DATA 690 NATURAL LANGUAGE PROCESSING ¶

FINAL PROJECT

AMTRACK CHATBOT SERVICES

In [1]:
```
!pip install flask-ngrok
```

```
Collecting flask-ngrok
  Downloading https://files.pythonhosted.org/packages/af/6c/f54cb686ad1129e27
d125d182f90f52b32f284e6c8df58c1bae54fa1adbc/flask_ngrok-0.0.25-py3-none-any.w
hl
Requirement already satisfied: Flask>=0.8 in /usr/local/lib/python3.7/dist-pa
ckages (from flask-ngrok) (1.1.2)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-pack
ages (from flask-ngrok) (2.23.0)
Requirement already satisfied: Werkzeug>=0.15 in /usr/local/lib/python3.7/dis
t-packages (from Flask>=0.8->flask-ngrok) (1.0.1)
Requirement already satisfied: click>=5.1 in /usr/local/lib/python3.7/dist-pa
ckages (from Flask>=0.8->flask-ngrok) (7.1.2)
Requirement already satisfied: itsdangerous>=0.24 in /usr/local/lib/python3.
7/dist-packages (from Flask>=0.8->flask-ngrok) (1.1.0)
Requirement already satisfied: Jinja2>=2.10.1 in /usr/local/lib/python3.7/dis
t-packages (from Flask>=0.8->flask-ngrok) (2.11.3)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /us
r/local/lib/python3.7/dist-packages (from requests->flask-ngrok) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-
packages (from requests->flask-ngrok) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/
dist-packages (from requests->flask-ngrok) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.
7/dist-packages (from requests->flask-ngrok) (2020.12.5)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/d
ist-packages (from Jinja2>=2.10.1->Flask>=0.8->flask-ngrok) (1.1.1)
Installing collected packages: flask-ngrok
Successfully installed flask-ngrok-0.0.25
```

In [2]:

```
!pip install Flask-Mail
!pip install -U flask-cors
```

```
Collecting Flask-Mail
  Downloading https://files.pythonhosted.org/packages/05/2f/6a545452040c25565
59779db87148d2a85e78a26f90326647b51dc5e81e9/Flask-Mail-0.9.1.tar.gz (45kB)
     |████████████████████████████████| 51kB 1.8MB/s
Requirement already satisfied: Flask in /usr/local/lib/python3.7/dist-package
s (from Flask-Mail) (1.1.2)
Collecting blinker
  Downloading https://files.pythonhosted.org/packages/1b/51/e2a9f3b757eb802f6
1dc1f2b09c8c99f6eb01cf06416c0671253536517b6/blinker-1.4.tar.gz (111kB)
     |████████████████████████████████| 112kB 4.9MB/s
Requirement already satisfied: Werkzeug>=0.15 in /usr/local/lib/python3.7/dis
t-packages (from Flask->Flask-Mail) (1.0.1)
Requirement already satisfied: itsdangerous>=0.24 in /usr/local/lib/python3.
7/dist-packages (from Flask->Flask-Mail) (1.1.0)
Requirement already satisfied: Jinja2>=2.10.1 in /usr/local/lib/python3.7/dis
t-packages (from Flask->Flask-Mail) (2.11.3)
Requirement already satisfied: click>=5.1 in /usr/local/lib/python3.7/dist-pa
ckages (from Flask->Flask-Mail) (7.1.2)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/d
ist-packages (from Jinja2>=2.10.1->Flask->Flask-Mail) (1.1.1)
Building wheels for collected packages: Flask-Mail, blinker
  Building wheel for Flask-Mail (setup.py) ... done
  Created wheel for Flask-Mail: filename=Flask_Mail-0.9.1-cp37-none-any.whl s
ize=7568 sha256=9da3e159bad7ad904eb6651cfa888fcab7b1b20fb6d2b1d445a9d2e1353d7
458
  Stored in directory: /root/.cache/pip/wheels/eb/aa/d9/34b8f2f9bce7d06a4d07f
d46078770584d5504949ebfa286f5
  Building wheel for blinker (setup.py) ... done
  Created wheel for blinker: filename=blinker-1.4-cp37-none-any.whl size=1344
8 sha256=683f0f3b7120e1c518c720707dca217d9e65fbfe826c8f2e5f37a48231257658
  Stored in directory: /root/.cache/pip/wheels/92/a0/00/8690a57883956a301d91c
f4ec999cc0b258b01e3f548f86e89
Successfully built Flask-Mail blinker
Installing collected packages: blinker, Flask-Mail
Successfully installed Flask-Mail-0.9.1 blinker-1.4
Collecting flask-cors
  Downloading https://files.pythonhosted.org/packages/db/84/901e700de86604b1c
4ef4b57110d4e947c218b9997adf5d38fa7da493bce/Flask_Cors-3.0.10-py2.py3-none-an
y.whl
Requirement already satisfied, skipping upgrade: Flask>=0.9 in /usr/local/li
b/python3.7/dist-packages (from flask-cors) (1.1.2)
Requirement already satisfied, skipping upgrade: Six in /usr/local/lib/python
3.7/dist-packages (from flask-cors) (1.15.0)
Requirement already satisfied, skipping upgrade: Werkzeug>=0.15 in /usr/loca
l/lib/python3.7/dist-packages (from Flask>=0.9->flask-cors) (1.0.1)
Requirement already satisfied, skipping upgrade: itsdangerous>=0.24 in /usr/l
ocal/lib/python3.7/dist-packages (from Flask>=0.9->flask-cors) (1.1.0)
Requirement already satisfied, skipping upgrade: click>=5.1 in /usr/local/li
b/python3.7/dist-packages (from Flask>=0.9->flask-cors) (7.1.2)
Requirement already satisfied, skipping upgrade: Jinja2>=2.10.1 in /usr/loca
l/lib/python3.7/dist-packages (from Flask>=0.9->flask-cors) (2.11.3)
Requirement already satisfied, skipping upgrade: MarkupSafe>=0.23 in /usr/loc
al/lib/python3.7/dist-packages (from Jinja2>=2.10.1->Flask>=0.9->flask-cors)
(1.1.1)
Installing collected packages: flask-cors
Successfully installed flask-cors-3.0.10
```

In [3]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

In [5]:
```
import pandas as pd
import numpy as np
PATH="/content/drive/MyDrive/Source_destination_updated_test - Source_destinat
ion_updated_test.csv"


city_data=pd.read_csv(PATH)
city_data.head(80)
```

Out[5]:

	Source	Destination	Timings	Source_city	Destination_city	Day	price	name	arr_time
0	NYF	WAS	6:15	New York State Fair Station	Washington DC	Mon	25	New York State Fair Station - Washington express	6:1
1	NYF	WAS	6:15	New York State Fair Station	Washington DC	Mon	15	New York State Fair Station - Washington express	6:1
2	NYF	WAS	6:15	New York State Fair Station	Washington DC	Mon	12	New York State Fair Station - Washington express	6:1
3	NYF	WAS	18:15	New York State Fair Station	Washington DC	Mon	25	New York State Fair Station - Washington express	18:1
4	NYF	WAS	18:15	New York State Fair Station	Washington DC	Mon	15	New York State Fair Station - Washington express	18:1
...	...	...	...	...	...	...	...	...	..
75	NYF	COL	12:30	New York State Fair Station	Columbus	Sun	28	New York State Fair Station- Columbus express	12:3
76	NYF	COL	12:30	New York State Fair Station	Columbus	Sun	19	New York State Fair Station- Columbus express	12:3
77	NYF	COL	12:30	New York State Fair Station	Columbus	Sun	13	New York State Fair Station- Columbus express	12:3
78	BOS	WAS	16:30	Boston	Washington DC	Sun	34	Boston-Washington DC express	16:3
79	BOS	WAS	16:30	Boston	Washington DC	Sun	24	Boston-Washington DC express	16:3

80 rows × 11 columns

In [10]:
```
from flask_ngrok import run_with_ngrok
from flask import Flask,request, make_response, jsonify
from flask_mail import Mail, Message
from flask_cors import CORS, cross_origin
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
import random
import json
import datetime
import warnings
warnings.filterwarnings('ignore')

sender = 'travelassistantbot123@gmail.com'
receivers = [sender]
password='Qazwsx=123'

app = Flask(__name__)

#mail = Mail(app)
run_with_ngrok(app)

# default route
@app.route('/')
def index():
    return 'Hello World!'

# create a route for webhook
def send_mail(contact_list):
  try:
    s = smtplib.SMTP('smtp.gmail.com', 587)
    s.starttls()
    s.login(sender,password)
    recipient_mail=contact_list[2]
    name=contact_list[0]
    phone=contact_list[1]
    select_src=contact_list[3]
    select_dest=contact_list[4]
    select_arr_time=contact_list[5]
    select_dept_time=contact_list[6]
    select_price=contact_list[7]
    day_of_travel=contact_list[8]
    date_of_journey=contact_list[9]
    session_id=contact_list[10]
    arr_tim = datetime.datetime.strptime(select_arr_time,"%Y-%m-%dT%H:%M:%S%z"
)
    dept_tim = datetime.datetime.strptime(select_dept_time,"%Y-%m-%dT%H:%M:%S%
z")
    doj_tim = datetime.datetime.strptime(date_of_journey,"%Y-%m-%dT%H:%M:%S%z"
)
    new_format = "%I:%M"
    doj_format="%d-%m-%Y"
    arrival_time=arr_tim.strftime(new_format)
    dept_time=dept_tim.strftime(new_format)
    doj_date=doj_tim.strftime(doj_format)
    print(recipient_mail)
    print("-----")
    print(name)
    print("-----")
    print(phone)
    #recipient_ph=request.args.get('phone')
    ticket_id=random.randint(1000,9000)
    msg = MIMEMultipart()
    msg['From'] = sender
    msg['To'] = recipient_mail
    msg['Subject'] = 'Booking Details :BNG'+str(ticket_id)
    msg.attach(MIMEText("Hello "+ name+", \nHere is your Ticket details : \n T
icket ID : BNG"+str(ticket_id)+"\n Passenger name : "+name+"\n Date of Journey
: "+doj_date+"\n From : "+select_src+" To : "+select_dest+"\n Arrival : "+arri
val_time+" Departure : "+dept_time+"\n Price : "+str(select_price)+"$" , 'plai
n'))
    s.sendmail(sender,recipient_mail, msg.as_string())

   # req = request.get_json(force=True)

    # fetch action from json
    #action = req.get('queryResult').get('action')

    # return a fulfillment response
  #  return {'fulfillmentText': 'This is a response from webhook.'}
    return {

            "fulfillmentMessages": [
                {
                    "text": {
                        "text": [
                            "Hurray pack your bags .\n\n Your booking BNG"+str
(ticket_id)+" is confirmed .\n\n we have sent you the ticket details over mai
l."
                        ],


                    }
                },
                {
                    "payload" :{
                      "richContent": [
                        [
                          {
                            "options": [
                               {
                                   "text": "Book a new ticket"
                                 },
                               {
                                   "text": "Ask Query"
                               },
                             {
                                "text": "Nothing else"
                             }
                            ],
                            "type": "chips"
                          }
                        ]
                        ]
                      }
                  }


            ],
            "outputContexts": [
                                    {
                                      "name": "projects/test-davr/agent/sessio
ns/"+session_id+"/contexts/sendemail",
                                      "lifespanCount": 0

                                    }
                              ]
        }

  except Exception as e:
    #return(str(e))
    return {'fulfillmentText': str(e)}
#@app.route('/send_mail')
def fetch_details(travel_list):
  try:
    ddate=travel_list[2]
    src=travel_list[0]
    des=travel_list[1]
    train_class=travel_list[3]
    # print(src)
    # print("-----")
    # print(des)
    # print("-----")
    # print(ddate)
    sel_date = datetime.datetime.strptime(ddate,"%Y-%m-%dT%H:%M:%S%z")
    time_now = datetime.datetime.now()
    same_day_flag=False
    date_time_format = "%I:%M"
    twenty_four_format = "%H:%M:%S"
    hours_min_format="%H:%M"
    date_year_month_format="%d-%m-%Y"
    day_format="%a"
    query_day=sel_date.strftime(day_format)
    query_same_day_time=sel_date.strftime(hours_min_format)
    query_date_year=sel_date.strftime(date_year_month_format)
    current_date_year=time_now.strftime(date_year_month_format)
    print(query_same_day_time)
    if (query_date_year == current_date_year) :
      same_day_flag= True
    if (city_data[city_data['Source_city'].str.contains(src,case=False)].shape
[0] > 0):
      result_df=city_data[city_data['Source_city'].str.contains(src,case=False
)]
      if (result_df[result_df['Destination_city'].str.contains(des,case=False
)].shape[0] > 0 ):
        final_df_temp=result_df[result_df['Destination_city'].str.contains(des
,case=False)]

        if (final_df_temp[final_df_temp['Day'].str.contains(query_day,case=Fal
se)].shape[0] > 0):
```

```
                final_df=final_df_temp[final_df_temp['Day'].str.contains(query
_day,case=False)]
                if (same_day_flag):
                      # final_df['Timings'] = pd.to_datetime(dc.Timings,format
= '%H:%M')
                      # if (final_df[dc.Timings.dt.strftime('%H:%M').between
('00:00','23:59')].shape[0] > 0) :
                      #          final_df=final_df[dc.Timings.dt.strftime('%
H:%M').between('00:00','23:59')]
                      #          final_df.reset_index(inplace=True)
                                if (final_df[final_df['Class'].str.contains(tr
ain_class,case=False)].shape[0] > 0):
                                    final_df=final_df[final_df['Class'].str.co
ntains(train_class,case=False)]
                                    final_df.reset_index(inplace=True)
                                    webhooks=[]
                                    webhooks_text=[]
                                    rich_txt=[]
                                    for i in range(len(final_df)):
                                      webhooks_text.append({ "text": { "text":
["Train Name : "+str(final_df.loc[i,"name"])+" \n\n Source station : "+str(fin
al_df.loc[i,"Source_city"])+" \n\n Destination station : "+str(final_df.loc[i,
"Destination_city"])+"\n\n Route : "+str(final_df.loc[i,"Source"])+" ==> "+str
(final_df.loc[i,"Destination"])+"\n\n Price: "+str(final_df.loc[i,"price"])+"
 $"+" \n\n Arrival Time : "+final_df.loc[i,"arr_time"]+" \n\n Dep Time : "+fin
al_df.loc[i,"Dep_time"]+"\n\n Day : "+final_df.loc[i,"Day"]+" Class : "+final_
df.loc[i,"Class"]] } },)
                                      webhooks.append( { "text": ["Train Name
 : "+str(final_df.loc[i,"name"])+" \n\n Source station : "+str(final_df.loc[i,
"Source_city"])+" \n\n Destination station : "+str(final_df.loc[i,"Destination
_city"])+"\n\n Route : "+str(final_df.loc[i,"Source"])+" ==> "+str(final_df.lo
c[i,"Destination"])+"\n\n Price: "+str(final_df.loc[i,"price"])+" $"+" \n\n Ar
rival Time : "+final_df.loc[i,"arr_time"]+" \n\n Dep Time : "+final_df.loc[i,
"Dep_time"]+"\n\n Day : "+final_df.loc[i,"Day"]+" Class : "+final_df.loc[i,"Cl
ass"]] } ,)
                                      rich_txt.append( { "text": ["Route : "+s
tr(final_df.loc[i,"Source"])+" ==> "+str(final_df.loc[i,"Destination"])+" Time
: "+final_df.loc[i,"arr_time"]+" -> "+final_df.loc[i,"Dep_time"]+" Price:"+str
(final_df.loc[i,"price"])+" $"+" Day : "+final_df.loc[i,"Day"]+" Class : "+fin
al_df.loc[i,"Class"]] } ,)
                                    #for i in range(len(final_df)):
                                # webhook= "text": { "text": ["Source station
 : "+final_df.loc[i,"Source_city"]+"\n \n destination station : "+final_df.loc
[i,"Destination_city"]+"\n \n Timing : "+"\n \n Price: "+final_df.loc[i,"pric
e"]] };
                                # return {
                                #     "fulfillmentMessages": [ i for i in webh
ooks]
                                #       }
                                else :
                                    #final_df=final_df[final_df['Class'].str.c
ontains(train_class,case=False)]
                                    final_df.reset_index(inplace=True)
                                    webhooks=[]
                                    webhooks_text=[]
                                    rich_txt=[]
                                    for i in range(len(final_df)):
                                        webhooks_text.append({ "text": { "text":
["Train Name : "+str(final_df.loc[i,"name"])+" \n\n Source station : "+str(fin
al_df.loc[i,"Source_city"])+" \n\n Destination station : "+str(final_df.loc[i,
"Destination_city"])+"\n\n Route : "+str(final_df.loc[i,"Source"])+" ==> "+str
(final_df.loc[i,"Destination"])+"\n\n Price: "+str(final_df.loc[i,"price"])+"
 $"+" \n\n Arrival Time : "+final_df.loc[i,"arr_time"]+" \n\n Dep Time : "+fin
al_df.loc[i,"Dep_time"]+"\n\n Day : "+final_df.loc[i,"Day"]+" Class : "+final_
df.loc[i,"Class"]] } },)
                                        webhooks.append( { "text": ["Train Name
 : "+str(final_df.loc[i,"name"])+" \n\n Source station : "+str(final_df.loc[i,
"Source_city"])+" \n\n Destination station : "+str(final_df.loc[i,"Destination
_city"])+"\n\n Route : "+str(final_df.loc[i,"Source"])+" ==> "+str(final_df.lo
c[i,"Destination"])+"\n\n Price: "+str(final_df.loc[i,"price"])+" $"+" \n\n Ar
rival Time : "+final_df.loc[i,"arr_time"]+" \n\n Dep Time : "+final_df.loc[i,
"Dep_time"]+"\n\n Day : "+final_df.loc[i,"Day"]+" Class : "+final_df.loc[i,"Cl
ass"]] } ,)
                                        rich_txt.append( { "text": ["Route : "+s
tr(final_df.loc[i,"Source"])+" ==> "+str(final_df.loc[i,"Destination"])+" Time
: "+final_df.loc[i,"arr_time"]+" -> "+final_df.loc[i,"Dep_time"]+" Price:"+str
(final_df.loc[i,"price"])+" $"+" Day : "+final_df.loc[i,"Day"]+" Class : "+fin
al_df.loc[i,"Class"]] } ,)

                      # else:
                      #         return {
                      #                     "fulfillmentMessages": [
                      #                         {
                      #                             "text": {
                      #                                 "text": [
                      #                                     "Sorry , our train
s were already left on selected time !!!.Please book a new ticket on alternati
ve day"
                      #                                 ]
                      #                             }
                      #                         },
                      #                         {
                      #                           "payload" :{
                      #                             "richContent": [
                      #                               [
                      #                                 {
                      #                                   "options": [
                      #                                     {
                      #                                       "text": "Book
 a new ticket"
                      #                                     },
                      #                                     {
                      #                                       "text": "Ask Q
uery"
                      #                                     },
                      #                                   {
                      #                                     "text": "Nothing
else"
                      #                                   }
                      #                                 ],
                      #                                 "type": "chips"
                      #                               }
                      #                             ]
                      #                           ]
```

```
                    #                            }
                    #                        }
                    #                      ]
                    #                  }           #for i in range(len(final_
df)):
                else :
                    if (final_df[final_df['Class'].str.contains(train_class,ca
se=False)].shape[0] > 0):
                        final_df=final_df[final_df['Class'].str.contains(tra
in_class,case=False)]
                        final_df.reset_index(inplace=True)
                        webhooks=[]
                        webhooks_text=[]
                        rich_txt=[]
                        for i in range(len(final_df)):
                          webhooks_text.append({ "text": { "text": ["Train N
ame : "+str(final_df.loc[i,"name"])+" \n\n Source station : "+str(final_df.loc
[i,"Source_city"])+" \n\n Destination station : "+str(final_df.loc[i,"Destinat
ion_city"])+"\n\n Route : "+str(final_df.loc[i,"Source"])+" ==> "+str(final_df
.loc[i,"Destination"])+"\n\n Price: "+str(final_df.loc[i,"price"])+" $"+" \n\n
Arrival Time : "+final_df.loc[i,"arr_time"]+" \n\n Dep Time : "+final_df.loc[i
,"Dep_time"]+"\n\n Day : "+final_df.loc[i,"Day"]+" Class : "+final_df.loc[i,"C
lass"]] } },)
                          webhooks.append( { "text": ["Train Name : "+str(fi
nal_df.loc[i,"name"])+" \n\n Source station : "+str(final_df.loc[i,"Source_cit
y"])+" \n\n Destination station : "+str(final_df.loc[i,"Destination_city"])+"
\n\n Route : "+str(final_df.loc[i,"Source"])+" ==> "+str(final_df.loc[i,"Desti
nation"])+"\n\n Price: "+str(final_df.loc[i,"price"])+" $"+" \n\n Arrival Time
: "+final_df.loc[i,"arr_time"]+" \n\n Dep Time : "+final_df.loc[i,"Dep_time"]+
"\n\n Day : "+final_df.loc[i,"Day"]+" Class : "+final_df.loc[i,"Class"]] } ,)
                          rich_txt.append( { "text": ["Route : "+str(final_d
f.loc[i,"Source"])+" ==> "+str(final_df.loc[i,"Destination"])+" Time : "+final
_df.loc[i,"arr_time"]+" -> "+final_df.loc[i,"Dep_time"]+" Price:"+str(final_df
.loc[i,"price"])+" $"+" Day : "+final_df.loc[i,"Day"]+" Class : "+final_df.loc
[i,"Class"]] } ,)
                        #for i in range(len(final_df)):
                    # webhook= "text": { "text": ["Source station : "+final_
df.loc[i,"Source_city"]+"\n \n destination station : "+final_df.loc[i,"Destina
tion_city"]+"\n \n Timing : "+"\n \n Price: "+final_df.loc[i,"price"]] };
                    # return {
                    #     "fulfillmentMessages": [ i for i in webhooks]
                    #       }
                    else :
                        #final_df=final_df[final_df['Class'].str.contains(tr
ain_class,case=False)]
                        final_df.reset_index(inplace=True)
                        webhooks=[]
                        webhooks_text=[]
                        rich_txt=[]
                        for i in range(len(final_df)):
                          webhooks_text.append({ "text": { "text": ["Train N
ame : "+str(final_df.loc[i,"name"])+" \n\n Source station : "+str(final_df.loc
[i,"Source_city"])+" \n\n Destination station : "+str(final_df.loc[i,"Destinat
ion_city"])+"\n\n Route : "+str(final_df.loc[i,"Source"])+" ==> "+str(final_df
.loc[i,"Destination"])+"\n\n Price: "+str(final_df.loc[i,"price"])+" $"+" \n\n
Arrival Time : "+final_df.loc[i,"arr_time"]+" \n\n Dep Time : "+final_df.loc[i
,"Dep_time"]+"\n\n Day : "+final_df.loc[i,"Day"]+" Class : "+final_df.loc[i,"C
lass"]] } },)
                            webhooks.append( { "text": ["Train Name : "+str(fi
nal_df.loc[i,"name"])+" \n\n Source station : "+str(final_df.loc[i,"Source_cit
y"])+" \n\n Destination station : "+str(final_df.loc[i,"Destination_city"])+"
\n\n Route : "+str(final_df.loc[i,"Source"])+" ==> "+str(final_df.loc[i,"Desti
nation"])+"\n\n Price: "+str(final_df.loc[i,"price"])+" $"+" \n\n Arrival Time
: "+final_df.loc[i,"arr_time"]+" \n\n Dep Time : "+final_df.loc[i,"Dep_time"]+
"\n\n Day : "+final_df.loc[i,"Day"]+" Class : "+final_df.loc[i,"Class"]] } ,)
                            rich_txt.append( { "text": ["Route : "+str(final_d
f.loc[i,"Source"])+" ==> "+str(final_df.loc[i,"Destination"])+" Time : "+final
_df.loc[i,"arr_time"]+" -> "+final_df.loc[i,"Dep_time"]+" Price:"+str(final_df
.loc[i,"price"])+" $"+" Day : "+final_df.loc[i,"Day"]+" Class : "+final_df.loc
[i,"Class"]] } ,)
                          #for i in range(len(final_df)):
                return {
                        "fulfillmentMessages": [
                          {
                            "card": {
                              "title": "Here are the trains running ",

                              "buttons": [ i for i in webhooks]
                            }
                          },

                              #[ i for i in webhooks_text]

                               {
                    "text": {
                        "text": [
                            "Please choose from below trains"
                        ],


                    }
                },

                          {
                          "payload": {
                            "richContent": [
                              [
                              #  {
                              #    "type": "description",
                              #    "text": [
                              #      "Please choose from below trains."
                              #    ]
                              # },

                                {
                                  "options": [i for i in rich_txt],

                                  "type": "chips"
                                }
                              ]
                            ]

                          }
                        }
                        ]
                      }
                  # return {
                  #           "richContent" : [
                  #             [
                  #               {
                  #                 "type" : "chips",
                  #                 "options" : [
                  #                   {
                  #                   "text" : "Option 1"
                  #                   },
                  #                   {
                  #                   "text" : "Option 2"
                  #                   }
                  #                 ]
                  #               }
                  #             ]
                  #           ]
                  #         }
          else:


            final_df=final_df_temp.copy()

            if (same_day_flag):
                      # final_df['Timings'] = pd.to_datetime(dc.Timings,format
      = '%H:%M')
                      # if (final_df[dc.Timings.dt.strftime('%H:%M').between
      ('00:00','23:59')].shape[0] > 0) :
                      #           final_df=final_df[dc.Timings.dt.strftime('%
      H:%M').between('00:00','23:59')]
                      #           final_df.reset_index(inplace=True)
                                if (final_df[final_df['Class'].str.contains(tr
      ain_class,case=False)].shape[0] > 0):
                                    final_df=final_df[final_df['Class'].str.co
      ntains(train_class,case=False)]
                                    final_df.reset_index(inplace=True)
                                    webhooks=[]
                                    webhooks_text=[]
                                    rich_txt=[]
                                    for i in range(len(final_df)):
                                      webhooks_text.append({ "text": { "text":
      ["Train Name : "+str(final_df.loc[i,"name"])+" \n\n Source station : "+str(fin
      al_df.loc[i,"Source_city"])+" \n\n Destination station : "+str(final_df.loc[i,
      "Destination_city"])+"\n\n Route : "+str(final_df.loc[i,"Source"])+" ==> "+str
      (final_df.loc[i,"Destination"])+"\n\n Price: "+str(final_df.loc[i,"price"])+"
       $"+" \n\n Arrival Time : "+final_df.loc[i,"arr_time"]+" \n\n Dep Time : "+fin
      al_df.loc[i,"Dep_time"]+"\n\n Day : "+final_df.loc[i,"Day"]+" Class : "+final_
      df.loc[i,"Class"]] } },)
                                      webhooks.append( { "text": ["Train Name
       : "+str(final_df.loc[i,"name"])+" \n\n Source station : "+str(final_df.loc[i,
      "Source_city"])+" \n\n Destination station : "+str(final_df.loc[i,"Destination
      _city"])+"\n\n Route : "+str(final_df.loc[i,"Source"])+" ==> "+str(final_df.lo
      c[i,"Destination"])+"\n\n Price: "+str(final_df.loc[i,"price"])+" $"+" \n\n Ar
      rival Time : "+final_df.loc[i,"arr_time"]+" \n\n Dep Time : "+final_df.loc[i,
      "Dep_time"]+"\n\n Day : "+final_df.loc[i,"Day"]+" Class : "+final_df.loc[i,"Cl
      ass"]] } ,)
                                            rich_txt.append( { "text": ["Route : "+s
      tr(final_df.loc[i,"Source"])+" ==> "+str(final_df.loc[i,"Destination"])+" Time
      : "+final_df.loc[i,"arr_time"]+" -> "+final_df.loc[i,"Dep_time"]+" Price:"+str
      (final_df.loc[i,"price"])+" $"+" Day : "+final_df.loc[i,"Day"]+" Class : "+fin
      al_df.loc[i,"Class"]] } ,)
                                          #for i in range(len(final_df)):
                                      # webhook= "text": { "text": ["Source station
       : "+final_df.loc[i,"Source_city"]+"\n \n destination station : "+final_df.loc
      [i,"Destination_city"]+"\n \n Timing : "+"\n \n Price: "+final_df.loc[i,"pric
      e"]] };
                                      # return {
                                      #     "fulfillmentMessages": [ i for i in webh
      ooks]
                                      #       }
                                      else :
                                          #final_df=final_df[final_df['Class'].str.c
      ontains(train_class,case=False)]
                                          final_df.reset_index(inplace=True)
                                          webhooks=[]
                                          webhooks_text=[]
                                          rich_txt=[]
                                          for i in range(len(final_df)):
                                            webhooks_text.append({ "text": { "text":
      ["Train Name : "+str(final_df.loc[i,"name"])+" \n\n Source station : "+str(fin
      al_df.loc[i,"Source_city"])+" \n\n Destination station : "+str(final_df.loc[i,
      "Destination_city"])+"\n\n Route : "+str(final_df.loc[i,"Source"])+" ==> "+str
      (final_df.loc[i,"Destination"])+"\n\n Price: "+str(final_df.loc[i,"price"])+"
       $"+" \n\n Arrival Time : "+final_df.loc[i,"arr_time"]+" \n\n Dep Time : "+fin
      al_df.loc[i,"Dep_time"]+"\n\n Day : "+final_df.loc[i,"Day"]+" Class : "+final_
      df.loc[i,"Class"]] } },)
                                            webhooks.append( { "text": ["Train Name
       : "+str(final_df.loc[i,"name"])+" \n\n Source station : "+str(final_df.loc[i,
      "Source_city"])+" \n\n Destination station : "+str(final_df.loc[i,"Destination
      _city"])+"\n\n Route : "+str(final_df.loc[i,"Source"])+" ==> "+str(final_df.lo
      c[i,"Destination"])+"\n\n Price: "+str(final_df.loc[i,"price"])+" $"+" \n\n Ar
      rival Time : "+final_df.loc[i,"arr_time"]+" \n\n Dep Time : "+final_df.loc[i,
      "Dep_time"]+"\n\n Day : "+final_df.loc[i,"Day"]+" Class : "+final_df.loc[i,"Cl
      ass"]] } ,)
                                            rich_txt.append( { "text": ["Route : "+s
      tr(final_df.loc[i,"Source"])+" ==> "+str(final_df.loc[i,"Destination"])+" Time
      : "+final_df.loc[i,"arr_time"]+" -> "+final_df.loc[i,"Dep_time"]+" Price:"+str
      (final_df.loc[i,"price"])+" $"+" Day : "+final_df.loc[i,"Day"]+" Class : "+fin
      al_df.loc[i,"Class"]] } ,)

                            # else:
                            #         return {
                            #                     "fulfillmentMessages": [
                            #                         {
                            #                             "text": {
                            #                                 "text": [
                            #                                     "Sorry , our train
      s were already left on selected time !!!.Please book a new ticket on alternati
      ve day"
                            #                                 ]
                            #                             }
                            #                         },
                            #                             {
                            #                               "payload" :{
                            #                                 "richContent": [
                            #                                   [
                            #                                     {
                            #                                       "options": [
                            #                                         {
                            #                                           "text": "Book
       a new ticket"
                            #                                         },
                            #                                         {
                            #                                           "text": "Ask Q
      uery"
                            #                                         },
                            #                                       {
                            #                                         "text": "Nothing
      else"
                            #                                       }
                            #                                     ],
                            #                                     "type": "chips"
                            #                                   }
                            #                                 ]
                            #                               ]
                            #                             }
                            #                         }
                            #                       ]
                            #                 }           #for i in range(len(final_
      df)):
                else :
                          if (final_df[final_df['Class'].str.contains(train_class,ca
      se=False)].shape[0] > 0):
                                final_df=final_df[final_df['Class'].str.contains(tra
      in_class,case=False)]
                                final_df.reset_index(inplace=True)
                                webhooks=[]
                                webhooks_text=[]
                                rich_txt=[]
                                for i in range(len(final_df)):
                                  webhooks_text.append({ "text": { "text": ["Train N
      ame : "+str(final_df.loc[i,"name"])+" \n\n Source station : "+str(final_df.loc
      [i,"Source_city"])+" \n\n Destination station : "+str(final_df.loc[i,"Destinat
      ion_city"])+"\n\n Route : "+str(final_df.loc[i,"Source"])+" ==> "+str(final_df
      .loc[i,"Destination"])+"\n\n Price: "+str(final_df.loc[i,"price"])+" $"+" \n\n
      Arrival Time : "+final_df.loc[i,"arr_time"]+" \n\n Dep Time : "+final_df.loc[i
      ,"Dep_time"]+"\n\n Day : "+final_df.loc[i,"Day"]+" Class : "+final_df.loc[i,"C
      lass"]] } },)
                                  webhooks.append( { "text": ["Train Name : "+str(fi
      nal_df.loc[i,"name"])+" \n\n Source station : "+str(final_df.loc[i,"Source_cit
      y"])+" \n\n Destination station : "+str(final_df.loc[i,"Destination_city"])+"
      \n\n Route : "+str(final_df.loc[i,"Source"])+" ==> "+str(final_df.loc[i,"Desti
      nation"])+"\n\n Price: "+str(final_df.loc[i,"price"])+" $"+" \n\n Arrival Time
      : "+final_df.loc[i,"arr_time"]+" \n\n Dep Time : "+final_df.loc[i,"Dep_time"]+
      "\n\n Day : "+final_df.loc[i,"Day"]+" Class : "+final_df.loc[i,"Class"]] } ,)
                                  rich_txt.append( { "text": ["Route : "+str(final_d
      f.loc[i,"Source"])+" ==> "+str(final_df.loc[i,"Destination"])+" Time : "+final
      _df.loc[i,"arr_time"]+" -> "+final_df.loc[i,"Dep_time"]+" Price:"+str(final_df
      .loc[i,"price"])+" $"+" Day : "+final_df.loc[i,"Day"]+" Class : "+final_df.loc
```

```
[i,"Class"]] } ,)
                          #for i in range(len(final_df)):
                      # webhook= "text": { "text": ["Source station : "+final_
df.loc[i,"Source_city"]+"\n \n destination station : "+final_df.loc[i,"Destina
tion_city"]+"\n \n Timing : "+"\n \n Price: "+final_df.loc[i,"price"]] };
                      # return {
                      #     "fulfillmentMessages": [ i for i in webhooks]
                      #       }
                    else :
                          #final_df=final_df[final_df['Class'].str.contains(tr
ain_class,case=False)]
                          final_df.reset_index(inplace=True)
                          webhooks=[]
                          webhooks_text=[]
                          rich_txt=[]
                          for i in range(len(final_df)):
                            webhooks_text.append({ "text": { "text": ["Train N
ame : "+str(final_df.loc[i,"name"])+" \n\n Source station : "+str(final_df.loc
[i,"Source_city"])+" \n\n Destination station : "+str(final_df.loc[i,"Destinat
ion_city"])+"\n\n Route : "+str(final_df.loc[i,"Source"])+" ==> "+str(final_df
.loc[i,"Destination"])+"\n\n Price: "+str(final_df.loc[i,"price"])+" $"+" \n\n
Arrival Time : "+final_df.loc[i,"arr_time"]+" \n\n Dep Time : "+final_df.loc[i
,"Dep_time"]+"\n\n Day : "+final_df.loc[i,"Day"]+" Class : "+final_df.loc[i,"C
lass"]] } },)
                            webhooks.append( { "text": ["Train Name : "+str(fi
nal_df.loc[i,"name"])+" \n\n Source station : "+str(final_df.loc[i,"Source_cit
y"])+" \n\n Destination station : "+str(final_df.loc[i,"Destination_city"])+"
\n\n Route : "+str(final_df.loc[i,"Source"])+" ==> "+str(final_df.loc[i,"Desti
nation"])+"\n\n Price: "+str(final_df.loc[i,"price"])+" $"+" \n\n Arrival Time
: "+final_df.loc[i,"arr_time"]+" \n\n Dep Time : "+final_df.loc[i,"Dep_time"]+
"\n\n Day : "+final_df.loc[i,"Day"]+" Class : "+final_df.loc[i,"Class"]] } ,)
                            rich_txt.append( { "text": ["Route : "+str(final_d
f.loc[i,"Source"])+" ==> "+str(final_df.loc[i,"Destination"])+" Time : "+final
_df.loc[i,"arr_time"]+" -> "+final_df.loc[i,"Dep_time"]+" Price:"+str(final_df
.loc[i,"price"])+" $"+" Day : "+final_df.loc[i,"Day"]+" Class : "+final_df.loc
[i,"Class"]] } ,)
                          #for i in range(len(final_df)):
                    return {
                        "fulfillmentMessages": [
                          {
                            "card": {
                              "title": "Here are the trains running ",

                              "buttons": [ i for i in webhooks]
                            }
                          },

                              #[ i for i in webhooks_text]


                              {
                    "text": {
                        "text": [
                            "We are not running trains for this city on this d
ate. The next available trains are shown below. Please choose from below train
s if you want to travel on these dates"
                        ],
                    }
                },

                        {
                        "payload": {
                          "richContent": [
                            [
                            #  {
                            #    "type": "description",
                            #    "text": [
                            #      "Please choose from below trains."
                            #    ]
                            # },

                              {
                                "options": [i for i in rich_txt],

                                "type": "chips"
                              }
                            ]
                          ]

                        }
                      }

                      ]
                    }

      else:

        return {

              "fulfillmentMessages": [
                  {
                      "text": {
                          "text": [
                              "We are sorry to inform you that we are not oper
ating for this route. We only provide our service in the following routes, Bal
timore-Virginia Beach, Baltimore-Philadelphia, Boston-Columbus, Boston-Washing
ton DC, Columbus-New York, Columbus-Boston, New York-Washington DC, New York-C
olumbus, Virginia Beach-Baltimore, Washington DC-Boston, Washington DC-New Yor
k,Philadelphia-Baltimore.Please note that these are non-stop trains."
                          ]

                      }
                  },
                  {
                    "payload" :{
                      "richContent": [
                        [
                          {
                            "options": [
                                {
                                  "text": "Book a new ticket"
                                 },
                                {
                                    "text": "Ask Query"
                                 },
                               {
                                 "text": "Nothing else"
                               }
                             ],
                             "type": "chips"
                           }
                         ]
                       ]
                     }
                 }


               ]
           }



    else :
      return {

            "fulfillmentMessages": [
                {
                    "text": {
                        "text": [
                        "We are sorry to inform you that we are not operating
 for this route. We only provide our service in the following routes, Baltimor
e-Virginia Beach, Baltimore-Philadelphia, Boston-Columbus, Boston-Washington D
C, Columbus-New York, Columbus-Boston, New York-Washington DC, New York-Columb
us, Virginia Beach-Baltimore, Washington DC-Boston, Washington DC-New York,Phi
ladelphia-Baltimore.Please note that these are non-stop trains."
                          ]

                    }
                },
                  {
                    "payload" :{
                      "richContent": [
                        [
                          {
                            "options": [
                                {
                                    "text": "Book a new ticket"
                                 },
                                {
                                    "text": "Ask Query"
                                },
                              {
                                 "text": "Nothing else"
                               }
                             ],
                             "type": "chips"
                           }
                         ]
                       ]
                     }
                    }

                ]
            }
        #recipient_ph=request.args.get('phone')
        # ticket_id=random.randint(1000,9000)
        # msg = MIMEMultipart()
        # msg['From'] = sender
        # msg['To'] = recipient_mail
        # msg['Subject'] = 'Booking Details :BNG'+str(ticket_id)
        # msg.attach(MIMEText("Hello "+ name+", \n  Here is your Ticket details : 
   \n Ticket ID : BNG"+str(ticket_id)+"\n Phone num : "+str(phone) , 'plain'))
        # s.sendmail(sender,recipient_mail, msg.as_string())

       # req = request.get_json(force=True)

        # fetch action from json
        #action = req.get('queryResult').get('action')

        # return a fulfillment response
      #  return {'fulfillmentText': 'This is a response from webhook.'}


    except Exception as e:
        #return(str(e))
        return {'fulfillmentText': str(e)}

@app.route('/webhook', methods=[ 'POST'])
@cross_origin()
def webhook():
    req = request.get_json(silent=True, force=True)
    res = processRequest(req)
    res = json.dumps(res, indent=4)
    r = make_response(res)
    r.headers['Content-Type'] = 'application/json'
    return r
    # return response
    #return make_response(jsonify(send_mail()))
def processRequest(req):
    # dbConn = pymongo.MongoClient("mongodb://localhost:27017/")  # opening a 
 connection to Mongo
    #log = Conversations.Log()
    sessionID = req.get('responseId')
    result = req.get("queryResult")
    intent = result.get("intent").get('displayName')
    query_text = result.get("queryText")
    parameters = result.get("parameters")
    cust_name = parameters.get("name")
    cust_contact = parameters.get("phone")
    cust_email = parameters.get("email")
    cust_source_city=parameters.get("from-city")
    cust_des_city=parameters.get("to-city")
    cust_class=parameters.get("classtype")

    global select_src_st
    global select_dest_st
    global select_arr_time
    global select_dept_time
    global select_price
    global day_of_travel
    global cust_date


    print("******")
    #print(cust_date)
    print("******")
   # print(select_dest_st)
   # print(select_arr_time)
    #print(select_dept_time)
    #print(select_price)

    if intent == "trainselection":
        select_src_st=parameters.get("selectsourcestations")
        select_dest_st=parameters.get("selectdeststations")
        select_arr_time=parameters.get("selectarr-time")
        select_dept_time=parameters.get("selectdept-time")
        select_price=parameters.get("price")
        day_of_travel=parameters.get("day")
        # print("@@@@@@@")
        # print(select_src_st)
        # print("@@@@@@@")
        return
    if intent == "sendemailticket":
        fulfillmentText = result.get("fulfillmentText")
        #log.saveConversations(sessionID, "Sure send email", fulfillmentText,
 intent, db)
       # val = log.getcasesForEmail("country", "", db)

        print("===>")
        print(select_src_st)
        print(select_dest_st)
        print(select_arr_time)
        print(select_dept_time)
        print(select_price)
        print(cust_date)
        return send_mail([cust_name, cust_contact, cust_email,select_src_st,se
lect_dest_st,select_arr_time,select_dept_time,select_price,day_of_travel,cust_
date,sessionID])
    if intent == "bookTrainTicket":
        cust_date=parameters.get("date-time")
        fulfillmentText = result.get("fulfillmentText")
        #log.saveConversations(sessionID, "Sure send email", fulfillmentText,
 intent, db)
       # val = log.getcasesForEmail("country", "", db)
        print("===>",cust_class)
        return fetch_details([cust_source_city, cust_des_city, cust_date,cust_
class])
#def prepareEmail(contact_list):

app.run()
```

```
 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deplo
yment.
   Use a production WSGI server instead.
 * Debug mode: off

 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

 * Running on http://811c8a425924.ngrok.io
 * Traffic stats available on http://127.0.0.1:4040

127.0.0.1 - - [11/May/2021 05:32:31] "POST /webhook HTTP/1.1" 200 -

******
******
===> Business
12:00

127.0.0.1 - - [11/May/2021 05:33:32] "POST /webhook HTTP/1.1" 200 -

******
******
******
******
===>
BAL
PDA
2021-05-11T11:00:00-04:00
2021-05-11T09:00:00-04:00
30.0
2021-05-12T12:00:00-04:00
aslambaigus@gmail.com
-----
Aslam
-----
4434688203

127.0.0.1 - - [11/May/2021 05:33:55] "POST /webhook HTTP/1.1" 200 -
```