

A Chatbot with Microsoft Azure

1. Introduction:

A chatbot is an application that can initiate and continue a conversation using auditory and/or textual methods as a human would do. A chatbot can be either a simple rule-based engine or an intelligent application leveraging Natural Language Understanding. Many organizations today have started using chatbots extensively. Chatbots are becoming famous as they are available 24*7, provide a consistent customer experience, can handle several customers at a time, are cost-effective and hence, result in a better overall customer experience.

1.1 Uses

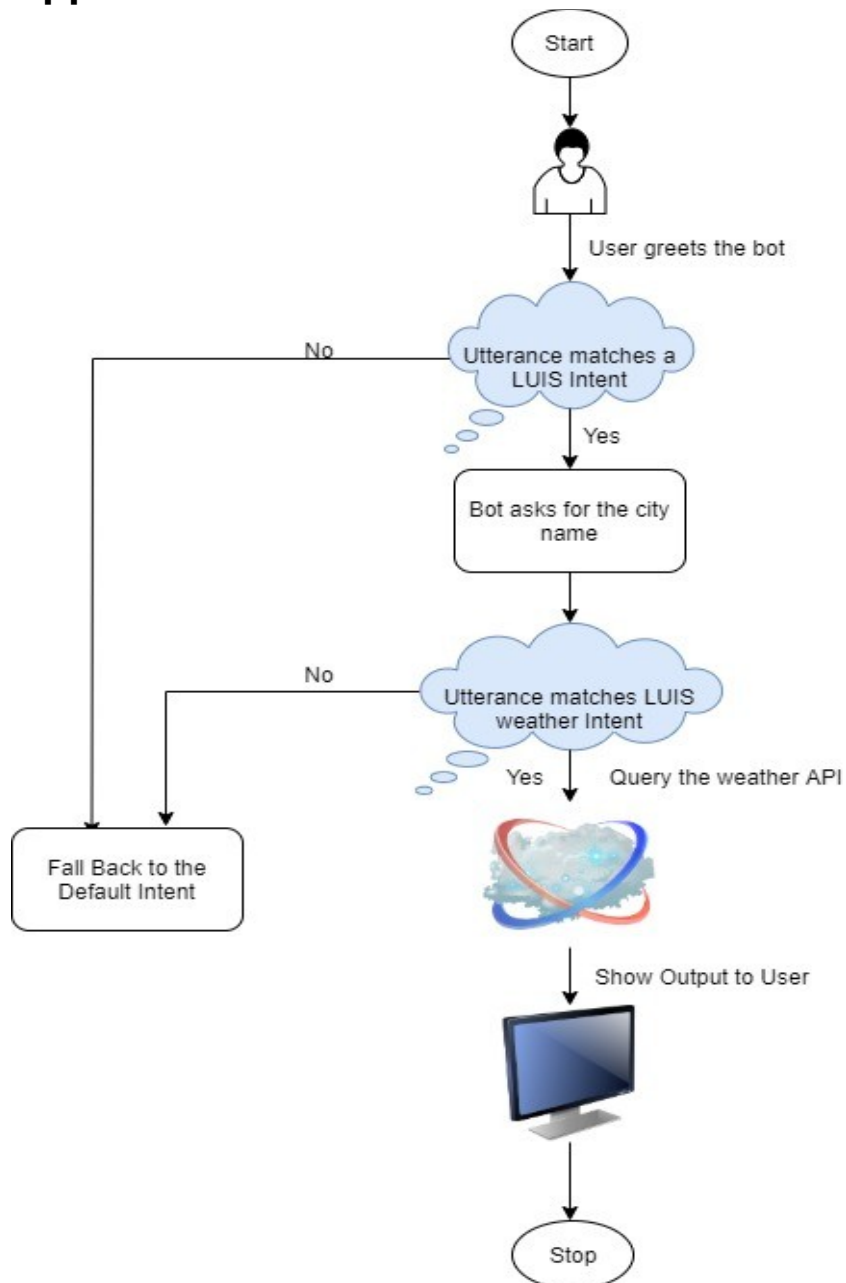
- Customer support
- Frequently Asked Questions
- Addressing Grievances
- Appointment Booking
- Automation of routine tasks
- Address a query

2. Prerequisites

The prerequisites for developing and understanding a chatbot using Microsoft Azure are:

- An Azure account.
- A fundamental understanding of python and flask

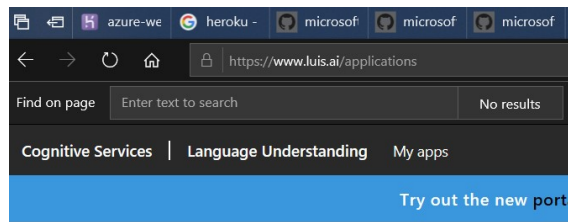
3. Application Architecture



4. Implementation

4.1 Creating a LUIS App

- Go to <https://www.luis.ai> and create an account if you already don't have one.
- Click on 'create new app' to create a new app by as shown:



My Apps ?

+ Create new app ↑ Import new app

☐ Name

[weatherBot](#) (V 0.1)

[MovieTicketBooking](#) (V 0.1)

- Provide the following details and click 'Done'.

Create new app

Name (Required)

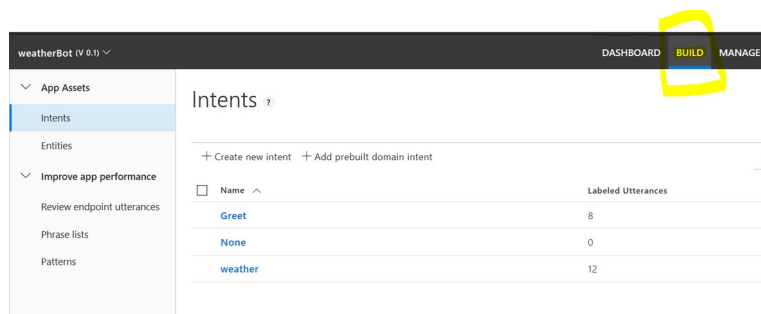
Culture (Required)

** Culture is the language that your app understands and speaks, not the interface language.

Description

Done Cancel

- Once created, open your app, select build, and click 'Create new intent' to create a new intent.



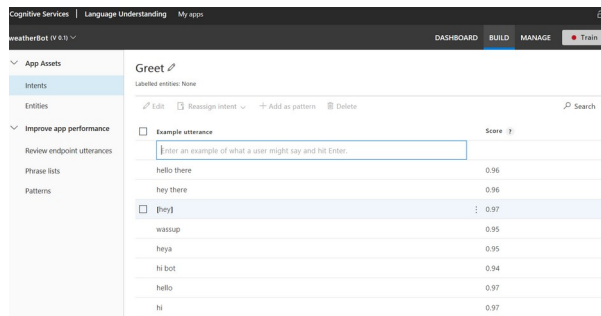
- Enter the name of the intent.

Create new intent

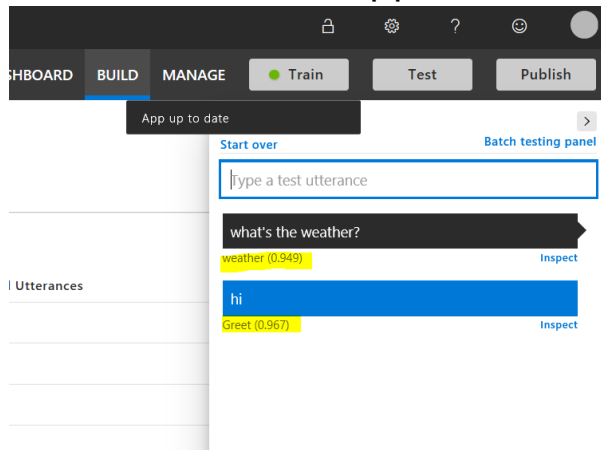
Intent name (Required)

Done Cancel

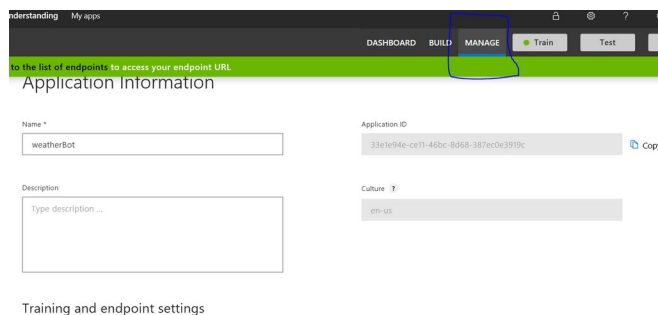
- Enter the user utterances and then click 'train' to train the LUIS app.



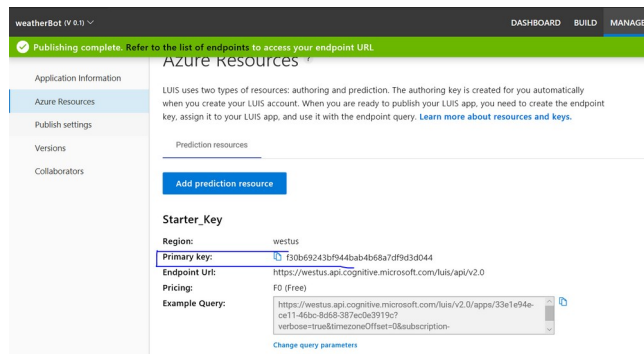
- Click 'Test' to test the intent and see the confidence of the app for various utterances.



- If satisfied with the test, click 'publish' and select the environment to make the LUIS app ready for consumption.
- Go to the *Manage* section of the published app and copy the *Application ID*. It will act as the 'LUIS App ID.'



- Go to the *Azure Resources* section and copy the *Primary key*. It will serve as the LUIS API KEY.



4.2 Create a python app

4.2.1 Subscribing to the weather API

- Go to <https://home.openweathermap.org/>, sign in/signup, and create an API Key for calling the *current weather data* API.
- This will act as the *weather_api_key*.

4.2.2 App creation and Integration with LUIS

- Create a folder for your chatbot called azurePythonBot.
- Open the folder through pycharm.
- Create a file called app.py and put the following code.

```
from flask import Flask, request, Response
from botbuilder.core import BotFrameworkAdapter,
BotFrameworkAdapterSettings,
ConversationState, MemoryStorage
from botbuilder.schema import Activity
import asyncio
from luis.luisApp import LuisConnect
import os
from logger.logger import Log

app = Flask(__name__)
loop = asyncio.get_event_loop()

bot_settings = BotFrameworkAdapterSettings("", "")
bot_adapter = BotFrameworkAdapter(bot_settings)

#CON_MEMORY = ConversationState(MemoryStorage())
luis_bot_dialog = LuisConnect()

@app.route("/api/messages", methods=["POST"])
def messages():
    if "application/json" in request.headers["content-
type"]:
        log=Log()
        request_body = request.json
        user_says = Activity().deserialize(request_body)
```

```

log.write_log(sessionID='session1',log_message="user
says: "+str(user_says))
    authorization_header =
(request.headers["Authorization"] if "Authorization" in
request.headers else "")

    async def call_user_fun(turncontext):
        await luis_bot_dialog.on_turn(turncontext)

    task = loop.create_task(
        bot_adapter.process_activity(user_says,
authorization_header, call_user_fun)
    )
    loop.run_until_complete(task)
    return ""
else:
    return Response(status=406) # status for Not
Acceptable

if __name__ == '__main__':
    #app.run(port= 3978)
    app.run()

```

- Create a folder logger and create logger.py inside as shown below:

```

from datetime import datetime
class Log:
    def __init__(self):
        pass

    def write_log(self, sessionID, log_message):
        self.file_object =
open("conversationLogs/"+sessionID+".txt", 'a+')
        self.now = datetime.now()
        self.date = self.now.date()
        self.current_time = self.now.strftime("%H:%M:%S")
        self.file_object.write(
            str(self.date) + "/" + str(self.current_time)
+ "\t\t" + log_message + "\n")
        self.file_object.close()

```

- Create a folder luis and create luisApp.py inside as shown below:

```

from botbuilder.core import TurnContext,ActivityHandler
from botbuilder.ai.luis import
LuisApplication,LuisPredictionOptions,LuisRecognizer
import json
from weather.weatherApp import WeatherInformation
from config.config_reader import ConfigReader
from logger.logger import Log

```

```

class LuisConnect(ActivityHandler):
    def __init__(self):
        self.config_reader = ConfigReader()
        self.configuration =
self.config_reader.read_config()

self.luis_app_id=self.configuration['LUIS_APP_ID']
        self.luis_endpoint_key =
self.configuration['LUIS_ENDPOINT_KEY']
        self.luis_endpoint =
self.configuration['LUIS_ENDPOINT']
        self.luis_app =
LuisApplication(self.luis_app_id,self.luis_endpoint_key,s
self.luis_endpoint)
        self.luis_options =
LuisPredictionOptions(include_all_intents=True,include_in
stance_data=True)
        self.luis_recognizer =
LuisRecognizer(application=self.luis_app,prediction_optio
ns=self.luis_options,include_api_results=True)
        self.log=Log()

    async def
on_message_activity(self,turn_context:TurnContext):
        weather_info=WeatherInformation()
        luis_result = await
self.luis_recognizer.recognize(turn_context)
        result = luis_result.properties["luisResult"]
        json_str =
json.loads((str(result.entities[0])).replace("'", "\'"))

weather=weather_info.get_weather_info(json_str.get('entit
y'))

self.log.write_log(sessionID='session1',log_message="Bot
Says: "+str(weather))
        await turn_context.send_activity(f"{weather}")

```

- Create a folder weather and create weatherApp.py inside it as shown below:

```

import pyowm
from config.config_reader import ConfigReader

class WeatherInformation():
    def __init__(self):
        self.config_reader = ConfigReader()
        self.configuration =
self.config_reader.read_config()
        self.owmapikey =
self.configuration['WEATHER_API_KEY']
        self.owm = pyowm.OWM(self.owmapikey)

    def get_weather_info(self,city):
        self.city=city

```



```

        observation = self.owm.weather_at_place(city)
        w = observation.get_weather()
        latlon_res = observation.get_location()
        lat = str(latlon_res.get_lat())
        lon = str(latlon_res.get_lon())

        wind_res = w.get_wind()
        wind_speed = str(wind_res.get('speed'))

        humidity = str(w.get_humidity())

        celsius_result = w.get_temperature('celsius')
        temp_min_celsius =
str(celsius_result.get('temp_min'))
        temp_max_celsius =
str(celsius_result.get('temp_max'))

        fahrenheit_result =
w.get_temperature('fahrenheit')
        temp_min_fahrenheit =
str(fahrenheit_result.get('temp_min'))
        temp_max_fahrenheit =
str(fahrenheit_result.get('temp_max'))
        self.bot_says = "Today the weather in " + city
+ ".\n Maximum Temperature :"+temp_max_celsius+ " Degree
Celsius"+ ".\n Minimum Temperature :"+temp_min_celsius+ "
Degree Celsius" + ": \n" + "Humidity :" + humidity + "%"
        return self.bot_says

```

- Create a file config.ini and put the following details:

```

[DEFAULT]
WEATHER_API_KEY=your weather api key
LUIS_APP_ID=your LUIS app id
LUIS_ENDPOINT_KEY=your LUIS endpoint key
LUIS_ENDPOINT=your LUIS endpoint

```

- Create a folder config and and create config_reader.py inside as shown below:

```

import configparser

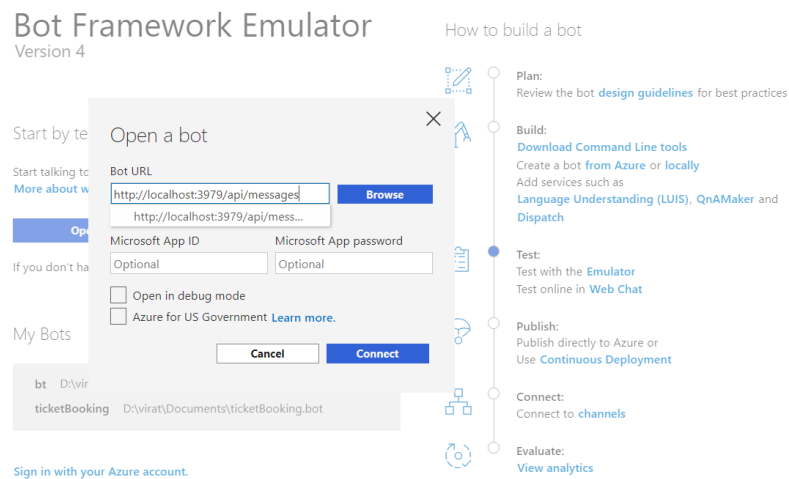
class ConfigReader:
    def __init__(self):
        self.filename = 'config.ini'
    def read_config(self):
        self.config = configparser.ConfigParser()
        self.config.read(self.filename)
        self.configuration=self.config['DEFAULT']
        return self.configuration

```

4.3 Install Bot Emulator and test

- Go to <https://github.com/Microsoft/BotFramework-Emulator/releases> and download the Bot Emulator setup file based on your computer.

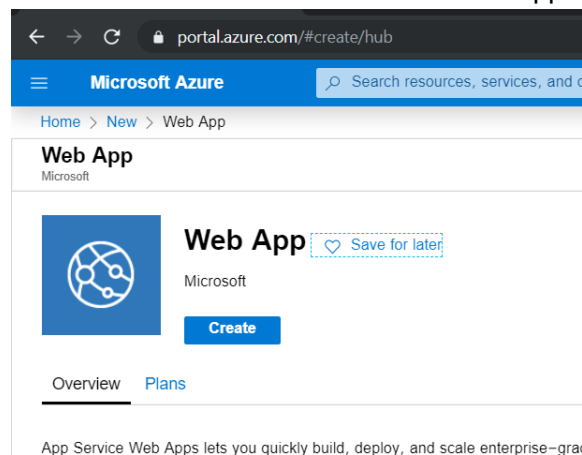
- Once the download is completed, double click the installation file and it'll automatically install the Bot Emulator.
- Run the bot emulator and connect to the already running bot file(app.py) as shown:



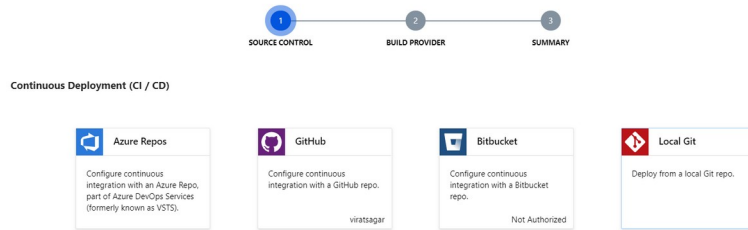
Give the port number on which your app is running.

4.4 Deploy to azure

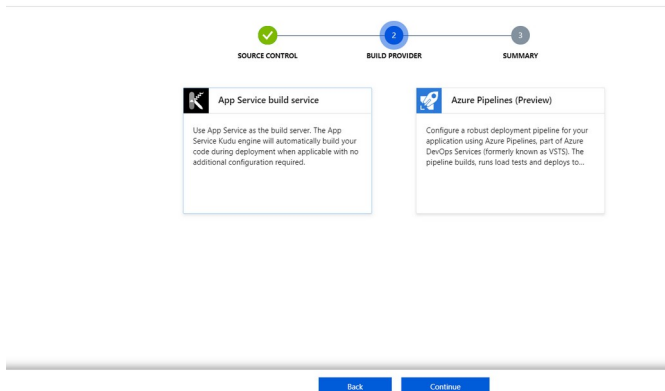
- Go to the Azure account and create a web app.



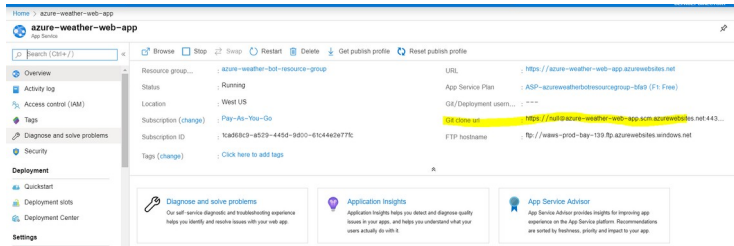
- Provide the app name, resource group(create new if necessary), runtime stack(Python <version>), region, select the 1 GB size, which is free to use. Click **Review+create** to create the web app.
- Once the deployment is completed, open the app and go to the 'Deployment Center' option. Select 'local git' for source control and click continue.




- Select the kudo 'App service build provider' as the build provider and click continue.



- Click 'Finish' to complete the setup.
- Go to the overview section of the app, and the Git link now will be visible.



- Open a command prompt and navigate to your project folder.
- Run **git init** to initialize an empty git repository
- Create a new remote git alias using the command: **git remote add <alias> <git clone url>**
- Use **git add .** to add all the files to the local git repository.
- Use **git commit -m "First Commit"** to commit the code to the git repo.
- Push the code to the remote repo using **git push <alias> master**
- This prompts for a username and password. Go to the 'Deployment Credentials' section and copy the username and password to enter in the prompt.



Deployment Credentials

✕

App Service supports multiple technologies to access, publish and modify the content of your app. Deployment credentials can be scoped to the application or the user.

Refresh
Get publish profile
Reset publish profile

Local Git

FTP/FTPS

Git Clone URL

https://azure-weather-web-app.scm.azurewebsites.net:443/azure-weather-web-
Copy

Scope

App Credentials
User Credentials

Application Credentials are auto-generated and provide access only to this specific app or deployment slot. These credentials can be used with FTP, Local Git and WebDeploy. They cannot be configured manually, but can be reset anytime. [Learn more](#)

Username

\$azure-weather-web-app
Copy

Password


.....
Show Copy

- Once the credentials are correctly entered, the app deployment to azure is completed.

4.5 Create bot channel Registration

- Now in the Azure portal, create a bot channel registration.
- Provide the bot handle, resource group and other fields.
- For Message endpoint, provide: <URL from the web app created above>api/messages. Click Create to create the bot.
- Once your web channel registration gets done, open the bot and then click 'test in web chat.' If the chat works fine, our deployment is a success.

Home > azure-weather-bot - Test in Web Chat


azure-weather-bot - Test in Web Chat
Bot Channels Registration

Search (Ctrl+/)
Test
Start over

Overview
Activity log
Access control (IAM)
Tags

Bot management
Test in Web Chat
Analytics
Channels
Settings
Speech priming
Bot Service pricing

Support + troubleshooting
New support request

hi
Just now

Hello! This is a weather bot. I can help you know the weather of any city.
Just now

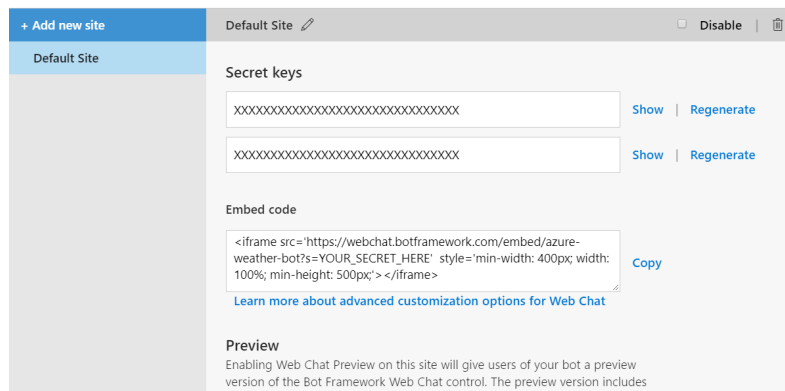
What's the weather in Mumbai?
Just now

The weather details for Mumbai is-- Maximum Temperature(in Degree Celsius): 29, Minimum Temperature(in Degree Celsius): 29, Humidity : 66%, Forecast: haze, Wind Speed: 2.6 Km/h
Just now

Enter your message

4.6 Deployment

- Go to the channels section of your bot.
- The bot can be deployed as an embedding to an existing HTML page by selecting the get bot embedded code option



The screenshot shows the 'Default Site' configuration page for a bot. It includes a sidebar with '+ Add new site' and 'Default Site' options. The main content area has a 'Secret keys' section with two input fields containing masked text and 'Show'/'Regenerate' links. Below is an 'Embed code' section with a text area containing an iframe snippet and a 'Copy' button. At the bottom is a 'Preview' section with explanatory text.

4.6.1.1 Telegram Deployment

- Open the telegram app and search for botfather(it is an inbuilt bot used to create other bots)
- Start a conversation with botfather and enter `/newbot` to create a newbot.
- Give a name to your bot
- Give a username to your bot, which must end in `_bot`.
- This generates an access token. Enter that access token after clicking the telegram channel in your bot app and click save.

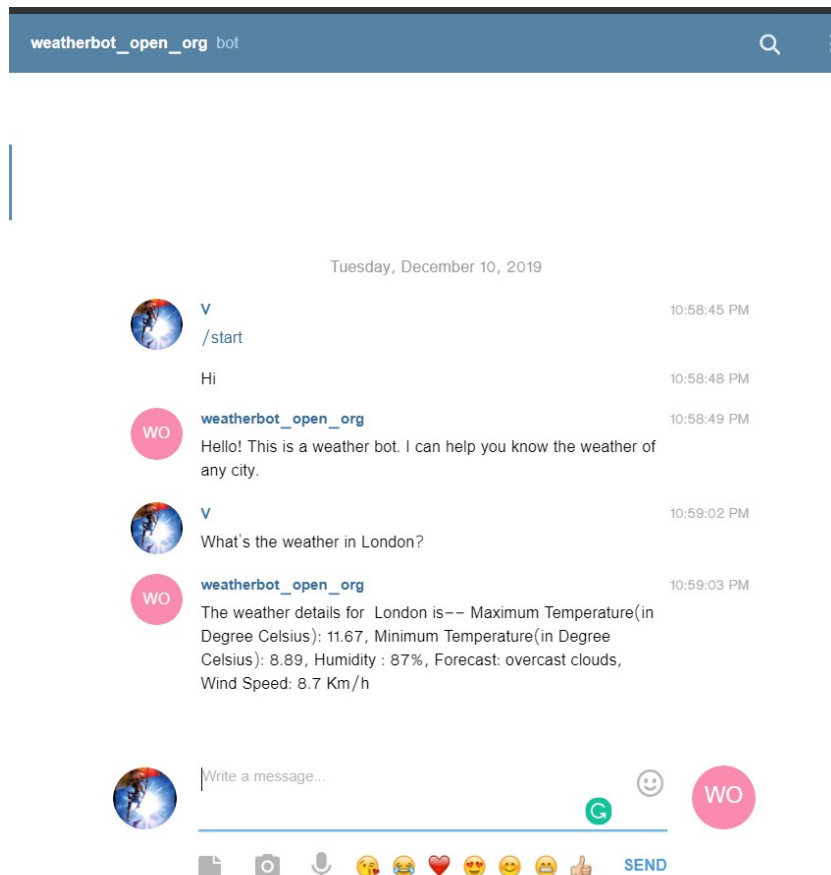
Configure Telegram



Enter your Telegram credentials
[Step-by-step instructions to add the bot to Telegram.](#)

Access Token *

- Now, search the username of the bot in telegram and start conversation with your bot.



Thank You!

4.7