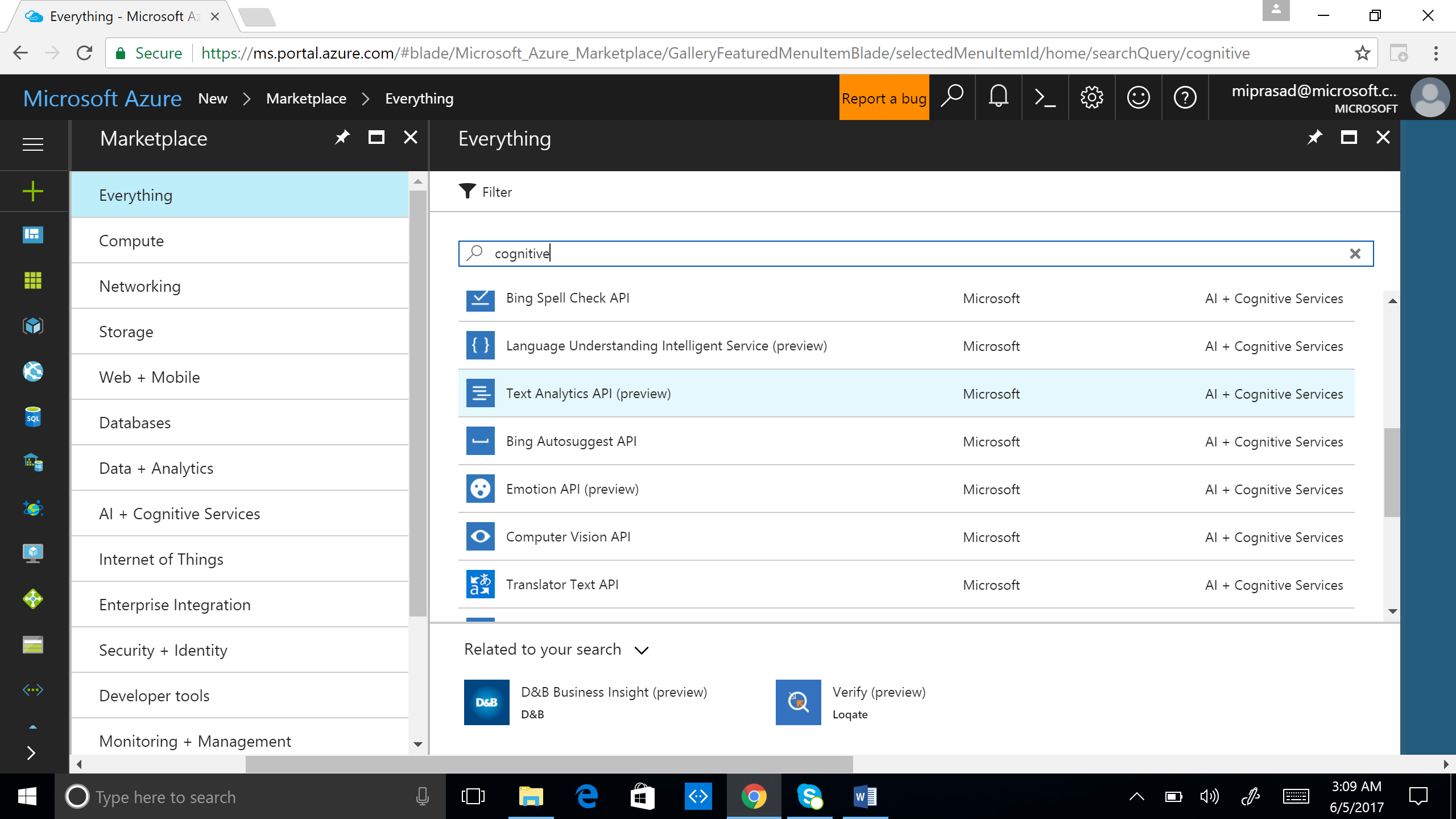
The lab is adapted from:

<https://docs.botframework.com/en-us/bot-intelligence/language/#example-emotional-bot>

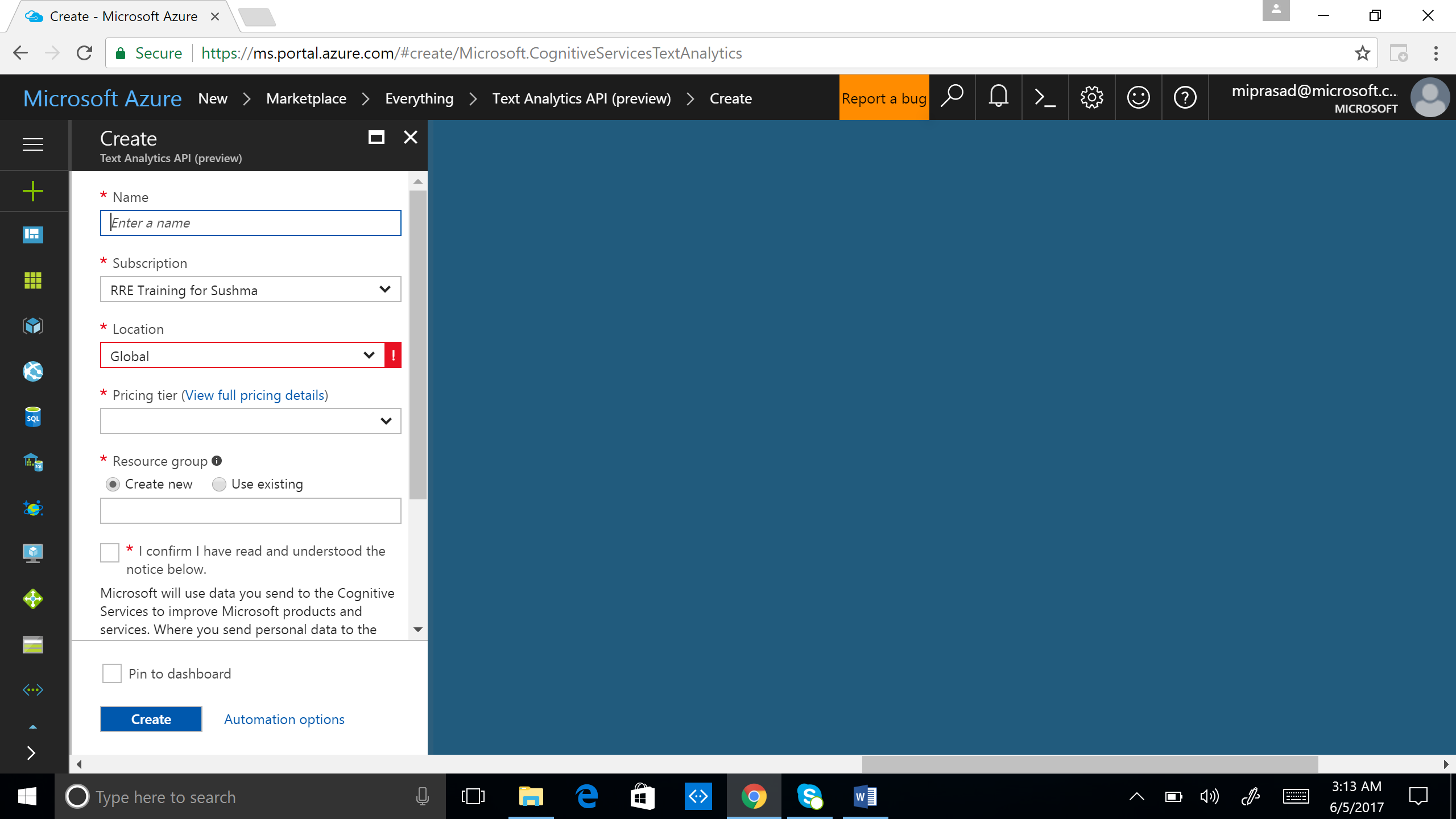
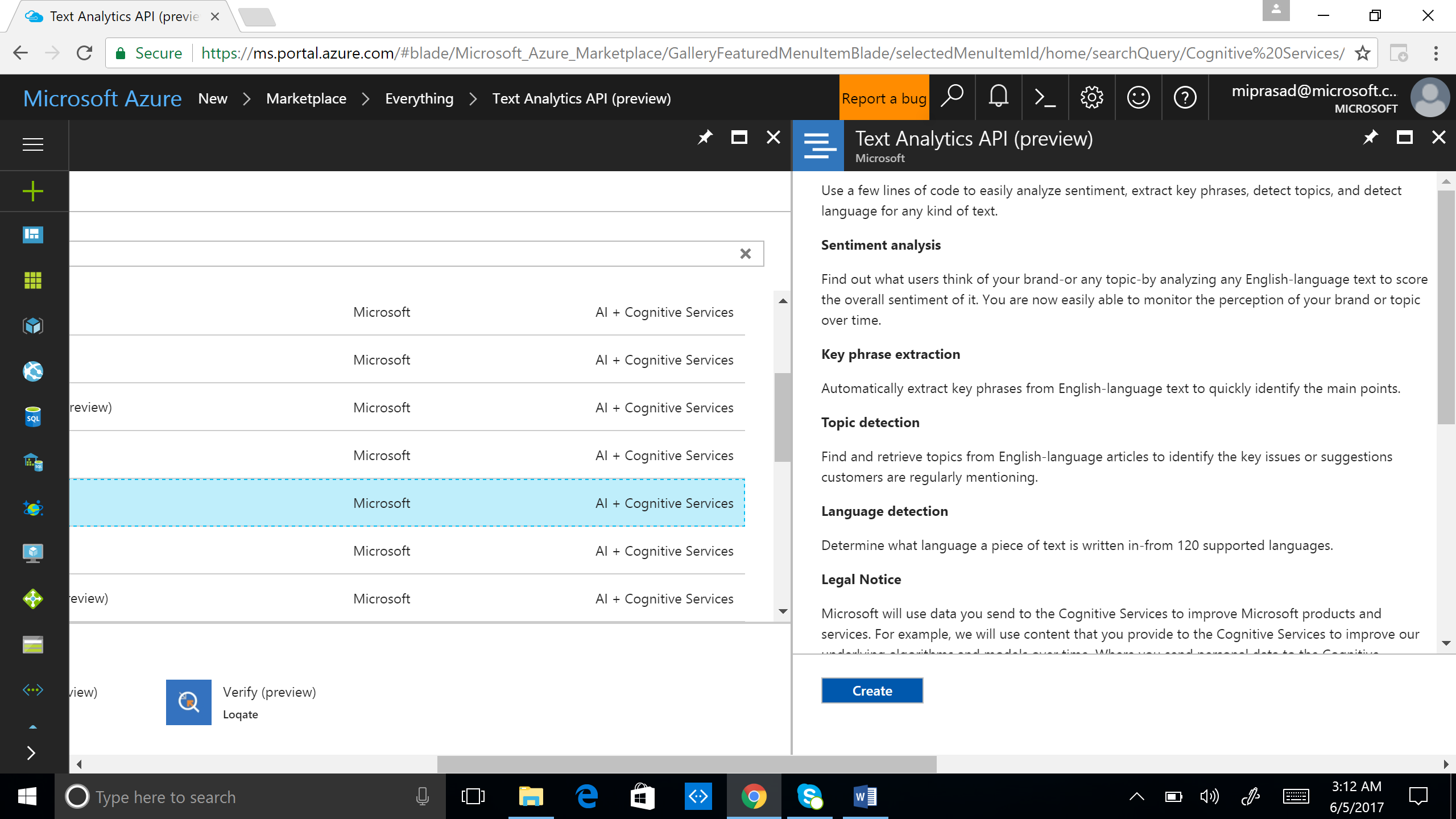
.

Setup

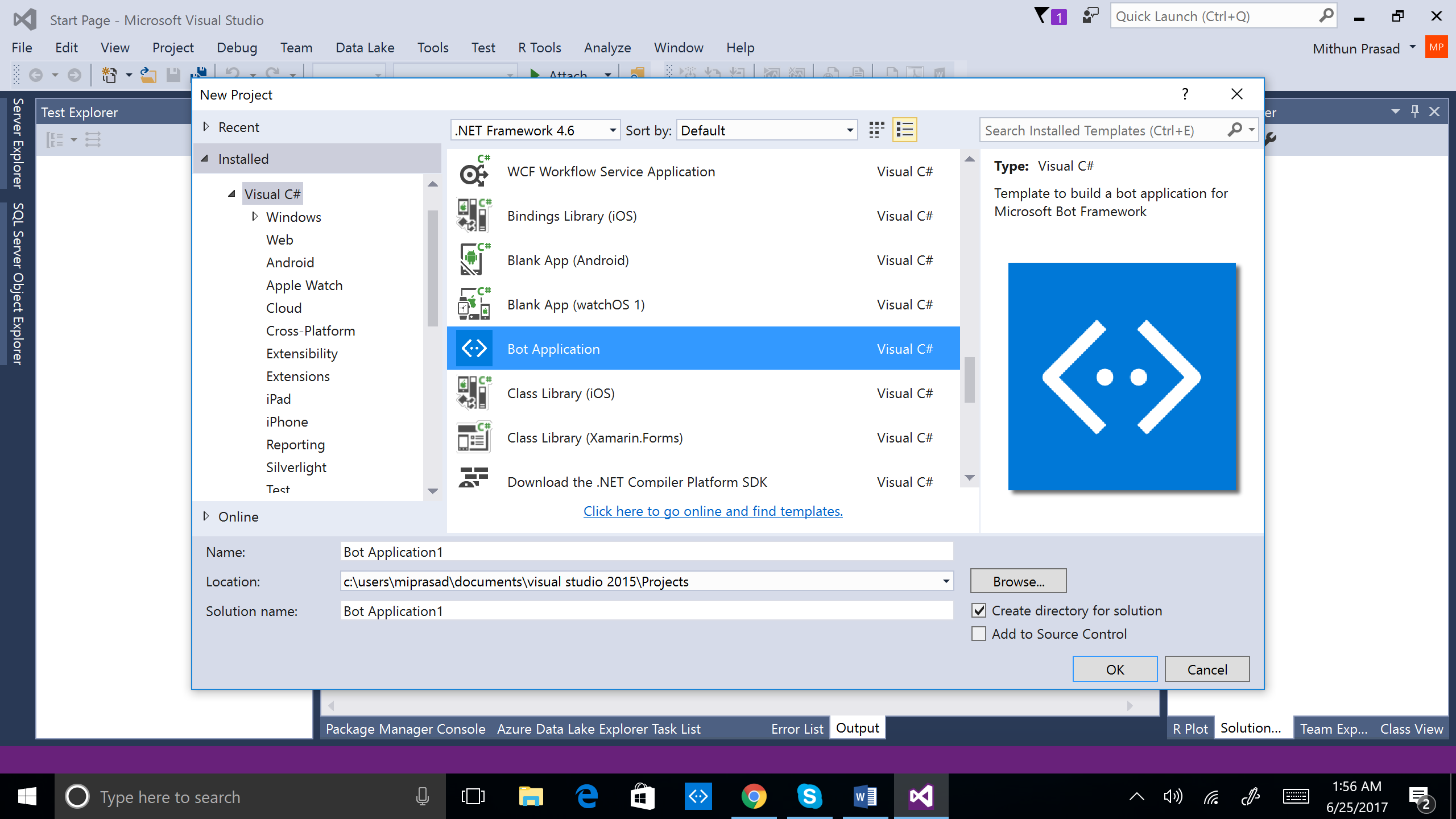
1. Text Analytics API Key:
   1. Go to [portal.azure.com](https://azure.microsoft.com/en-in/services/cognitive-services/text-analytics/) and sign in. Sign up for an account if you do not have one.
   2. Select the Text Analytics API as shown below:



* 1. Create an account by filling out all the mandatory fields after clicking “create”. Copy the key from the blade after you successfully create.



2. Create a new Bot Project using Bot Application Template in Visual Studio as shown below:



SentiBot

Create a new C# class file (Sentiment.cs) with the following code. The class will serve as our model for the JSON input/output of the Text Analytics API.

namespace TextSentiBot

{

// Classes to store the input for the sentiment API call

public class BatchInput

{

public List<DocumentInput> documents { get; set; }

}

public class DocumentInput

{

public double id { get; set; }

public string text { get; set; }

}

// Classes to store the result from the sentiment analysis

public class BatchResult

{

public List<DocumentResult> documents { get; set; }

}

public class DocumentResult

{

public double score { get; set; }

public string id { get; set; }

}

}

Next, go to MessagesController.cs and add the following namespaces if needed.

using System;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Description;

using Microsoft.Bot.Connector;

using Newtonsoft.Json;

using System.Collections.Generic;

using System.Net.Http.Headers;

using System.Text;

The code receives the user message, calls the sentiment analysis endpoint and responds accordingly to the user. Replace the Post method in MessageController.cs with the following code.

The apiKey in the below code would need to be replaced with your key.

public async Task<HttpResponseMessage> Post([FromBody]Activity activity)

{

ConnectorClient connector = new ConnectorClient(new   
 Uri(activity.ServiceUrl));

if (activity == null || activity.GetActivityType() !=   
 ActivityTypes.Message)

{

//add code to handle errors, or non-messaging activities

}

const string apiKey = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";

string queryUri = "https://westus.api.cognitive.microsoft.com/text/analytics/v2.0/sentiment";

HttpClient client = new HttpClient();

client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key", apiKey);

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

BatchInput sentimentInput = new BatchInput();

sentimentInput.documents = new List<DocumentInput>();

sentimentInput.documents.Add(new DocumentInput()

{

id = 1,

text = activity.Text

});

var sentimentJsonInput = JsonConvert.SerializeObject(sentimentInput);

byte[] byteData = Encoding.UTF8.GetBytes(sentimentJsonInput);

var content = new ByteArrayContent(byteData);

content.Headers.ContentType = new   
 MediaTypeHeaderValue("application/json");

var sentimentPost = await client.PostAsync(queryUri, content);

var sentimentRawResponse = await   
 sentimentPost.Content.ReadAsStringAsync();

var sentimentJsonResponse =   
 JsonConvert.DeserializeObject<BatchResult>(sentimentRawResponse);

double sentimentScore = sentimentJsonResponse.documents[0].score;

var replyMessage = activity.CreateReply();

replyMessage.Recipient = activity.From;

replyMessage.Type = ActivityTypes.Message;

if (sentimentScore > 0.9)

{

replyMessage.Text = $"This appears quite positive to me.";

}

else if (sentimentScore < 0.1)

{

replyMessage.Text = $"This appears quite negative to me.";

}

else

{

replyMessage.Text = $"I can not decipher the sentiment here. Please   
 try again or add more information.";

}

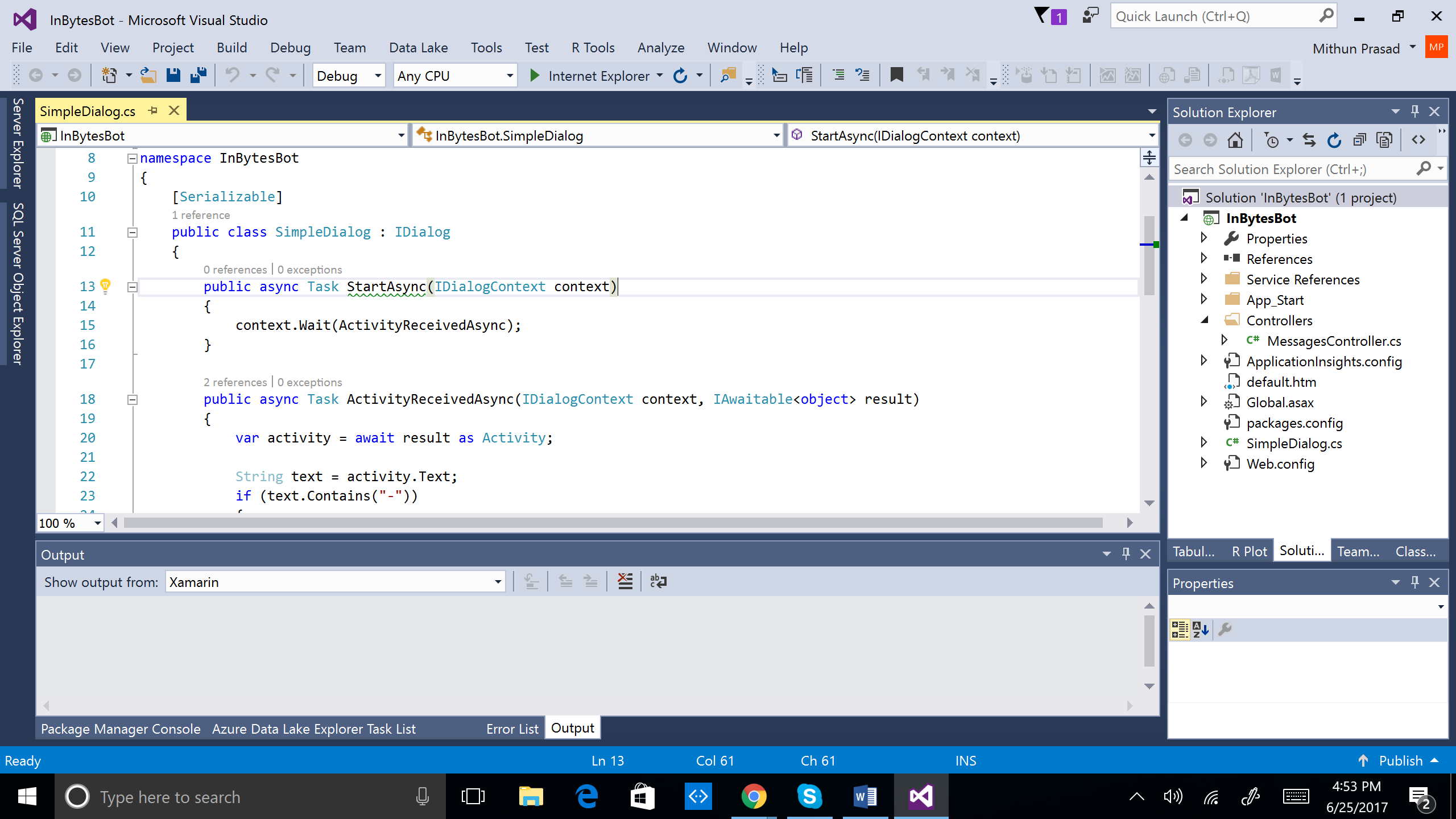
await connector.Conversations.ReplyToActivityAsync(replyMessage);

var response = Request.CreateResponse(HttpStatusCode.OK);

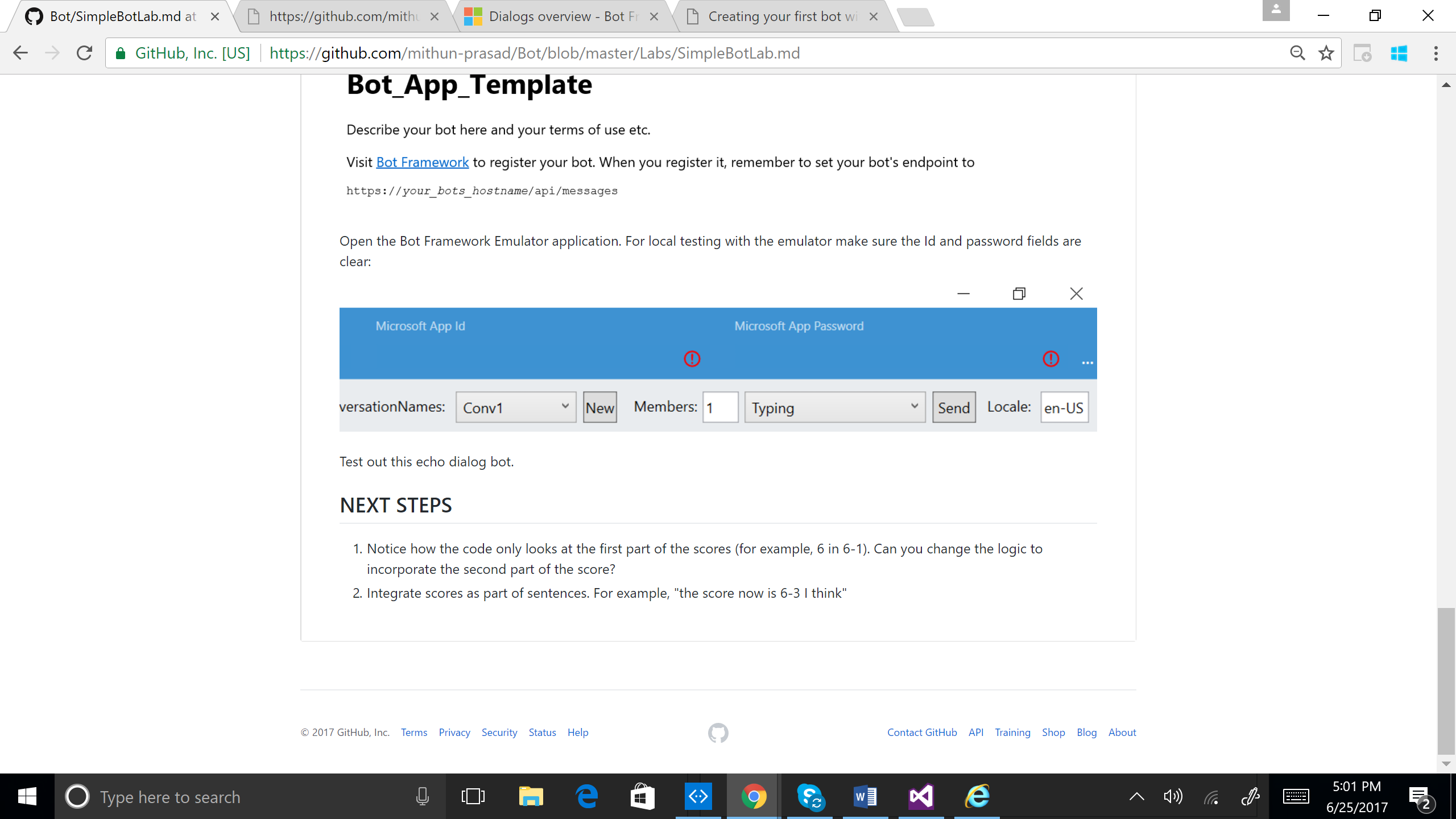
return response;

}

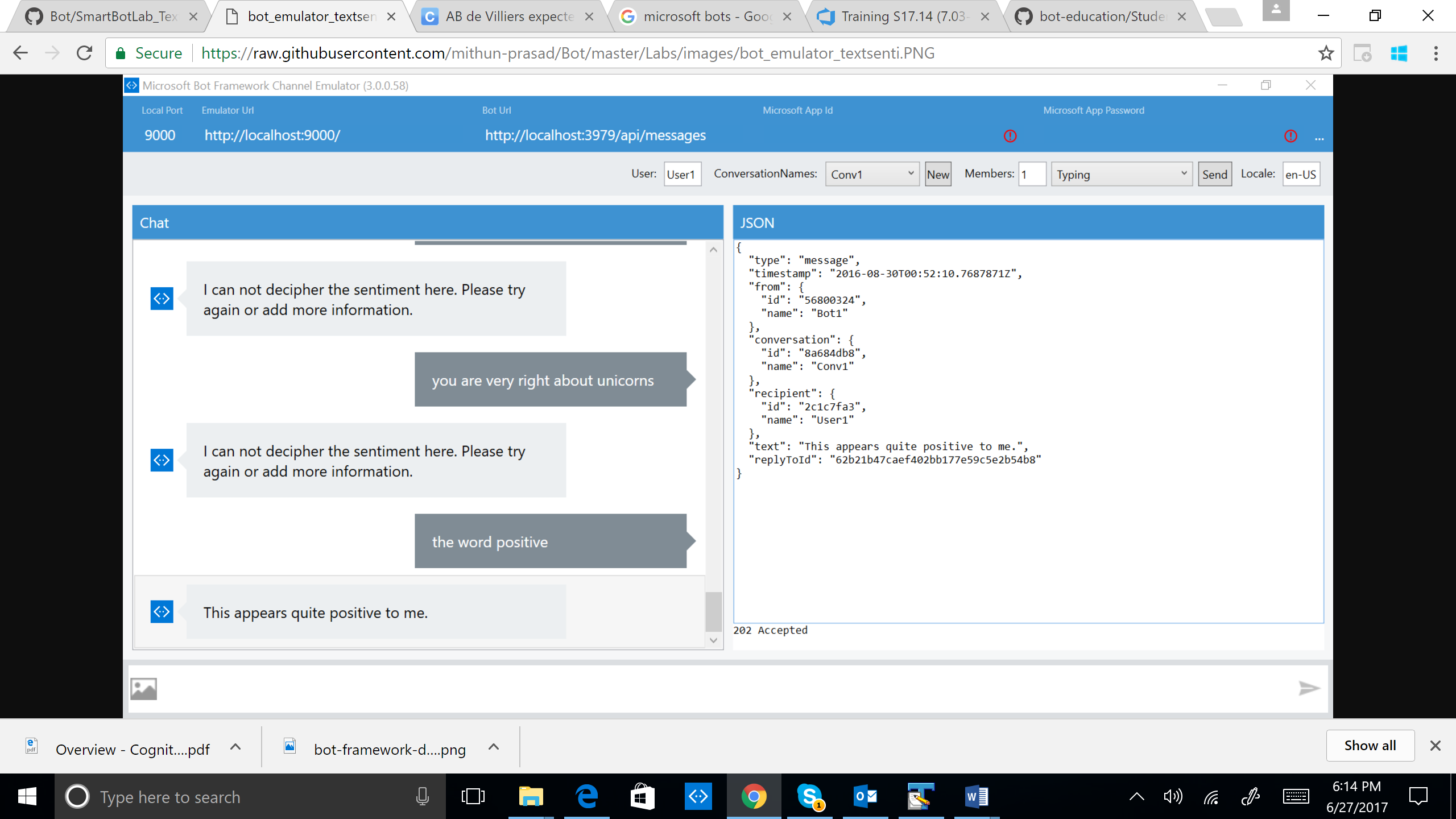
Run from Visual Studio by clicking *Internet Explorer* as shown below. This should open a browser window with the endpoint (showing the default.htm file from project).



Open the bot emulator and change the Bot Url to <http://localhost:port/api/messages>. You will need to add /api/messages to the Url. Ensure the fields Id and password fields are clear.



Test it in the emulator by sending very positive and very negative messages:



Exercise

1. You will find that the sentiment of the sentences has to be quite extreme for the app to decipher. Can you change the sentiment thresholds in the code for the app to be more sensitive?
2. Can you integrate the key phrase extraction API to extract keywords from any text sent to the bot?