```
from flask import Flask, request, jsonify, render_template
import os
import requests
import dialogflow
import requests
import json
app = Flask(__name__)
@app.route('/')
def index():
    return '<h1>sports APP</h1>'
@app.route('/webhook', methods=['POST'])
def webhook():
    data = request.get_json(silent=True)
    #k=data['queryResult']['parameters']["tournament"]
    intent=data['queryResult']['intent']['displayName']
    #sen=data['queryResult']['queryText']
    #if('meme' in data["intent"]["dislayName"]):
    if('meme' in intent):
        link=meme()
        resp= {"fulfillmentMessages":[{
        "image": {
          "imageUri": link
        },
        "platform": "FACEBOOK"
      },]
              }
    elif('youtube' in intent):

        query=data['queryResult']['parameters']["teamName"]+' on
'+data['queryResult']['parameters']["matchDate"]
        k=youtube_search(query)
        resp = {

            "fulfillmentText":k
            }
        ''' if(k!='nope'):
            resp = {"fulfillmentMessages": [
    {
      "payload": {
        "facebook": {
          "attachment": {
            "payload": {
              "elements": [
```

```
            {
              "image_url": k[2],
              "title": k[0],
              "default_action": {
                "url": "https://www.youtube.com/watch?v="+k[1],
                "type": "web_url"
              },
              "buttons": [
                {
                  "type": "web_url",
                  "url": "https://www.youtube.com/watch?v="+k[1],
                  "title": "view"
                }
              ],

            }
          ],
          "template_type": "generic"
        },
        "type": "template"
      }
    }
  },
  "platform": "FACEBOOK",
  "lang": "en"
},
{
  "text": {
    "text": [
      ""
    ]
  },
  "lang": "en"
}
],}
    else:'"


elif('score' in intent):
    team1=data['queryResult']['parameters']["hometeam"]
    team2=data['queryResult']['parameters']["awayteam"]
    dateT=data['queryResult']['parameters']["Matchdate"]
    k=score(team1,team2,dateT,dateT)
```

```
    resp = {"fulfillmentMessages": [
  {
    "payload": {
     "facebook": {
       "attachment": {
         "payload": {
          "elements": [
           {
             "image_url": k[6],
             "title": k[2]+' vs '+k[3],
             "default_action": {
               "url": "https://www.cricbuzz.com/",
               "type": "web_url"
             },
             "buttons": [
               {
                 "type": "web_url",
                 "url": "https://www.cricbuzz.com/",
                 "title": "view"
               }
             ],
             "subtitle": k[4]+" half time\n"+k[5] +'full time'
           }
          ],
          "template_type": "generic"
         },
         "type": "template"
       }
     }
    },
    "platform": "FACEBOOK",
    "lang": "en"
  },
  {
    "text": {
     "text": [
       ""
     ]
    },
    "lang": "en"
  }
],}
```

```
elif('highlights' in intent):
    match_date=data['queryResult']['parameters']["matchdate"]
    teamName=data['queryResult']['parameters']["teamname"]
elif('trend' in intent):
    k=''
    try:
        k=data['queryResult']['parameters']["search"]
    except:
        pass
    if(len(k)!=0):
        resp = {

        "fulfillmentText": trending(k),
        }

    else:
        resp = {

        "fulfillmentText": trendin(),
        }
elif('gossip' in intent):
    k=gossip()
    resp={

"fulfillmentMessages": [
  {
    "payload": {
      "facebook": {
        "attachment": {
          "type": "template",
          "payload": {
            "elements": [
              {
                "image_url": k[1],

                "title": "have ou seen this?",
        "subtitle":k[0],

                "default_action": {
                  "type": "web_url",
                  "url": k[2]
                },
                "buttons": [
```

```
                {
                  "type": "web_url",
                  "url": k[2],
                  "title": "view"
                },

              ],

            }
          ],
          "template_type": "generic"
        }
      }
    }
  },
  "platform": "FACEBOOK"
 },
 {
  "text": {
   "text": [
    ""
   ]
  }
 }
]}
elif('things' in intent):
    resp={
        "fulfillmentText":'''you can ask me questions like
        1.get score
        2.get gossip
        3.get meme
        4.whats trending
        5.what is trending about a particular topic
        6. jokes
        '''
        }
elif('joke' in intent):
    k=jokes()
    resp = {
        "fulfillmentText": k,
    }

elif('reddit' in intent):
```

```
        resp = {
            "fulfillmentText": reddit_topics(),
        }
    elif('emotion' in intent):
        k=data['queryResult']['parameters']["topic"]

        resp = {
            "fulfillmentText": reddit_sentiment(k),
        }

    else:

        resp = {
            "fulfillmentText": "sorry ! could not process the request",
        }



    return jsonify(resp)


def jokes():
    k=requests.get("http://api.icndb.com/jokes/random")
    return k.json()["value"]["joke"]

def trendin():
    import pandas as pd
    from pytrends.request import TrendReq
    pytrend = TrendReq()
    df =pytrend.today_searches()
    return 'People are searching about'+'\n'.join((list((df).to_frame()["query"])))
'''def highlight(team1,team2,matchDate):
    team1=get_eventId(team1,team2,matchDate)
   '''
def trending(k):

    import pandas as pd
    from pytrends.request import TrendReq
    pytrend = TrendReq()
    pytrend.build_payload(kw_list=[k])
    related_queries = pytrend.related_queries()
```

```
    return '\n'.join((list((((related_queries)[k])["top"])["query"].to_frame()["query"])))



def meme(query=""):
    import random
    from firebase import firebase
    firebase = firebase.FirebaseApplication('XXXXXXXXX', None)
    res=firebase.get('XXXXX', '')
    result = res[random.choice(list(res.keys()))]
    return result['url']
def gossip():
    import random
    from firebase import firebase
    firebase = firebase.FirebaseApplication('XXXXXXXXXXXX', None)
    res=firebase.get('XXXXXXXX, '')
    result = res[random.choice(list(res.keys()))]
    #result2=res[random.choice(list(res.keys()))]
    k=(result['para'],result['photo'],result['url'])
    return k



def score(awayteam,hometeam,dateT,dateT1):

resp=requests.get("https://allsportsapi.com/api/football/?met=Fixtures&APIkey=xxx&from="+dat
eT+"&to="+dateT1+"&countryId=62")


    for x in resp.json()["result"]:
        if((awayteam.lower()==x["event_away_team"].lower() and
hometeam.lower()==x["event_home_team"].lower()) or
(awayteam.lower()==x["event_home_team"].lower() or hometeam.lower()==
x["event_away_team"].lower())):
            k=x["event_final_result"].encode('ascii', 'replace').decode('utf-8').replace('?','')
            index=k.index('-')
            team1=int(k[:index])
            team2=int(k[index+1:])
            if(team1>team2):
                return
(x["event_key"],x["event_date"],x["event_home_team"],x["event_away_team"],x["event_halftime
```

```
_result"].encode('ascii',
'replace').decode('utf-8').replace('?',''),x["event_final_result"].encode('ascii',
'replace').decode('utf-8').replace('?',''),x["home_team_logo"])
        elif(team2>team1):
            return
(x["event_key"],x["event_date"],x["event_home_team"],x["event_away_team"],x["event_halftime
_result"].encode('ascii',
'replace').decode('utf-8').replace('?',''),x["event_final_result"].encode('ascii',
'replace').decode('utf-8').replace('?',''),x["away_team_logo"])
        else:

            return
(x["event_key"],x["event_date"],x["event_home_team"],x["event_away_team"],x["event_halftime
_result"].encode('ascii',
'replace').decode('utf-8').replace('?',''),x["event_final_result"].encode('ascii',
'replace').decode('utf-8').replace('?',''),x["league_logo"])



#print([(x["event_key"],x["event_date"],x["event_home_team"],x["event_away_team"],x["event_h
alftime_result"].encode('ascii',
'replace').decode('utf-8').replace('?',''),x["event_final_result"].encode('ascii',
'replace').decode('utf-8').replace('?','')) for x in resp.json()["result"]])

def senti(k,sen):
    import urllib.request as r

sol=json.loads(requests.get("https://apiv2.apifootball.com/?action=get_leagues&APIkey=xxxx").t
ext)
    sol=[x for x in sol if(x["league_name"].lower()==k.lower())]
    sentiment=sen
    dic=dict()
    dic["text"]=sentiment
    sentiment=json.loads(requests.post("http://text-processing.com/api/sentiment/",dic).text)
    #(' '.join(sol[0].values())+
    return sentiment['label']
def detect_intent_texts(project_id, session_id, text, language_code):
    session_client = dialogflow.SessionsClient()
    session = session_client.session_path(project_id, session_id)

    if text:
        text_input = dialogflow.types.TextInput(
            text=text, language_code=language_code)
```

```
        query_input = dialogflow.types.QueryInput(text=text_input)
        response = session_client.detect_intent(
            session=session, query_input=query_input)
        return response.query_result.fulfillment_text
@app.route('/send_message', methods=['POST'])
def send_message():
    message = request.form['message']
    project_id = os.getenv('DIALOGFLOW_PROJECT_ID')
    fulfillment_text = detect_intent_texts(project_id, "unique", message, 'en')
    response_text = { "message":  fulfillment_text }
    return jsonify(response_text)
# run Flask app
def youtubeVideo(query):
    try:

        return youtube_search(query)
    except HttpError:
        return "nope"

def youtube_search(query):
    import random
    from googleapiclient.discovery import build
    from googleapiclient import discovery
    DEVELOPER_KEY = 'xxxxxx'
    YOUTUBE_API_SERVICE_NAME = 'youtube'
    YOUTUBE_API_VERSION = 'v3'
    youtube = build(YOUTUBE_API_SERVICE_NAME, YOUTUBE_API_VERSION,
    developerKey=DEVELOPER_KEY)


  # Call the search.list method to retrieve results matching the specified
  # query term.
    search_response = youtube.search().list(
    q=query,
    part='id,snippet',
    maxResults=20
    ).execute()

    videos = []


  # Add each result to the appropriate list, and then display the lists of
  # matching videos, channels, and playlists.
```

```
    for search_result in search_response.get('items', []):
        if search_result['id']['kind'] == 'youtube#video':
            videos.append((search_result['snippet']['title'],

search_result['id']['videoId'],search_result["snippet"]["thumbnails"]["medium"]["url"]))
    return len(search_response["items"])
def reddit_topics():
    import praw
    import requests,time
    import pandas as pd
    reddit = praw.Reddit(client_id='xxxxxxxxxx', \
                         client_secret='xxxxxxxxxxxxx', \
                         user_agent='xxxxxxxxxxxx', \
                         username='xxxxxxxxxxxxxx', \
                         password='xxxxxxxxxxxxxxxx')
    subreddit = reddit.subreddit('IndianFootball')
    top_subreddit = subreddit.top(limit=100)
    print(top_subreddit)
    c=[]
    for submission in subreddit.top(limit=10):
        c.append(submission.title+' '+ submission.id)
    return '\n'.join([x for x in c])
def reddit_sentiment(query):
    c=get_comments(query)
    if(c=='nope'):
        return "not enough comments to draw conclusion sorry"
    p=0
    n=0
    for i in c:
        if(i=='pos'):
            p+=1
        elif(i=='neg'):
            n+=1
        else:
            pass
    if(p>n):
        return 'people are happy about it'
    elif(n>p):
        return 'people are not looking so happy about it'
    else:
        return 'not enough comments to draw conclusion sorry'

def get_comments(IDL):
```

```
    c=[]
    import praw
    reddit = praw.Reddit(client_id='XXXXX', \
                         client_secret='XXXXXXXX', \
                         user_agent='XXXXXXX', \
                         username='XXXXXXXXX', \
                         password='XXXXXXXXXX')
    submission = reddit.submission(IDL)
    from praw.models import MoreComments
    submission.comments.replace_more(limit=None)
    comment_queue = submission.comments[:]
    # Seed with top-level
    if(len(comment_queue)==0):
        return "nope"
    while comment_queue:
        comment = comment_queue.pop(0)
        c.append(sentiment(comment.body)["label"])
        #comment_queue.extend(comment.replies)
    return c
def sentiment(sentence):
    import json
    dic=dict()
    dic["text"]=sentence
    senti=json.loads(requests.post("http://text-processing.com/api/sentiment/",dic).text)
    return senti



if __name__ == "__main__":



    app.config['INTEGRATIONS'] = ['ACTIONS_ON_GOOGLE']

    app.run(debug=True)
```