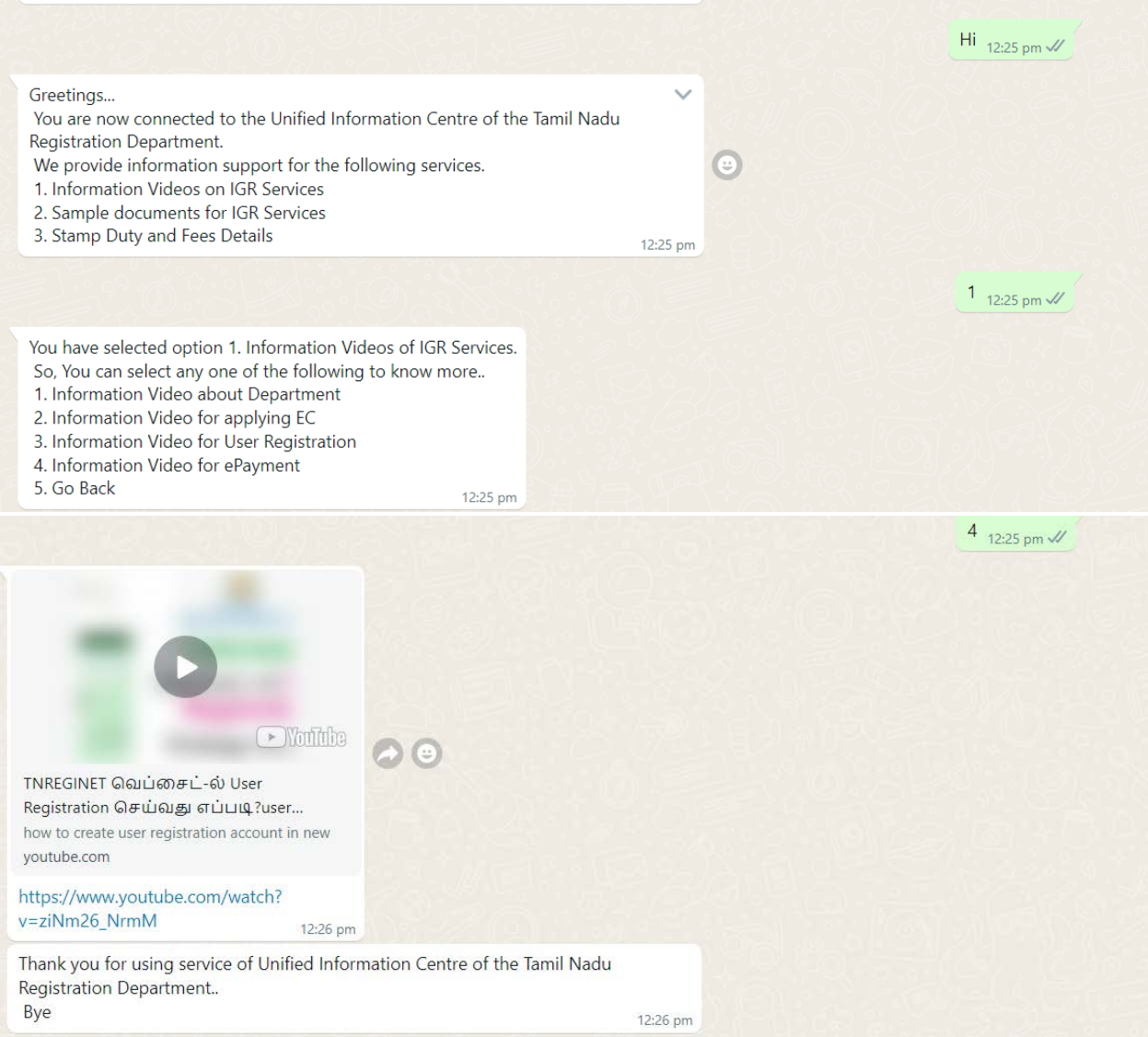
This is the Whatsapp bot is operating with options .



To create a whatsapp chatbot needs 3 following things

1] RASA open source 3.x framework

2]Whatsapp Sandbox credentials (Twillo Authentication token and password) .

3]Web site tunneling app (ngrok,pagekite,localtunnel).

1] RASA open soure 3.x framework

This chatbot is working with option so, we need to create working option using rasa forms.
For more information https://rasa.com/docs/rasa/forms/

First add following intent in nlu.yml file

```
- intent: selection_1
  examples: |
    - [1](options)
    - [2](options)
    - [3](options)
    - [4](options)
    - [5](options)
```

In above file 1,2,3,4,5 are the entities(options) given by the user.
You can check more about intents ansd entities on followin link https://rasa.com/docs/rasa/nlu-training-data/

Now edit domain.yml file as follows:

```
intents:
- affirm
- deny
- goodbye
- greet
- mood_great
- mood_unhappy
- nlu_fallback
- selection_1
- selection_2

entities:
- options

slots:
  choices_1:
    type: text
    influence_conversation: true
    mappings:
    - type: from_entit
      entity: options
```

```
  choices_2:
    type: text
    influence_conversation: true
    mappings:
    - type: from_entity
      entity: options

forms:
  bot_form:
    required_slots:
    - choices_1
    - choices_2




responses:

  utter_greet:
  - text: Hey! How are you?

  utter_first_layer:
  - text: |
      Greetings...
       You are now connected to the Unified Information Centre of the Tamil Nadu Registration
Department.
       We provide information support for the following services.
       1. Information Videos on IGR Services
       2. Sample documents for IGR Services
       3. Stamp Duty and Fees Details

  utter_ask_2nd_layer_1:
  - text: |-
      You have selected option 1. Information Videos of IGR Services.
       So, You can select any one of the following to know more..
       1. Information Video about Department
       2. Information Video for applying EC
       3. Information Video for User Registration
       4. Information Video for ePayment

      5. Go Back

  utter_ask_2nd_layer_2:
```

```
  - text: |-
      You have selected option 2. Information Videos of IGR Services.
       So, You can select any one of the following to know more..
       1. Link for Registration Department
       2. Sample formats of deeds
       3. Stamp Duty and Fees
       4. Go Back

  utter_ask_2nd_layer_3:
  - text: |-
      You have selected option 2. Information Videos of IGR Services.
       So, You can select any one of the following to know more..
       1. stamp Duty
       2. Stamp Fees
       3. Go Back

  utter_did_that_help:
  - text: Did that help you?

  utter_happy:
  - text: Great, carry on!

  utter_submit:
  - text: |-
      Thank you for using service of Unified Information Centre of the Tamil Nadu Registration
Department..
       Bye

actions:
- utter_first_layer
- utter_greet
- utter_submit
- validate_bot_form

session_config:
  session_expiration_time: 60
  carry_over_slots_to_new_session: true
```

Now in this project we are using rasa form which is in-built form in rasa..
For that we have to create form and slots in domain.yml file as follows.
But first add intent selection_1 in the domain.yml file as follows.
For more information https://rasa.com/docs/rasa/forms/

```
intents:
- affirm
- deny
- goodbye
- greet
- mood_great
- mood_unhappy
- nlu_fallback
- selection_1
entities:
- options
```

We have add new entities also
Now , we have to make slots which act as a memory and save the input options as user is give in whatsapp.

```
slots:
  choices_1:
    type: text
    influence_conversation: true
    mappings:
    - type: from_entity
      entity: options

  choices_2:
    type: text
    influence_conversation: true
    mappings:
    - type: from_entity
      entity: options
```

In above code we creating two slots because we are taking two inputs from users.
“Type: text ” rasa have some data type which directly related to entities but unfortunately rasa don’t have “int” data type so we have to code int as a string.
For more information: https://rasa.com/docs/rasa/domain#slot-types

“Influence_conversation: true” means slot value is going to influence the conversation during chatbot is running.
For more information: https://rasa.com/docs/rasa/domain#slots-and-conversation-behavior

Mappings are how slot value is going to map from entities.
For more informations : https://rasa.com/docs/rasa/domain#mapping-conditions

```
forms:
  bot_form:
    required_slots:
    - choices_1
    - choices_2
```

Now create rasa form as above and give the slots name which are going to work with that specific form.
For our bot chioces_1 and chioces_2 are going to work with bot_form(form name).
We can build multiple form in single project.
For more information : https://rasa.com/docs/rasa/forms/

```
responses:

  utter_greet:
  - text: Hey! How are you?

  utter_first_layer:
  - text: |
      Greetings...
       You are now connected to the Unified Information Centre of the
Tamil Nadu Registration Department.
       We provide information support for the following services.
       1. Information Videos on IGR Services
       2. Sample documents for IGR Services
       3. Stamp Duty and Fees Details


  utter_ask_2nd_layer_1:
  - text: |-
      You have selected option 1. Information Videos of IGR Services.
       So, You can select any one of the following to know more..
       1. Information Video about Department
       2. Information Video for applying EC
       3. Information Video for User Registration
       4. Information Video for ePayment
       5. Go Back
```

```
  utter_ask_2nd_layer_2:
  - text: |-
      You have selected option 2. Information Videos of IGR Services.
       So, You can select any one of the following to know more..
       1. Link for Registration Department
       2. Sample formats of deeds
       3. Stamp Duty and Fees
       4. Go Back


  utter_ask_2nd_layer_3:
  - text: |-
      You have selected option 2. Information Videos of IGR Services.
       So, You can select any one of the following to know more..
       1. stamp Duty
       2. Stamp Fees
       3. Go Back


  utter_did_that_help:
  - text: Did that help you?

  utter_happy:
  - text: Great, carry on!

  utter_submit:
  - text: |-
      Thank you for using service of Unified Information Centre of the
Tamil Nadu Registration Department..
       Bye
```

In above code all are utterances which chatbot is going to show to the user during conversation..
Final utterance is link related to option selected (which is in actions.py file) by user and utter_submit .
For more informations : https://rasa.com/docs/rasa/responses/

```
actions:
- utter_first_layer
- utter_greet
- utter_submit
- validate_bot_form
```

Now we have add actions what we are creating in actions.py file .
We adding some utterances also so their should not get any error during training.
As shown in above code validate_bot_form is also added because rasa have inbuilt action called validate_form action. Which is really helpful for create flow of chat bot and when and which slots is going to invok so the flow of the chatbot remain unchanged.

Now edit action.py file as follows:

```
import time

from typing import Any, Text, Dict, List

from rasa_sdk import Action , Tracker , FormValidationAction
from rasa_sdk.events import EventType
from rasa_sdk.types import DomainDict
from rasa_sdk.executor import CollectingDispatcher
from rasa_sdk.events import SlotSet


first_layer =
{'1':'second_layer_1','2':'second_layer_2','3':'second_layer_3'}
second_layer_1 =
{'1':'https://www.youtube.com/watch?v=ziNm26_NrmM','2':'https://www.youtub
e.com/watch?v=ziNm26_NrmM','3':'https://www.youtube.com/watch?v=ziNm26_Nrm
M','4':'https://www.youtube.com/watch?v=ziNm26_NrmM','5':'Go Back'}
second_layer_2 =
{'1':'https://tnreginet.gov.in/portal/webHP?requestType=ApplicationRH&acti
onVal=homePage&screenId=114&UserLocaleID=en&_csrf=67510e41-b245-4318-b89d-
79bd2d19a8ae','2':'https://docs.google.com/document/d/101_lifBdmg3K_d9zaeR
Lzp-N-
sCAQ9Olcca31n4Cm40/edit?usp=sharing','3':'https://docs.google.com/document
/d/101_lifBdmg3K_d9zaeRLzp-N-sCAQ9Olcca31n4Cm40/edit?usp=sharing','4':'Go
Back'}
second_layer_3 =
{'1':'https://docs.google.com/document/d/1FecqT2lS2mWyqgDlgv0F6OEUiBpog-
PsI_C1eieIRQc/edit?usp=sharing','2':'https://docs.google.com/document/d/1F
```

ecqT2lS2mWyqgDlgv0F6OEUiBpog-PsI_C1eieIRQc/edit?usp=sharing','3':'Go Back'}

```
v = ''


class ValidateBotForm(FormValidationAction):
    def name(self) -> Text:
        return "validate_bot_form"

    def validate_choices_1(
        self,
        slot_value: Any,
        dispatcher: CollectingDispatcher,
        tracker: Tracker,
        domain: DomainDict,
    ) -> Dict[Text, Any]:
        """Validate `choices_1` value."""

        global v
        slot_1 = tracker.get_slot('choices_1')
        first_selection = first_layer.get( slot_1)

        if first_selection == 'second_layer_1' or first_section ==
'second_layer_2' or first_selection == 'second_layer_3':
            if first_selection == 'second_layer_1':
                v = 'second_layer_1'
                dispatcher.utter_message(response =
"utter_ask_2nd_layer_1")
            elif first_selection == 'second_layer_2':
                v = 'second_layer_2'
                dispatcher.utter_message(response =
"utter_ask_2nd_layer_2")
            elif first_selection == 'second_layer_3':
                v = 'second_layer_3'
                dispatcher.utter_message(response =
"utter_ask_2nd_layer_3")

        else:

            dispatcher.utter_message(text = "Please give valid input")
            return {"choices_1": None}
```

```
        return {"choices_1": slot_value}


    def validate_choices_2(
        self,
        slot_value: Any,
        dispatcher: CollectingDispatcher,
        tracker: Tracker,
        domain: DomainDict,
    ) -> Dict[Text, Any]:
        """Validate `choices_2` value."""

        slot_2 = tracker.get_slot('choices_2')
        second_selection_1 = second_layer_1.get(slot_2)
        second_selection_2 = second_layer_2.get(slot_2)
        second_selection_3 = second_layer_3.get(slot_2)

        if v == 'second_layer_1':
            if slot_2 in list(second_layer_1.keys()):
                if slot_2 == '5':
                    dispatcher.utter_message(response ='utter_first_layer'
)
                    return {"choices_2": None,"choices_1": None}
                else:
                    dispatcher.utter_message(text = second_selection_1)
                    time.sleep(3)
            else:
                dispatcher.utter_message(text = "Please give valid input")
                return {"choices_2": None}
        elif v == 'second_layer_2':
            if slot_2 in list(second_layer_2.keys()):
                if slot_2 == '4':
                    dispatcher.utter_message(response ='utter_first_layer'
)
                    return {"choices_2": None,"choices_1": None}
                else:
                    dispatcher.utter_message(text = second_selection_2)
                    time.sleep(3)

            else:
                dispatcher.utter_message(text = "Please give valid input")
                return {"choices_2": None}
        elif v == 'second_layer_3':
            if slot_2 in list(second_layer_3.keys()):
                if slot_2 == '3':
```

```
                    dispatcher.utter_message(response ='utter_first_layer'
)
                    return {"choices_2": None,"choices_1": None}
                else:
                    dispatcher.utter_message(text = second_selection_3)
                    time.sleep(3)
            else:
                dispatcher.utter_message(text = "please give valid input")
                return {"choices_2": None}


        print(tracker.sender_id,tracker.events)

        return {"choices_2": slot_value}
```

Now in first lines as follows

```
import time
```

We are importing time library to get exact time in code.

```
from typing import Any, Text, Dict, List

from rasa_sdk import Action , Tracker , FormValidationAction
from rasa_sdk.events import EventType
```

We are importing above library to track the all chat going on or events going on during chat sessions . we are also calling FormValidationAction to validate our bot_form.

```
from rasa_sdk.types import DomainDict
```

Now we are importing domain file in function

```
from rasa_sdk.executor import CollectingDispatcher
```

Collecting Dispacher show output to the user which is given in the actions.py file.

```
from rasa_sdk.events import SlotSet
```

Slotset is the the function to set the values to the slot.

```
first_layer =
{'1':'second_layer_1','2':'second_layer_2','3':'second_layer_3'}
second_layer_1 =
{'1':'https://www.youtube.com/watch?v=ziNm26_NrmM','2':'https://www.youtub
e.com/watch?v=ziNm26_NrmM','3':'https://www.youtube.com/watch?v=ziNm26_Nrm
M','4':'https://www.youtube.com/watch?v=ziNm26_NrmM','5':'Go Back'}
```

```
second_layer_2 =
{'1':'https://tnreginet.gov.in/portal/webHP?requestType=ApplicationRH&actionVal=homePage&screenId=114&UserLocaleID=en&_csrf=67510e41-b245-4318-b89d-79bd2d19a8ae','2':'https://docs.google.com/document/d/101_lifBdmg3K_d9zaeRLzp-N-sCAQ9Olcca31n4Cm40/edit?usp=sharing','3':'https://docs.google.com/document/d/101_lifBdmg3K_d9zaeRLzp-N-sCAQ9Olcca31n4Cm40/edit?usp=sharing','4':'Go Back'}
second_layer_3 =
{'1':'https://docs.google.com/document/d/1FecqT2lS2mWyqgDlgv0F6OEUiBpog-PsI_C1eieIRQc/edit?usp=sharing','2':'https://docs.google.com/document/d/1FecqT2lS2mWyqgDlgv0F6OEUiBpog-PsI_C1eieIRQc/edit?usp=sharing','3':'Go Back'}
```

Above Dictionaries have values as a links which is our last response from chat bot.

```
v = ''
```

We have 1 global variable to hold the values.-

```
class ValidateBotForm(FormValidationAction):
    def name(self) -> Text:
        return "validate_bot_form"
```

We introducing class which have same name as bot_form given in domain.yml file.
This is the in-built syntax to validate RASA form.

```
    def validate_choices_1(
        self,
        slot_value: Any,
        dispatcher: CollectingDispatcher,
        tracker: Tracker,
        domain: DomainDict,
    ) -> Dict[Text, Any]:
        """Validate `choices_1` value."""
```

Now in above function validate slot one by one as our flow.
This is in-built function for RASA form.

```
        global v
        slot_1 = tracker.get_slot('choices_1')
        first_selection = first_layer.get( slot_1)
```

Now in above code first we are calling global variable into function.
slot_1 is place holder for choices_1 slot value when user select or give option in chatbot.

Then it will select value from dictionary first_layer and save to the first_selections

```
               if first_selection == 'second_layer_1' or first_section ==
'second_layer_2' or first_selection == 'second_layer_3':
            if first_selection == 'second_layer_1':
                v = 'second_layer_1'
                dispatcher.utter_message(response =
"utter_ask_2nd_layer_1")
            elif first_selection == 'second_layer_2':
                v = 'second_layer_2'
                dispatcher.utter_message(response =
"utter_ask_2nd_layer_2")
            elif first_selection == 'second_layer_3':
                v = 'second_layer_3'
                dispatcher.utter_message(response =
"utter_ask_2nd_layer_3")

        else:

            dispatcher.utter_message(text = "Please give valid input")
            return {"choices_1": None}



        return {"choices_1": slot_value}
```

Now in above code first if loop iterate through the every value from first_layer dictionary else give “Please give invalid input” as a output and set slot value as null or None.
Second if loop giving output utterances given in domain.yml file and change the value of global variable v .
It will select the flow of first layer of chat bot as per user.

```
 def validate_choices_2(
        self,
        slot_value: Any,
        dispatcher: CollectingDispatcher,
        tracker: Tracker,
        domain: DomainDict,
    ) -> Dict[Text, Any]:
        """Validate `choices_2` value."""

        slot_2 = tracker.get_slot('choices_2')
        second_selection_1 = second_layer_1.get(slot_2)
        second_selection_2 = second_layer_2.get(slot_2)
        second_selection_3 = second_layer_3.get(slot_2)
```

Now above code is same as explained before.
Only difference is slot value is iterating through 3 different dictionaries.
Which is also depends upon the first selection by user.

```
        if v == 'second_layer_1':
            if slot_2 in list(second_layer_1.keys()):
                if slot_2 == '5':
                    dispatcher.utter_message(response ='utter_first_layer'
)
                    return {"choices_2": None,"choices_1": None}
                else:
                    dispatcher.utter_message(text = second_selection_1)
                    time.sleep(3)
            else:
                dispatcher.utter_message(text = "Please give valid input")
                return {"choices_2": None}
```

Now in above code first if loop is invoke through first user input which save values in global variable v.
Second if loop is iterate through all keys in given dictionary as second option is selected by user and if its not in dictionary it will show “Please give valid input” as output.
Now third if loop for “Go back” option. As in this code when go back option got select by user it will make slot_1 and slot_2 value null or None, so it reset the form and user have to give input from start , else it will show the link or other output given in dictionary.
This process is same for the all remaining code.

Now add in stories.yml file as follows:

Now In stories.yml file, we can write multiple stories as per our need .
Stories are needed to be given for flow of chatbot. Although our form will set the flow but it’s compulsory to give the stories for it otherwise it will not work.
In case if we have multiple chat flows then stoaries will decide which flow is going to present in chat.

```
stories:


- story: interactive_story_1
  steps:
  - intent: greet
  - action:  utter_first_layer
  - action: bot_form
  - active_loop: bot_form
  - slot_was_set:
```

```
    - requested_slot: choices_1
  - slot_was_set:
    - choices_1: options
  - slot_was_set:    - requested_slot: choices_2
  - slot_was_set:
    - choices_2: options
  - slot_was_set:
    - requested_slot: null
  - active_loop: null
  - action: utter_submit
  - action: action_restart
```

In above code
All the code written in RASA in-bult syntax. This code is showing when bot form is staring and when its going to end . As per bot_form when and which slot is going to fetch the value from user. How slot going map the value from entities.
For more information: https://rasa.com/docs/rasa/stories/

Now add rules in rules.yml file as follows:

Rules are necessary when we are working with forms . During chat if we have multiple forms and stories then we have to clarify that which form is activating by which action so it will decide the flow of chat bot.
Stories and rule should be match with each other otherwise RASA will not get train and it will through error.

For more information: https://rasa.com/docs/rasa/rules/

```
- rule: Activate bot Form
  steps:
  - action: utter_first_layer
  - action: bot_form
  - active_loop: bot_form

- rule: Submit bot Form
  condition:
  - active_loop: bot_form
  steps:
  - action: bot_form
  - active_loop: null
  - slot_was_set:
    - requested_slot: null
  - action: utter_submit
  - action: action_restart
```

In above code , we can see that when bot form is activating and when it is deactivating.

Now uncomment the following in config.yml file

```
# The config recipe.
# https://rasa.com/docs/rasa/model-configuration/
recipe: default.v1

# Configuration for Rasa NLU.
# https://rasa.com/docs/rasa/nlu/components/
language: en

pipeline:
# # No configuration for the NLU pipeline was provided. The following default pipeline was used to train your model.
# # If you'd like to customize it, uncomment and adjust the pipeline.
# # See https://rasa.com/docs/rasa/tuning-your-model for more information.
  - name: WhitespaceTokenizer
  - name: RegexFeaturizer
  - name: LexicalSyntacticFeaturizer
  - name: CountVectorsFeaturizer
  - name: CountVectorsFeaturizer
    analyzer: char_wb
    min_ngram: 1
    max_ngram: 4
  - name: DIETClassifier
    epochs: 100
    constrain_similarities: true
  - name: EntitySynonymMapper
  - name: ResponseSelector
    epochs: 100
    constrain_similarities: true
  - name: FallbackClassifier
    threshold: 0.3
    ambiguity_threshold: 0.1

# Configuration for Rasa Core.
# https://rasa.com/docs/rasa/core/policies/
policies:
# # No configuration for policies was provided. The following default policies were used to train your model.
# # If you'd like to customize them, uncomment and adjust the policies.
```

```
# # See https://rasa.com/docs/rasa/policies for more information.
  - name: MemoizationPolicy
  - name: RulePolicy
  - name: UnexpecTEDIntentPolicy
    max_history: 5
    epochs: 100
  - name: TEDPolicy
    max_history: 5
    epochs: 100
    constrain_similarities: true
```

Now uncomment the following in endpoints.yml

```
# This file contains the different endpoints your bot can use.

# Server where the models are pulled from.
# https://rasa.com/docs/rasa/model-storage#fetching-models-from-a-server

#models:
#  url: http://my-server.com/models/default_core@latest
#  wait_time_between_pulls:  10   # [optional](default: 100)

# Server which runs your custom actions.
# https://rasa.com/docs/rasa/custom-actions

action_endpoint:
  url: "http://localhost:5055/webhook"

# Tracker store which is used to store the conversations.
# By default the conversations are stored in memory.
# https://rasa.com/docs/rasa/tracker-stores

#tracker_store:
#    type: redis
#    url: <host of the redis instance, e.g. localhost>
#    port: <port of your redis instance, usually 6379>
#    db: <number of your database within redis, e.g. 0>
#    password: <password used for authentication>
#    use_ssl: <whether or not the communication is encrypted, default false>

#tracker_store:
#    type: mongod
#    url: <url to your mongo instance, e.g. mongodb://localhost:27017>
#    db: <name of the db within your mongo instance, e.g. rasa>
```

```
#    username: <username used for authentication>
#    password: <password used for authentication>

# Event broker which all conversation events should be streamed to.
# https://rasa.com/docs/rasa/event-brokers

#event_broker:
#  url: localhost
#  username: username
#  password: password
#  queue: queue
```

Now make changes in credentials.yml file

```
twilio:
  account_sid: "******************************"
  auth_token: "*******************************"
  twilio_number: "whatsapp:+****************"
```

To get credential follow Step 2].

Now go the cmd and type

```
rasa train
```

It will train the model as per our requirement.
Then type

```
rasa shell
```

to load the bot into command line.

Now open another command line and type

```
rasa run actions
```

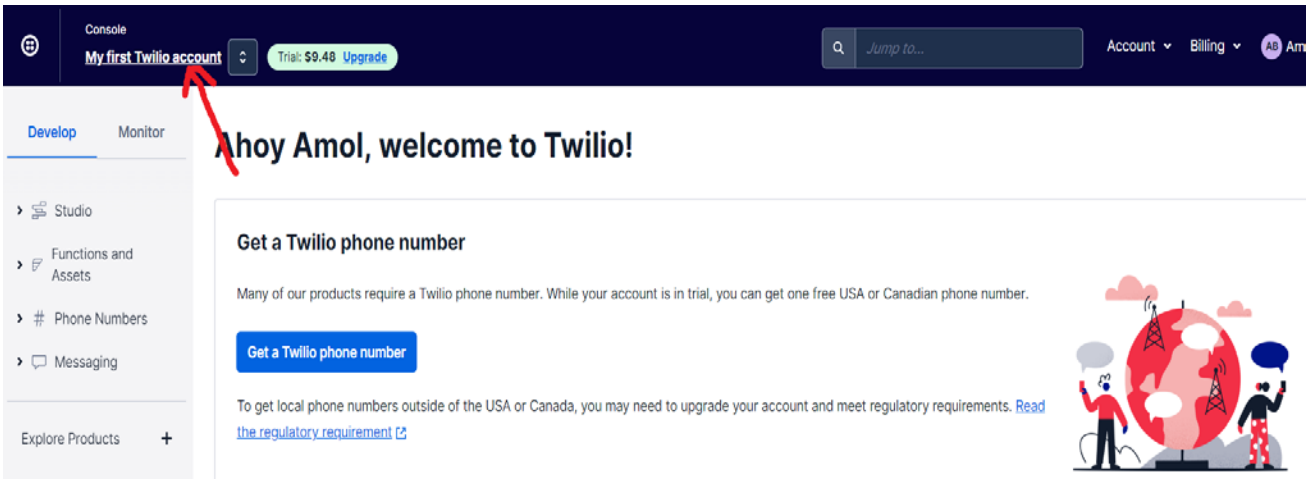
To run actions which wrote in actions.py file.

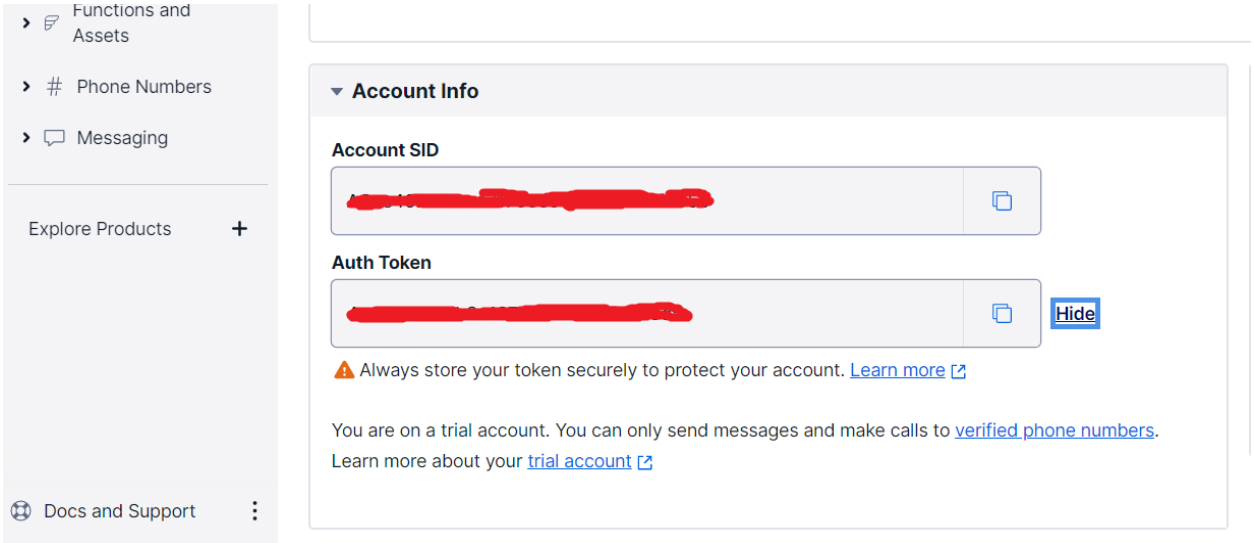
2]Whatsapp Sandbox credentials (Twillo Authentication token and password):

First we to create account on whatsapp sandbox eg.(Twillo, Aisensy)
We are using Twillo here so create an account on https://www.twilio.com/try-twilio .
Follow the step given in https://www.youtube.com/watch?v=b0gVRlbyViY .

Get credentials from Twillo

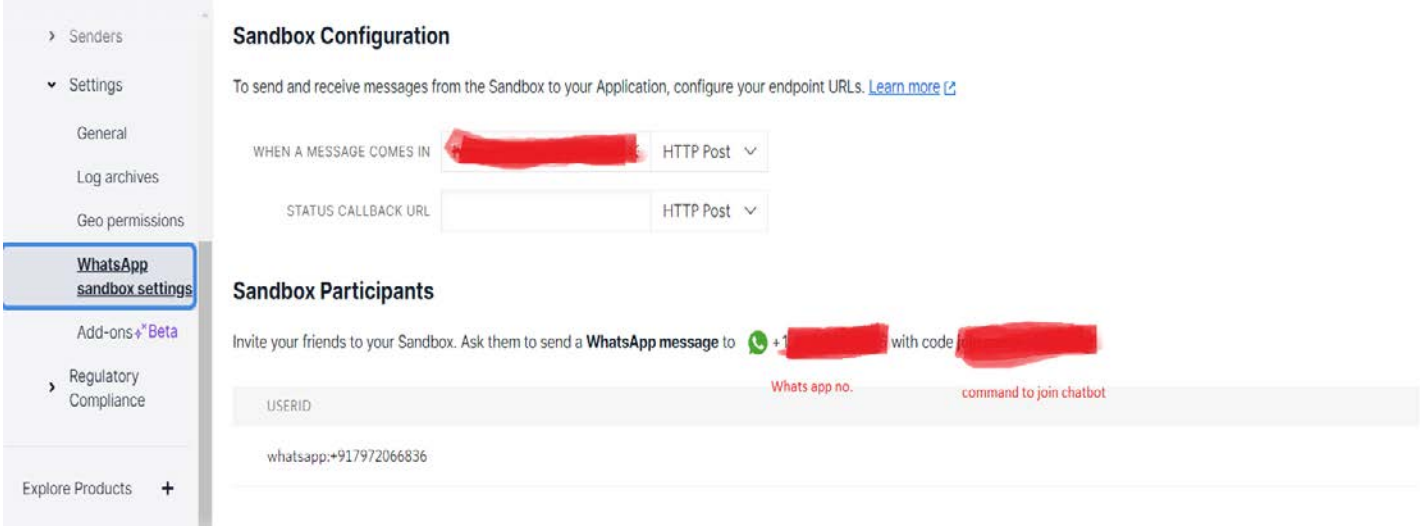
Go in to twillo account and click on your project as follows :



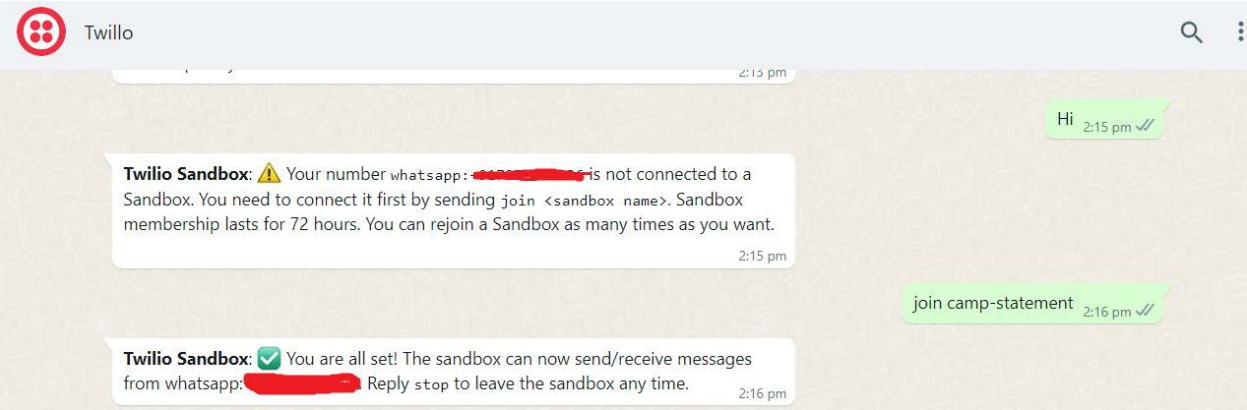


Click on show in Auth Token and Copy both Account SID , Auth Token paste in credentials.yml file of rasa as shown in Step1].

Now go to the option Messaging → Settings → Whatsapp Sandbox Setting



You can get whats app no. and command to chatbot .
You can check whether no. is responding or not by saving no in your phone and go to the whatsapp and type give command which is given. It will respond as follows.



3]]Web site tunneling app (ngrok,pagekite,localtunnel).

This project is working on localtunnel software

Installation of localtunnel software

Localtunnel works on nodejs and npm software.
First install nodejs and npm by following steps https://www.geeksforgeeks.org/installation-of-node-js-on-linux/ .

For installing localtunnel follow the steps given in
https://techmonger.github.io/13/localtunnel-ubuntu/

Steps for Configure Whatsapp Chatbot

First Copy all Credentials from twillo and paste it in credentials.yml file.
Then we have load train model to the rasa server. For that in terminal type

rasa run -m enable-api –cors “*”

Now open another terminal and run action server as follows .

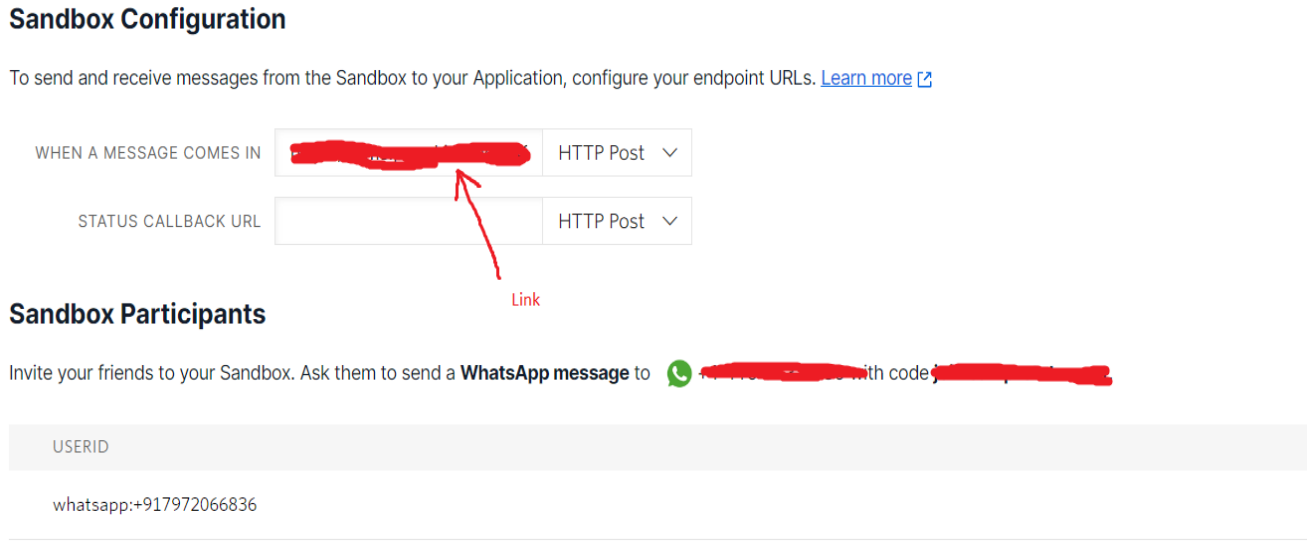
rasa run actions

Now open another terminal and type

lt –port 5005

Don’t press ctrl+c or termate any terminal .

It will give some link copy that link and paste it in Twillo site → messaging → settings →whatsapp sandbox settings.



Now edit the link as follows

Your link starts with https or http/webhooks/twilio/webhook

Now click on save .

Now go to the whats app and try chat on twillo no.