\_\_AUTHORS\_\_=**"Devna Chaturvedi"** and **"GIREESH KUMAR Muppalla"** and **'Sai Harshavardhan Mylavarapu'** and **'Yusuf'**\  
 **"UMKC ChatBot"** \  
 **"------------------ BUSINESS LOGIC FILE-----------------------"  
  
"""  
This python file will process the user message received from skype  
and dispatch corresponding message for the next action.  
"""**

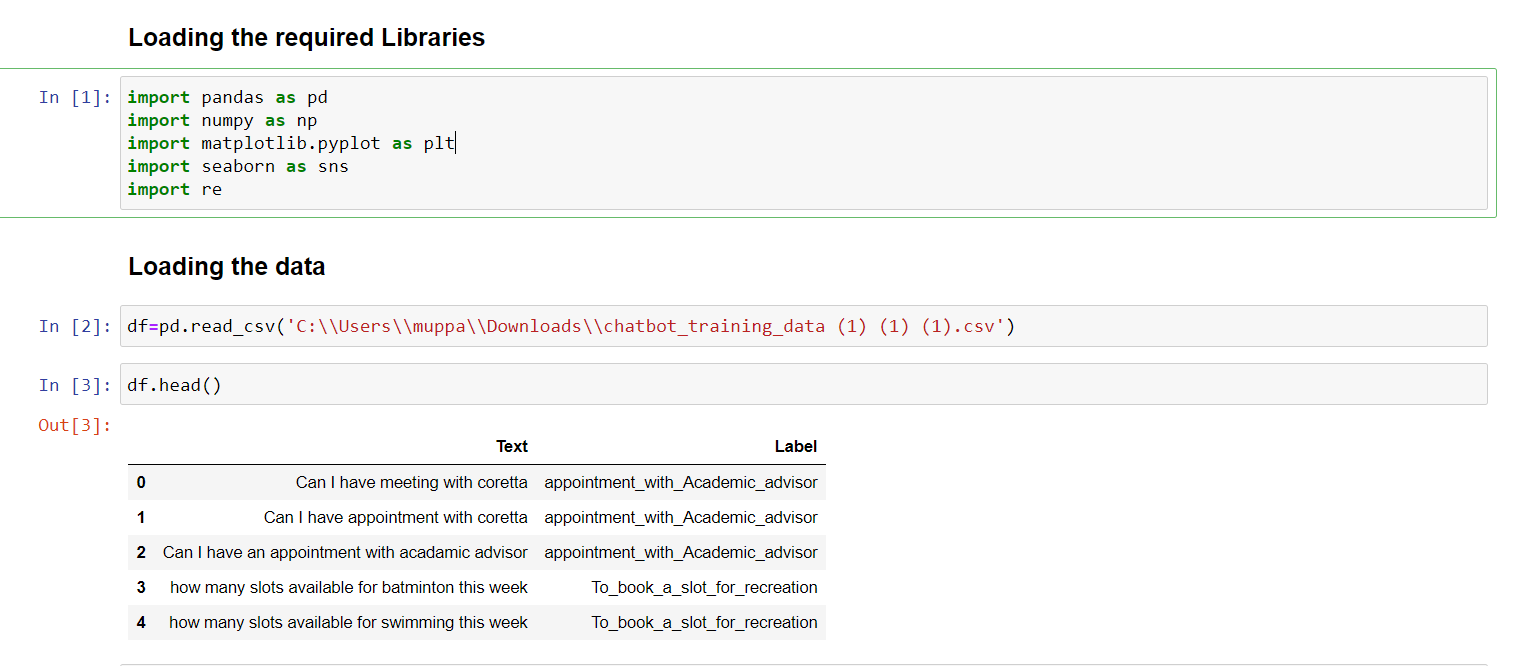
**Python filename: message\_processor.py**

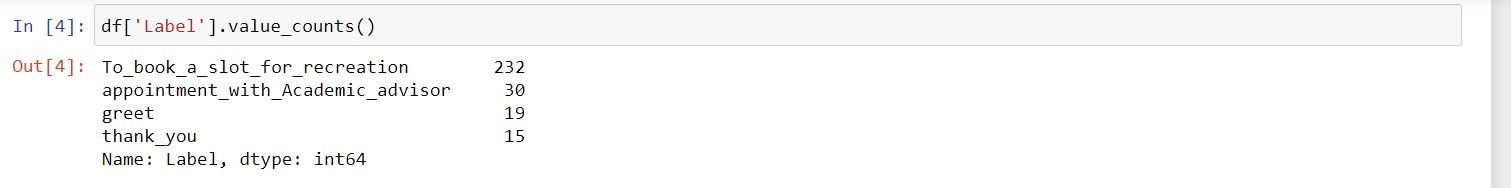
import re  
global ver  
import random  
from datetime import date  
  
today = date.today()  
import sys  
import re  
from datetime import datetime  
import config\_file\_processor  
from session import ConversationSessionManager  
import pandas as pd  
from rasa\_nlu.model import Interpreter  
import pickle  
from nltk.corpus import stopwords  
stop = list(stopwords.words(**'english'**))  
  
bot=config\_file\_processor.chatbot()  
event\_data = Interpreter.load(**"./models/default/event\_data"**)  
intent\_classifier = Interpreter.load(**"./models/default/Tamak's\_intents\_data"**)  
intent\_classifier1 = pickle.load(**'C:**\\**Users**\\**muppa**\\**Downloads**\\**UMKC\_Bot\_V1**\\**models**\\**chat\_bot\_final.sav'**)  
vectorizer = pickle.load(**'C:**\\**Users**\\**muppa**\\**Downloads**\\**UMKC\_Bot\_V1**\\**models**\\**vectorizer.sav'**)  
  
*#-------------------SESSION CREATION-----------------------*conversation\_sessions = ConversationSessionManager()  
  
*#------------------------------------------FILES PATH----------------------------------------*log\_path = **'C:**\\**Users**\\**muppa**\\**Downloads**\\**UMKC\_Bot\_V1'***### Method for writing the chat logs of user to call for every user message and append to the file created*def logs\_user(filename, username, text):  
 with open(filename, **"a"**, encoding=**'utf-8'**) as myfile: *#,encoding='utf-8'* myfile.write(  
 **"["** + str(datetime.now().strftime(**'%Y-%m-%d %H:%M:%S'**)) + **"]"** + username + **"**\n**"** + str(text) + **"**\n\n**"**)  
  
*## Method for writing the chat logs of Bot*def logs\_bot(filename, t):  
 with open(filename, **"a"**) as myfile:  
 myfile.write(**"["** + str(datetime.now().strftime(**'%Y-%m-%d %H:%M:%S'**)) + **"]"** + **"BOT"** + **"**\n**"** + str(t) + **"**\n\n**"**)  
  
  
  
*### Method for dell business logic of chat flow which will be called in dell\_bot\_api file for every message from user from skype*def inp(text, sess\_id):  
 *# try:* sessionId = sess\_id  
 t=None  
 m=None  
 current\_session = conversation\_sessions.get(sessionId)  
 if current\_session is None:  
 print(**"its new session"**)  
 current\_session = conversation\_sessions.new(sessionId)  
 *# current\_session.filesession = str(username) + "\_" + str(current\_session.id)* current\_session.date\_time = datetime.now().strftime(**'%Y-%m-%d\_%H\_%M\_%S'**)  
 *# current\_session.emailid = email* filename = log\_path + **"UMKC\_Roo\_Bot\_Logs"** + **"\_"** + str(current\_session.date\_time) + **".txt"** *#To create the file name with user email id and current time with .txt extension* print(**"previous response"**,current\_session.res\_sess)  
 *#print(bot.get\_response(text.lower()).confidence, "bot confidence for the current statement")  
  
 ################## text cleaning ##########################################* text1 = **" "**.join(text.lower() for x in text.split() if x not in stop)  
 text1=text1.str.replace(**'[^\w\s]'**, **''**)  
 text1=re.sub(**r"\d+"**, **""**, text1).strip()  
   
 *#### feature extraction #################* vects=vectorizer.transform(text)  
   
 *## Intent Extraction ##################* intent2=intent\_classifier1.predict(vects)  
 logs\_user(filename, **"User"**, text)  
 print(text,**"above"**)  
 text=**' '**.join(text.split(**'**\n**'**))  
 print(text,**"bellow"**)  
 print(intent2)  
  
 text=text.replace(**" tommorrow "**,**" "**+str()+**"0"**+str(today.month)+**'/'**+str(today.day+1)+**' '**)  
 text = text.replace(**" today "**, **" "** + str() + **"0"** + str(today.month) + **'/'** + str(today.day)+**' '**)  
 text = text.lower().replace(**"'"**, **''**).replace(**"?"**, **' '**).replace(**' '**,**' '**).replace(**"="**,**' '**)  
 text = re.sub(**" +"**,**" "**,text)  
 print(text, **"after cleaning"**)  
 intent=intent\_classifier.parse(text)[**'intent'**]  
 result = event\_data.parse(text.lower())  
 print(result,**"filters"**)  
 print(intent,**"intent"**)  
 for i in result[**'entities'**]:  
 print(**'entities'**,i)  
 if i[**'entity'**]==**'academic\_advisor'**:  
 current\_session.advisor\_name=i[**'value'**]  
 if i[**'entity'**]==**'recreation'** and i[**'confidence'**]>0.85:  
 current\_session.gamename = i[**'value'**]  
 if i[**'entity'**] == **'date'**:  
 current\_session.date = i[**'value'**]  
  
  
 intent1=intent[**'name'**]  
 if (intent1==**'greet'** and intent[**'confidence'**]>0.96) or text==**'triggerit'** :  
 text1=**"Hi, I can help you to book a slot for recreational activities and to schedule a meeting with academic advisor?"** elif current\_session.pre\_input==**"When do you want to book the slot, Please mention the data and time (mm/dd HH:MM) in CST ?"** and current\_session.intent==**'appointment\_with\_Academic\_advisor'**:  
 current\_session.datetime=text  
 text1=**"Your slot is booked with Coretta on "**+current\_session.datetime+ **" for 15 minutes. Thanks"** elif intent1==**'appointment\_with\_Academic\_advisor'** and intent[**'confidence'**]>0.75:  
 text1=**"When do you want to book the slot, Please mention the data and time (mm/dd HH:MM) in CST ?"** current\_session.pre\_input=text1  
 current\_session.intent = **'appointment\_with\_Academic\_advisor'** elif current\_session.pre\_input==**"When do you want to book the slot, Please mention the data and time (mm/dd HH:MM) in CST ?"** and current\_session.date\_time is None:  
 current\_session.datetime=text  
 text1=**"Your slot is booked with Coretta on "**+current\_session.datetime+ **"for 15 minutes. Thanks"** current\_session=None  
  
 elif intent1==**'To\_book\_a\_slot\_for\_recreation'** and intent[**'confidence'**]>0.70 and current\_session.gamename is None:  
 text1=**"Please mention the game that you want to play"** print(**"in game is none"**)  
 current\_session.pre\_input=text1  
 current\_session.intent = **'To\_book\_a\_slot\_for\_recreation'** elif intent1==**'To\_book\_a\_slot\_for\_recreation'** and intent[**'confidence'**]>0.70 and current\_session.gamename is not None and current\_session.date is None:  
 text1=**"When do you want to play the "** +current\_session.gamename+**", Please mention the date in (MM/DD) format"** current\_session.pre\_input=text1  
 current\_session.intent = **'To\_book\_a\_slot\_for\_recreation'** elif current\_session.intent==**'To\_book\_a\_slot\_for\_recreation'** and current\_session.pre\_input==**"Please mention the game that you want to play"** and current\_session.gamename is None and current\_session.date is None:  
 current\_session.gamename = text  
 text1=**"When do you want to play the "** +current\_session.gamename+**", Please mention the date in (MM/DD) format"** current\_session.pre\_input=text1  
  
  
 elif current\_session.intent==**'To\_book\_a\_slot\_for\_recreation'** and current\_session.pre\_input==**"When do you want to play the "** +current\_session.gamename+**", Please mention the date in (MM/DD) format"** and current\_session.gamename is not None and current\_session.date is None:  
 current\_session.date = text  
 text1=**"Your slot for "** + current\_session.gamename + **" on "** + current\_session.date + **" at "**+str(random.randint(1,8)) +**" PM is booked. Thanks"** current\_session.pre\_input=text1  
 current\_session=None  
  
 import pyodbc  
 conn = pyodbc.connect(**'Driver={SQL Server};'  
 'Server=localhost;'  
 'Database=UMKC\_Bot;'  
 'Trusted\_Connection=yes;'**)  
  
 cursor = conn.cursor()  
 cursor.execute(**'INSERT INTO bot\_logs VALUES('**+current\_session.gamename+**','**+current\_session.date+**','**+current\_session.date\_time+**"'"**)  
  
 elif intent1==**'thankyou'**:  
 text1=random.choice([**"You are Welcome"**,**"No Problem"**, **"Np"**, **"That's Alright"**, **"No Worries"**])  
  
 else:  
 text1=**"Didn't get you, Could you please re-phrase it"** *# print(intent['confidence'], "confidence")* print(text,**"given text"**)  
 *# print(current\_session.pre\_input, "previous\_text")* print(intent1)  
 *# print("game, date",current\_session.gamename, current\_session.date)* return text1

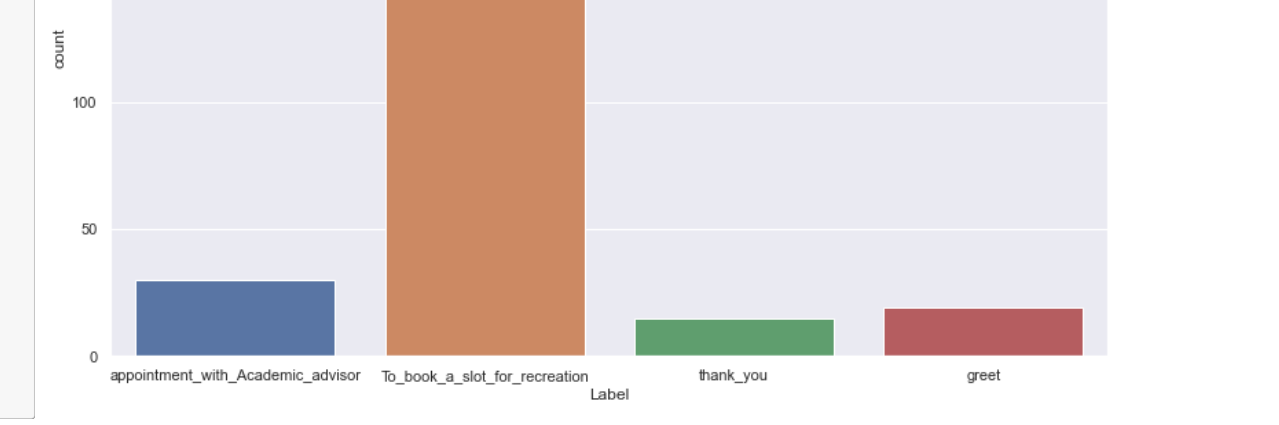
**Python\_File\_2: bot\_api.py**

import sys  
from flask import Flask, jsonify  
import json, requests  
from flask\_cors import CORS, cross\_origin  
from flask import json  
from flask.globals import request  
*# import os  
# import email\_notification as en*from datetime import datetime  
import message\_processor  
import re  
import traceback  
  
*# --------------------------------------------------------------------------------------------*app = Flask(\_\_name\_\_)  
CORS(app)  
  
*#Method to stop the Flask\_API before changes are done in config file, calling this in start\_bot.BAT file*def shutdown\_server():  
 func = request.environ.get(**'werkzeug.server.shutdown'**)  
 if func is None:  
 raise RuntimeError(**'Not running with the Werkzeug Server'**)  
 func()  
  
*# # ----------------------------FAQ Classifier Called--------------------------------------------------------------------------*@app.route(**'/tamak-bot'**, methods=[**'POST'**])  
@cross\_origin()  
def GetFAQ():  
 if request.method != **'POST'**:  
 return json.dumps({**"Status"**: **"ERROR"**, **"DATA"**: None, **"Reason"**: **"Only accept POST request"**})  
 if not request.headers[**'Content-Type'**] == **'application/json'**:  
 return json.dumps({**"Status"**: **"ERROR"**, **"DATA"**: None, **"Reason"**: **"Only accept Content-Type:application/json"**})  
 if not request.is\_json:  
 return json.dumps({**"Status"**: **"ERROR"**, **"DATA"**: None,  
 **"Reason"**: **'Expecting json data in the form {"data":"VALUE"}'**})  
 data = request.json  
 print(data)  
 if **'message'** not in data:  
 return json.dumps({**"Status"**: **"ERROR"**, **"DATA"**: None, **"Reason"**: **'Expecting key as data'**})  
 try:  
 statement = data[**'message'**]  
 print(statement)  
 statement=statement.replace(**','**,**" "**)  
 statement = statement.replace(**'.'**, **" "**)  
 print(**"data receiving from gui bot"**,data)  
 except Exception as e:  
 print(**"there is some issue"**)  
 return json.dumps({**"DATA"**: None,  
 })  
 try:  
  
 data=message\_processor.inp(re.sub(**" +"**,**" "**,statement.strip()), data[**'sender'**])*#, data['sess\_id']#,data["email\_id"],data["username"],data["Session\_start\_time"],data['Hit\_count'])#data['Hit\_count']* print(**"------------------------------------------------------------------------------"**)  
 except Exception as e:  
 *#print(os.getcwd())* k = traceback.format\_tb(e.\_\_traceback\_\_)  
 with open(**"Input\_exceptions.txt"**, **"a"**, encoding=**'utf-8'**) as myfile: *# ,encoding='utf-8'* myfile.write(  
 **"["** + str(datetime.now().strftime(**'%Y-%m-%d %H:%M:%S'**)) + **"]"** +**'('**+str(data[**"sender"**])+**')'** + **"**\n**"** +**"User\_Input: "** + str(statement) + **'**\n**'**+**'Error\_Details: '** + str(e) +**'**\n**'**+**"Traceback\_details: "**+str(k)+**"**\n\n**"**)  
 *# print(en.send\_email(e,statement,data["username"],data["email\_id"]))* print(e)  
 return json.dumps({**'Status": "ERROR", "DATA": None, "Reason": "Internal server error'**})  
 *# print(session\_id,"sessionid")  
 # print(type(session\_id), "sessionid")  
 # print(data,"data")* return json.dumps([{**"Status"**: **"SUCCESS"**, **"text"**: data}])*#, "recipient\_id":session\_id }])*@app.route(**'/shutdown'**, methods=[**'GET'**])  
def shutdown():  
 shutdown\_server()  
 return **'Restarting the UMKC ROO BOT...'***# ---------------------------------------------------------------------------------------------------*def startAPIs():  
 try:  
 app.run(**"192.168.1.240"**, port=(5004), debug=False, threaded=False)  
 app.run()  
 except Exception as e:  
 raise (**"APIs not started Exception (startAPIs ) at : "** + str(**"192.168.1.240"**) + **":"** + str(5004) + **" due to :"** + str(  
 e))  
if \_\_name\_\_ == **'\_\_main\_\_'**:  
 startAPIs()

**## KDM\_Project-1.ipynb**

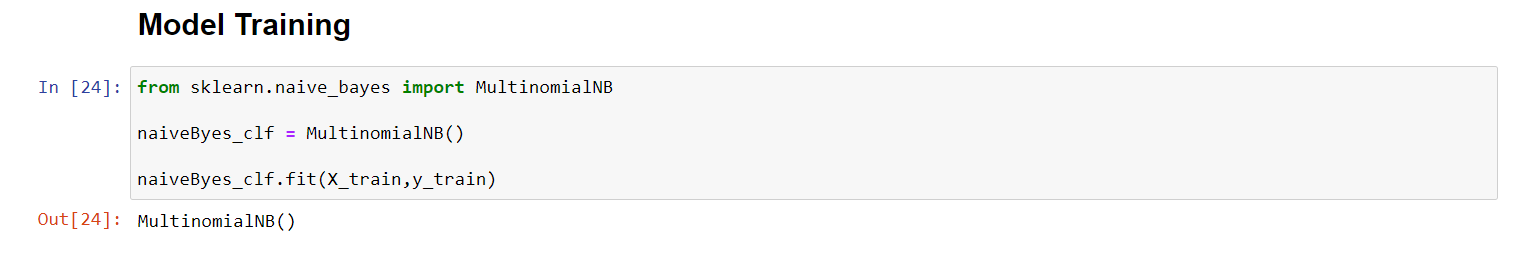


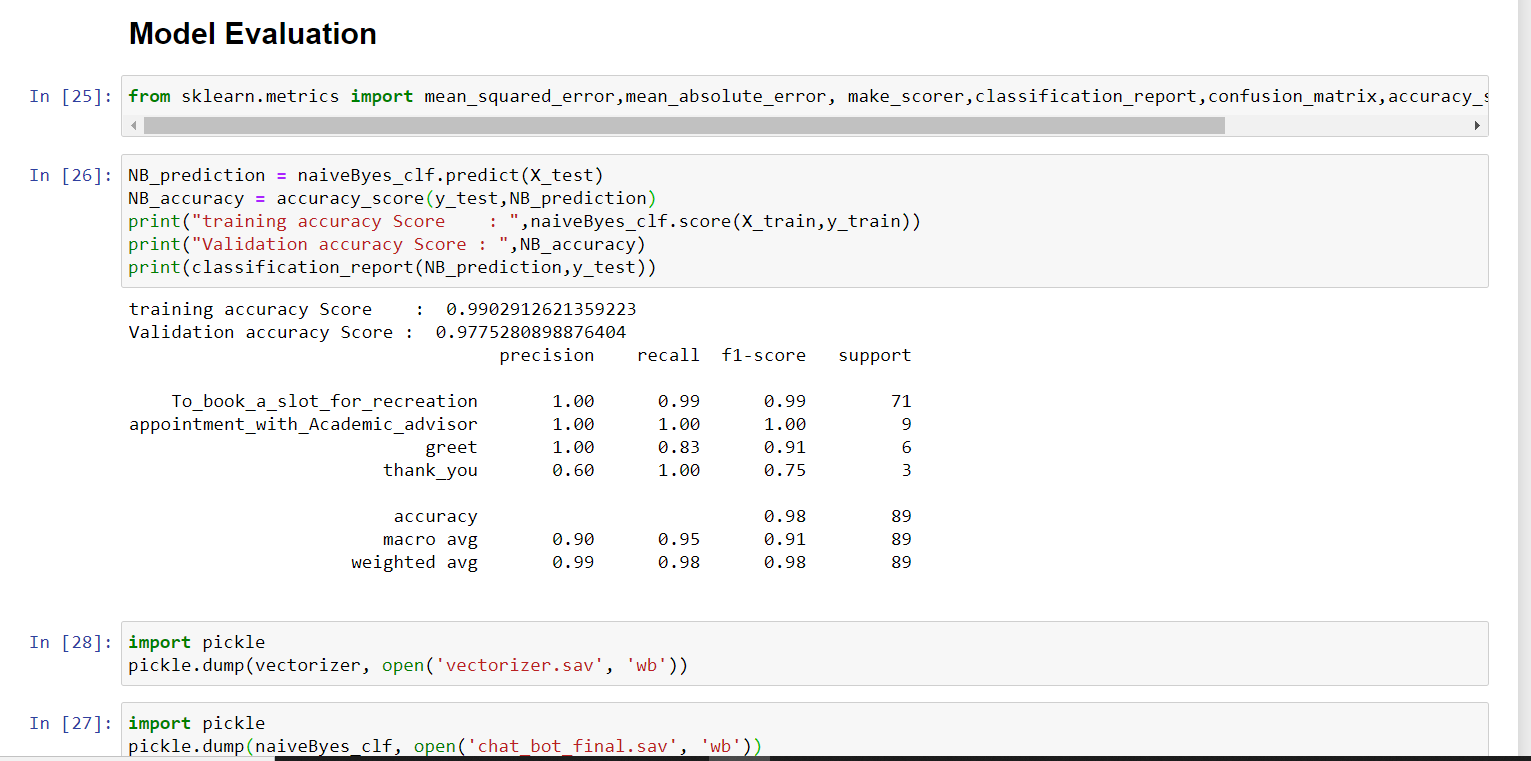












**## Chat.html**

<html>  
<head>  
 *<!-- https://bootsnipp.com/tags/chat?page=2 -->* <title>Chat box</title>  
 <link href="/static/css/bootstrap.min.css" rel="stylesheet" id="bootstrap-css">  
 <link rel='stylesheet prefetch' href='https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css'>  
 <link rel='stylesheet prefetch' href='https://fonts.googleapis.com/icon?family=Material+Icons'>  
 <link rel="stylesheet" href="/static/css/chat.css">  
 <link href="/static/css/jquery.tagit.css" rel="stylesheet" type="text/css">  
 <link href="/static/css/tagit.ui-zendesk.css" rel="stylesheet" type="text/css">  
  
</head>  
  
<body>  
<div id="body">  
  
<div id="chat-circle" class="btn btn-raised hvr-float" data-toggle="tooltip" title="Hey, How can I help you ?">  
 <div id="chat-overlay"></div>  
 <i class="material-icons">forum</i>  
 </div>  
  
 <div class="chat-box">  
  
 <div class="chat\_window">  
 <div class="top\_menu">  
 <div class="title">Welcome to UMKC Roo  
 <span class="chat-box-toggle"><i class="material-icons">cancel</i></span>  
 </div>  
 </div>  
 <div class="chat-bg">  
 <ul class="messages"></ul>  
 <div class="notification\_wrapper">  
 <p class="error"></p>  
 </div>  
 <div class="bottom\_wrapper clearfix">  
 <div class="message\_input\_wrapper">  
 <textarea class="message\_input" id="message\_input" placeholder="Type your message here..." /></textarea>  
 </div>  
 </div>  
 </div>  
 </div>  
 <div class="message\_template">  
 <li class="message" style="opacity: 0;">  
 <div class="avatar"></div>  
 *<!-- <p class="hr\_typing">Nainy is typing <span></span><span></span><span></span><span></span></p> -->  
 <!--<img class="bot\_typing" src="images/icon\_typing.gif" style="display:block;" />-->* <div class="text\_wrapper">  
 <div class="text"></div>  
 </div>  
 </li>  
 </div>  
 </div>  
  
</div>  
 <script src="/static/js/jquery-1.11.1.min.js"></script>  
 <script src="/static/js/jquery-ui.min.js"></script>  
 <script src="/static/js/bootstrap.min.js"></script>  
 <script src="/static/js/tag-it.js"></script>  
 <script src="/static/js/config.js"></script>  
 <script src="/static/js/chat.js"></script>  
</body>  
</html>