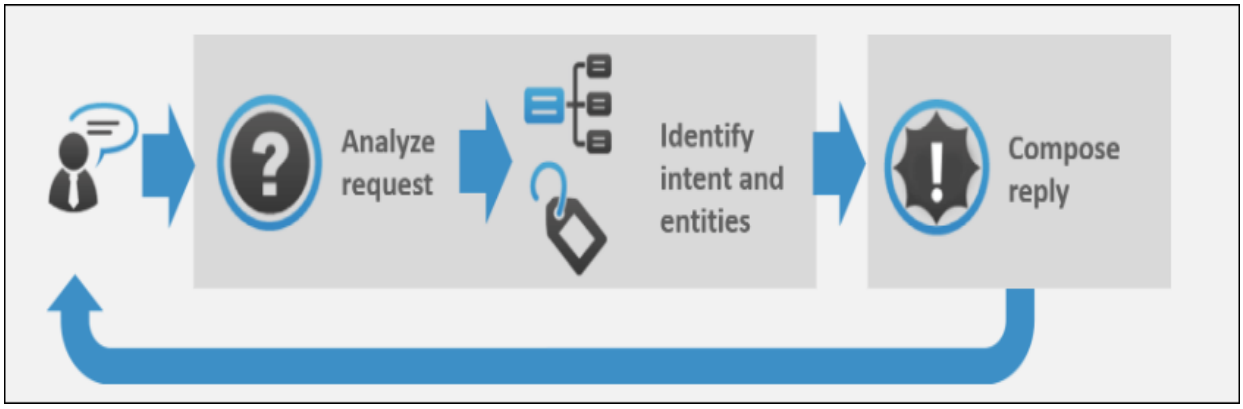
Building A ChatBot With Amazon Lex

Chatbot Concept

What are chatbots?

A Chatbots is an intelligent software that can simulate a conversation (or a chat) with a user in natural language through messaging applications, websites, mobile phones or telephone. They can be programmed to respond to simple keywords or prompts to complex conversations.

A Chatbot has two different tasks at the core: user request analysis (identifying the intent of the user) and returning the response.



There are two types of chatbots: one functions on a set of rules and the other more advanced one uses artificial intelligence.

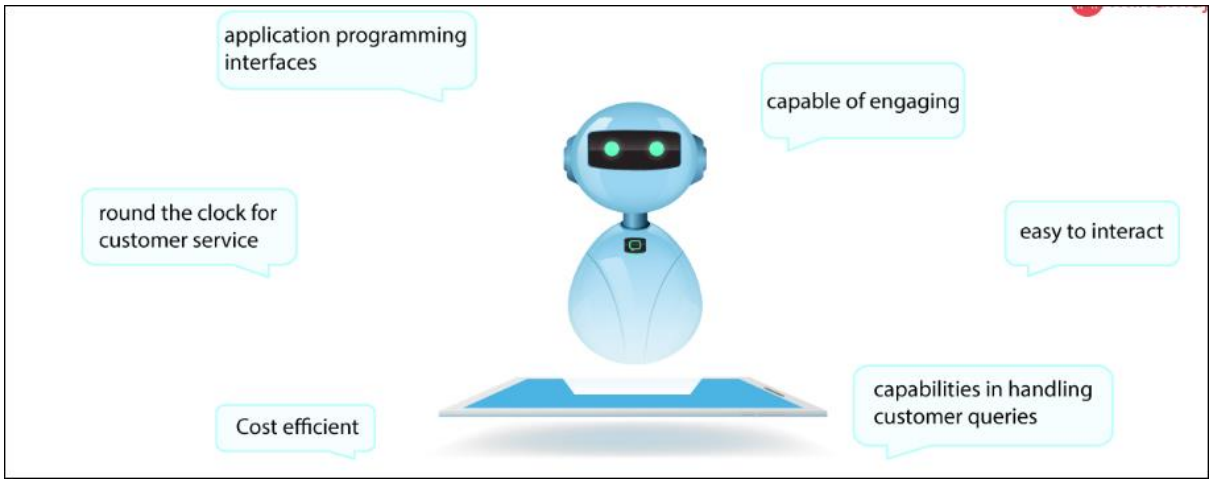
1. **Chatbots that function on rules**: It is easy to build. Though simplistic, it can get basics tasks done with limitations to specific commands. This bot is only as smart as it is programmed.
2. **Chatbots that function on machine learning**: It understands languages and not just commands and provides relevant responses. It communicates through speech or text relying on AI technologies like machine learning and NLP. It gets smarter as it learns from the conversations from users.

Use cases

A chatbot can be used in many different ways. Any business or industry having a website or an app can have a chatbot. Following are some of the chatbot applications, out of the infinite possibilities:

1. A takeaway restaurant that allows customers to order from their office or home.
2. A chatbot that answers customer service questions and provides help with different tasks.
3. A chatbot that allows the customer to book a flight and receive relevant information.
4. A chatbot that helps a customer with ecommerce purchases.
5. A marketing campaign chatbot that asks questions to the customer.
6. A health bot that provides services for booking a Doctor consultation.
7. A movie ticket booking bot.

Advantages of Chatbots



The advantages of chatbots are as follows:

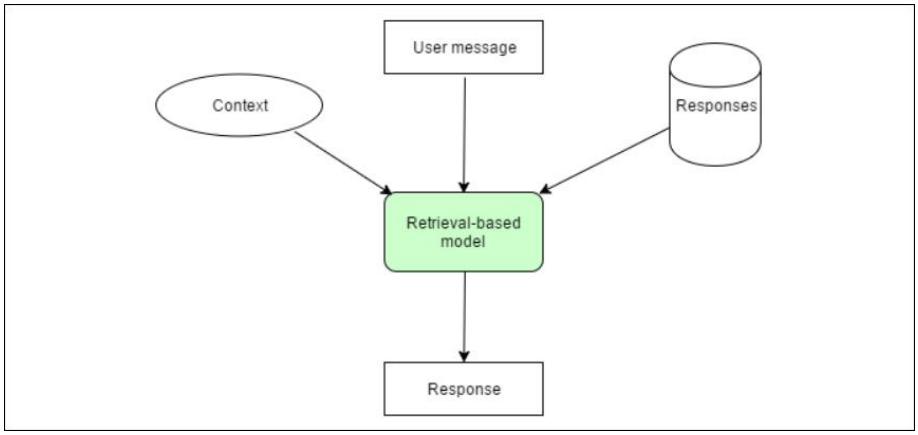
- They are capable of engaging customers in a friendly manner.
- A simple interface makes it easier to interact with.
- Interactions are made possible via social media platforms like Twitter, Facebook, etc. Chats are served through application programming interfaces (APIs).
- Chatbots have extraordinary capabilities in handling customer queries.
- Chatbots work efficiently round the clock for customer service.
- Cost-efficient and are easy to build.

Models for a Chatbot

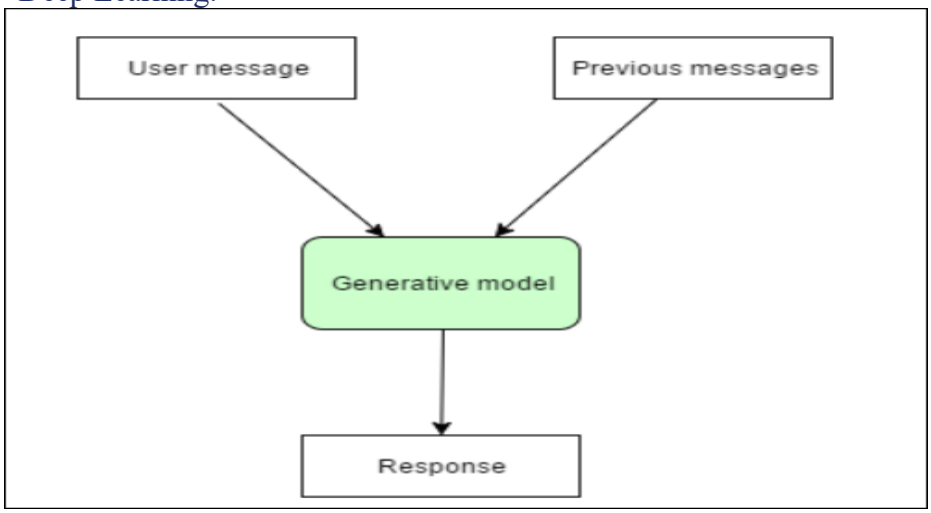
When creating a chatbot, the goal is to automate the process altogether and to reduce human intervention. The first step in creating a chatbot involves inputting thousands of existing interactions between customers and customer service representatives to teach the machine which words/phrases are sensitive to the business. The next step is preprocessing which involves incorporating grammar to identify any misspellings.

Then the next step is to identify the kind of chatbot we would like to implement. There mainly are two models:

1. **Retrieval-based model**- These models are much easy to build and give more predictable responses. They make use of context in the conversation for selecting the best response from a predefined list of messages that they got trained from. The context includes all the previous conversations, and the saved variables.



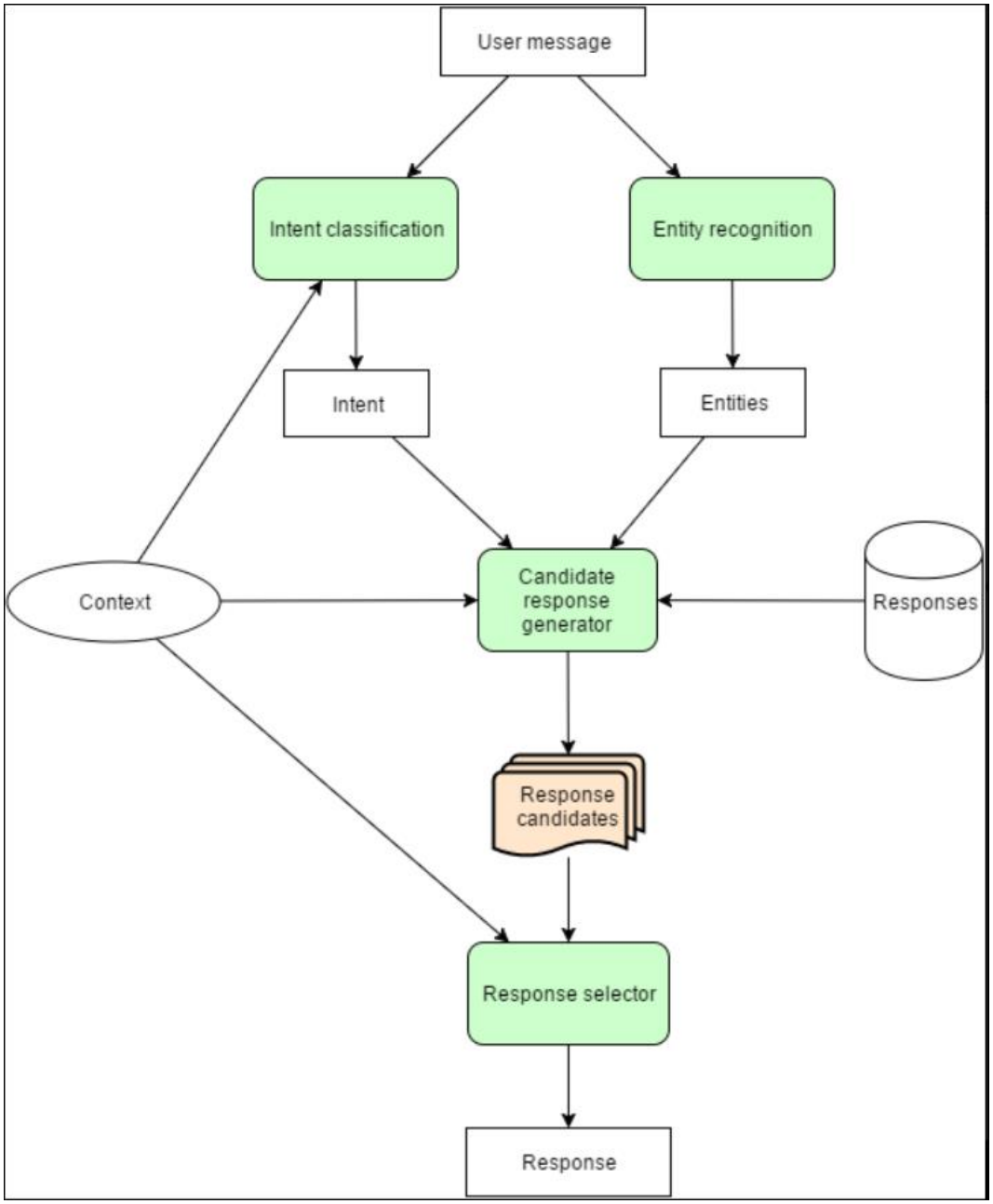
2. **Generative based model**: A generative model chatbot doesn't use any predefined repository. This kind of chatbot is more advanced because it learns from scratch using "Deep Learning."



Pros and Cons:

Generative models	Selective models
+ Can possibly generate arbitrary answer (more similar to general AI) + Can generate answer in correct grammar form (e.g. with correct speaker gender) - Can generate answer with incorrect grammar/syntax - Prone to "general answer" problem - Difficult to impose properties on model replies (e.g. no obscene words, speak like some specific person), but possible!	- Restricted pool of answers which can not cover all dialogue topics - For context "What is your name, girl?" can select "My name is Stephen." (inconsistency) + Predefined answers have good grammar/syntax + Less prone to "general answer" problems + You can customize answers for your own needs (without obscenities, kind answers)

Architecture of AI chatbot



Message processing begins with understanding what the user is talking about.

1. An intent classification module identifies the intent of the user messages. Typically, it is a selection of one out of many predefined intents, though more sophisticated bots can identify multiple intents from one message.

2. An entity recognition module extracts structured bits of information from the message. The weather bot can extract the location and date.

3. The candidate response generator is doing all the domain-specific calculations to process the user request. It can use different algorithms, call a few external APIs, or even ask a human to help with response generation. The result of these calculations is a list of response candidates. All these responses should be correct according to the domain-specific logic. They can't be just tons of random responses. The response generator must use the context of the conversation as well as intent and entities extracted from the last user message. Otherwise, it can't support multi-message conversations.

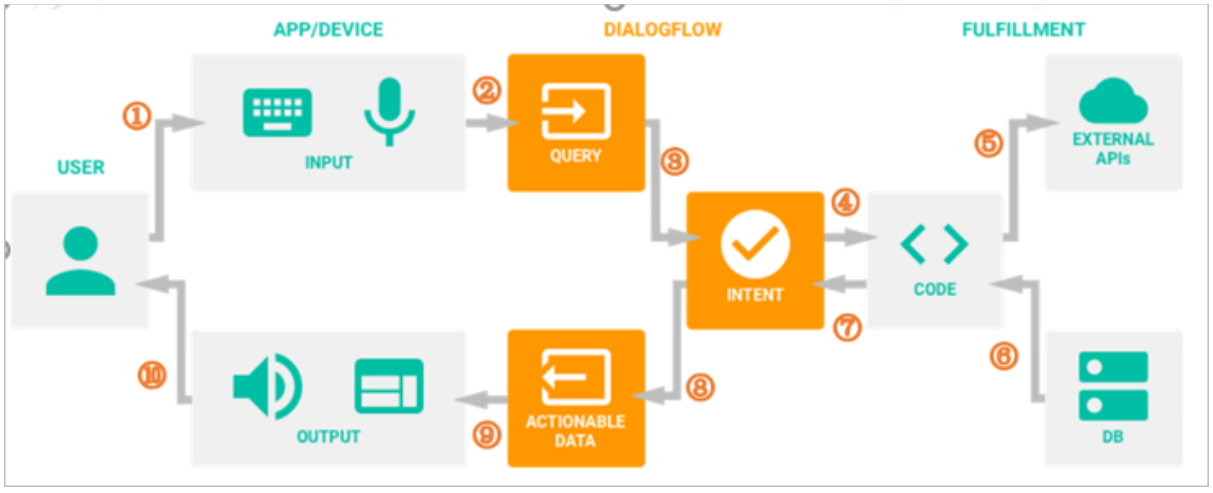
4. The response selector just scores all the response candidate and selects a response which should work better for the user.

Amazon Lex

Introduction

Amazon Lex is an AWS service for building conversational interfaces for applications using voice and text.

How do Chatbots work?



Detailed steps:

1. A user sends a text/voice message to a device or an App
2. The App/Device transfers the message to Lex
3. The message is categorized and matched to a corresponding intent (Intents are defined manually by developers in Lex)
4. We define actions for each intent in the corresponding fulfillment (Webhook).
5. When Lex finds a specific intent, the webhook will use external APIs to find a response from external databases.
6. The external databases send back the required information to the webhook.
7. Webhook sends a formatted response to the intent.

8. Intent generates actionable data according to different channels.
9. The actionable data go to output Apps/Devices.
10. The user gets a text/image/voice response.

What Is Amazon Lex?

Amazon Lex is an AWS service for building conversational interfaces for applications using voice and text. With Amazon Lex, the same conversational engine that powers Amazon Alexa is now available to any developer, enabling you to build sophisticated, natural language chatbots into your new and existing applications. Amazon Lex provides the deep functionality and flexibility of natural language understanding (NLU) and automatic speech recognition (ASR) so you can build highly engaging user experiences with lifelike, conversational interactions, and create new categories of products.

Amazon Lex enables any developer to build conversational chatbots quickly. With Amazon Lex, no deep learning expertise is necessary—to create a bot, you just specify the basic conversation flow in the Amazon Lex console. Amazon Lex manages the dialogue and dynamically adjusts the responses in the conversation. Using the console, you can build, test, and publish your text or voice chatbot. You can then add the conversational interfaces to bots on mobile devices, web applications, and chat platforms (for example, Facebook Messenger).

Amazon Lex provides pre-built integration with AWS Lambda, and you can easily integrate with many other services on the AWS platform, including Amazon Cognito, AWS Mobile Hub, Amazon CloudWatch, and Amazon DynamoDB. Integration with Lambda provides bots access to pre-built serverless enterprise connectors to link to data in SaaS applications, such as Salesforce, HubSpot, or Marketo.

Some of the benefits of using Amazon Lex include:

- **Simplicity** – Amazon Lex guides you through using the console to create your own chatbot in minutes. You supply just a few example phrases, and Amazon Lex builds a complete natural language model through which the bot can interact using voice and text to ask questions, get answers, and complete sophisticated tasks.

- **Democratized deep learning technologies** – Powered by the same technology as Alexa, Amazon Lex provides ASR and NLU technologies to create a Speech Language Understanding (SLU) system. Through SLU, Amazon Lex takes natural language speech and text input, understands the intent behind the input, and fulfills the user intent by invoking the appropriate business function.

Speech recognition and natural language understanding are some of the most challenging problems to solve in computer science, requiring sophisticated deep learning algorithms to be trained on massive amounts of data and infrastructure. Amazon Lex puts deep learning technologies within reach of all developers, powered by the same technology as Alexa. Amazon Lex chatbots convert incoming speech to text and understand the user intent to generate an intelligent response, so you can focus on building your bots with differentiated value-add for your customers, to define entirely new categories of products made possible through conversational interfaces.

- **Seamless deployment and scaling** – With Amazon Lex, you can build, test, and deploy your chatbots directly from the Amazon Lex console. Amazon Lex enables you to easily publish your voice or text chatbots for use on mobile devices, web apps, and chat services (for example, Facebook Messenger). Amazon Lex scales automatically so you don't need to worry about provisioning hardware and managing infrastructure to power your bot experience.

- **Built-in integration with the AWS platform** – Amazon Lex has native interoperability with other AWS services, such as Amazon Cognito, AWS Lambda, Amazon CloudWatch, and AWS Mobile Hub. You can take advantage of the power of the AWS platform for security, monitoring, user authentication, business logic, storage, and mobile app development.

- **Cost-effectiveness** – With Amazon Lex, there are no upfront costs or minimum fees. You are charged only for the text or speech requests that are made. The pay-as-you-go pricing and the low cost per request make the service a cost-effective way to build conversational interfaces. With the Amazon Lex free tier, you can easily try Amazon Lex without any initial investment.

Amazon Lex: How It Works

Amazon Lex enables you to build applications using a speech or text interface powered by the same technology that powers Amazon Alexa. Following are the typical steps you perform when working with Amazon Lex:

1. Create a bot and configure it with one or more intents that you want to support. Configure the bot so it understands the user's goal (intent), engages in conversation with the user to elicit information, and fulfills the user's intent.
2. Test the bot. You can use the test window client provided by the Amazon Lex console.

3. Publish a version and create an alias.
4. Deploy the bot. You can deploy the bot on platforms such as mobile applications or messaging platforms such as Facebook Messenger.

Before you get started, familiarize yourself with the following Amazon Lex core concepts and terminology:

- **Bot** – A bot performs automated tasks such as ordering a pizza, booking a hotel, ordering flowers, and so on. An Amazon Lex bot is powered by Automatic Speech Recognition (ASR) and Natural Language Understanding (NLU) capabilities, the same technology that powers Amazon Alexa.

  Amazon Lex bots can understand user input provided with text or speech and converse in natural language. You can create Lambda functions and add them as code hooks in your intent configuration to perform user data validation and fulfillment tasks.

- **Intent** – An intent represents an action that the user wants to perform. You create a bot to support one or more related intents. For example, you might create a bot that orders pizza and drinks. For each intent, you provide the following required information:

  - Intent name– A descriptive name for the intent. For example, OrderPizza.
  - Sample utterances – How a user might convey the intent. For example, a user might say "Can I order a pizza please" or "I want to order a pizza".
  - How to fulfill the intent – How you want to fulfill the intent after the user provides the necessary information (for example, place order with a local pizza shop). We recommend that you create a Lambda function to fulfill the intent.

  You can optionally configure the intent so Amazon Lex simply returns the information back to the client application to do the necessary fulfillment.

  In addition to custom intents such as ordering a pizza, Amazon Lex also provides built-in intents to quickly set up your bot. For more information, see Built-in Intents and Slot Types.

- **Slot** – An intent can require zero or more slots or parameters. You add slots as part of the intent configuration. At runtime, Amazon Lex prompts the user for specific slot values. The user must provide values for all *required* slots before Amazon Lex can fulfill the intent.

For example, the OrderPizza intent requires slots such as pizza size, crust type, and number of pizzas. In the intent configuration, you add these slots. For each slot, you provide slot type and a prompt for Amazon Lex to send to the client to elicit data from the user. A user can reply with a slot value that includes additional words, such as "large pizza please" or "let's stick with small." Amazon Lex can still understand the intended slot value.

- **Slot type** – Each slot has a type. You can create your custom slot types or use built-in slot types. For example, you might create and use the following slot types for the OrderPizza intent:

  - Size – With enumeration values Small, Medium, and Large.
  - Crust – With enumeration values Thick and Thin.

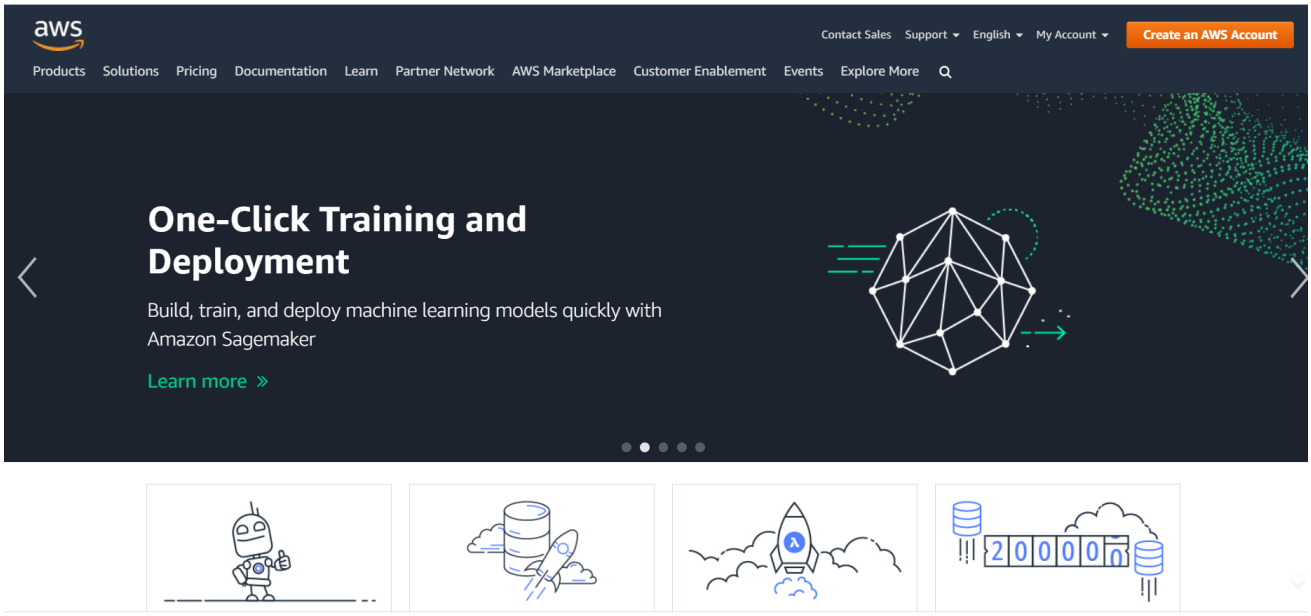
  Amazon Lex also provides built-in slot types. For example, AMAZON.NUMBER is a built-in slot type that you can use for the number of pizzas ordered. For more information, see Built-in Intents and Slot Types.

  Currently, Amazon Lex supports only US English language. That is, Amazon Lex trains your bots to understand only US English.

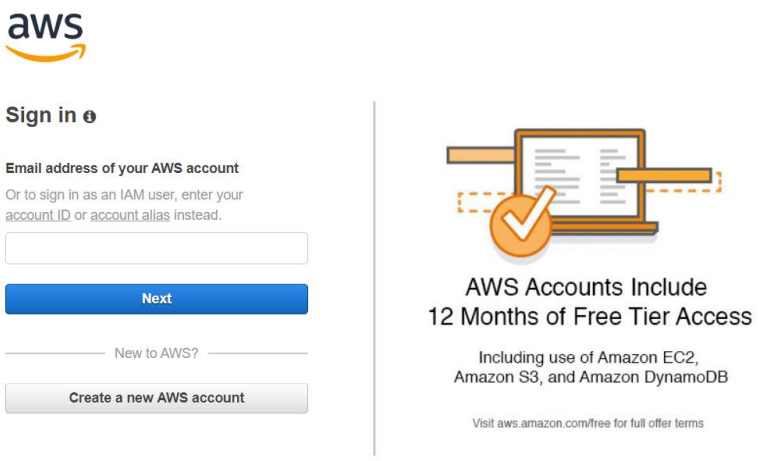
  (Source: https://docs.aws.amazon.com/lex/latest/dg/what-is.html)

Build your first chatbot

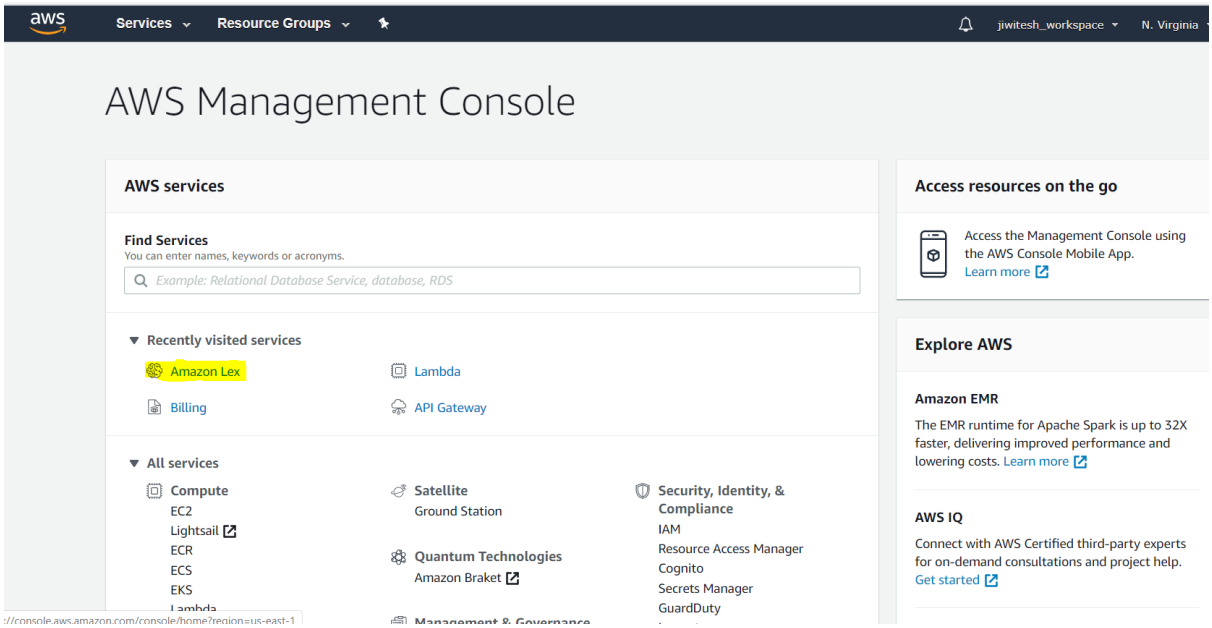
1) Login to AWS portal **https://signin.aws.amazon.com/**



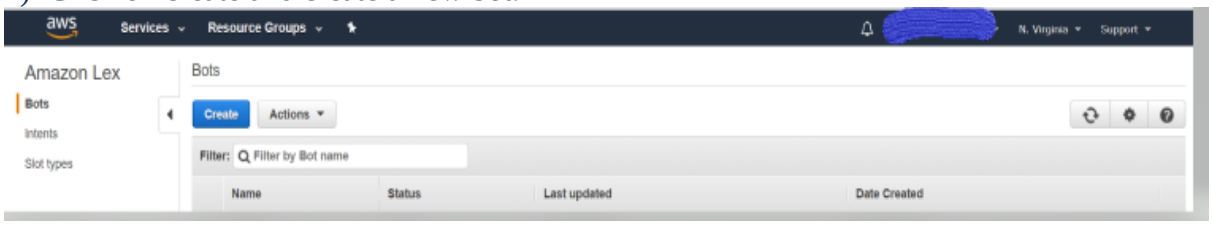
2) Click on “create an AWS Account”. Sign in with your credentials or create a new account.



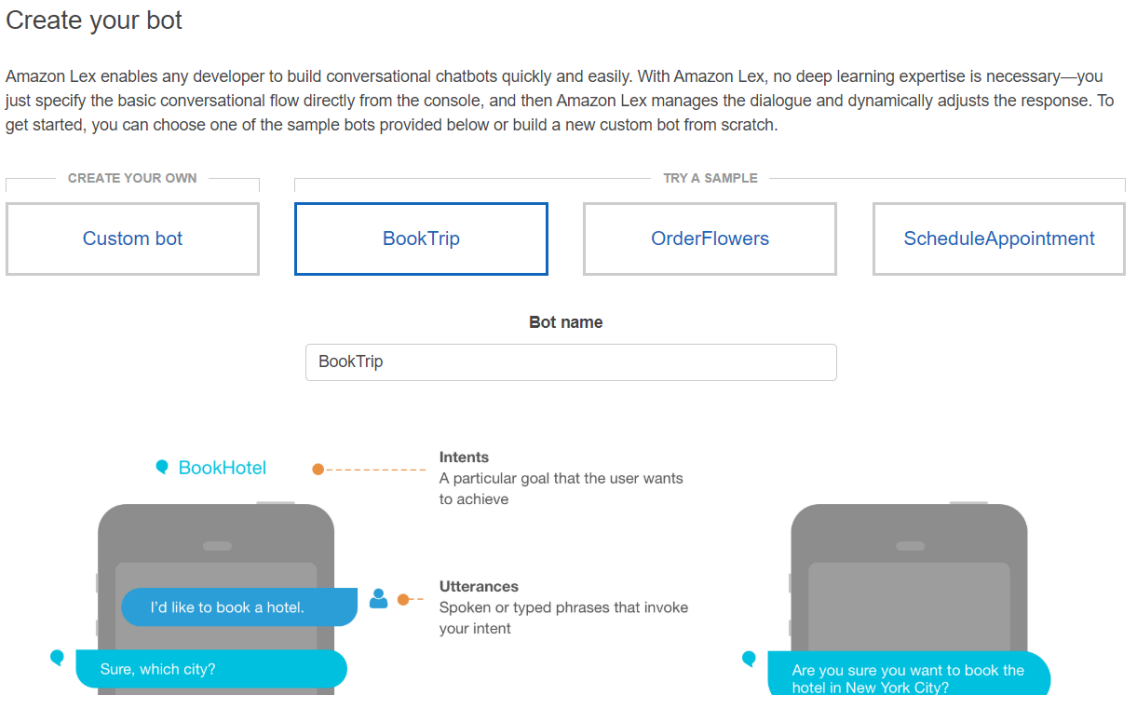
3) Once logged in, click on Amazon Lex to start creating your bot.



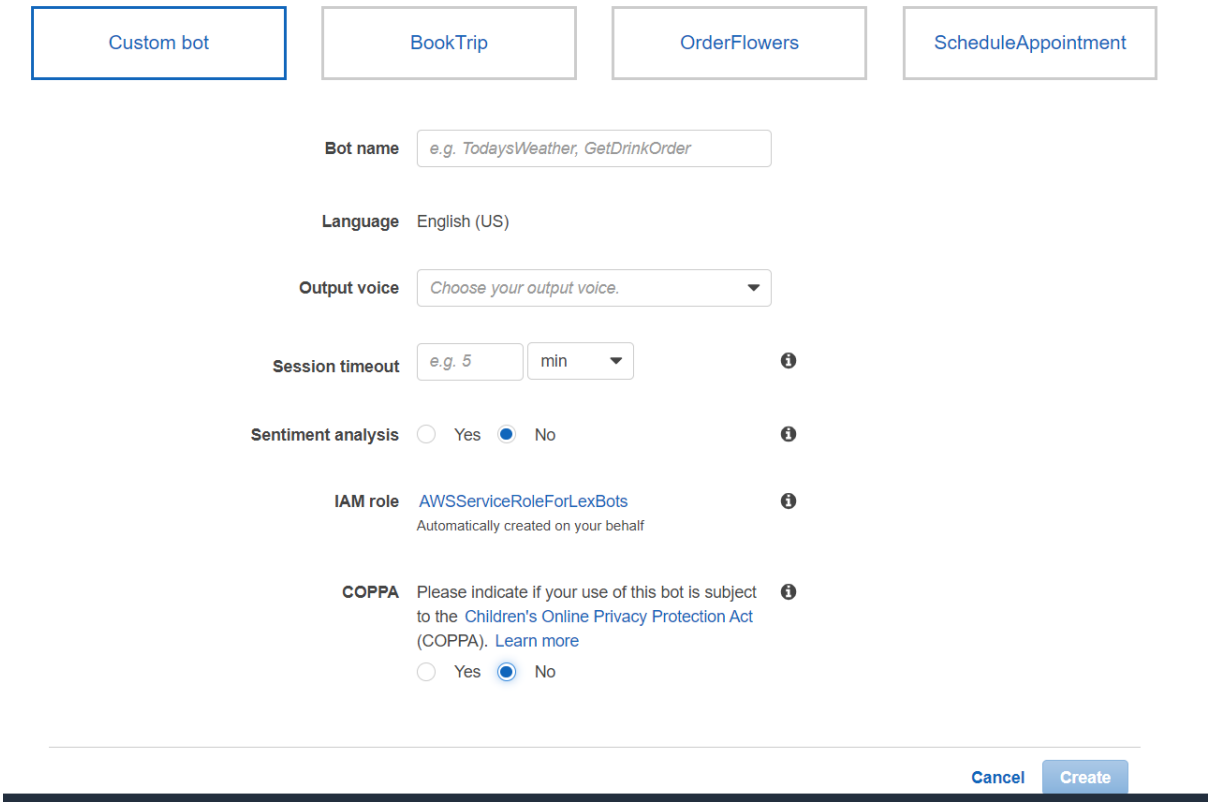
4) Click on create and create a new bot.



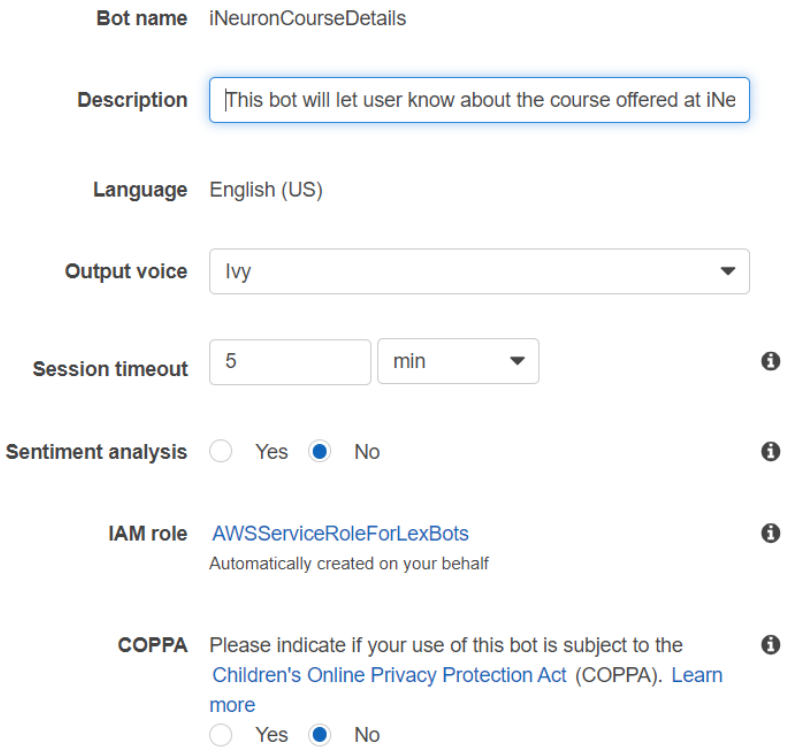
5) Once you click the create button, you will see a window like this. Click on the "custom bot" to make your own bot. The other bots are example bots and you can go through them to get a glance on some higher details.



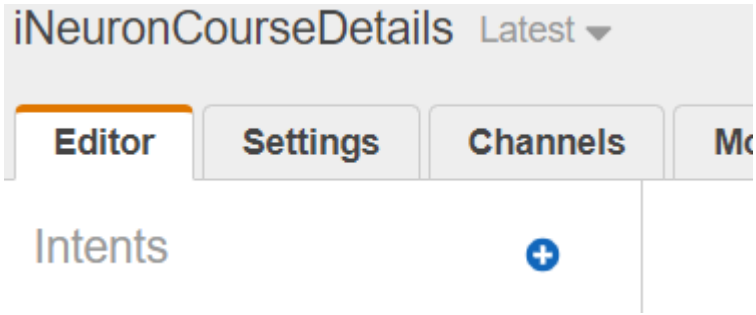
6) Fill up all the details and create on the create button.



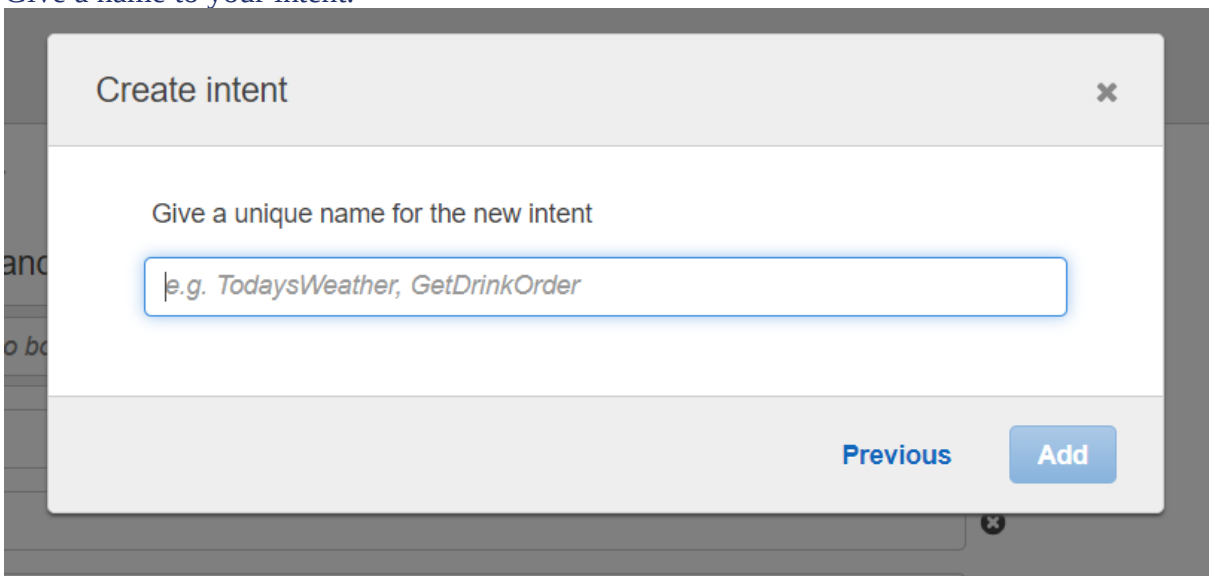
7) We have create a bot as follows:



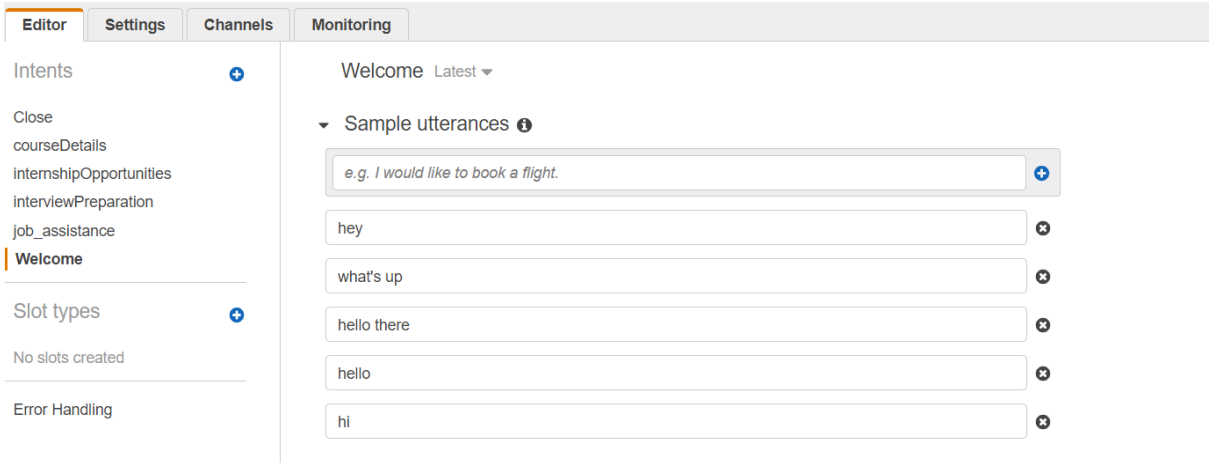
8) We will start by creating intents for our bot. Click on the plus button next to intents.

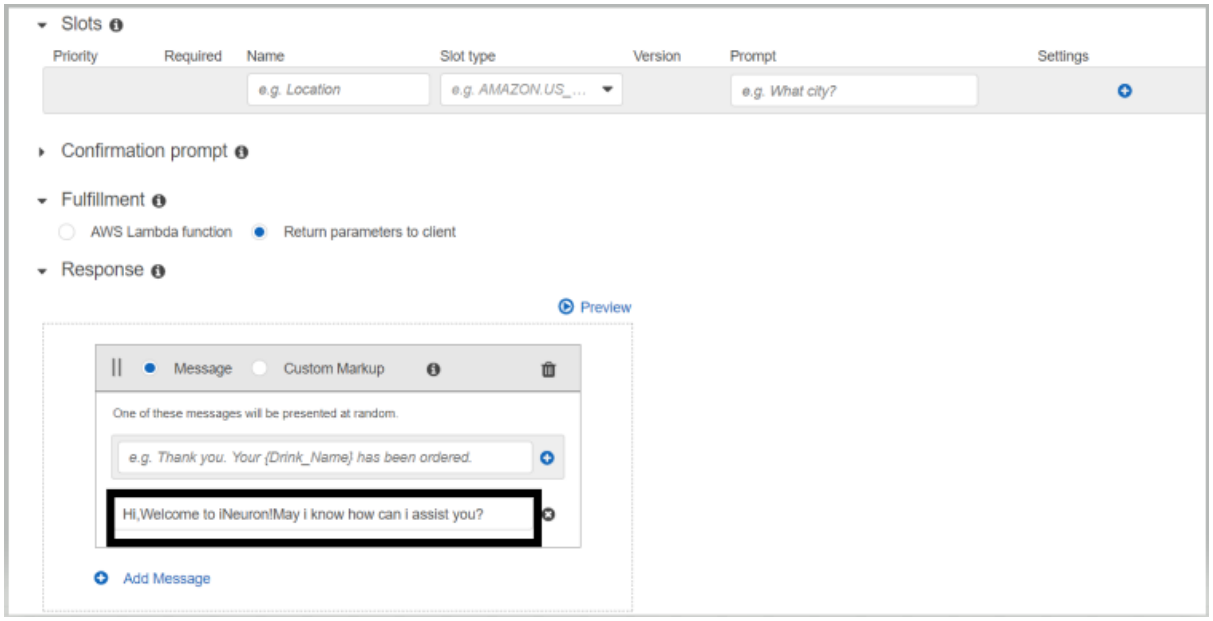


Give a name to your intent.



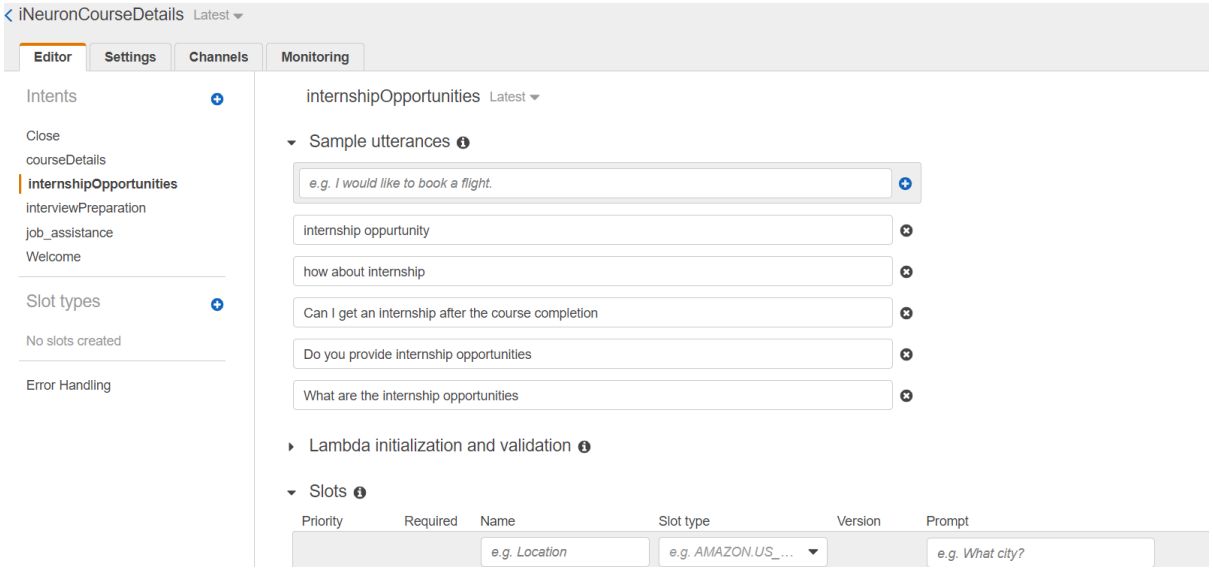
We are going to create a “Welcome” intent which will greet the user on hearing/reading the defined utterances.

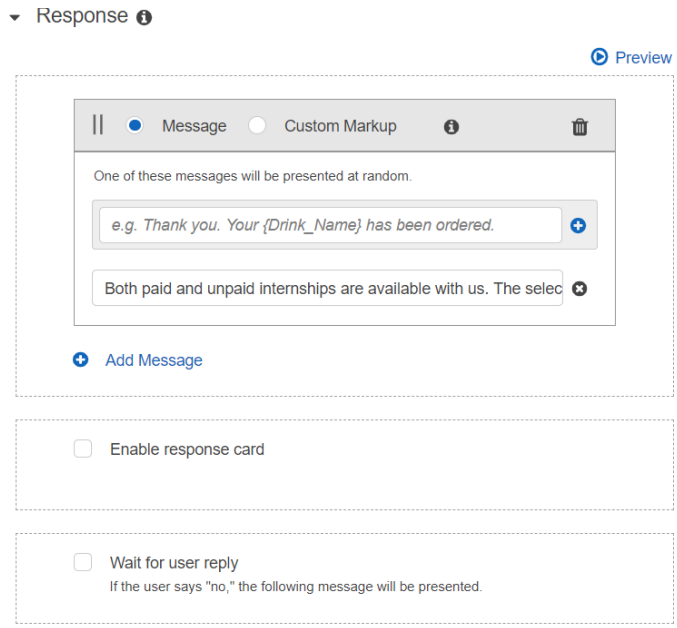




We are not filling the slots for this intent and keeping the “Fullfillment” as return parameters to client. We will see the use of slots and use of lambda function for the next intent.
In the response section, we are giving the response that the bot is going to give after hearing all the utterances we have provided.

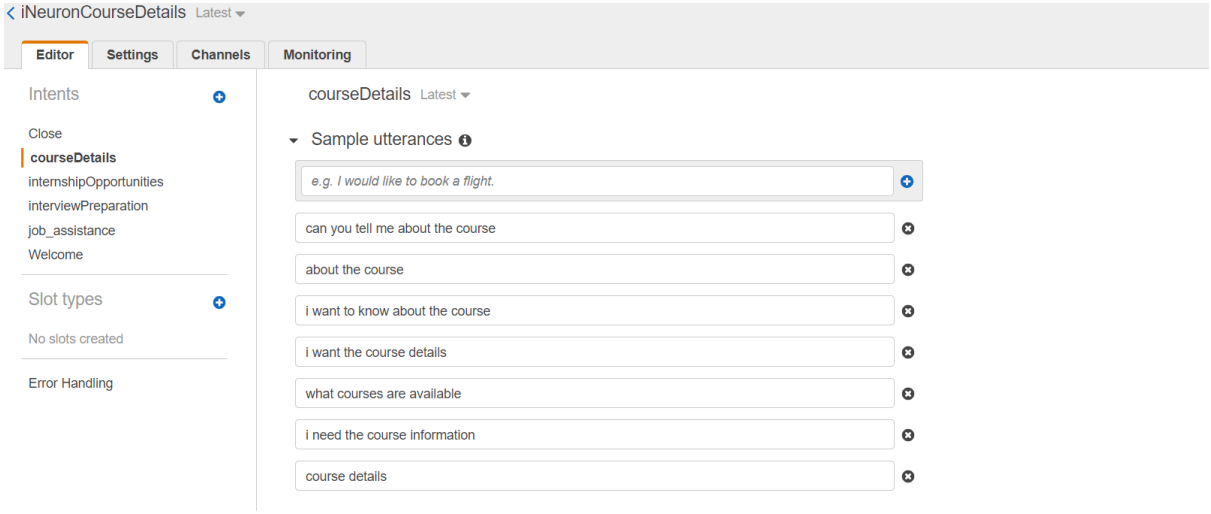
9) Similarly we are creating new intents as per our requirements that we need the bot to respond to.



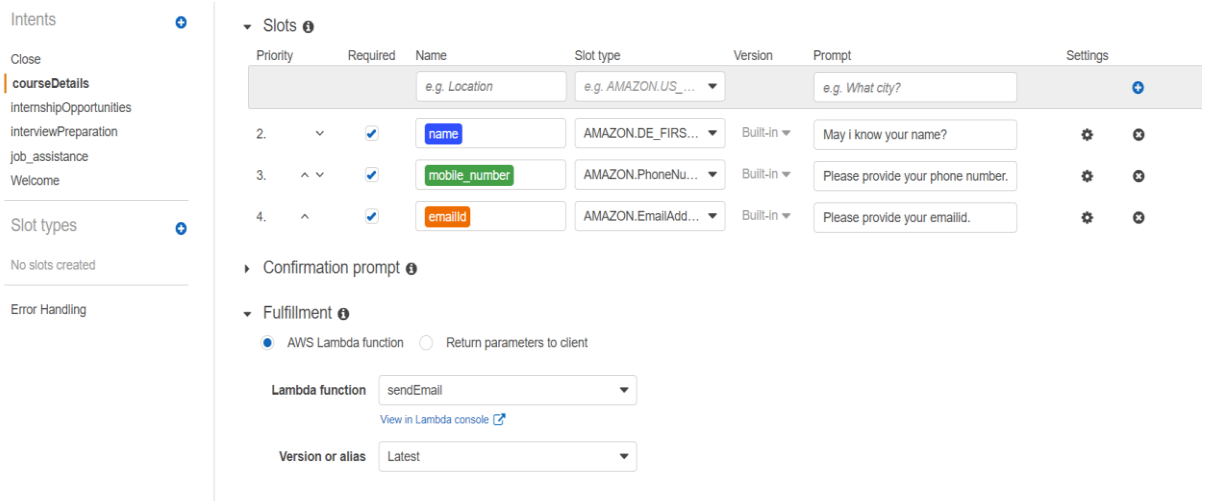


10) let's see the Fulfillment using lambda function. We are creating one intent with sample utterances and once our bot receives that intent, it will ask the user for its details like name, mobile number and email. Then it will send a mail to the user with the course details attached and it will also send an email to the support team with all the details of the user requesting a callback.

Let's see how we are going to do it.



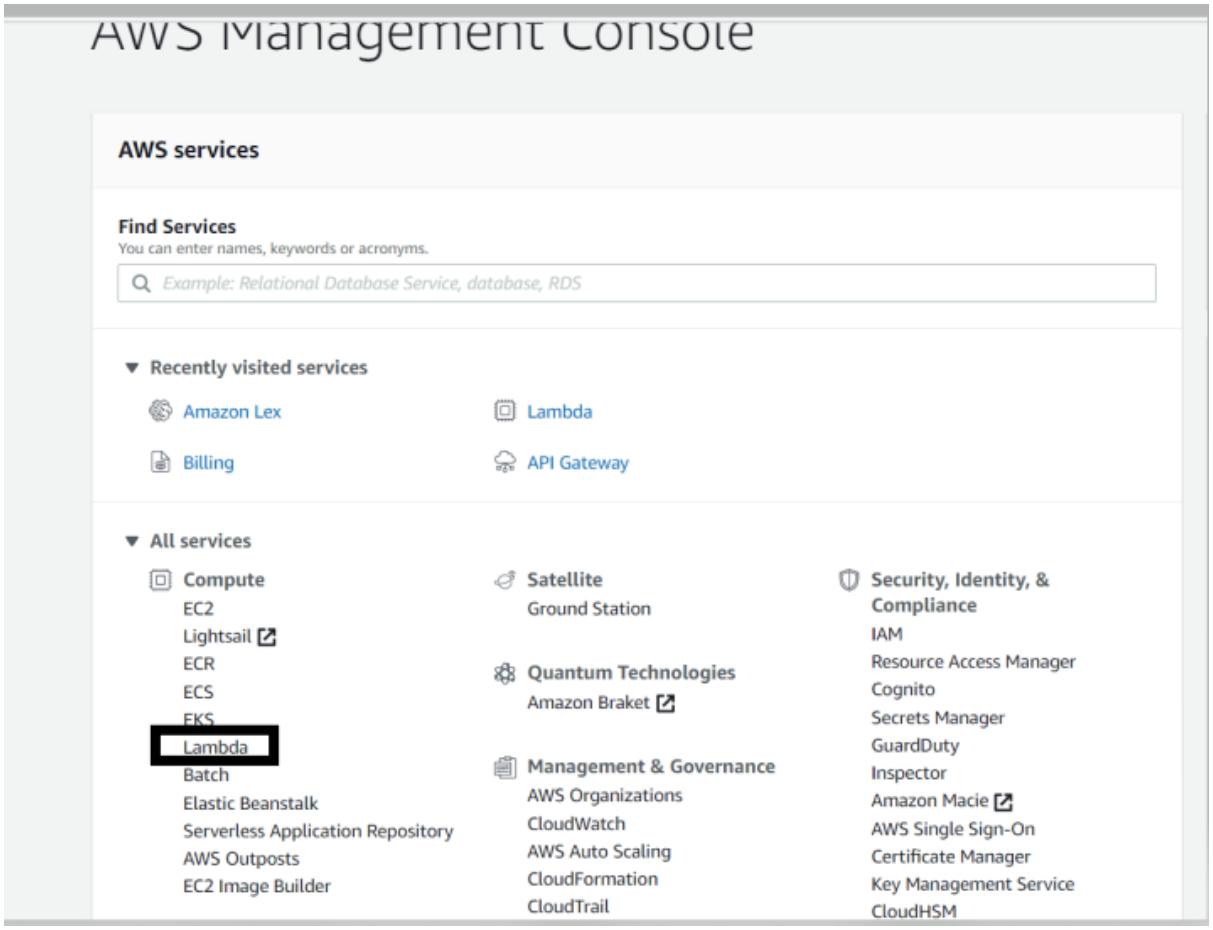
Once we have filled up the sample utterances. Let's fill the slots that we will be prompting the user for getting the user details.



In the slot name we give the values that we want from the user. The slot_type is the default validation parameter in Lex and we can use that to validate the values user gives against that slot. In the prompt we are going to prompt the user with the specific questions to get the slot values.
In the Fulfillment section, we are using Lambda function to send an email to the user on the given email id and to the support team with all the user details.

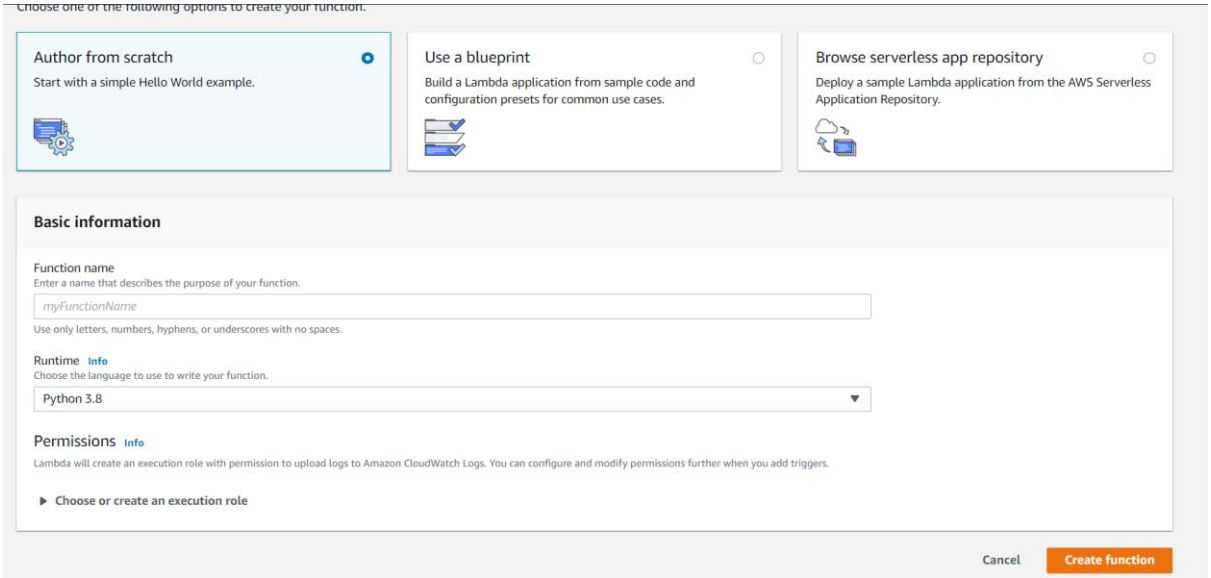
Let’s see how to create the lamda function.

Go to the AWS console and click on “Lambda” inside the ”compute” section.

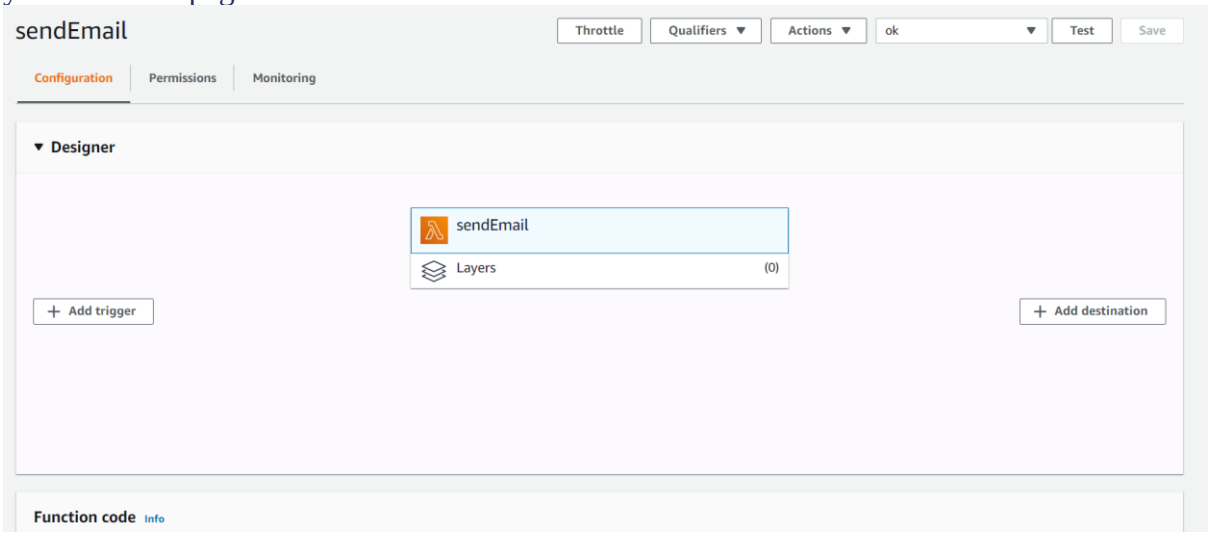


Click on the “create function” tab.

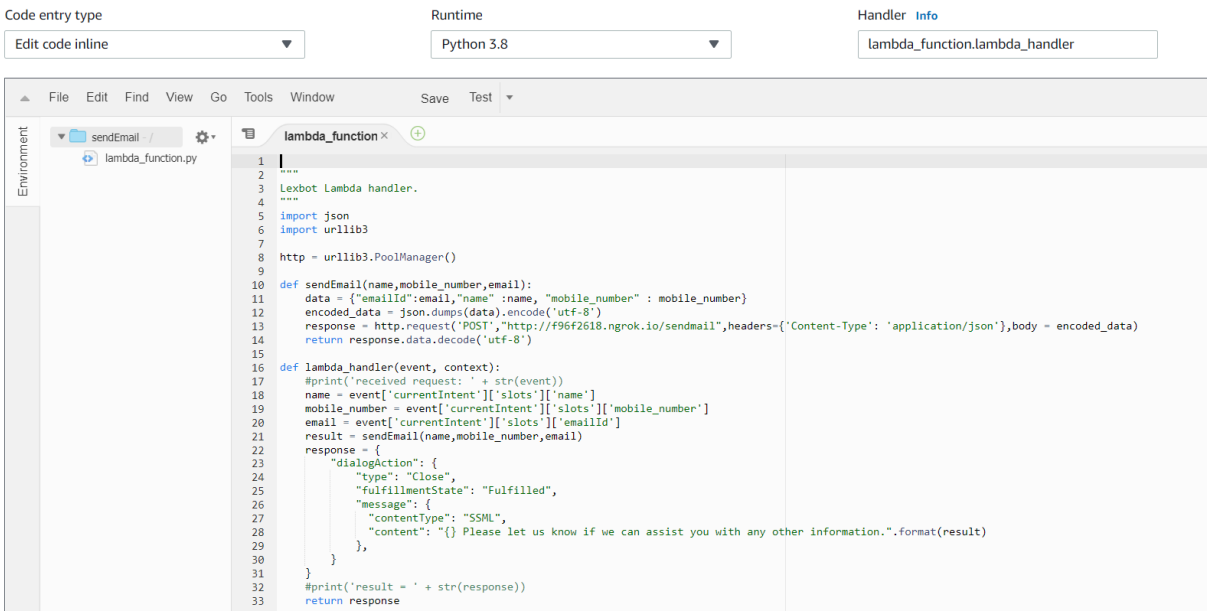
Given the name of the function and select the runtime as “python”, you can select other runtimes as well. Click on the create function.



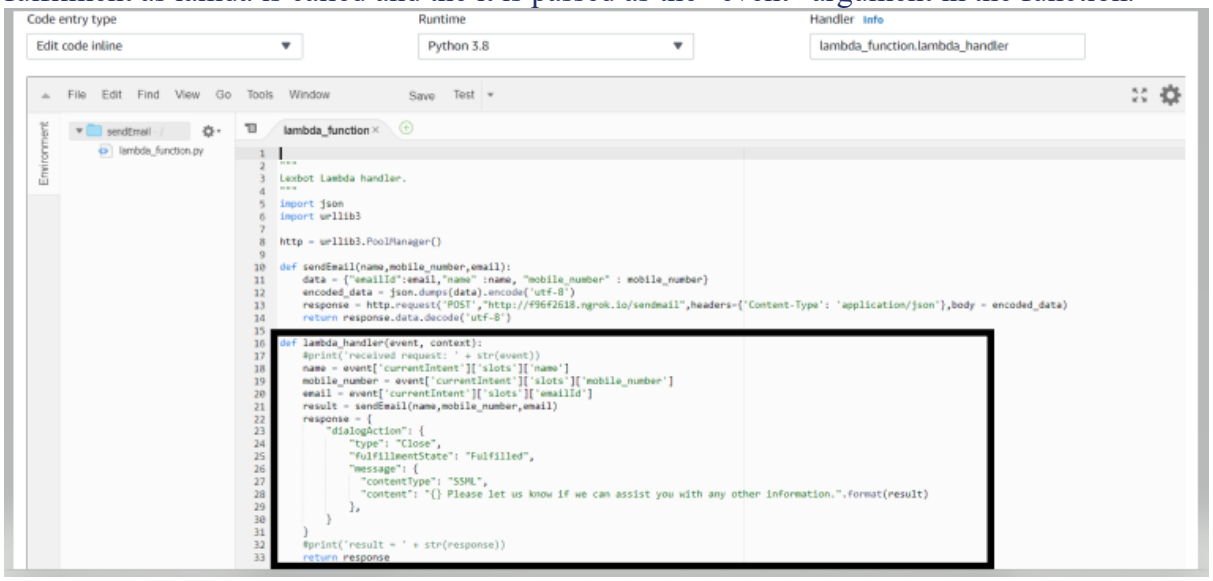
We have created a function named “sendEmail”. After clicking the “create function” button, you will see a page like this :



Go to the “Function code” section:



The function with name “lambda_handler” is the function first called once an intent with fulfilment as lamda is called and the it is passed as the “event” argument in the function.



From the “event” argument we can derive our values in the slots like below :

```
def lambda_handler(event, context):
    #print('received request: ' + str(event))
    name = event['currentIntent']['slots']['name']
    mobile_number = event['currentIntent']['slots']['mobile_number']
    email = event['currentIntent']['slots']['emailId']
```

Once we have got the values in our slots for the intent, we are going to pass these values as an argument in an another function we have written to send the mails.

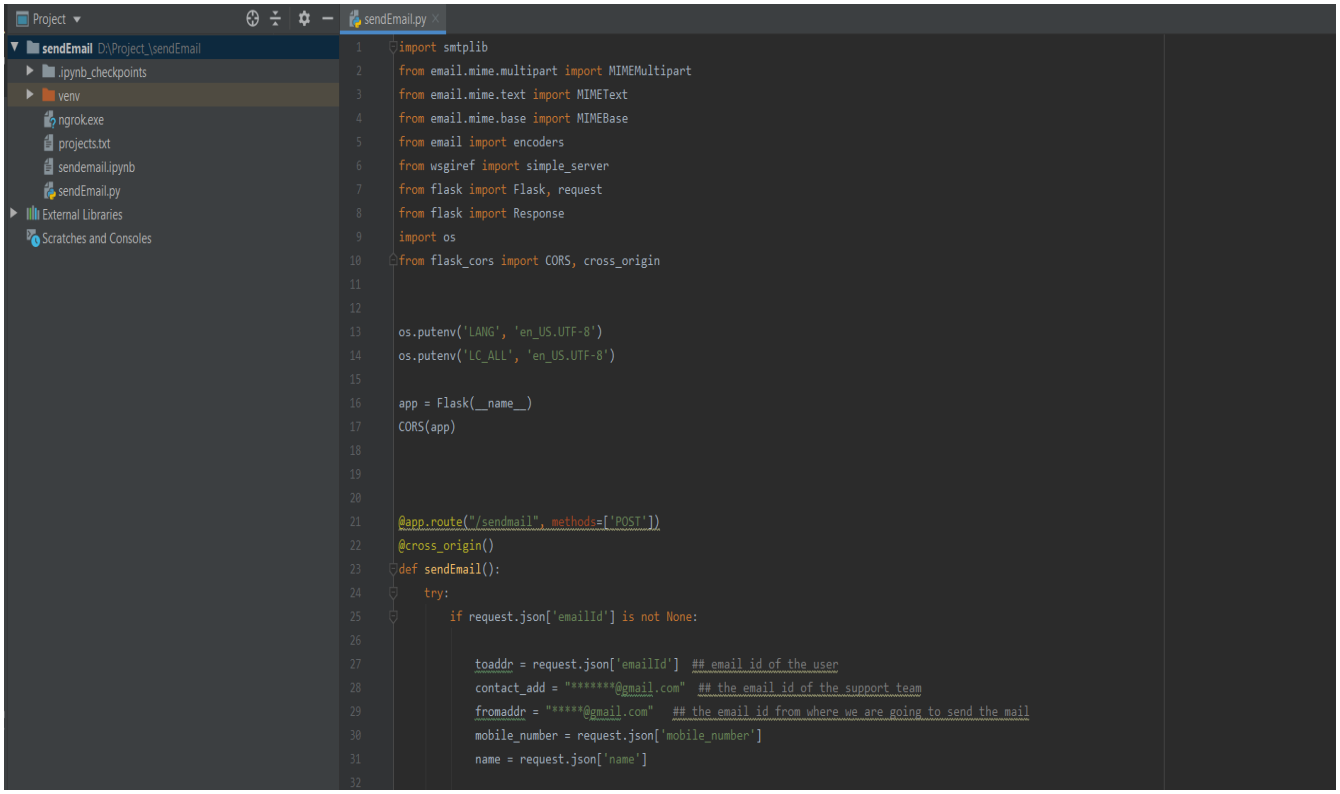
```
def lambda_handler(event, context):
    #print('received request: ' + str(event))
    name = event['currentIntent']['slots']['name']
    mobile_number = event['currentIntent']['slots']['mobile_number']
    email = event['currentIntent']['slots']['emailId']
    result = sendEmail(name,mobile_number,email)
```

Let's see what does this function "sendEmail" does:

```
import urllib3

http = urllib3.PoolManager()

def sendEmail(name,mobile_number,email):
    data = {"emailId":email,"name" :name, "mobile_number" : mobile_number}
    encoded_data = json.dumps(data).encode('utf-8')
    response = http.request('POST',"http://f96f2618.ngrok.io/sendmail",headers={'Content-Type': 'application/json'},body = encoded_data)
    return response.data.decode('utf-8')
```

This function hits an "Api" using the post method. We pass all the user details in the api in a json format. The Api returns a message on successfully sending the mails.

We have written a python code and deployed it as an "Api" which we are directly calling throught this "sendEmail" function. Let's see the code written for creating the "Api".



sendEmail.py

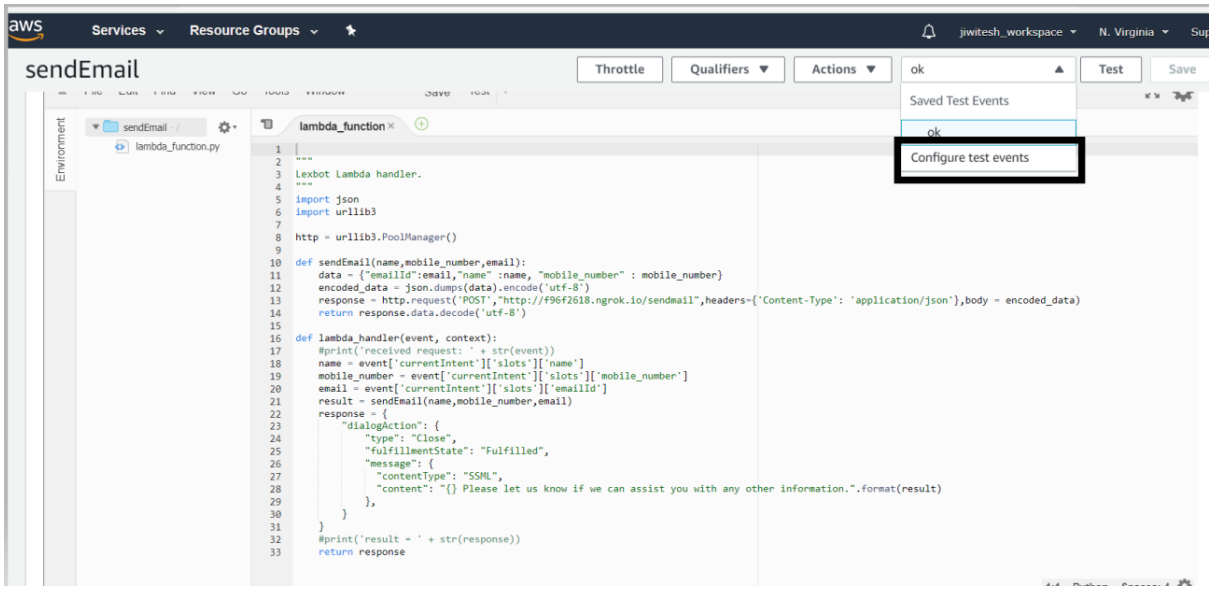
```
        # instance of MIMEMultipart
        msg = MIMEMultipart()

        # storing the senders email address
        msg['From'] = fromaddr

        # storing the receivers email address
        msg['To'] = ",".join(toaddr)
        # msg['To'] = toaddr

        # storing the subject
        msg['Subject'] = "Test bot email"

        # string to store the body of the mail
        body = "This will contain attachment"

        # attach the body with the msg instance
        msg.attach(MIMEText(body, 'plain'))

        # open the file to be sent
        filename = "projects.txt"
        attachment = open(filename, "rb")

        # instance of MIMEBase and named as p
        p = MIMEBase('application', 'octet-stream')

        # To change the payload into encoded form
        p.set_payload((attachment).read())

        # encode into base64
        encoders.encode_base64(p)

        p.add_header('Content-Disposition', "attachment; filename= %s" % filename)
```

```
# attach the instance 'p' to instance 'msg'
msg.attach(p)

# creates SMTP session
s = smtplib.SMTP('smtp.gmail.com', 587)

# start TLS for security
s.starttls()

# Authentication
s.login(fromaddr, "****password*****")# give your password here

# Converts the Multipart msg into a string
text = msg.as_string()
# sending the mail
s.sendmail(fromaddr, toaddr, text)

# send mail with username , mobile number and email id to concerned team

# instance of MIMEMultipart
msg1 = MIMEMultipart()
# storing the subject
msg1['Subject'] = "Query For Course Details"

# string to store the body of the mail
body1 = "person with name {0} have some queries regarding our course details." \
        "Please reach to {0} at his mobile number {1} and email id {2}".format(name,mobile_number,toaddr)

# attach the body with the msg instance
msg1.attach(MIMEText(body1, 'plain'))

text = msg1.as_string()
```

```
            text = msg1.as_string()
            # sending the mail
            s.sendmail(fromaddr, contact_add, text)

            # terminating the session
            s.quit()

            return Response("Course details is sent to the mail id %s" % toaddr +"\t " +
                            "and your details have been forwarded to the concerned team.")

    except ValueError:
        return Response("Error Occurred! %s" %ValueError)
    except KeyError:
        return Response("Error Occurred! %s" %KeyError)
    except Exception as e:
        return Response("Error Occurred! %s" %e)

#port = int(os.getenv("PORT"))
if __name__ == "__main__":
    host = '0.0.0.0'
    port = 5000
    httpd = simple_server.make_server(host, port, app)
    print("Serving on %s %d" % (host, port))
    httpd.serve_forever()
```

We are keeping the Lambda function as simple as possible and we will not add any more functionality. Let's test our function written in the lambda.

For running the test, firs save our lambda function, then click on the "configure test events"



You will see fields like this, create a new test event and set default values for the slots for testing the lamda function:

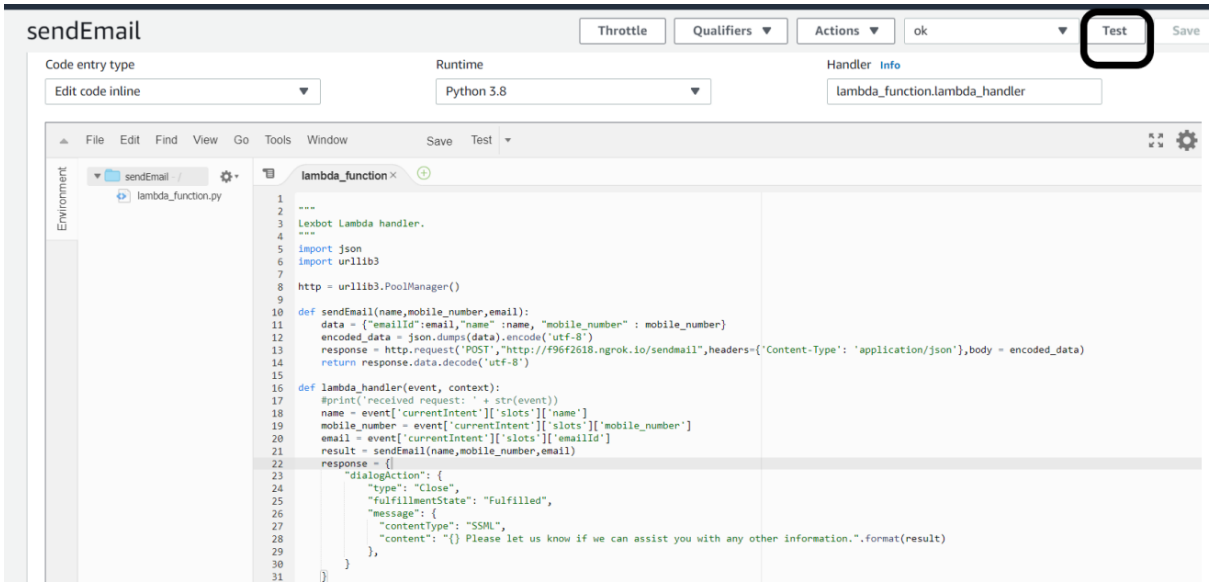
Saved test event

ok

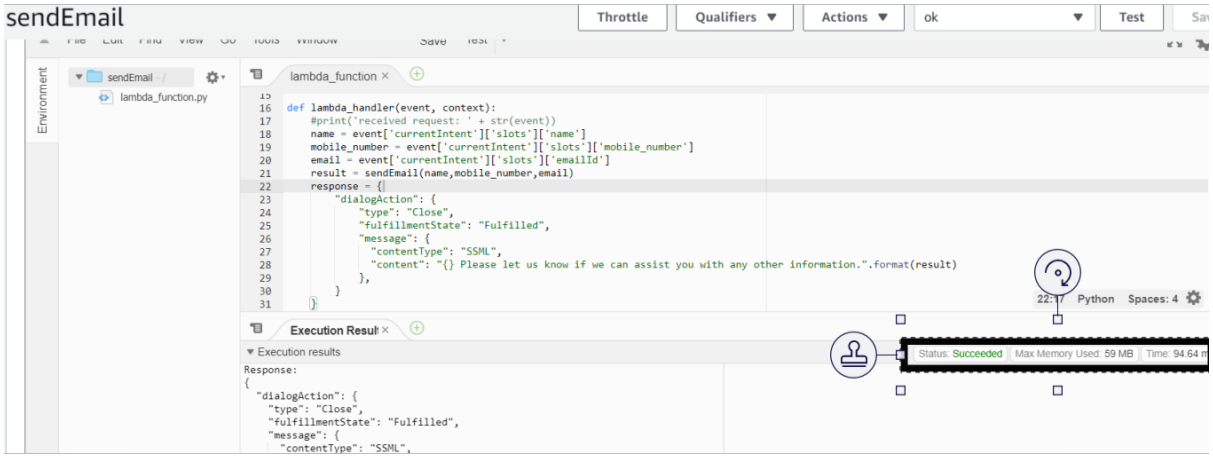
```
{
  "messageVersion": "1.0",
  "invocationSource": "DialogCodeHook",
  "userId": "John",
  "sessionAttributes": {},
  "bot": {
    "name": "sendEmail",
    "alias": "$LATEST",
    "version": "$LATEST"
  },
  "outputDialogMode": "Text",
  "currentIntent": {
    "name": "courseDetails",
    "slots": {
      "name": "Raj",
      "mobile_number": "8956785432",
      "emailId": [redacted]@gmail.com"
    },
    "confirmationStatus": "None"
  }
}
```

Delete Cancel Save

Once the default values are set. Let's test our bot. Click on the test bot.



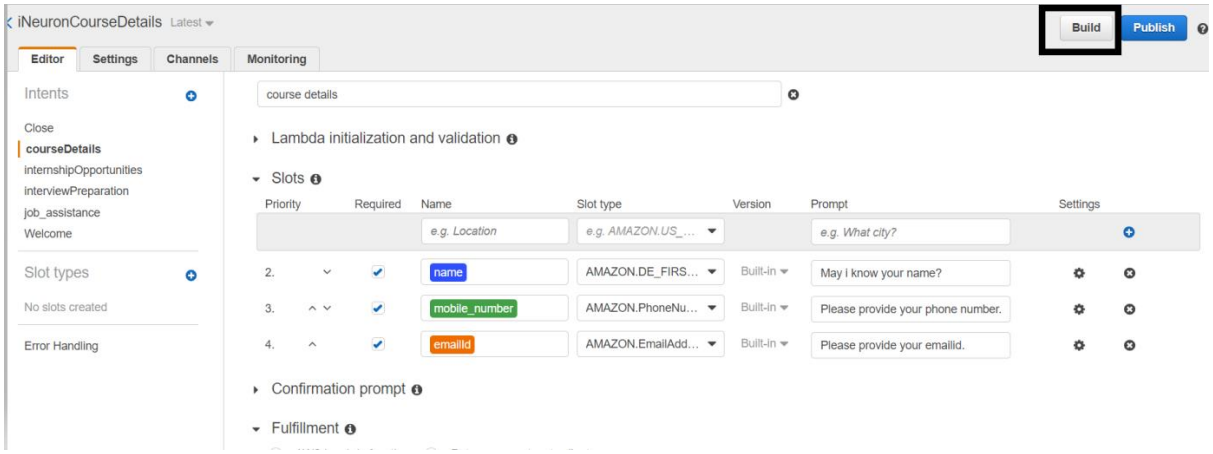
You will see a success message if your test works like this:



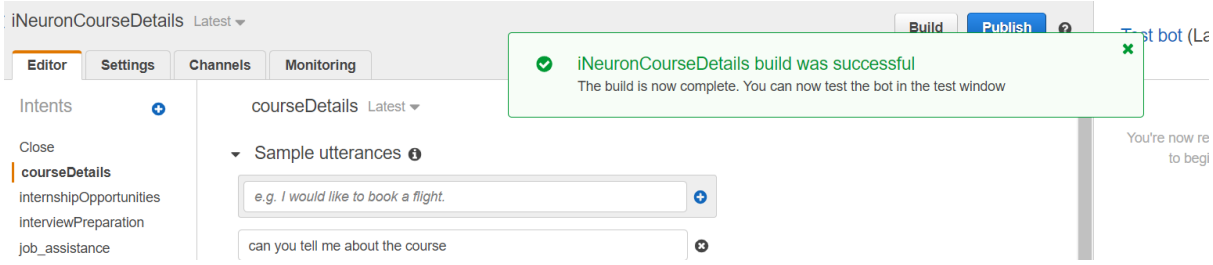
You will receive an error status if the function doesn't work properly.

Now that we have created all of our intents. You can test your bot with the intents by "building" you bot.

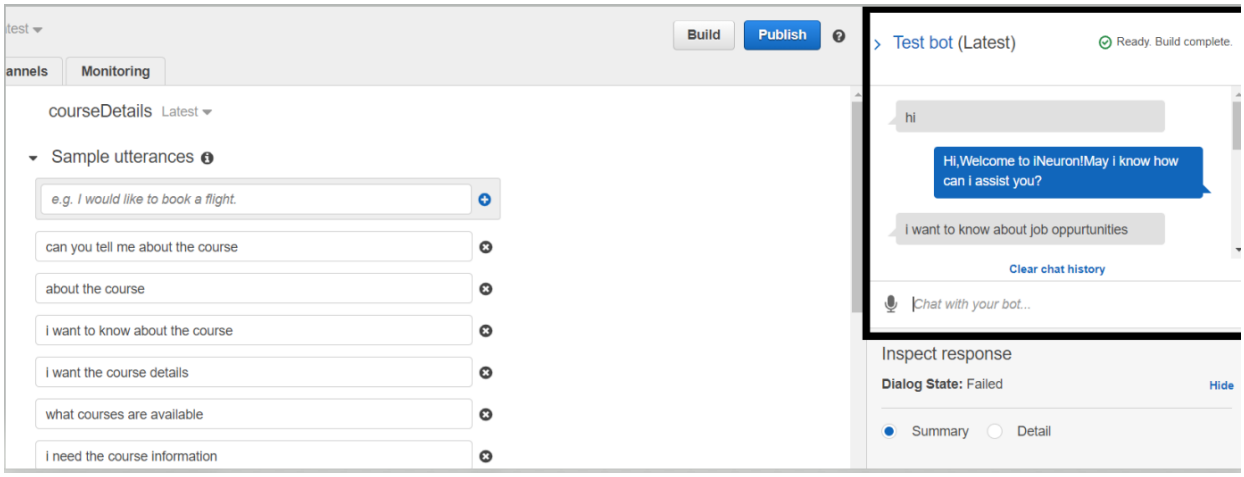
Click on the "build" button to build the bot:

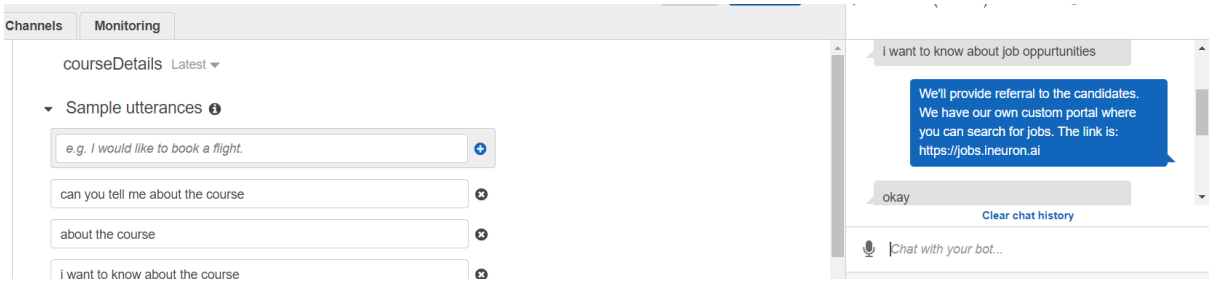


Wait till you get the message that you bot is ready for testing.



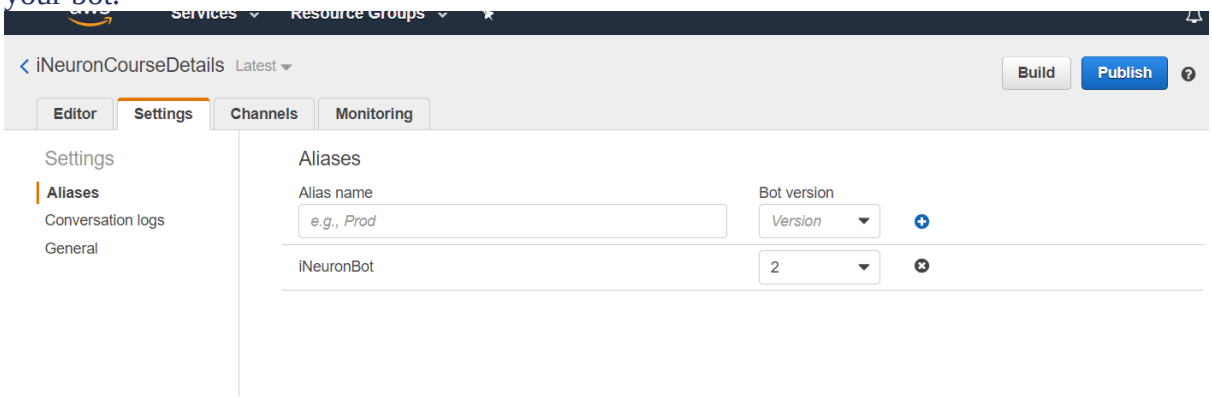
Let's test our bot in the "Test Bot" window on the right.



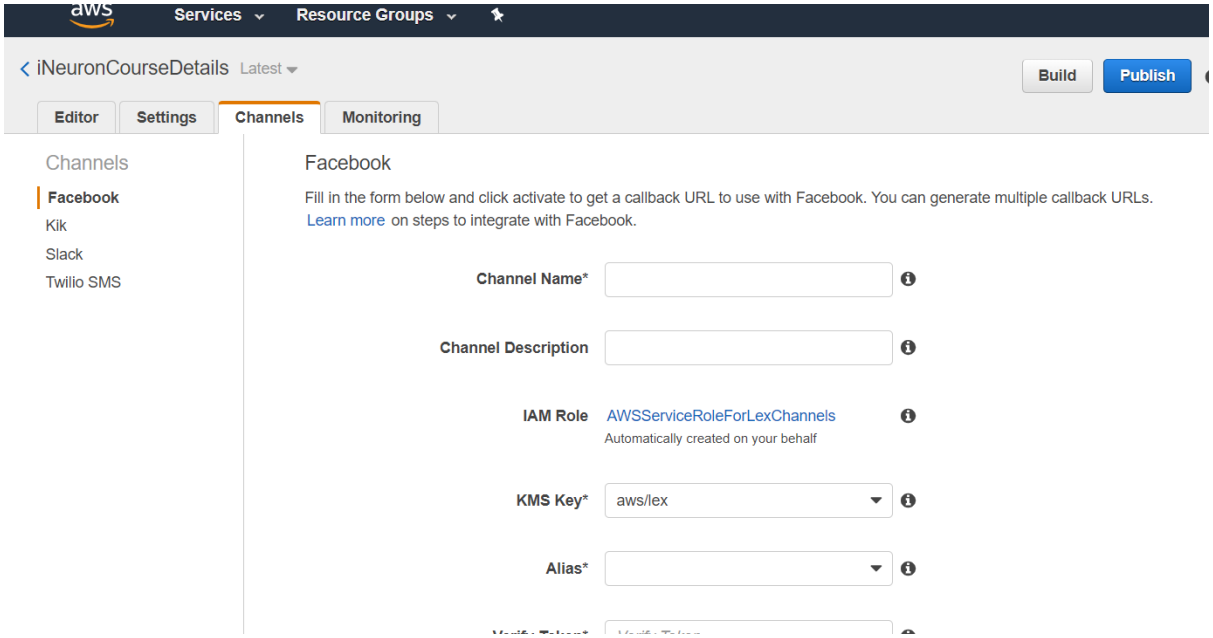


Now that we have built our bot and it's working fine. Let's go ahead with publishing the bot.

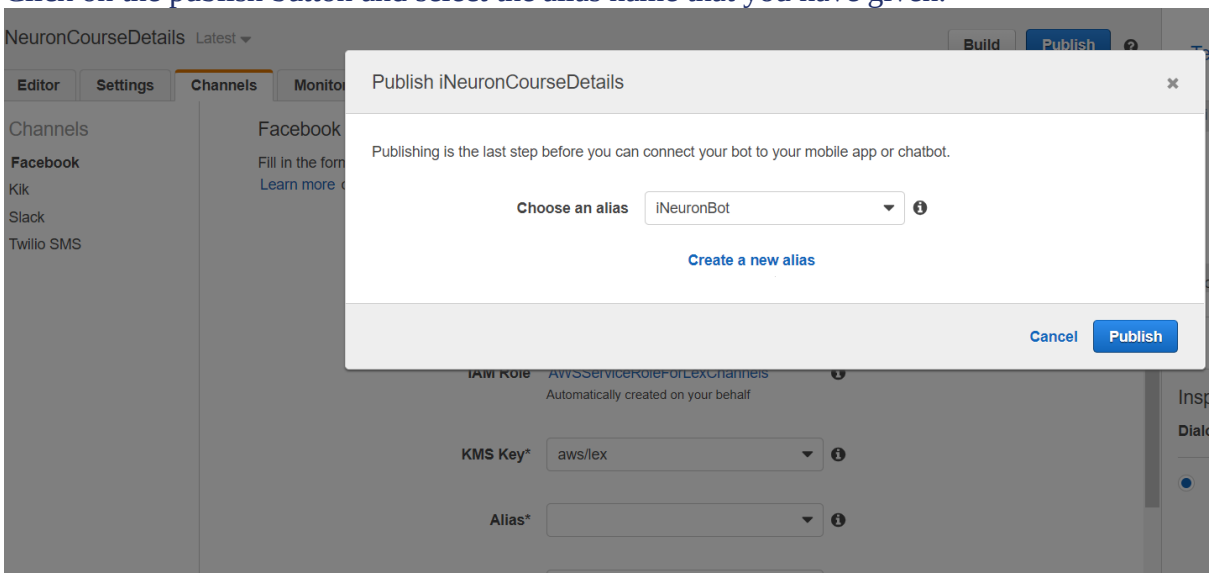
Before publishing the bot, go the settings tab and give an alias name to the bot. Again, build your bot.



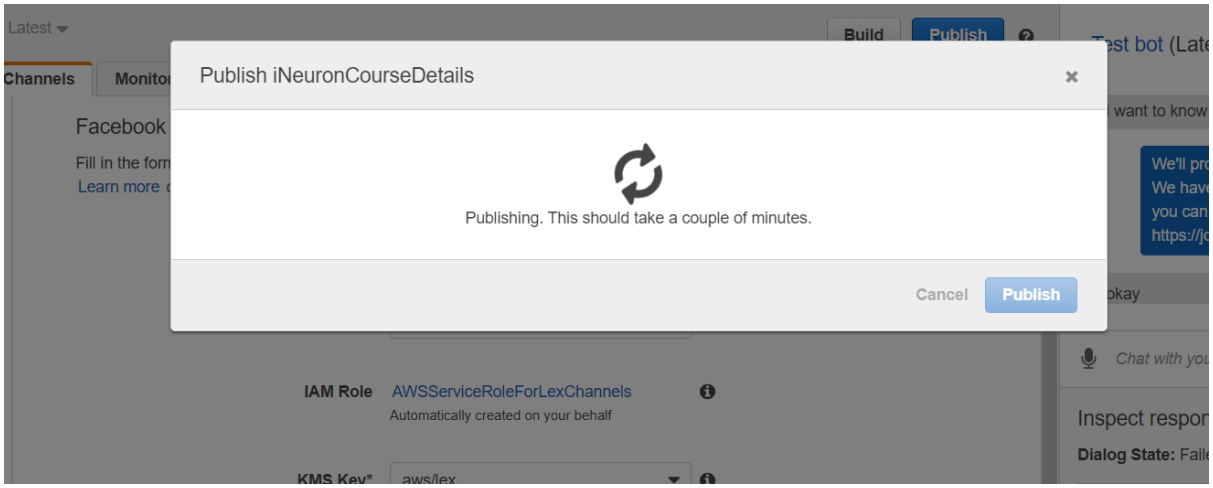
We will talk about deployment after publishing the bot, we will use this channel tab then.



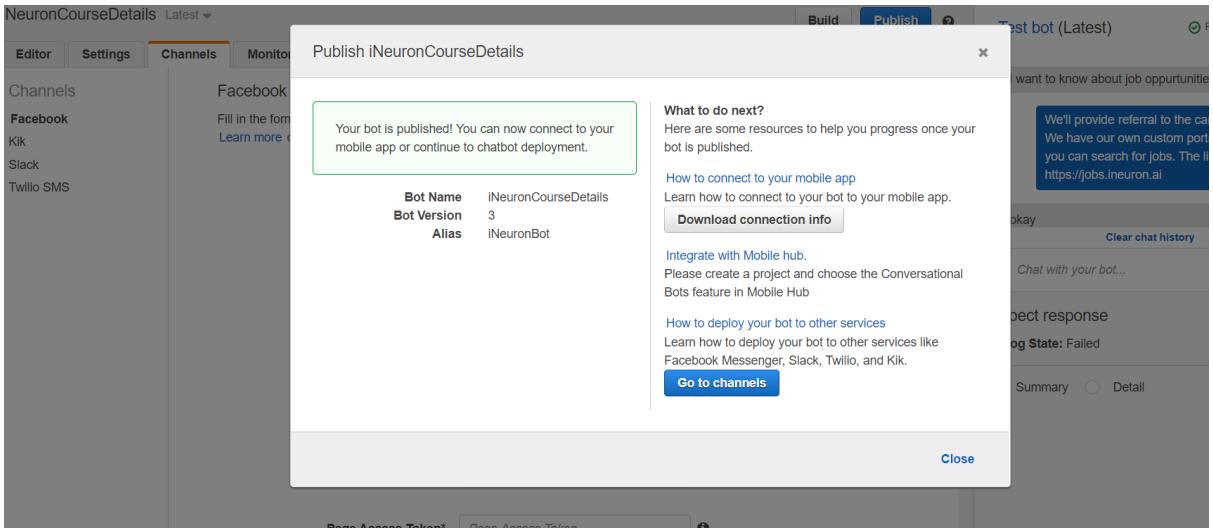
Click on the publish button and select the alias name that you have given.



Click on the publish button.

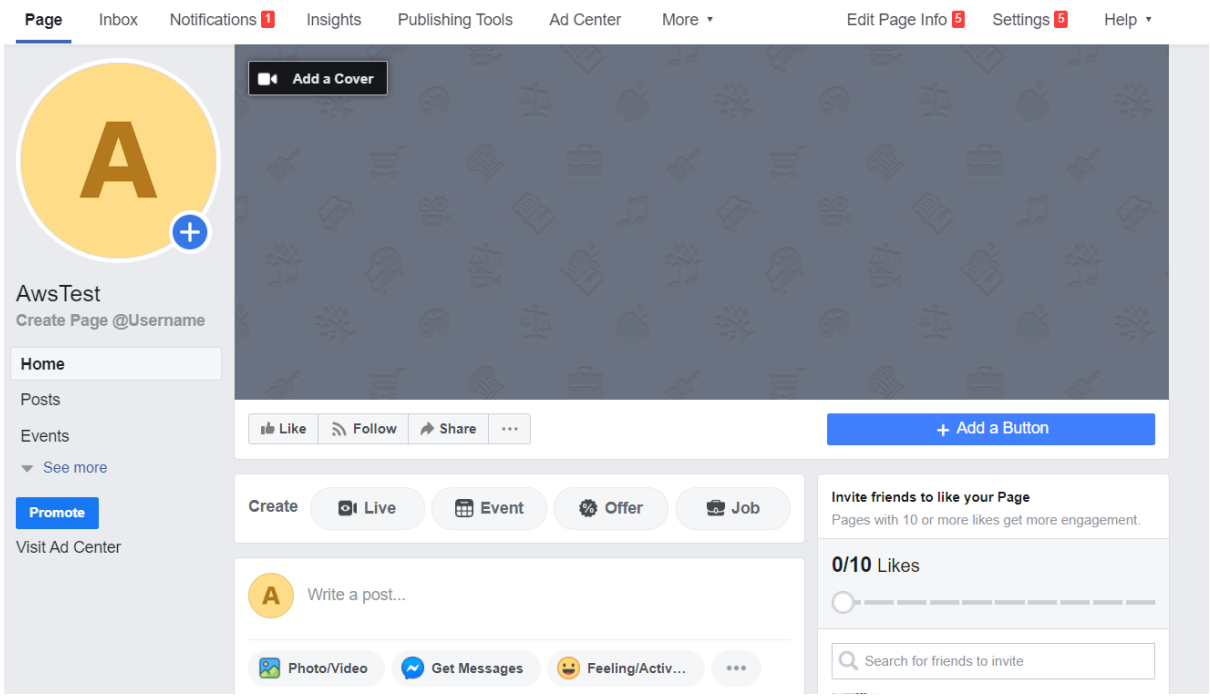


Now your bot is published. Let's deploy our bot on Facebook. Clik on the "Go to Channels".

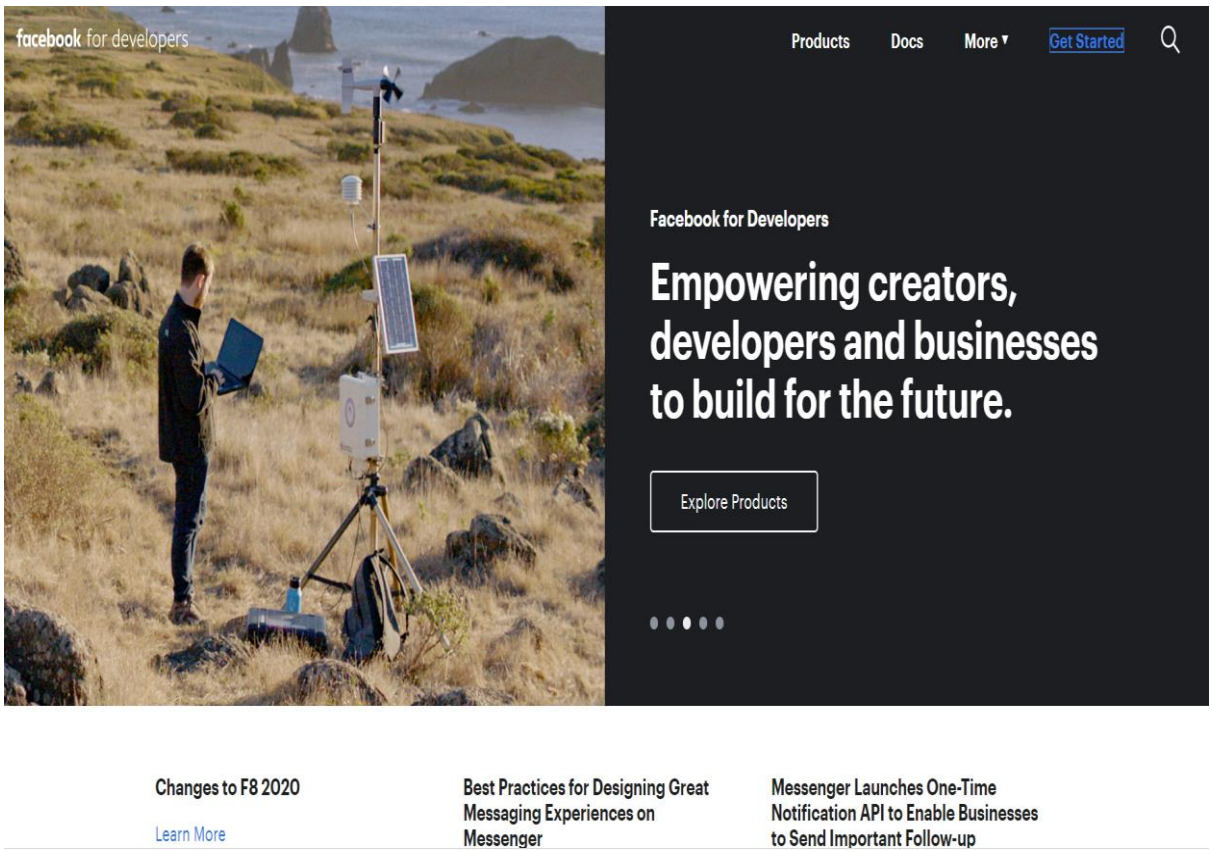


DEPLOYMENT ON FACEBOOK

Go to Facebook and create a page on which we will be integrating our chatbot. We have "AwsTest" page created already on facebook.

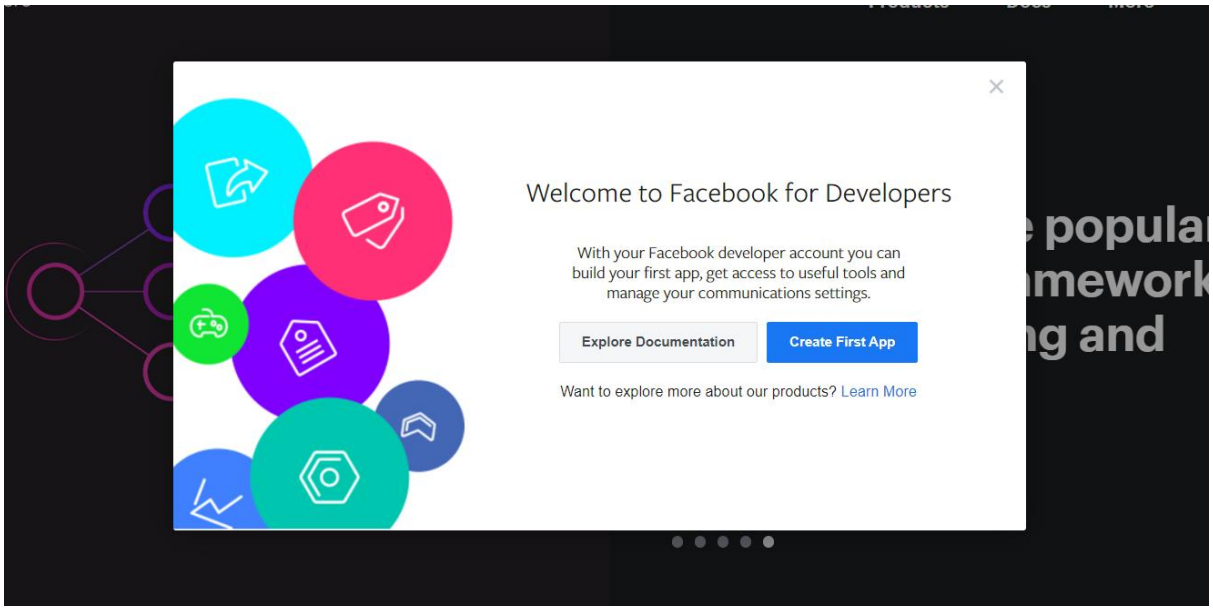


Go to Facebook for developers and click on Get started.

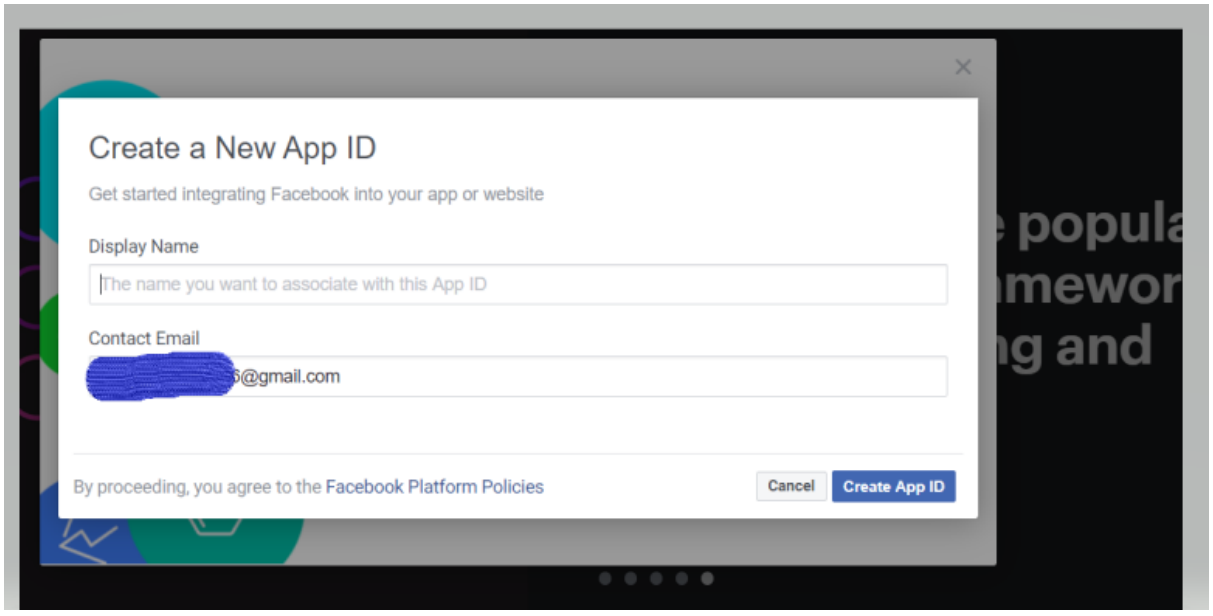


If you are a new user, create an account on facebook for developer and then proceed.

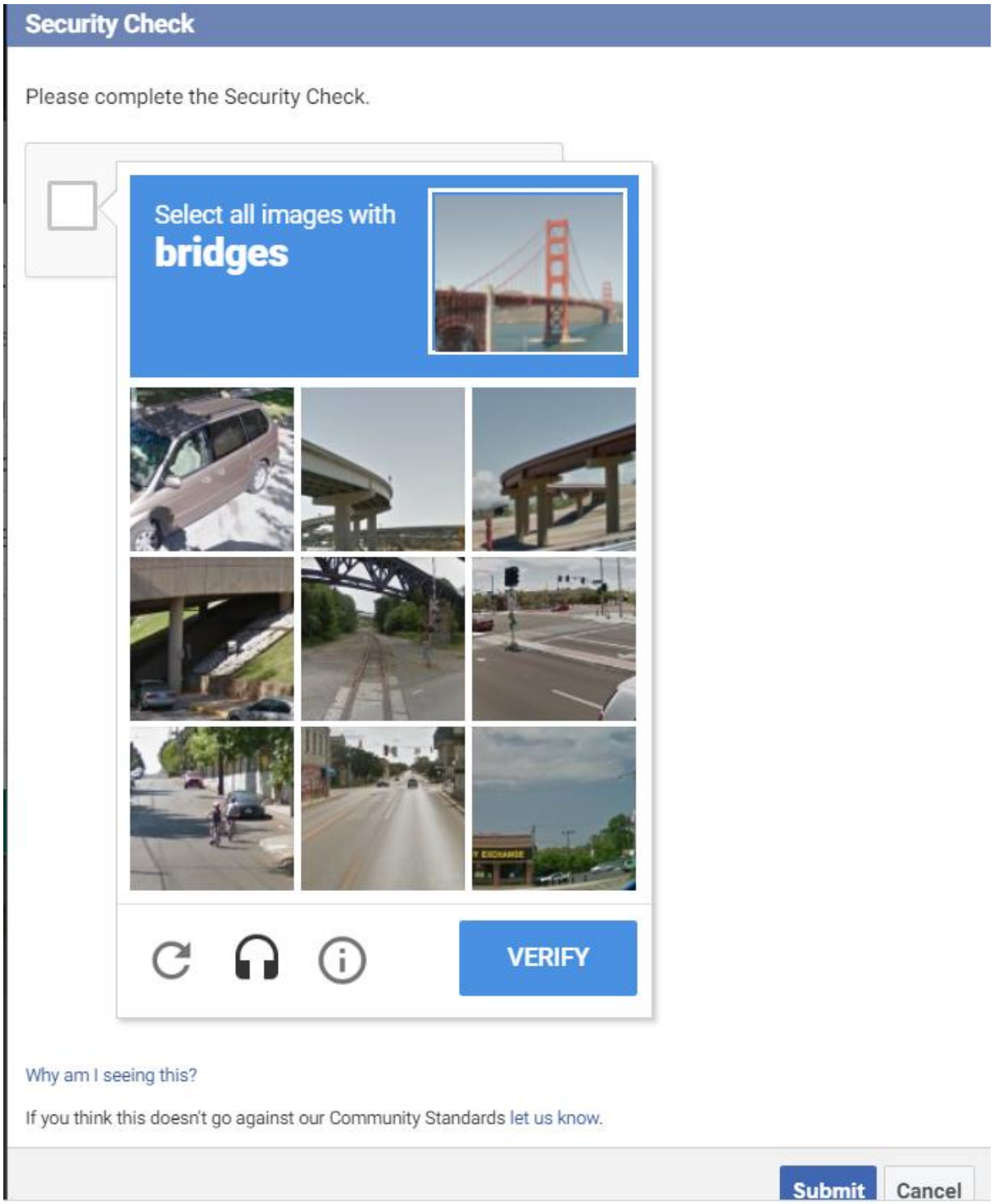
Click on “create first App”



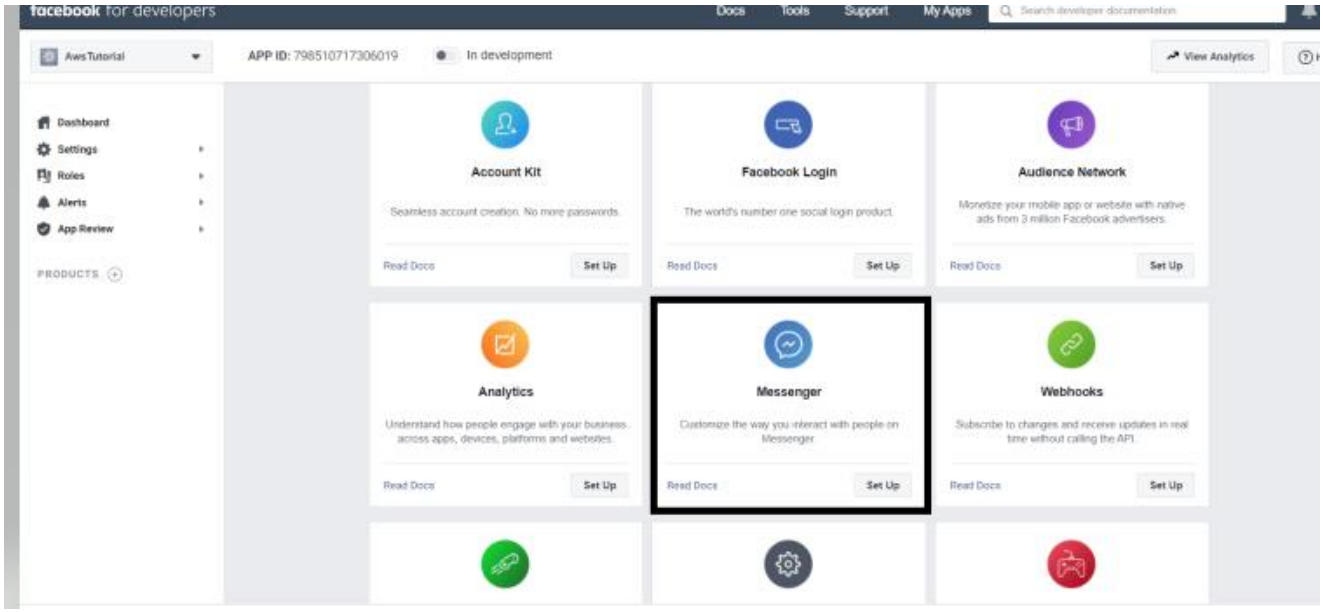
Give a display name



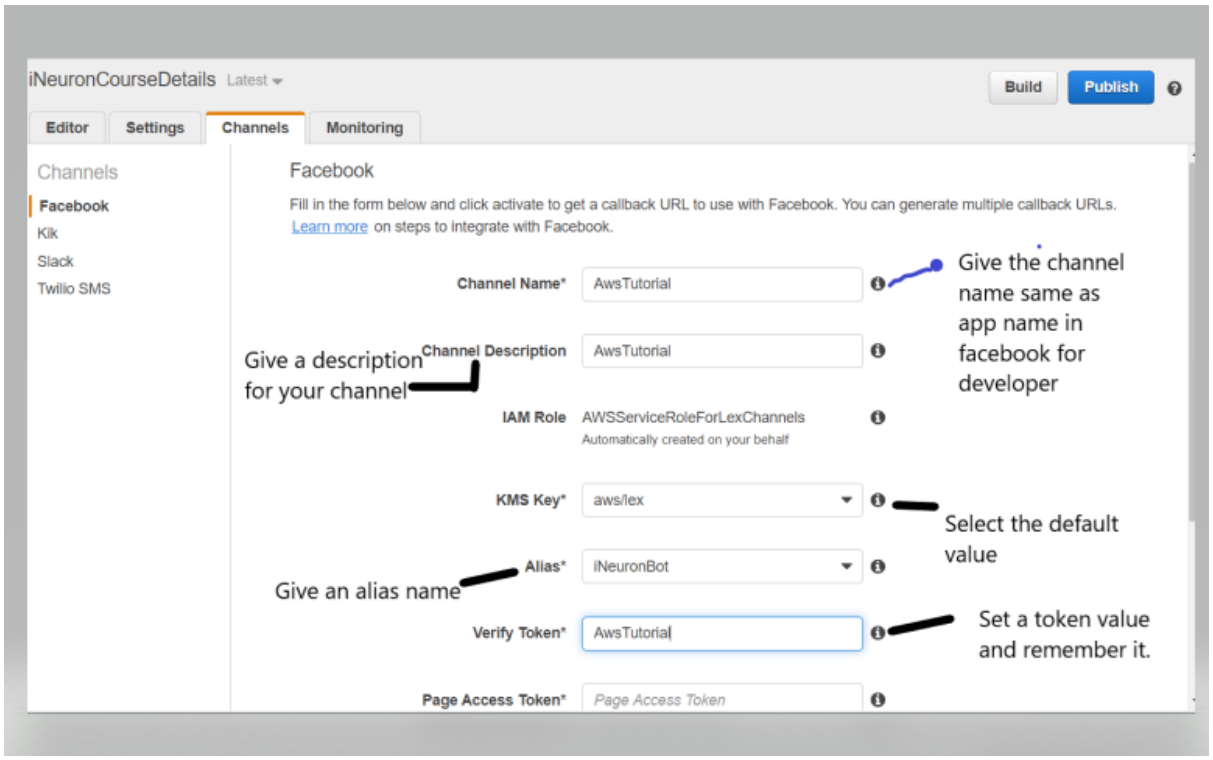
Complete the security check:



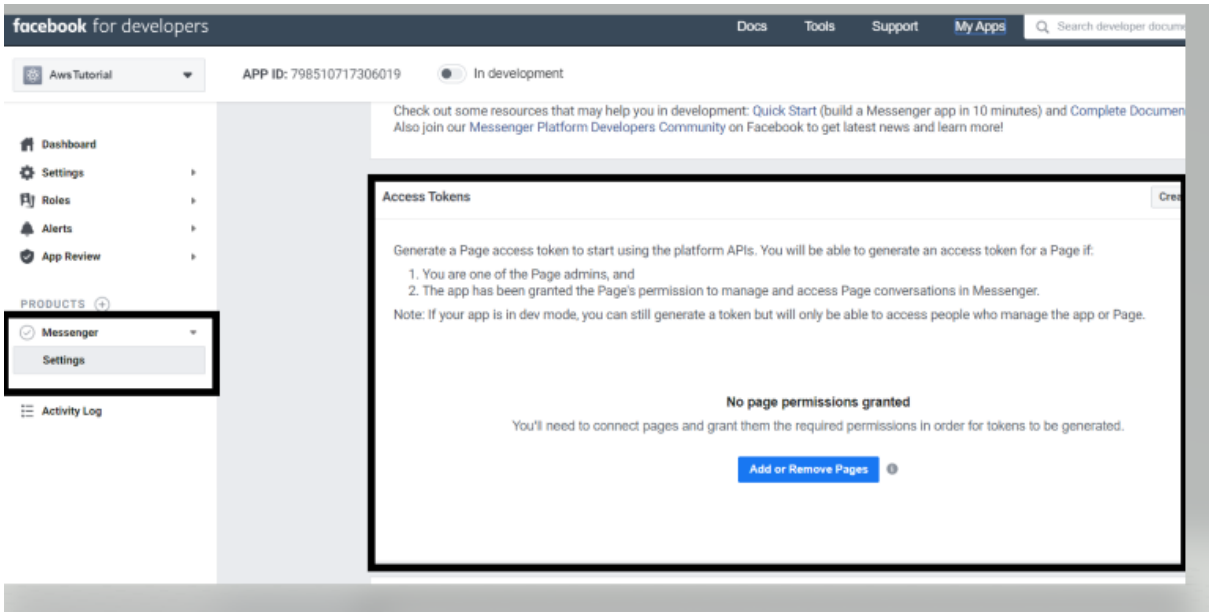
You will see the below page once the app is created. Click on the messenger tab and then on the setup button:



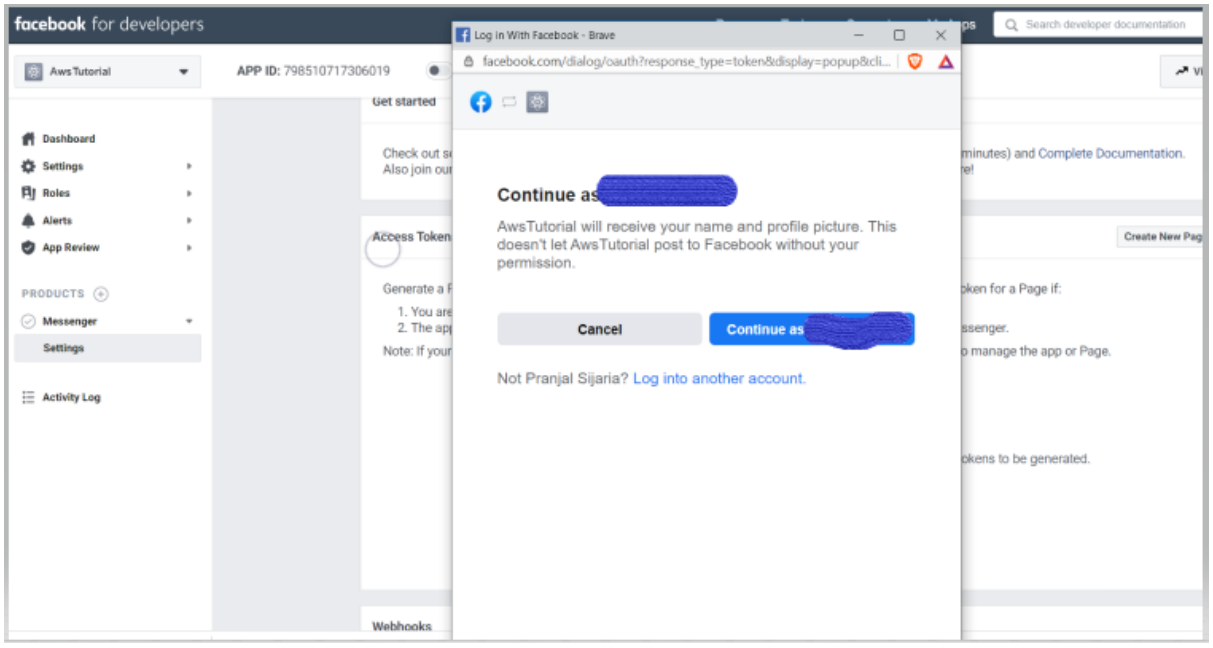
Now go back to your Lex bot page and enter the details as shown below:



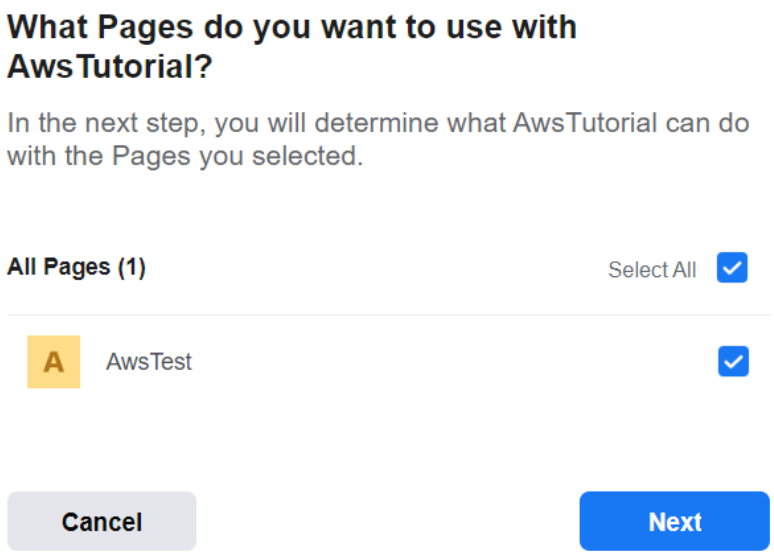
To get the page access token, go back to your app in Facebook for developers, click on settings below messenger and then scroll down to come to the access token section:



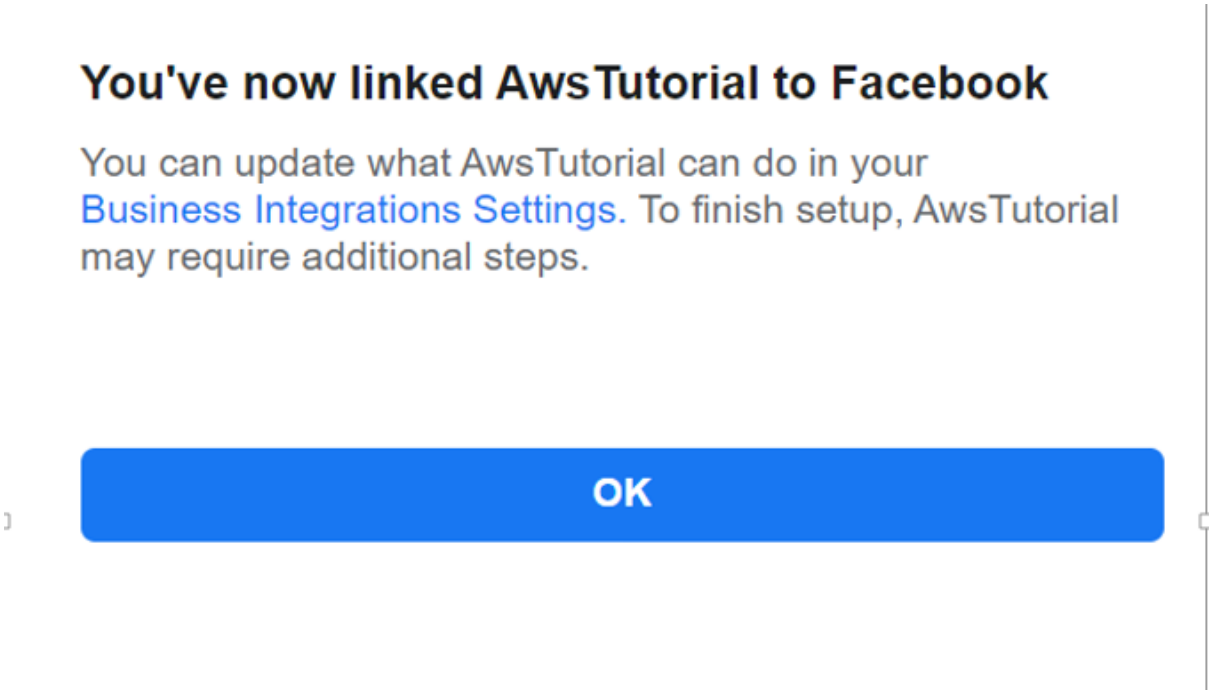
Click on the Add or remove pages button to add the page created on facebook. You will see a popup like this, press continue as:



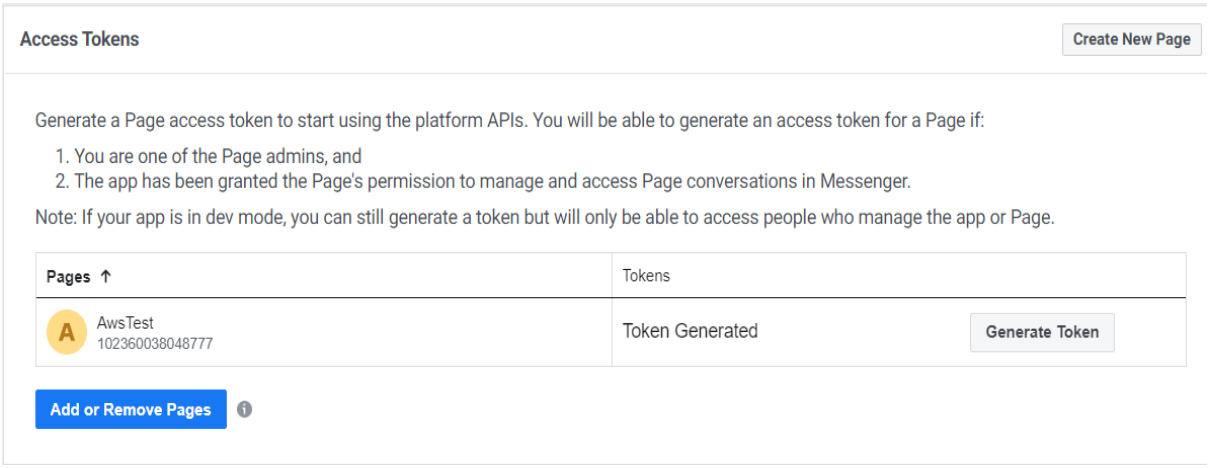
Select the app from the list :



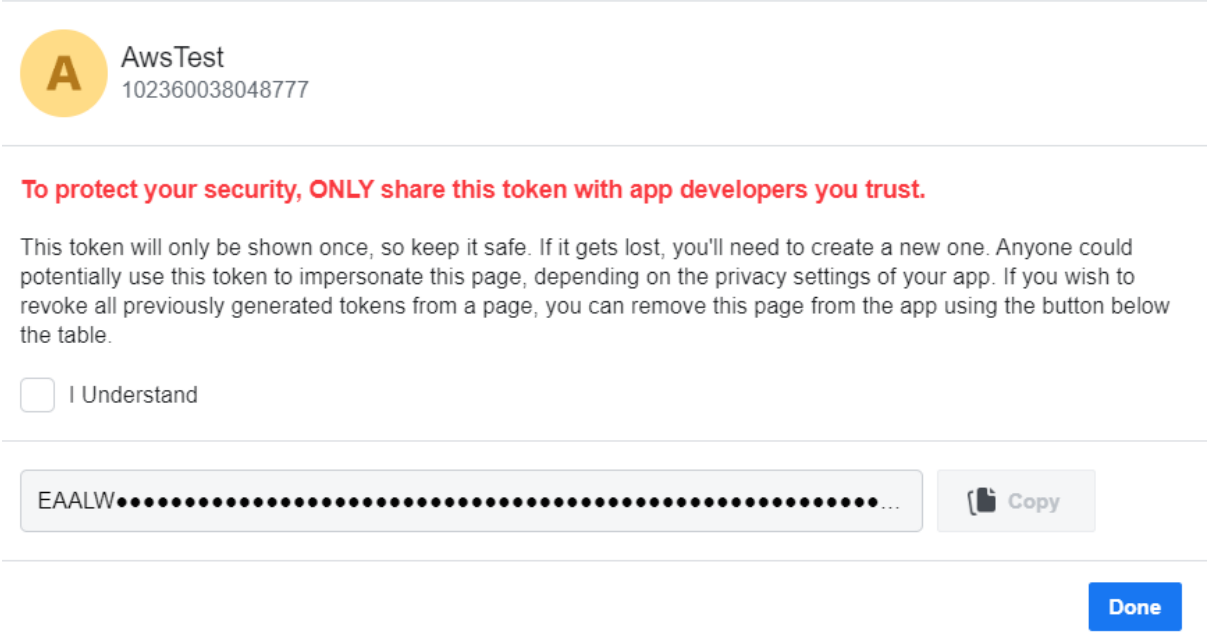
You will get a successful message:



You will see the access token section changed like this :

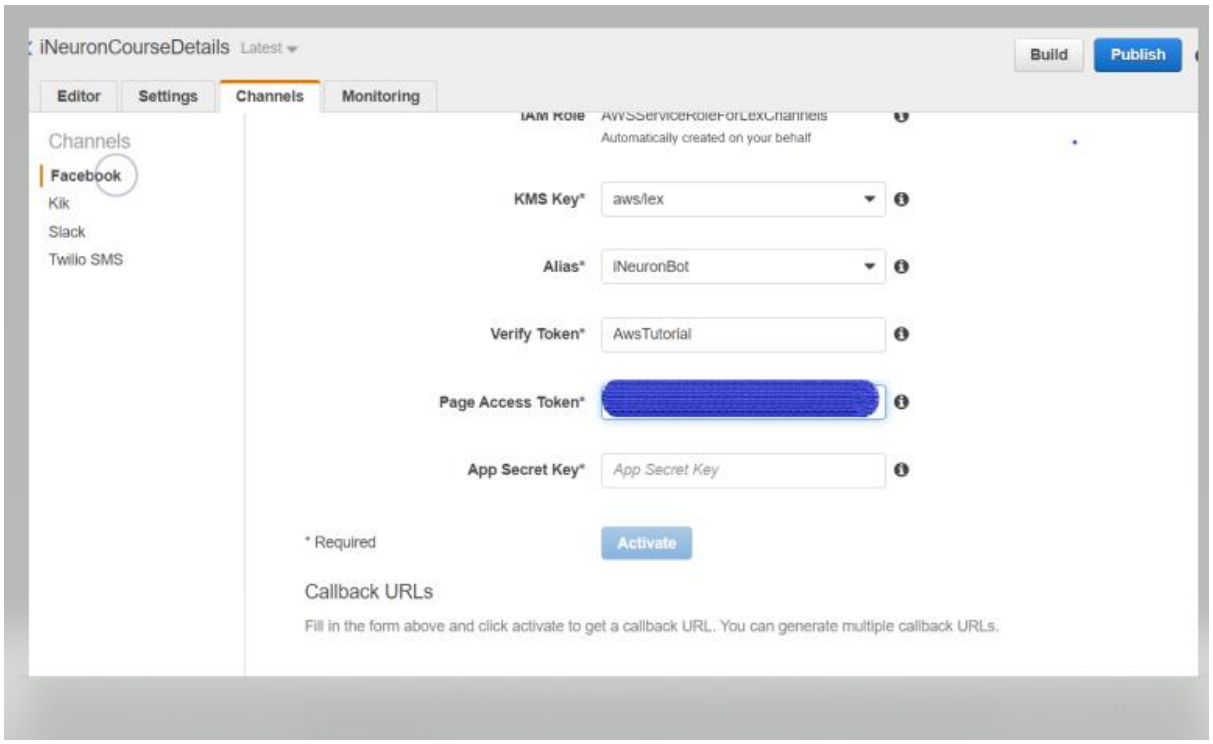


Click on the Generate Token. You will get a message like this:

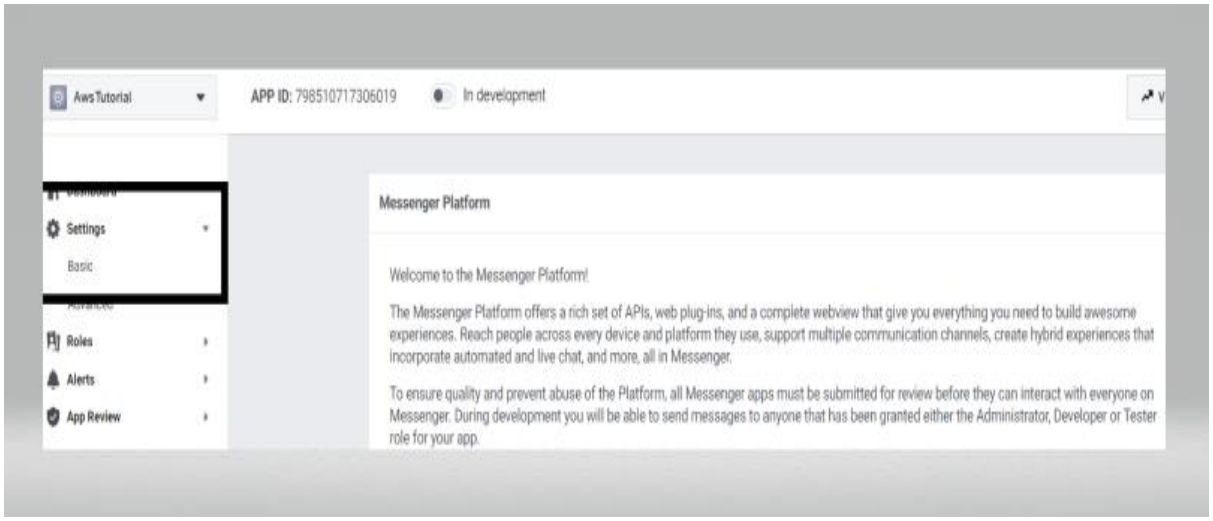


Click on “I understand” and then copy the given code.

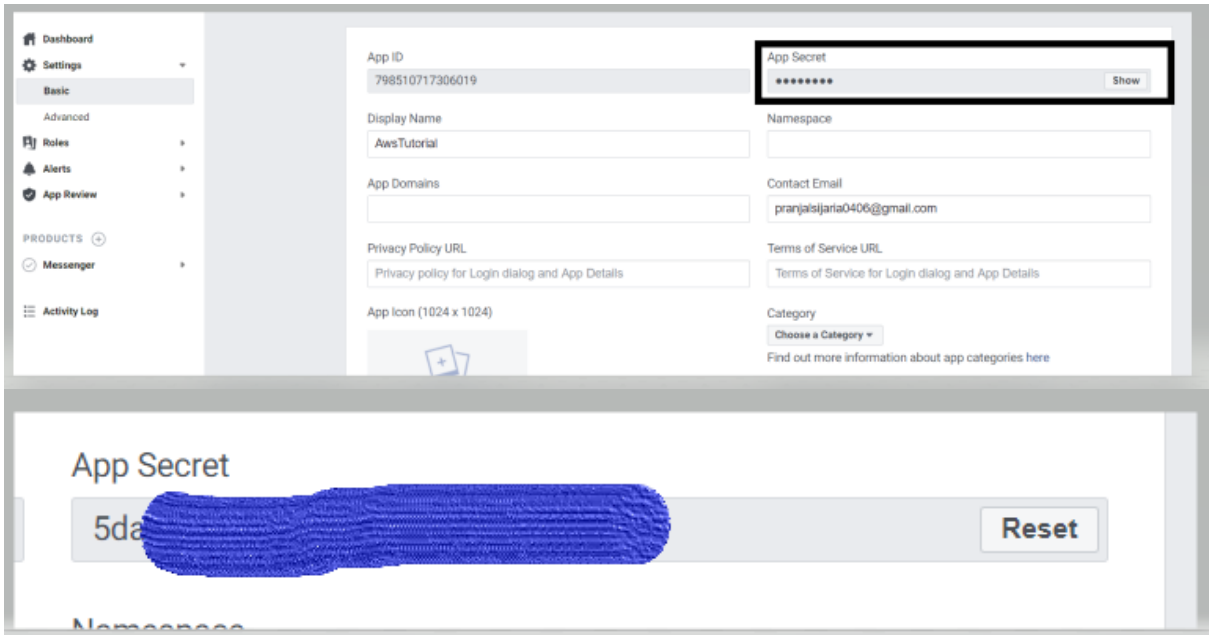
Paste the code to the page access token section of your Bot channel page:



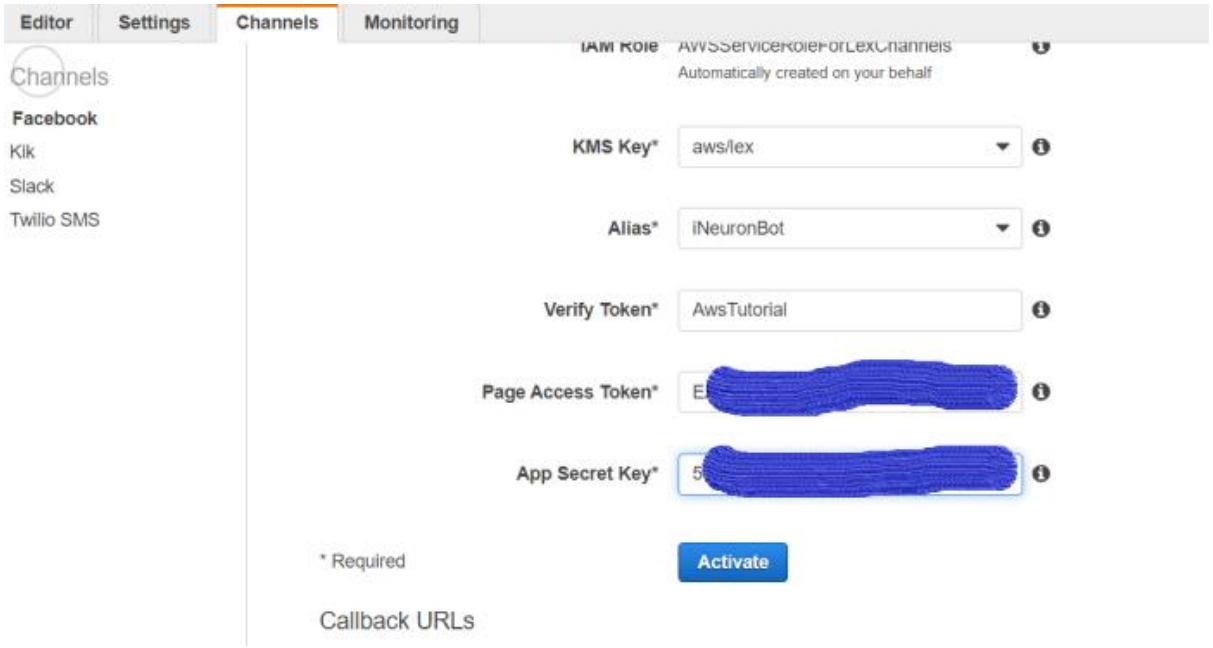
Now for app secret key, go back to facebook for developers and click on basic tab below settings:



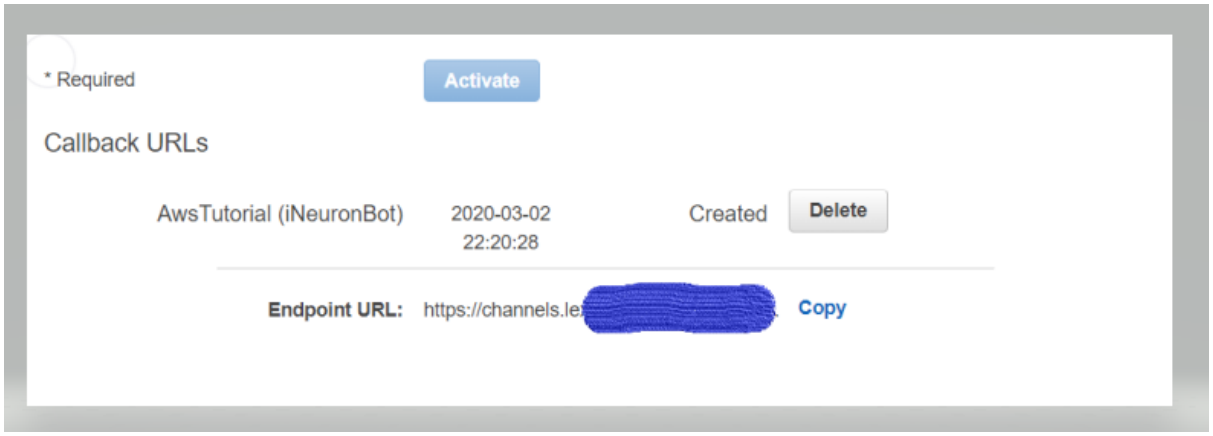
You will see a page like this after clicking. Click on the show button, you will prompted to enter your password for facebook, enter your password and you can copy the code:



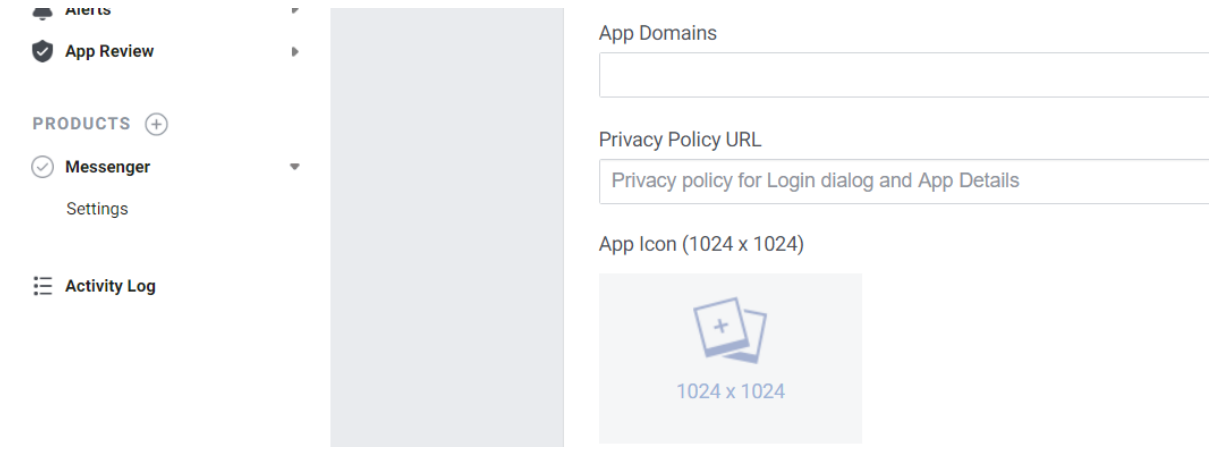
Copy the app secret and come back to your bot channel page and paste in the app secret field:



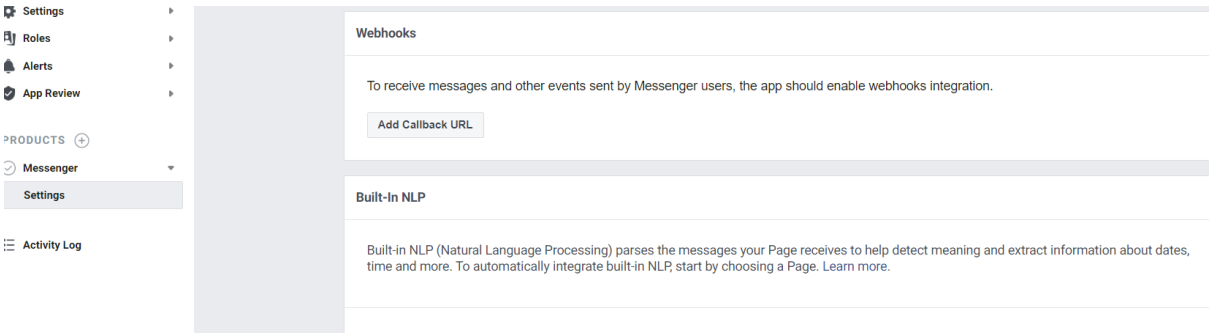
Click the activate button after filling all the details. You will get an endpoint url as shown below after activation. Copy the url:



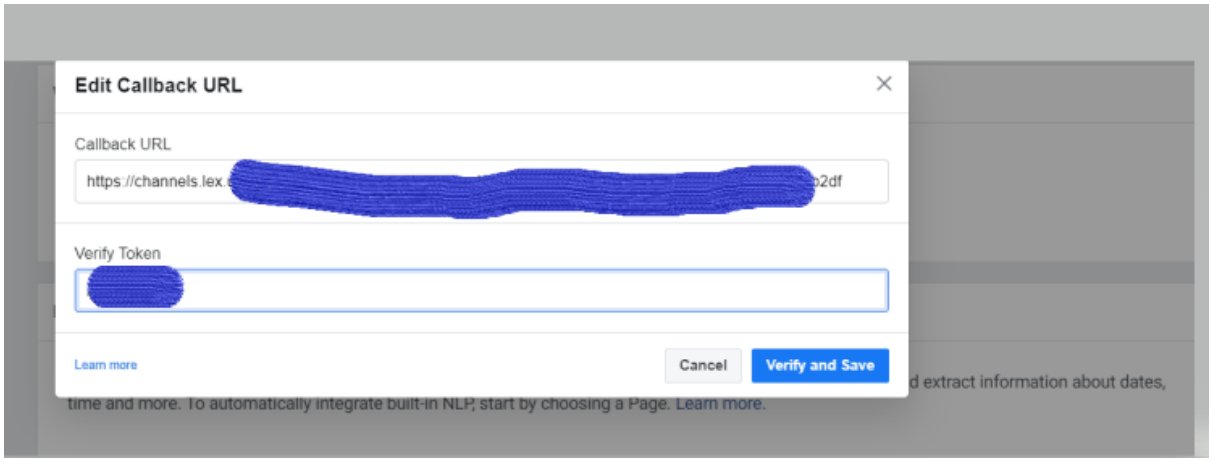
Go back to facebook for developer page in messenger settings tab:



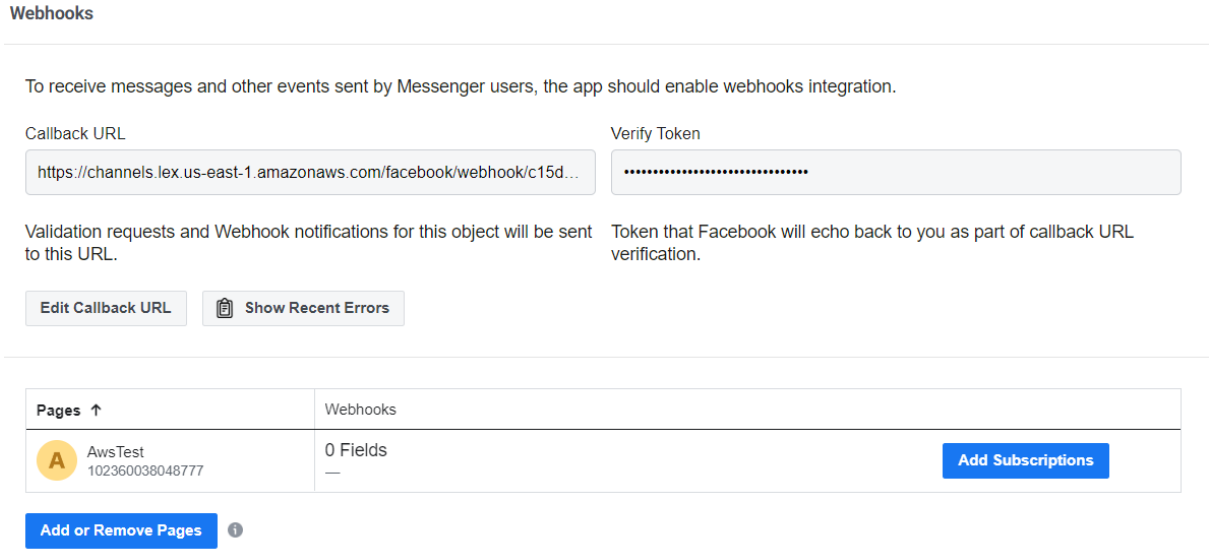
Click on add callback url under webhooks section:



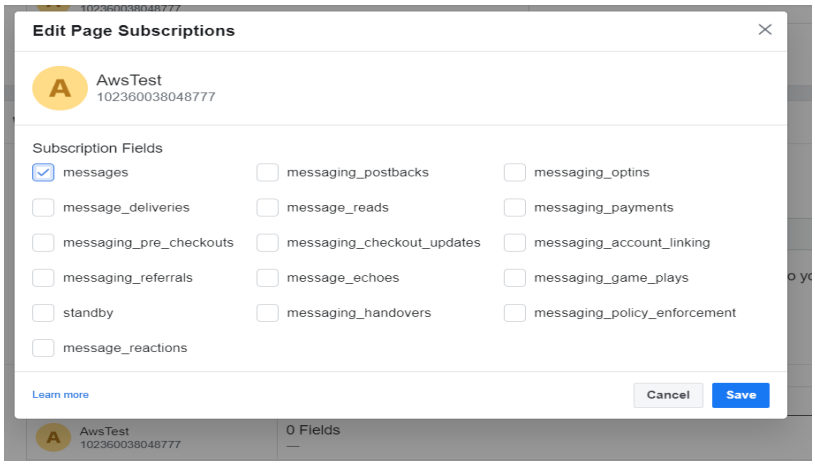
Enter all the details, paste the url in the first box and in second field enter the token value created at the bot channel page:



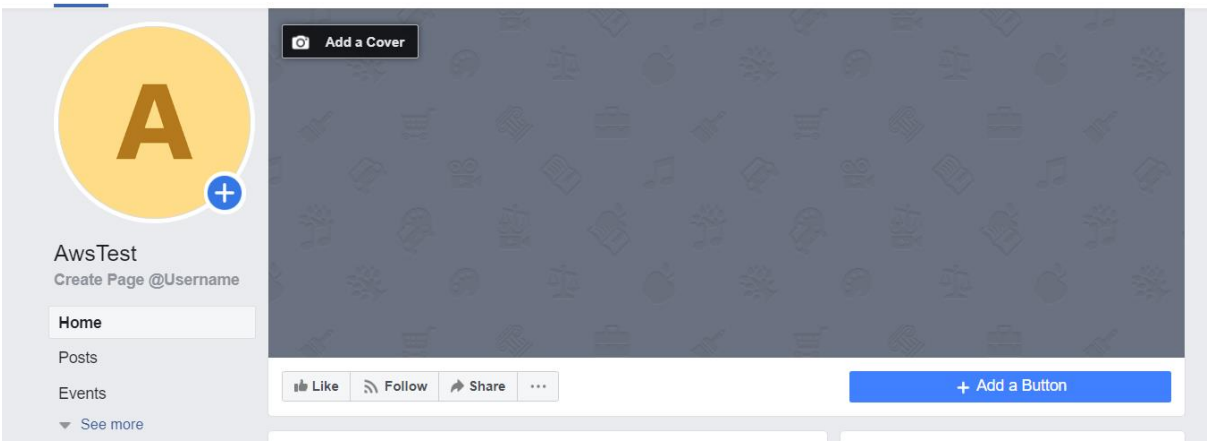
After clicking erify and save, the webhook section will be updated and you can see an Add subscription button like below. Click on it.



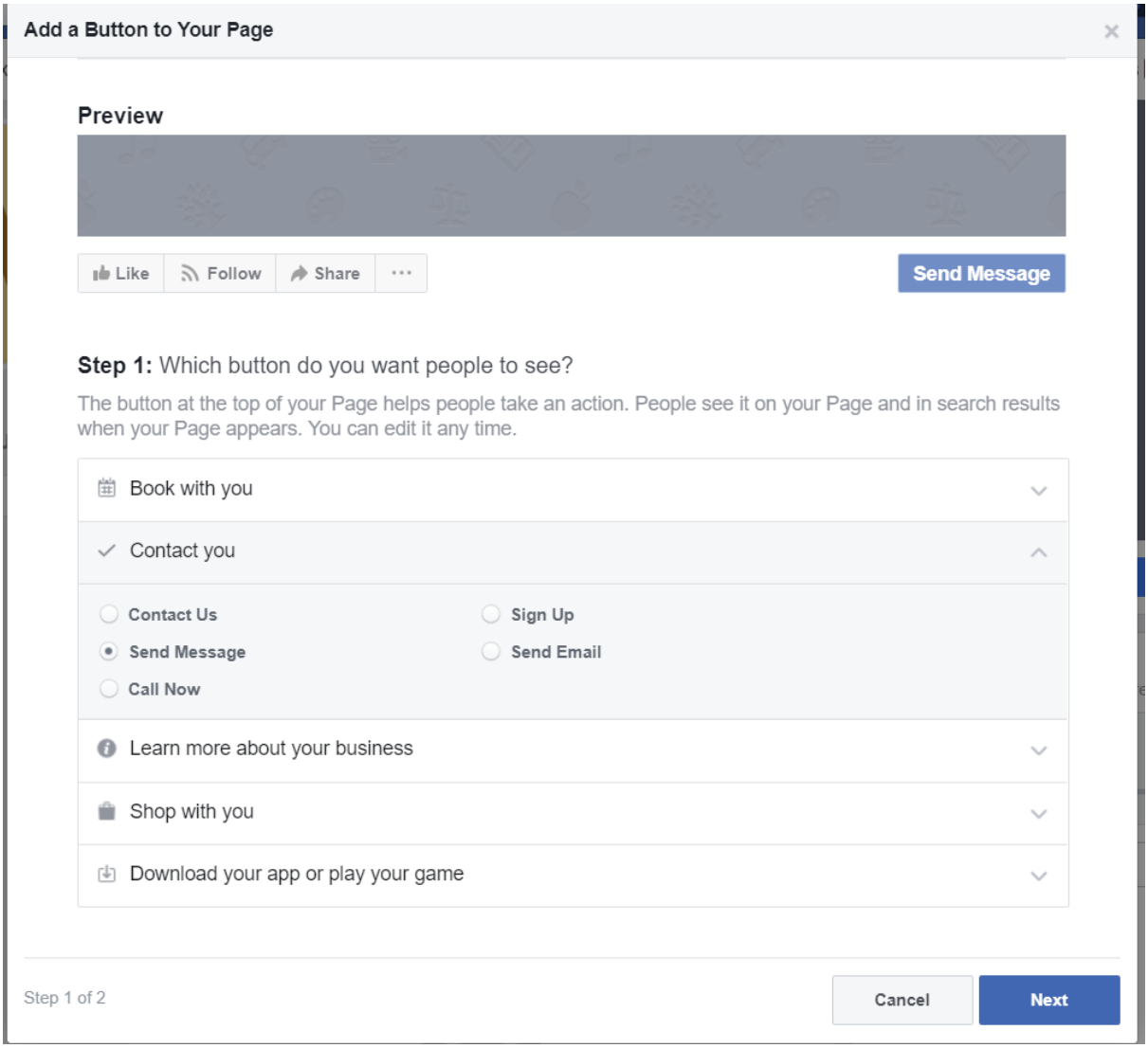
After clicking, check the messages box and save.



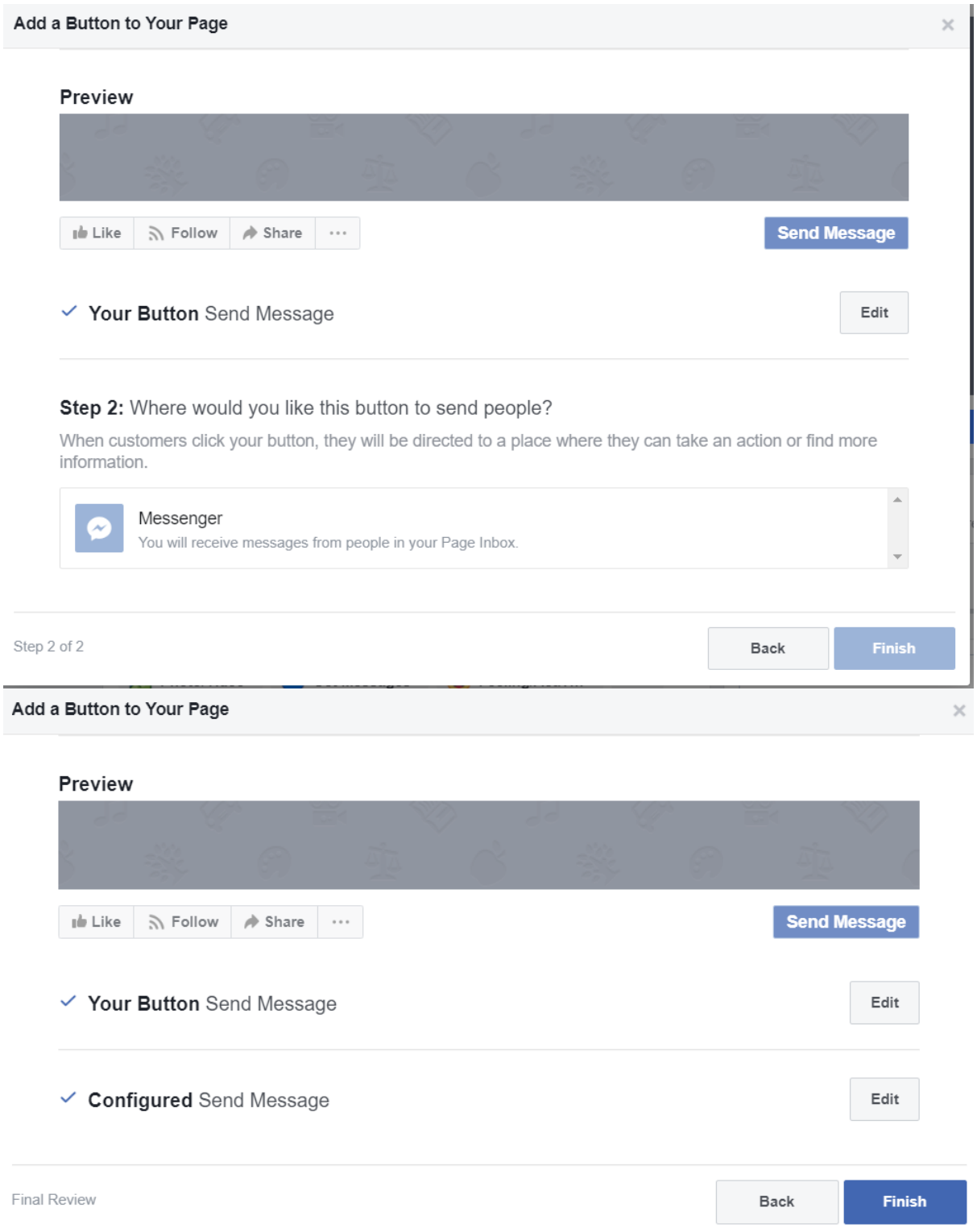
Now the setup is completed. Go to your page on facebook and hit refresh.



Click on Add a button.
Select the “send message” button in the “Contact you” tab and click next.

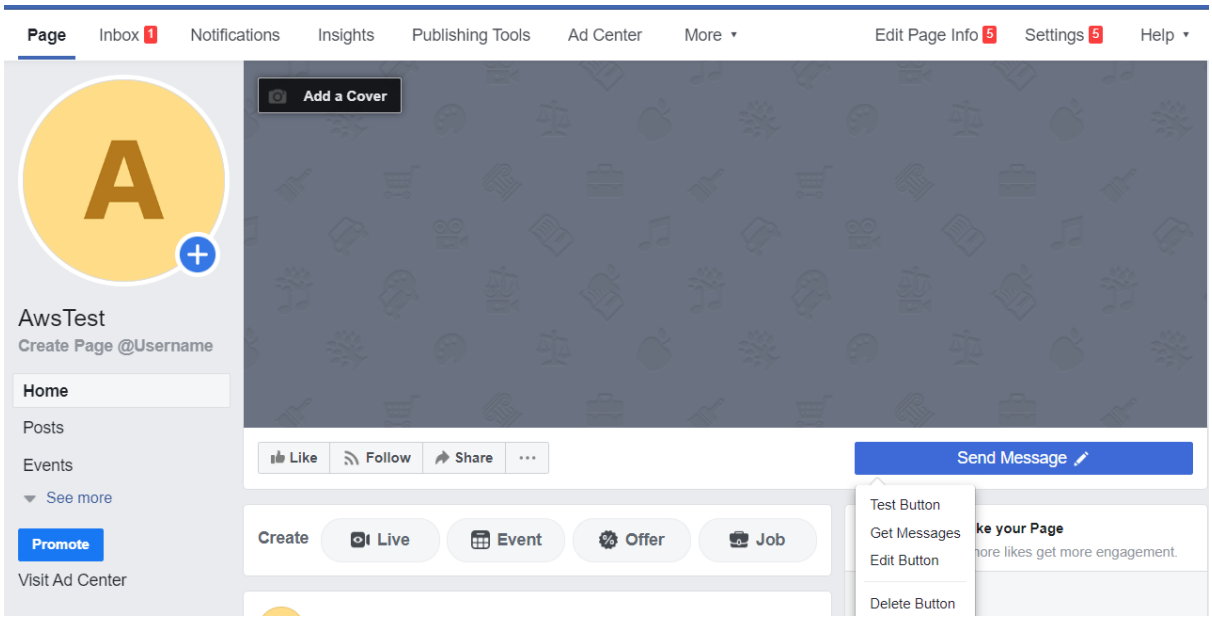


Click on messenger on the next page and click finish.

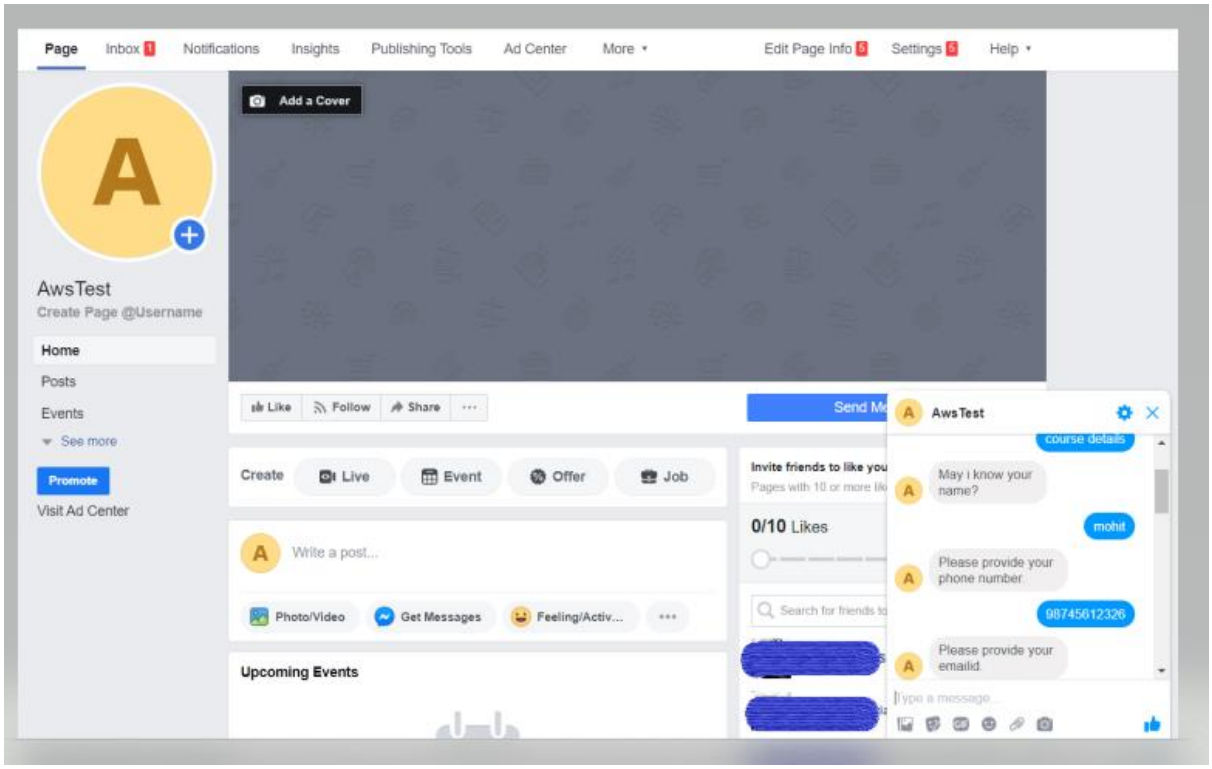


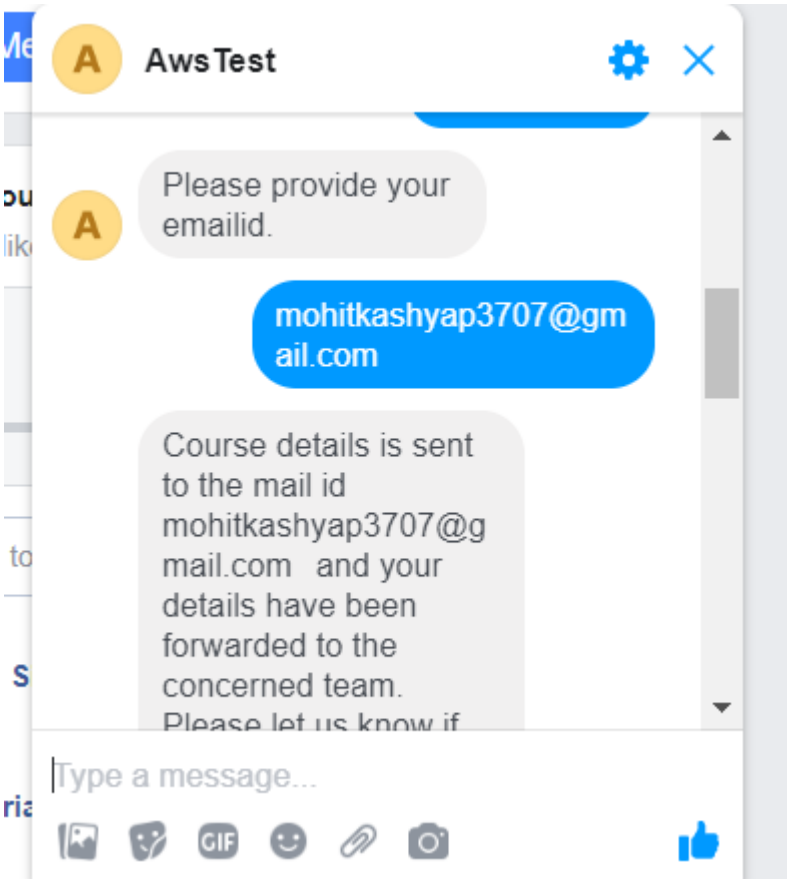
Now your chat bot is ready to be used.

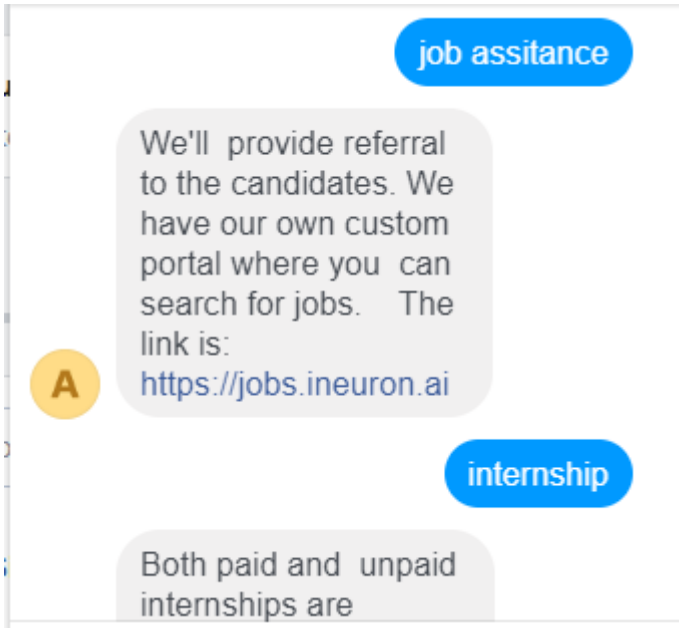
Hover the mouse over send message button and click on test button:

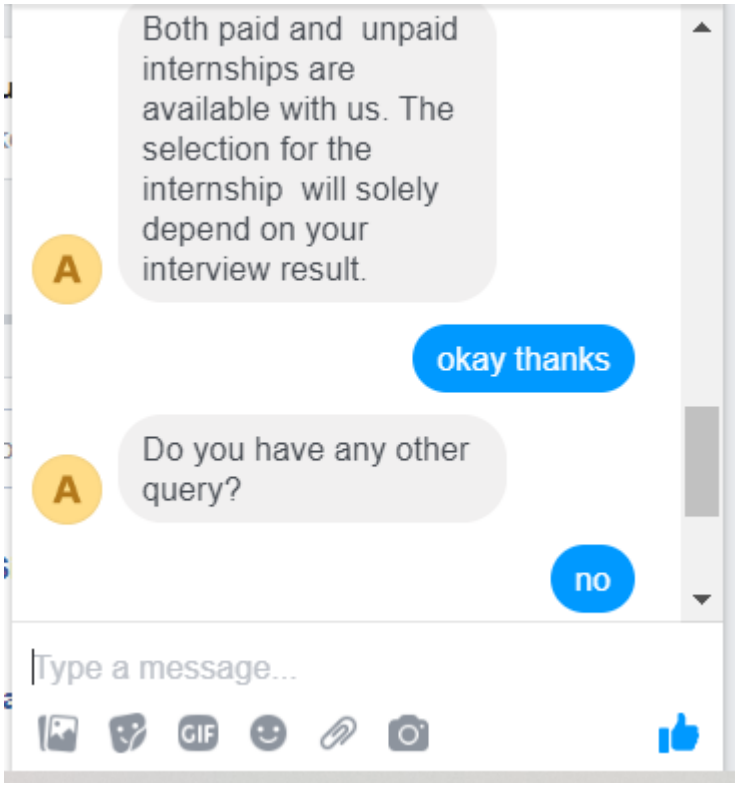


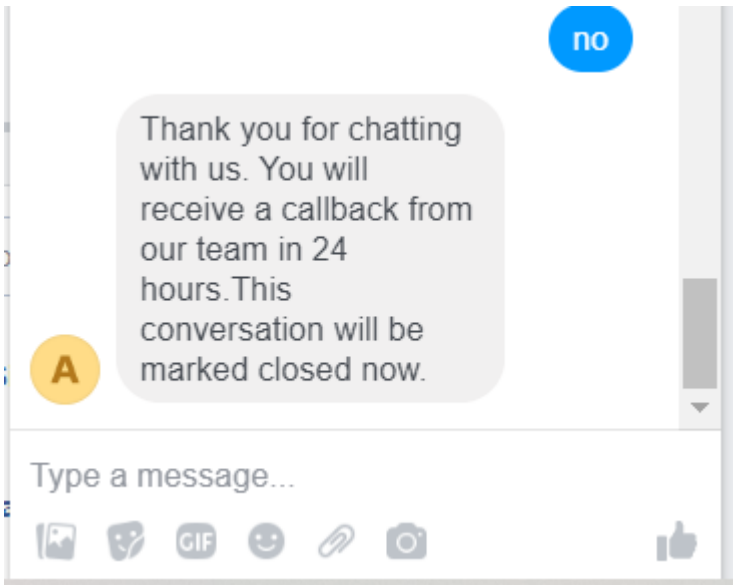
Start chatting.




AwsTest
Please provide your emailid.
mohitkashyap3707@gmail.com
Course details is sent to the mail id mohitkashyap3707@gmail.com and your details have been forwarded to the concerned team. Please let us know if
Type a message...


job assitance
We'll provide referral to the candidates. We have our own custom portal where you can search for jobs. The link is: https://jobs.ineuron.ai
internship
Both paid and unpaid internships are





Now your Bot is deployed on facebook and is ready for use.

Happy Learning!!!!!!