Appendix

First Meeting (Transcript of Interview)

Jerry (Haoran) - Hello my name is Jerry would you like to introduce yourself.
Ms. D - Hey my name is Dania.
Jerry - So since we're here, what is the general purpose of the program that you would like me to design?
Ms. D - So I would love to have a program that would give me the general feeling of the search and the general feelings about certain cryptos according to the tweets that are made.
Jerry - So how would you like to filter your tweets?
Ms. D - Yeah I would like to use the “cancella” sign, the hashtag.
Jerry - Okay yeah that would be perfect and how would you like your data to be visually displayed via an app or a website.
Ms. D - I'd rather have it by an app. Okay that's so that I can have it easily accessible.
Jerry - Yeah so from my understanding how many types of cryptocurrencies would you like to be collected.
Ms. D - Definitely definitely at least the top twenties if that's possible.
Jerry - That would indeed be possible, so you said you want to have the sentiment analysis of cryptos, what other data do you want to look at?
Ms. D - So this sentiment is really about the feelings people have towards the crypto right, which would help me I understand, but if there would be also something used about eventual changes that are like you know for example when there's a crypto coming and there's a new update happening or reprogramming like you know for example what had happened last time with Cardono. So, maybe this can help to pay more attention than to the tweets. Do you understand what I'm saying so when there's like maybe a little when there is talk about the possible upgrade or update of a certain crypto yeah? To understand so that will raise my alerts as to looking then about the sentiments of the crypto if that’s possible.
Jerry - So yeah so from my understanding this could be done by selecting the hour of tweets like the numbers of tweets discussed about like for example Cardano in an hour which this I think shows maybe there's some major event going on or there's a very high sentiment like of highly of positive is a highly negative feeling about of like a cryptocurrency, so is that okay? like showing the frequency of like tweets.
Ms. D - Yeah sure.
Jerry - So it basically just like shows like how much people talk about it in an hour which I think it indicates for what you want.
Ms. D - Yeah I think it's it's great because obviously if you will if there's a lot of talk about it. That's definitely an indication.
Jerry - Yeah okay that's nice. So how after collecting the sentiment analysis data how would you like it to be visually displayed through like a pie chart or a diagram of yours like a specific type.
Ms. D - I would imagine it's more like a bar graph yeah and where there are different percentages and the subjects if we're talking about so like bad feeling or maybe like you know like positive is green and negative feelings like bearish bullish. Maybe like yeah the neutral and then so I can see. Yeah, in a pie I don't know, no I'd rather have it in a bar graph.

Jerry - That would be good and are there any more other functionality that you wish to be added to the program.
Ms. D - Let me think about it. so the sentiments for sure, maybe if you could give them the advice as to where to get, can you add this, like the best places to get those.
Jerry - Like the best places to like exchange crypto or the best.
Ms. D - To buy, to exchange like maybe some advice maybe collect also. I don't know if it's possible but some news about it. You know like.
Jerry - I think that could be doable and so you would like some suggestions for like where to buy crypto, information on crypto, those and.
Ms. D - On that crypto, let's say like more details about okay so if I'm someone that doesn't know at all about crypto and I want to use your application and I hear that Solana, for example, all of a sudden it's really positive but I don't know about it. It would be cool to have access to some general information and news about it.
Jerry - okay.
Ms. D - And maybe.
Jerry - Yeah it is a hundred percent doable.

Email Exchange of Approval

Twitter Sentiment Analysis Program Outline Inbox ×

Haoran [redacted]
to Dania
Sun, Mar 13, 9:40 PM

Dear Ms. D,
After the consultation, I came up with the following criteria for the program. Can you confirm if these criteria are appropriate and suitable for the program that you asked for?

1. This system is able to collect tweets about the top 5 popular cryptocurrencies on Twitter.
2. This system is able to analyze whether or not each tweet is neutral, bullish, or bearish using a machine learning algorithm.
3. This system has a clear Graphical User Interface (GUI) in python for the user to interact with.
4. This system has options to view each specific tweet collected under each type of cryptocurrency and their corresponding sentiment.
5. This system can display the percentage of neutral, bullish, or bearish of a cryptocurrency
6. This system can display the total tweets collected.
7. This system can display the most discussed cryptocurrency over an hour.
8. This system can display basic information about each cryptocurrency (cautions).

Dania [redacted]
to me
Mon, Mar 14, 11:29 AM

Dear Haoran,

Looks awesome. Excited to see the final product.

Thank you, I do confirm that these are all the criteria's I had requested.

Sincerely,

Dania [redacted]

•••

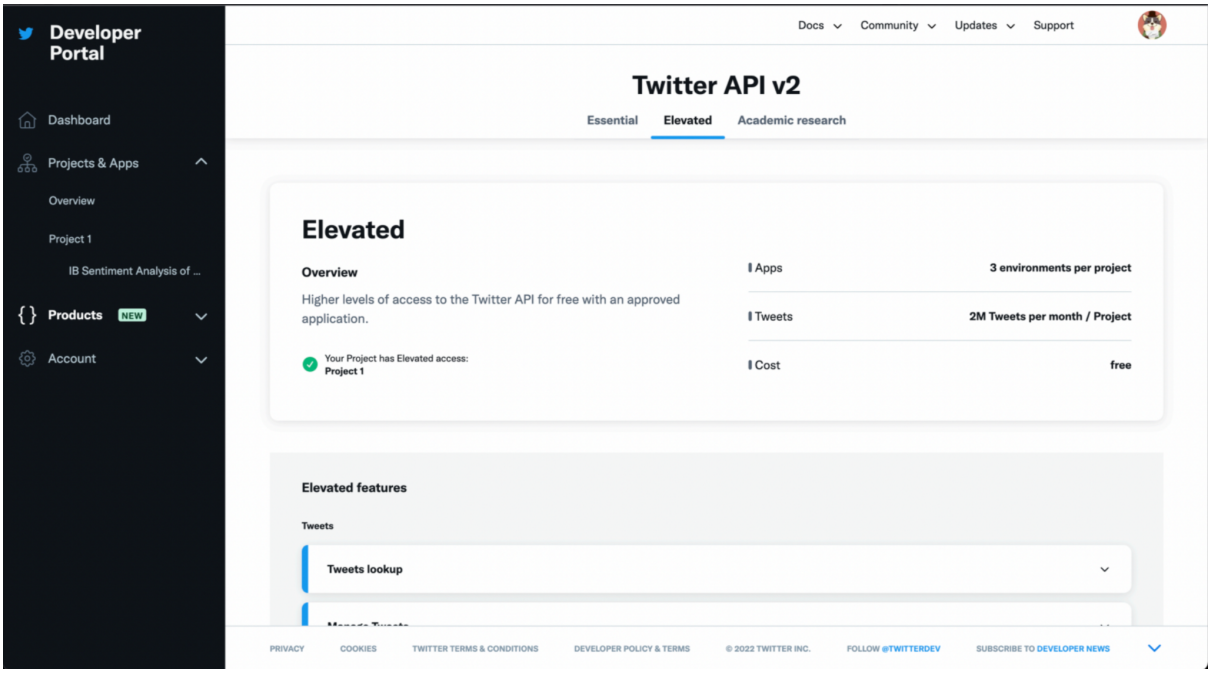
--

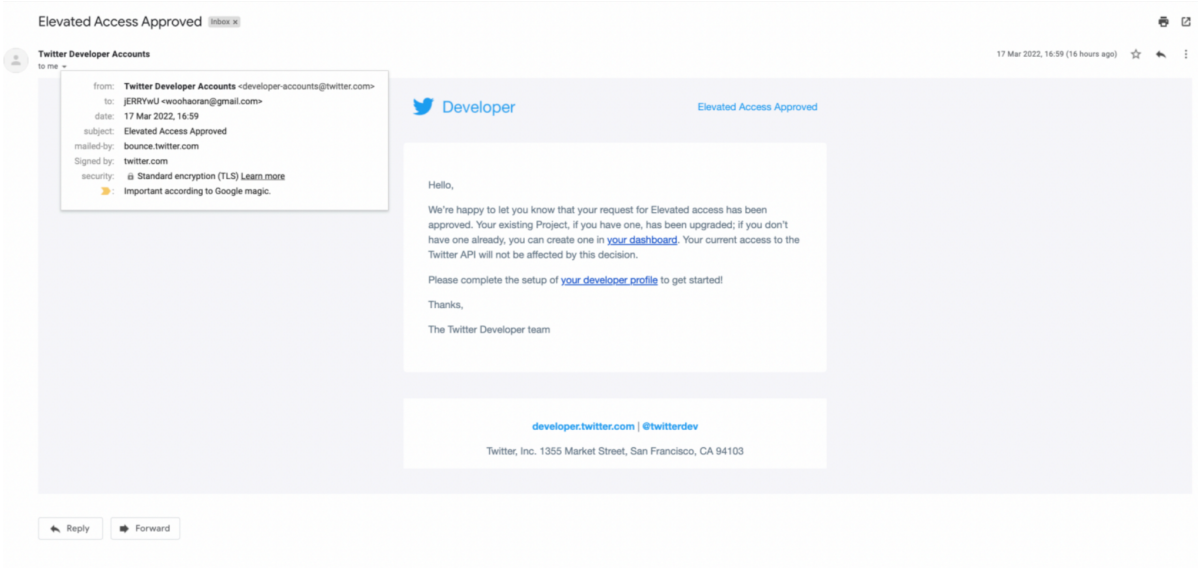
Dania [redacted]

Upper School Administrative Assistant

[redacted]

Twitter API Elevated Access





Last Meeting (Transcript of Interview #2)

Jerry - Do you have any like feedback and request for the future?

Ms.D - Yeah so I really love it. I think it's very simple. It would be awesome but that's a big challenge. If we would get alarmed. If there's a sudden change like what you told me like I would love to have like “ok caution” all of a sudden there's been a change of for example as Solana came in the first place. So that this gives me the hints that okay take a look at what's going on and then you can decide that would be awesome other than that I think it's complete while being concise. So..... I’m Impressed!

Jerry - Thank you, thank you!

Ms. D - You did a great job.

Jerry - Thank you!

Twitter Developer Agreement

Developer Agreement

Effective: March 10, 2020

This Twitter Developer Agreement ("Agreement") is made between you (either an individual or an entity, referred to herein as "you") and Twitter (as defined below) and governs your access to and use of the Licensed Material (as defined below).Your use of Twitter's websites, SMS, APIs, email notifications, applications, buttons, embeds, ads, and our other covered services is governed by our general Terms of Service and Privacy Policy.

PLEASE READ THE TERMS AND CONDITIONS OF THIS AGREEMENT CAREFULLY, INCLUDING ANY LINKED TERMS REFERENCED BELOW, WHICH ARE PART OF THIS LICENSE AGREEMENT. BY USING THE LICENSED MATERIAL, YOU ARE AGREEING THAT YOU HAVE READ, AND THAT YOU AGREE TO COMPLY WITH AND TO BE BOUND BY THE TERMS AND CONDITIONS OF THIS AGREEMENT AND ALL APPLICABLE LAWS AND REGULATIONS IN THEIR ENTIRETY WITHOUT LIMITATION OR QUALIFICATION. IF YOU DO NOT AGREE TO BE BOUND BY THIS AGREEMENT, THEN YOU MAY NOT ACCESS OR OTHERWISE USE THE LICENSED MATERIAL. THIS AGREEMENT IS EFFECTIVE AS OF THE FIRST DATE THAT YOU USE THE LICENSED MATERIAL ("EFFECTIVE DATE").

IF YOU ARE AN INDIVIDUAL REPRESENTING AN ENTITY, YOU ACKNOWLEDGE THAT YOU HAVE THE APPROPRIATE AUTHORITY TO ACCEPT THIS AGREEMENT ON BEHALF OF SUCH ENTITY. YOU MAY NOT USE THE LICENSED MATERIAL AND MAY NOT ACCEPT THIS AGREEMENT IF YOU ARE NOT OF LEGAL AGE TO FORM A BINDING CONTRACT WITH TWITTER, OR YOU ARE BARRED FROM USING OR RECEIVING THE LICENSED MATERIAL UNDER APPLICABLE LAW.

I. Twitter API and Twitter Content

A. Definitions

1. Broadcast ID - A unique identification number generated for each Periscope Broadcast.
2. Developer Site – Twitter's developer site located at **https://developer.twitter.com**
3. End Users – Users of your Services.
4. Licensed Material – A collective term for the Twitter API, Twitter Content and Twitter Marks.
5. Periscope Broadcast - A live or on-demand video stream that is publicly displayed on Twitter Applications and is generated by a user via Twitter's Periscope Producer feature (as set forth at **https://help.periscope.tv/customer/en/portal/articles/2600293**).
6. Services – Your services, websites, applications and other offerings (including research) that display Twitter Content or otherwise use the Licensed Material.
7. Tweet ID – A unique identification number generated for each Tweet.
8. Tweet – a posting made on Twitter Applications.

9. Twitter Content – Tweets, Tweet IDs, Twitter end user profile information, Periscope Broadcasts, Broadcast IDs and any other data and information made available to you through the Twitter API or by any other means authorized by Twitter, and any copies and derivative works thereof.
10. "Twitter" means Twitter, Inc., with an office located at 1355 Market Street, Suite 900, San Francisco, CA, 94103, USA. If you enter into this Agreement or an Order outside of the United States, Canada or Latin America, Twitter International Company with its registered offices at One Cumberland Place, Fenian Street, Dublin 2, D02 AX07, Ireland ("TIC") is the contracting entity.
11. Direct Message - A message that is privately sent on Twitter Applications by one end user to one or more specific end user(s) using Twitter's Direct Message function.
12. Twitter API – The Twitter Application Programming Interface ("API"), Software Development Kit ("SDK") and/or the related documentation, data, code, and other materials provided by Twitter with the API, as updated from time to time, including without limitation through the Developer Site.
13. Twitter Marks – The Twitter name, trademarks, and logos that Twitter makes available to you, including via the Developer Site.
14. Twitter Applications – Twitter's consumer facing products, services, applications, websites, web pages, platforms, and other offerings, including without limitation, those offered via **https://twitter.com** and Twitter's mobile applications.

B. License from Twitter. Subject to the terms and conditions in this Agreement and the Developer Policy (as a condition to the grant below), Twitter hereby grants you and you accept a non-exclusive, royalty free, non-transferable, non-sublicensable, revocable license solely to:

1. **Use the Twitter API to integrate Twitter Content into your Services or conduct analysis of such Twitter Content, as explicitly approved by Twitter;**
2. **Copy a reasonable amount of and display the Twitter Content on and through your Services to End Users, as permitted by this Agreement;**
3. **Modify Twitter Content only to format it for display on your Services; and**
4. **Use and display Twitter Marks, solely to attribute Twitter's offerings as the source of the Twitter Content, as set forth herein.**

C. License to Twitter You hereby grant Twitter and Twitter accepts a non-exclusive, royalty free, non-transferable, non-sublicensable revocable license to access, index, and cache by any means, including web spiders and/or crawlers, any webpage or applications on which you display Twitter Content using **embedded Tweets** or **embedded timelines**.

D. Incorporated Terms. Your use of the Licensed Material is further subject to and governed by the following terms and conditions:

1. the **Twitter Developer Policy**;
2. the **API Restricted Use Rules**;
3. the **Twitter Rules**;

4. as it relates to your display of any of the Twitter Content, the **Display Requirements**;
5. as it relates to your use and display of the Twitter Marks, the **Twitter Brand Resources**;
6. as it relates to taking automated actions on your account, the **Automation Rules**;
7. as it relates to your use of Periscope, the **Periscope Community Guidelines**, and the **Periscope Trademark Guidelines**.

The **Developer Policy**, **API Restricted Use Rules**, **Twitter Rules**, **Display Requirements**, **Brand Resources**, **Automation Rules**, **Periscope Community Guidelines**, and **Periscope Trademark Guidelines** are collectively referred to herein as the "Incorporated Developer Terms". You agree to the Incorporated Developer Terms, which are hereby incorporated by reference and are available in hardcopy upon request to Twitter. In the event of a conflict between the Incorporated Developer Terms and this Agreement, this Agreement shall control. None of the Incorporated Developer Terms expand or extend the license to the Twitter API, Twitter Content or Twitter Marks granted in this Agreement.

...]

login.txt


```
06tyUTKLcumWX0QHV0cEs3XQX
Pqoif26KVuSgP4CDzlqGbgNKhpqlmjWnuCTnerHuLzsv4s7Xl5
1069594080353624064-5Y9iFu61cLxl4CzEbeCNih54TgFMhd
2DreqDPPbQctMBc3OQ5EQ7bIE09unIeFnCuhMsvVeRtkJ
AAAAAAAAAAAAAAAAAAAAAH0TZAEAAAAAifa8zAtzO6yNqKLUhpOeNL5BBas%3DTtgLbUMQ2
iT0wRkyk9ZlLHoSwZalkC4wUcZ9mnPNwyNHgT7wNx
```


Code

```
#   import libraries
import os
import re
from regex import D
import sys
import tweepy
import datetime
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import PySimpleGUI as sg
from textblob import TextBlob
from tweepy import OAuthHandler
from matplotlib.ticker import NullFormatter
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import matplotlib
sg.theme("Reddit")
```

```
#   import keys for Twitter API Elevated Access (personal)

application_path = os.path.abspath('login.txt')
keys = open(f"{application_path}",'r')
lines = keys.readlines()
consumer_key = lines[0].rstrip()
consumer_secret = lines[1].rstrip()
access_token = lines[2].rstrip()
access_token_secret = lines[3].rstrip()
bearer_token = lines[4].rstrip()

#   authenticate to the twitter API
auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth)

#   create a client to collect tweets
Client = tweepy.Client(bearer_token=bearer_token,
                    consumer_key= consumer_key,
                    consumer_secret= consumer_secret,
                    access_token= access_token,
                    access_token_secret=access_token_secret)

#   set the display option of tweets to 100 characters & show all the
rows of tweets collected
pd.options.display.max_colwidth = 100
pd.set_option('display.max_rows', 9999)

#   layout of the GUI, where there is a homepage and 5 seperated pages
for the 5 cryptocurrency
homepage = [[sg.Text('Twitter Sentiment Analysis on
Cryptocurrency',font = "Helvetica 20",size=(50,1)),sg.Button("Refresh
for More Tweets",font = "Helvetica 15",key="refresh")],
        [sg.Text("Cryptocurrency just like any other investment
involves risks, and has potential in losing all your money. (DYOR) 📈📈
", font = "Helvetica 10")],
        [sg.Text("This application is not any financial advice, but an
educational application to monitor the sentiment of Twitter users. ",
font = "Helvetica 10")],
        [sg.Text("(Application) Discord/Telegram/Twitter → Crypto
community", font = "Helvetica 10")],
        [sg.Text("(Website) CoinMarketCap → Live price indicator", font
= "Helvetica 10")],
        [sg.Text("(Website) Coinbase → Exchanging currencies to
cryptocurrencies", font = "Helvetica 10")],
        [sg.Text("Total tweets in an hour",font = "Helvetica
11",key="total_tweets")],
        [sg.Text("TOP #1 Crypto Currency",font = "Helvetica
11",key="T1")],
```

```
        [sg.Text("TOP #2 Crypto Currency",font = "Helvetica
11",key="T2")],
        [sg.Text("TOP #3 Crypto Currency",font = "Helvetica
11",key="T3")],
        [sg.Text("TOP #4 Crypto Currency",font = "Helvetica
11",key="T4")],
        [sg.Text("TOP #5 Crypto Currency",font = "Helvetica
11",key="T5")],
        [sg.Text("Tweets Collected",font = "Helvetica
11",key="collect")],
        [sg.Canvas(key='-CANVAS-')]]
bitcoin = [[sg.Text('Bitcoin Sentiment Analysis (#BTC)',font =
"Helvetica 20",size=(50,1)),sg.Button("Refresh for More Tweets",font =
"Helvetica 15",key="refresh")],
        [sg.Text('[number of tweets per hour]',font = "Helvetica
15",key="total_Bitcoin")],
        [sg.Text("[sentiment for Bitcoin]",font = "Helvetica
15",key="Bitcoin_pos")],
        [sg.Text("[sentiment for Bitcoin]",font = "Helvetica
15",key="Bitcoin_neu")],
        [sg.Text("[sentiment for Bitcoin]",font = "Helvetica
15",key="Bitcoin_neg")],
        [sg.Multiline("[tweets and their corresponding sentiment]",font
= "Helvetica 14",enter_submits=False,key="Bitcoin",size=(120,30))],]

ethereum = [[sg.Text('Ethereum Sentiment Analysis (#ETH)',font =
"Helvetica 20",size=(50,1)),sg.Button("Refresh for More Tweets",font =
"Helvetica 15",key="refresh")],
        [sg.Text('[number of tweets per hour]',font = "Helvetica
15",key="total_Ethereum")],
        [sg.Text("[sentiment for Ethereum]",font = "Helvetica
15",key="Ethereum_pos")],
        [sg.Text("[sentiment for Ethereum]",font = "Helvetica
15",key="Ethereum_neu")],
        [sg.Text("[sentiment for Ethereum]",font = "Helvetica
15",key="Ethereum_neg")],
        [sg.Multiline("[tweets and their corresponding sentiment]",font
= "Helvetica 14",enter_submits=False,key="Ethereum",size=(120,30))],]

binance = [[sg.Text('BinanceCoin Sentiment Analysis (#BNB)',font =
"Helvetica 20",size=(50,1)),sg.Button("Refresh for More Tweets",font =
"Helvetica 15",key="refresh")],
        [sg.Text('[number of tweets per hour]',font = "Helvetica
15",key="total_BNB")],
        [sg.Text("[sentiment for BinanceCoin]",font = "Helvetica
15",key="BNB_pos")],
        [sg.Text("[sentiment for BinanceCoin]",font = "Helvetica
15",key="BNB_neu")],
        [sg.Text("[sentiment for BinanceCoin]",font = "Helvetica
15",key="BNB_neg")],
```

```
        [sg.Multiline("[tweets and their corresponding sentiment]",font
= "Helvetica 14",enter_submits=False,key="BNB",size=(120,30))],]

solana = [[sg.Text('Solana Sentiment Analysis (#SOL)',font = "Helvetica
20",size=(50,1)),sg.Button("Refresh for More Tweets",font = "Helvetica
15",key="refresh")],
        [sg.Text('[number of tweets per hour]',font = "Helvetica
15",key="total_Solana")],
        [sg.Text("[sentiment for Solana]",font = "Helvetica
15",key="Solana_pos")],
        [sg.Text("[sentiment for Solana]",font = "Helvetica
15",key="Solana_neu")],
        [sg.Text("[sentiment for Solana]",font = "Helvetica
15",key="Solana_neg")],
        [sg.Multiline("[tweets and their corresponding sentiment]",font
= "Helvetica 14",enter_submits=False,key="Solana",size=(120,30))],]

ripple = [[sg.Text('Ripple Sentiment Analysis (#XRP)',font = "Helvetica
20",size=(50,1)),sg.Button("Refresh for More Tweets",font = "Helvetica
15",key="refresh")],
        [sg.Text('[number of tweets per hour]',font = "Helvetica
15",key="total_XRP")],
        [sg.Text("[sentiment for Ripple]",font = "Helvetica
15",key="XRP_pos")],
        [sg.Text("[sentiment for Ripple]",font = "Helvetica
15",key="XRP_neu")],
        [sg.Text("[sentiment for Ripple]",font = "Helvetica
15",key="XRP_neg")],
        [sg.Multiline("[tweets and their corresponding sentiment]",font
= "Helvetica 14",enter_submits=False,key="XRP",size=(120,30))],]

layout = [[sg.TabGroup([[sg.Tab("Homepage",homepage),
                        sg.Tab("Bitcoin",bitcoin),
                        sg.Tab("Ethereum",ethereum),
                        sg.Tab("Binance",binance),
                        sg.Tab("Solana",solana),
                        sg.Tab("Ripple",ripple)]],font="Helvetica
16")]]

#   define the window of the GUI, and enable updating for Graph
window = sg.Window('Twitter Sentiment Analysis on Cryptocurrencies',
layout, finalize=True, element_justification='center', font='Helvetica
18',size=(1300,1000))

#   define class for each cryptocurrency
class CryptoCurrency ():
    def __init__ (self,query):
        self.query = query
        self.pos = int(0)
        self.neg = int(0)
        self.neu = int(0)
```

```
        self.total = int(0)
        self.tweets = []

    #   cleant tweets for the sentiment analysis
    def clean_tweet(self,tweet):
        return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z
\t])|(\w+:\/\/\S+)", " ", tweet).split())

    #   using a pretrained model, determine the sentiment of the tweets
    def get_tweet_sentiment(self,tweet):
        analysis = TextBlob(self.clean_tweet(tweet))
        if analysis.sentiment.polarity > 0:
            return 'POSITIVE'
        elif analysis.sentiment.polarity == 0:
            return 'NEUTRAL'
        else:
            return 'NEGATIVE'

    #   collect tweets with the limit of the variable count
    def get_tweets (self,count):
        self.tweets = []
        pagination = []

        #   collect tweets using the Cursor function
        for status in tweepy.Cursor(api.search_tweets,f"{self.query}
-filter:retweets").items(count):
            pagination.append(status.text)

        #   append the text of the tweet and sentiment together into a
dictionary
        for tweet in range(len(pagination)):
            parsed_tweet = {}
            parsed_tweet['SENTIMENT'] =
self.get_tweet_sentiment(str(pagination[tweet]))
            parsed_tweet['TEXT'] = str(pagination[tweet])
            self.tweets.append(parsed_tweet)
        return self.tweets

    #   get the tweets posted under an hour for each specific
cryptocurrency
    def get_tweets_count (self,start,end):
        self.total = Client.get_recent_tweets_count(query =
self.query,start_time=start,end_time=end)
        return self.total

    #   dislay the the total tweets of that specific tweet in an hour,
and the # of positive, negative, and neutral tweeets
    def display(self):
            total_tweets = self.pos + self.neg + self.neu
            window[f"{self.query}_pos"].update(f"# of Positive:
{self.pos}({round(self.pos/total_tweets*100,2)}%)")
```

```
            window[f"{self.query}_neg"].update(f"# of Negative:
{self.neg}({round(self.neg/total_tweets*100,2)}%)")
            window[f"{self.query}_neu"].update(f"# of Neutral:
{self.neu}({round(self.neu/total_tweets*100,2)}%)")
            window[f"total_{self.query}"].update(f"Total Tweets for
{self.query}: {self.total}")
            #   display the tweets and their corresponding sentiment
            window[self.query].update(f"{self.df}")

    #   create a dataframe for tweets and their sentiment
    def data_analysis (self,count,pos,neu,neg):
        self.tweets = self.tweets + self.get_tweets(count)
        self.df = pd.DataFrame(self.tweets)

        #   calculate the number of positive, negative, and neutral
        for i in range(len(self.tweets)):
            if self.tweets[i]['SENTIMENT'] == 'POSITIVE':
                self.pos = self.pos +1
            elif self.tweets[i]['SENTIMENT'] == 'NEGATIVE':
                self.neg = self.neg +1
            else:
                self.neu = self.neu +1

        #   add the total number of positive, negative, neutral in a
list for further calculation
        pos.append(self.pos)
        neu.append(self.neu)
        neg.append(self.neg)
        return pos,neu,neg

    #   add each total tweet into the TOTAL Tweet of 5 cryptocurrencies
ALL TOGETHER
    def total_tweets (self,total_tweets,total_sort):
        start_time,end_time = get_time()
        self.total =
self.get_tweets_count(start_time,end_time)[3]["total_tweet_count"]
        total_tweets = total_tweets + int(self.total)

        #   creating a dictionary of the name of cryptocurrency and
number of tweets colelcted for them
        total_sort[self.query] = self.total
        return total_tweets,total_sort

#   getting the current time and time of an hour ago
def get_time ():
    #   gather current time in RFC3339 datetime style for tweepy
operation
    date = datetime.datetime.utcnow()
    time = (date.strftime("%Y-%m-%dT%H:%M:%S"))
```

```
    #   determine the end time, because it has to be 1 minute before
the current time
    #   if the the end time is 00, -1 would not be an appropriate time
    #   hence go back an hour and set the minute to 59
    if time[14:16] == "00":
        end_time =  time[:11] + str((int(time[11:13])-1))+ ":59"
+time[16:] + ".00Z"
    else:
        if len(str(int(time[14:16])-1)) != 2:
            end_time =  time[:14] + "0" +str((int(time[14:16])-1))+
time[16:] + ".00Z"
        else:
            end_time =  time[:14] + str((int(time[14:16])-1))+
time[16:] + ".00Z"

    #   determine the start time for the tweet collection (1 hour
before)
    start_time =  time[:11] + str((int(time[11:13])-1))+ time[13:] +
".00Z"
    return start_time, end_time

#   defining the figure, so a matplotlib graph can fit in a pysimplegui
format
def draw_figure(canvas, figure):
    figure_canvas_agg = FigureCanvasTkAgg(figure, canvas)
    figure_canvas_agg.draw()
    figure_canvas_agg.get_tk_widget().pack(side='top', fill='both',
expand=1)
    return figure_canvas_agg

#   setting the graphs in the homepage
def homepage_chart (pos,neg,neu):
    #   reset the graph when the refresh button is being pressed (avoid
overlapse)
    plt.close('all')
    crypto_name = ["BTC","ETH","BNB","SOL","XRP","TOTAL"]

    pos.append(np.average(pos))
    neg.append(np.average(neg))
    neu.append(np.average(neu))

    #   create a dictionary of each sentiment to an array of positive,
negative, neutral
    data = {
        "Positive":pos,
        "Neutral":neu,
        "Negative":neg}

    #   create a dataframe
    df = pd.DataFrame(data,index=crypto_name)
```

```
    #   define a stacked horizontal bar graph with different color for
each sentiment
    plots = df.plot(kind = 'barh',
            stacked = True,
            figsize=(10,6),
            color = ["#46B748","#FCDE02","#EA1D25"]
            )
    #   define the axis, and position of the graph.
    plt.legend(loc="upper left", ncol=2)
    plt.xlabel("Sentiments")
    plt.ylabel("Cryptocurrencies")

    #   define the figure, and update the graph in the homepage.
    fig = matplotlib.figure.Figure(figsize=(10, 6), dpi=100)
    fig = plt.gcf()
    fig_canvas_agg = draw_figure(window['-CANVAS-'].TKCanvas, fig)

    return fig_canvas_agg

#   homepage GUI
def homepage_info (total,rank):
    #update the total tweet collected in an hour
    window['total_tweets'].update(f"Total tweets posted about
cryptocurrencies: {total}")
    count = 0

    #   for loop to print out all the ranking of popularity of
cryptocurrencies
    #   rank is equal to the total_sorted dictionary, hence we need
keys to access value
    for key in rank:
        count = count + 1
        display = str(f"T{count}")
        window[display].update(f"{count}. {key}: {rank[key]}")

#   main function
def main():
    #   define each crypto class with their corresponding query for
searching
    BTC = CryptoCurrency("Bitcoin")
    ETH = CryptoCurrency("Ethereum")
    BNB = CryptoCurrency("BNB")
    SOL = CryptoCurrency("Solana")
    XRP = CryptoCurrency("XRP")
    #   number of sentiment of each cryptocurrency in a list
    pos = []
    neu = []
    neg = []
    #   combine all the class objects into an array for operation
    crypto = [BTC,ETH,BNB,SOL,XRP]
    #   the number of tweet to collect once the app is opened
```

```
    count = 100
    total_tweets = 0
    total_sort = {}
    collect = 0
    #   for each cryptocurrency, gather the number of positive,
neutral, negative tweets
    for type in range(len(crypto)):
        pos,neu,neg = crypto[type].data_analysis(count,pos,neu,neg)
        #   gather the total tweets of each crypto currency, add them
to the total tweets of ALL 5
        #   append to an dictionary where then it will be sorted
        total_tweets, total_sort =
crypto[type].total_tweets(total_tweets,total_sort)
    #   using the dictionary, where the key is the cryptocurrency, and
the value is the total tweets
    #   sort in reverse order from greatest to least, and store in a
new variable
    total_sorted = dict(sorted(total_sort.items(), key=lambda item:
item[1],reverse=True))
    #   display on the GUI
    homepage_info(total_tweets,total_sorted)
    #   display chart on GUI, and obtain the value of fig_agg
    fig_agg = homepage_chart(pos,neg,neu)
    for coin in crypto:
        coin.display()
    collect = collect + count*5
    window['collect'].update(f"Actual Tweets Collected:
{int(collect)}")
    #   GUI Loop
    while True:
        event, values = window.read()
        if event in (sg.WIN_CLOSED, 'Cancel'):
            break
        #   actions when the refresh button is pressed
        if event == "refresh":
            pos = []
            neu = []
            neg = []
            #   collect another 100 tweets
            for type in range(len(crypto)):
                pos,neu,neg =
crypto[type].data_analysis(100,pos,neu,neg)
            #   update information
            fig_agg.get_tk_widget().forget()
            fig_agg = homepage_chart(pos,neg,neu)
            for coin in crypto:
                coin.display()
            collect = collect + count*5
            window['collect'].update(f"Actual Tweets Collected:
{int(500)}")
```

```
main()
```