Introduction :

In this article, we will learn how to Build A WhatsApp Bot Using Python Bot using Python. A WhatsApp bot is a software application capable of communicating with users in both written and spoken formats. Our bot will use the Twilio API to connect with WhatsApp and perform tasks like sending and receiving messages, notifications, and more.

We'll use the Python library Flask to handle incoming and outgoing messages and Ngrok to expose our Flask application to the public internet so that Twilio can communicate with it.

Required Modules and Packages:

To build this WhatsApp bot, you will need the following:

1. **A Twilio Account**: Sign up at Twilio and get a phone number with WhatsApp enabled.
2. **Python 3.9 or Newer**: Make sure you have Python installed on your system.
3. **Flask**: A lightweight web framework for Python that will be used to handle incoming HTTP requests from Twilio.
4. **Ngrok**: A tool that will help expose your Flask application running on your local machine to the internet.

How To Run The Code :

Follow these steps to set up and run the WhatsApp bot:

Step 1: Set up a Twilio account by visiting Twilio WhatsApp. Sign up and verify your email ID and phone number.

Step 2: Once logged in, navigate to the **Develop** section in the left menu, select **Messaging**, and then choose **Try it out** -> **Send a WhatsApp message**. This will take you to a page for setting up the WhatsApp Sandbox.

Step 3: Configure the WhatsApp Sandbox by sending a message to the provided WhatsApp number (+14155238886) with a unique security code like `join <secret-code>`.

Step 4: Once you send the security code, you will receive a confirmation message in WhatsApp.

Step 5: Open the terminal and run the following commands to create a directory for the bot, set up a virtual environment, and install the required packages:

- **Create a Directory and Navigate to It**:

```
mkdir whatsapp-bot && cd whatsapp-bot
```

- **Create and Activate a Python Virtual Environment**:

```
python3 -m venv bot-env && source bot-env/bin/activate
```

- **Install Twilio, Flask, and Requests**:

  ```
  pip install twilio flask requests
  ```

Improved Code Details

1. **Flask Setup**: The script initializes a Flask application and sets up a single route `/bot` that listens for incoming HTTP POST requests from Twilio.
2. **Receiving Messages**: The script captures the incoming message from WhatsApp using `request.values.get('Body', '')` and stores it in the `user_msg` variable.
3. **Responding to Messages**: The script checks if the user's message contains certain keywords (like 'help') and responds with predefined messages. If the user asks a specific question, the script performs a Google search using the `googlesearch` package to find the top 5 relevant URLs and sends them back to the user.
4. **Sending Responses**: The `MessagingResponse` object from the Twilio library is used to send the response back to WhatsApp.

How to Run the Bot

1. **Run the Python Script**:

   ```
   python main.py
   ```

2. **Start Ngrok Server**: In a separate terminal window, run:

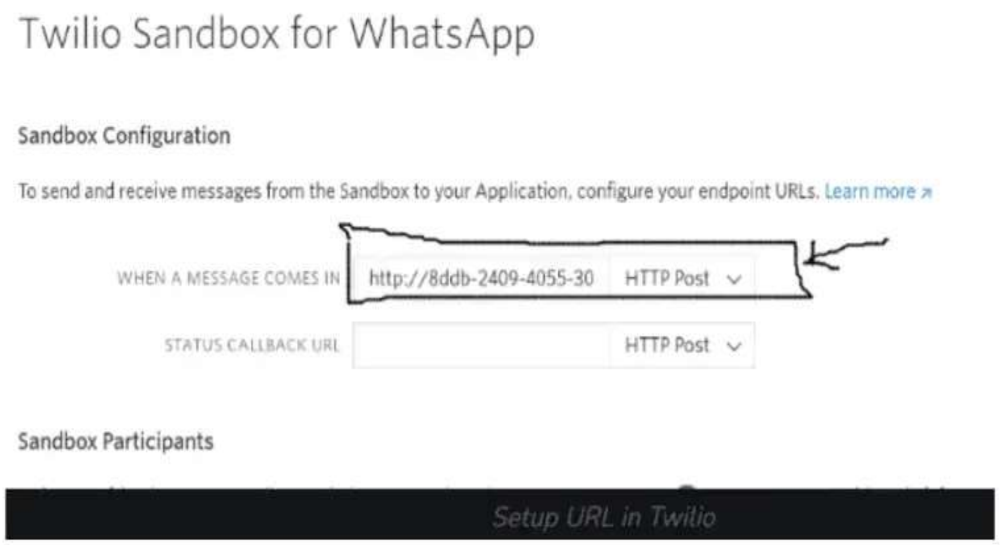
   ```
   ngrok http 5000
   ```

   This command will expose the Flask application to the internet, allowing Twilio to connect to it.

3. **Set Up the Forwarding URL**: Copy the forwarding URL provided by Ngrok (it will look like `http://<random-string>.ngrok.io`) and paste it into the **WhatsApp Sandbox** settings at Twilio Sandbox Settings.

Testing the WhatsApp Bot

Now that everything is set up, send a WhatsApp message to the Twilio number. The bot will receive your message, perform a Google search for the text you sent, and reply with the top 5 results.



Twilio Sandbox for WhatsApp

Sandbox Configuration

To send and receive messages from the Sandbox to your Application, configure your endpoint URLs. Learn more

WHEN A MESSAGE COMES IN: http://8ddb-2409-4055-30 HTTP Post

STATUS CALLBACK URL: HTTP Post

Sandbox Participants

Setup URL in Twilio

Conclusion

Congratulations! You have successfully built a WhatsApp bot using Python, Flask, and Twilio. This bot can receive messages, perform tasks like web searches, and send automated responses. You can further expand the bot’s capabilities by adding more complex logic, integrating it with different APIs, or connecting it to a database to store user data.