OCR
Oxford Cambridge and RSA

Candidate Authentication Statement

The completed form should be retained within the centre and should **not** be sent to the moderator or OCR unless specifically requested.

NOTICE TO CANDIDATE

The work you submit for assessment must be your own.

If you copy from someone else or allow another candidate to copy from you, or if you cheat in any other way, you may be disqualified from at least the subject concerned.

1. Any help or information you have received from people other than your subject teacher(s) must be clearly identified in the work itself.
2. Any books, information leaflets or other material (e.g. videos, software packages or information from the Internet) which you have used to help you complete this work must be clearly acknowledged in the work itself. To present material copied from books or other sources without acknowledgement will be regarded as deliberate deception.

Declaration by candidate

Centre name	LONDON BROOKES COLLEGE	**Centre no**	1 2 2 3 2
Series	June	**Year**	2 0 2 2
Specification or unit title	H446B		
Candidate name	KARRISON THIRRUPATHY	**Candidate number**	3 0 3 6

I have read and understood the **Notice to Candidate** (above). I have produced the work without any help from other people apart from that which I have declared in the work itself. I have acknowledged all source materials in the work itself.

Candidate's signature: Karrison.T

Date: 06/04/22

Notes

Once completed, the candidate authentication statement should be stored securely within the centre. A copy of this authentication form must be available upon request for each coursework/portfolio submission.

CREATING A TWITTER CRAWLER

Full name:	Karrison Thirrupathy
Candidate number:	3036
Centre name:	London Brookes College
Center number:	12232
Option code:	H446B
Season:	June 2022
Level:	GCE
UCI:	125400150223L

Contents

1 Analysis

1.1 Introduction to Problem

The problem I am tackling is a Twitter Scraper. I am trying to help people find users and tweets in certain areas regarding issues they are interested in. The reason I am doing this is to help all three of my stakeholders (John, Victoria and Albert) in different ways. For example, John is a highly experienced Information Technology (IT) worker, being involved for nearly 30 years and I want to help him find new customers in any location he chooses. For Victoria, she is a teacher of A-level science and I want to help her connect to other teachers and find the best resources online to expand not only her outlook but her students' outlooks as a result. Finally, for Albert, he isn't the most experienced online (especially on Twitter) therefore, I want to simplify his experience so he can find tweets about his own interests.

The Twitter Scraper, I will create, will search Twitter's databases using the representational state transfer application programming interface (REST API) and will find tweets about a certain topic in a certain location that the user has specified. Then, this can be outputted onto spreadsheet software, so the user is able to expand it and potentially use the hyperlinks attached, to probe further onto that topic.

1.2 Features that make the problem solvable by computational methods

Features	Justify why having a computational approach makes sense
Output results	First, I recognised a problem. The results would be outputted as a dictionary – it would be very hard to interpret the results, not only for the user but also for me when testing. Therefore, I used the Pandas data structure making it easier to process for the stakeholders as the columns are sorted into user_id, date, the actual tweet and the number of likes.
My Structured Query Languages (MySQL) tables inside database	I separated each process inside the database into tables. This means it is easier to understand what is inside each field, and easier to edit the tables if needed as it is sorted in a user-friendly manner applying visualisation.
Comma-separated values (CSV) file sorted in tables	This also applies visualisation as it has to be easy for the stakeholder, specifically for Victoria as she needs to access the csv file to find the hyperlink to find any more materials related to that Twitter thread.

Features	Justify why having a computational approach makes sense
Running through questions until User clicks “Yes”	This applies backtracking since I view “No” as a dead end and it would go onto another path/question and keep going until the user selects “Yes” which is viewed as a success. It could also apply enumeration as it goes through every word and if the user keeps selecting “No” then every case will have been searched. This is required since I need to provide the user with the best optimal search result in relation with their query and the question word allows me to narrow down the data mining as the crawler has to search through a large REST API database for tweets. This also applies abstraction in its running as it is simplified to just making the user press “YES” or “NO”, instead of typing exactly what they want – which may be lengthy. Instead this is a much more basic process to get the same solution.
Find the most likely pairing quickly	Applies automation as with enough user inputs over a period of time, a mathematical model can be created from this and then the artificial intelligence I implemented will provide the most likely question word to the user’s query word. E.g. Do you want to learn Python? This speeds the search process, optimising the user’s time, meaning they can find the results they desire faster (reducing any time complexities) – with more inputs increasing the speed rather than decreasing it – therefore the users will have a more positive view of the program. In all of this, I am using a heuristic approach as I wanted to find an optimal solution for the user in a reasonable time frame. Furthermore, this uses divide and conquer. It divides the question words first then once it is selected with the query word. The query word is searched on Twitter’s databases then the question word is used to find the best solutions for the user.

1.2.1 Research on other alternatives

Existing Solution	Identify approach	Justify
https://www.google.com/intl/en_uk/search/howsearchworks/	This searches through hundreds of billions of web pages and other content stored in their Search index using their Page Rank algorithm and also enables users to carry out data mining as it predicts user search results and places them at the top in regularly asked questions, since the user is probably asking themselves that same question with that search query. However, my solution will be different as instead of searching through Twitter's REST API globally, we ask the user his/her search area (the country and city they want to specifically search tweets in) and we provide local results in that results in a radius of 50 miles with search results from tweets there.	This means that users can not only get useful information much like Google's search engine, boosted by the crawlers implemented, but also information that is local and relevant to the area they would like. This means that those tweets are more likely to be much more relevant to the user as they would be in a place that they want to know about and specifically about a certain topic that is sorted by a question word in that area, providing a much more personal experience.
https://github.com/bisguzar/twitter-scraper	This enables you to retrieve tweet data from a specific user up to a certain limit (which is around 800 tweets). My solution will be different because I do not want a specific user's tweets but instead the general topic (query word) that the user entered allowing multiple tweets from multiple users to pop up.	This means that in my current solution, the user can search up a topic they want to know about on twitter and find users who tweeted about that whereas in this solution, it finds a specific user and only extracts a few tweets from their profile. Unlike in their solution, I can change the number of tweets that are shown but it is currently at ten to make the user experience more friendly.

Existing Solution	Identify approach	Justify
https://webscraper.io/	This is provided to the user as a web extension and can run on multiple websites to find certain products that the user would like to find. My crawler is similar to this as it does also find similar tweets to the user entered desire.	This has more flexibility than my solution due to its ability to work on multiple different websites unlike my solution which is catered to Twitter however mine is less complicated to understand for the user and far simpler to use as all requires are the inputs being fulfilled and the program being run. This, unlike mine, requires setting up and requires a further understanding of how the crawler works to fully utilise it so to a new user, this product would not be useful unlike my twitter crawler, which may be machine specific, but has clear simple instructions and inputs to make sure the user has a flawless experience.

Existing Solution	Identify approach	Justify
https://www.wolframalpha.com/	This is a type of search engine however instead it has a computation knowledge engine. It generates output by doing computations from the "Wolfram Knowledgebase" instead of traditionally searching the web and returning links. It also continually updates its data, in real time. It also tracks your Internet Protocol (IP) address to find your location within 5 miles.	This is similar to mine in how it continually updates depending on user responses however, Wolfram Alpha requires its code base to continually be updated to have changes whereas my code base does not have to be changed for its base functionality with the only part changing, being the user input and the artificial intelligence making decisions depending on that. Wolfram Alpha tracks your location unlike my program since we do not want to violate any privacy laws (like the Data Protection Act) or any users who may not want to share their location but are unaware that the IP address is being tracked. I let the users choose which location they want to choose to find the best results for them – this gives an option for the user to look anywhere which may allow adaptability. This is because, if the user was travelling somewhere in the not so distant future and wanted to find honest reviews about the place, they can use our location feature to find relevant tweets. By doing this, we are allowing for greater user hands-on capability and making them feel involved.

1.3 Stakeholder Analysis

Name of Stakeholder	Description	How it will help them?	Why is this suitable for them?
John	Involved in IT industry helping customers for nearly 30 years.	They can find people who need help on Twitter and contact them to find potential new customers. Due to their experience in the IT industry, they have a high expectation for a product before they can use it and will be impartial in their critique of the program. However, they lack coding experience so cannot directly help me in the programming but will provide direct guidance when needed.	Instead of manually trying to find potential new customers (which is very time-exhaustive), the search features enable convenience and efficiency for the stakeholder as they can find multiple twitter users needing IT help within a specific area even if they are not the most followed users.
Victoria	Teacher of A-level Science for young people.	They can find relevant educational resources easily from other users on Twitter, such as other teachers who may post useful resources to help them out.	Twitter is used by many people of different class, stature or age and they all have useful opinions. This will allow the teacher not only to gain useful resources but also further useful insight into her query, which will allow her to have an increased positive effect on her students, broadening their horizons as a result. E.g. if she comes across new technology due to our program, then the students in turn will now be informed about this new technology.

Name of Stakeholder	Description	How it will help them?	Why is this suitable for them?
Albert	My 60-year-old Client.	Uses Twitter however not the most acquainted with Twitter. This simplifies Twitter hashtag structure, instead with this search feature allowing them to find matters that they care about or interested in. The user themselves has a clear idea of a project they would like, however in terms of using abstraction to simplify that problem and giving exact criticism and ways on how to improve, they are not the best due to their lack of understanding of technology.	They rely heavily on Twitter's suggested feed, many times not providing an enjoyable experience of Twitter as their feed doesn't align with their interests. The search mechanism makes sure they can find exactly what they are interested in and other users who tweet and are fans of the same interests as they are.

1.4 Measurable Success Criteria

No.		Identify what makes this successful	Justify
1		Program loads without error.	It won't work if it doesn't load.
2		Design form appears when program is run	The user is able to see that the program is now available to work with
	2.1	User can enter in some inputs into the text boxes	This is their search word – the word they want to enquire using my crawler. Without this input, the user cannot search for their chosen topic
	2.2	Submit button can be clicked.	For the program to progress, the user has to be able to click the button, otherwise the program will not know when the user has finished their input
	2.3	New window is created	This allows for a cleaner user interface instead of adding all commands onto one window

No.		Identify what makes this successful	Justify
	2.4	Old window closes as soon as new window is created	This means there is less clutter on the screen for the user contributing to less confusion as it is easier to interpret and uses up less memory as one window is now already closed
	2.5	Validation	If the user input does not meet the criteria for a correct input, then it must be told to the user and they should be able to try input again until they get a permitted input. The criteria the query input has to fulfil: — Not null — Query word is less than 100 characters in length — English syntax (English characters only, no special characters) — Greater than 2 characters — No abbreviations
	2.6	Question statement appears	The user can now review the question they generated – to see if it links to what they want
	2.7	Radio-button appears	This provides an opportunity for the user to respond to the question
	2.8	User can select either of the radio-buttons	A radio-button eliminates the element of doubt since the user is only limited to a selection of 'yes' or 'no'
	2.9a	The question does not change if the user clicked 'Yes'	The user has found the question for what they are specifically trying to find. Therefore, it requires no more searching as the user's search has finished
	2.9b	The question now changes if the user clicked 'No'	The user has still not found what they are looking for, so as long as the user keeps selecting 'no', the question should keep changing until the user selects 'yes'
3		User inputting their City and Country they want to search	This is important since it gives us the input to know where to run location searches on tweets
4		Twitter development account has been approved by Twitter	Without Twitter approving the development account, I am unable to access Twitter's databases so I would be unable to search for tweets without it being approved

No.		Identify what makes this successful	Justify
	4.1	Connect to the Twitter REST API	This is the basic part of running processes using Twitter's API. If I can connect to Twitter then I can find tweets relating to the user's search query. Now I have gained access to Twitter's databases
	4.2	Connecting Twitter to Python	Using the credentials as a login, does Twitter allow access to Python to run my code using my created Twitter Development Account as a proxy
	4.2a	Writing the Twitter Credentials in a file	A subclass should write the twitter credentials into a file so it can be accessed by the main program. It is in a file for security purposes so someone cannot gain access to my twitter credentials without them also having access to the file
	4.2b	Reading the Twitter Credentials from a file	The main program should be able to read and open the file written by the previous subclass and extract the necessary information from there like the consumer key for example
	4.2c	Loading the Twitter Credentials file	Once the file is read then it should be loaded up and the information requested by the code should be made usable in programs to carry out functions with that information
5		Run and load up MySQL	Am I able to connect to MySQL without any problems? This is where I am going to create a database. MySQL loads up with no errors
	5.1	Creating tables in MySQL	When I create tables then it provides the data structure to store any data that I would get from the data processing
	5.2	Entering in the 100+ question words as the variable 'words' into the action table	This is what is used to specify the user's search query. These words are 100 of the most common English words therefore the user's search query is likely to be involved using one of these 100 words or any words to that effect.
	5.3	Creating the structure of the other tables	This should when connected to Python, mean that the user input should slot in the variable 'phrase' and be stored within the tables

No.		Identify what makes this successful	Justify
	5.4	Integrating MySQL into Python	This allows us to connect user input and place that into the table and for us to use any tables previously created in MySQL in Python. This is useful for both storing and getting data
	5.5	Load up and store data in the database	Run MySQL instructions in Python using cursor – which is a variable that allows me to carry out MySQL instructions in Python such as SELECT, which means I can specifically call data from certain tables
6		Importing libraries	I need to use externally created libraries for some parts of the project instead of spending time manually crafting out already made functions
	6.1	Importing mysql. connector	This is a library that allows us to connect to MySQL and without it I cannot connect Python to MySQL. This is similar to 3.3 however that is more specified towards carrying out functions with the data from MySQL compared to this, which is specifically to providing the bridge between MySQL and Python
	6.2	Importing Twython	This is a Python library which has been created to provide a way to access Twitter data. This is important for us to use as I need to access Twitter lists of Tweets being sent out and data on those tweets
	6.3	Importing json	JSON helps us to store the Twitter Credentials like the API key in a human-readable format but able to also be translated and understood by the main program
	6.4	Importing pandas	Pandas allows us to sort through all the data that I have gathered from Tweets quickly and efficiently as it is a data frame as well as a library to allow for analysis and manipulation of data
	6.5	Importing Nominatim	I use this library to find the user entered location specifically the longitude and latitude and then use that to sort through the tweets filtering by distance from user entered area

No.		Identify what makes this successful	Justify
	6.6	Importing Tkinter	Tkinter is a library specifically designed for aiding Graphical User Interface design and this will help me in creating a User Interface to help the user visualise better what the program can do and what they, the user can do in progressing the program
	6.7	Importing time	Instead of rapid movement of text, time allows me to slow it down such as for the updating questions when No is pressed, time will help to slow it down by a few seconds so the user can see it has changed
7		Incorporating Artificial Intelligence	By incorporating some sort of AI, this will allow us to become much more efficient providing the user a much more flawless experience as time goes on
	7.1	Storing count of number of times each word was selected	Whenever the user clicks “Yes” to a question, the count for that ‘word’ increments by one, this will then be useful for finding the most common query words
	7.2	User words being stored with the question word chosen with it	The query words will then be stored with the question words that was chosen. The less common words (e.g. if hospital and play were never paired together) will be then eventually be sorted out of the list, to reduce time.
	7.3	Pairing common query words and question words together	This will then create a pair and the more pairs that are created then the more likely the AI can find the most common question words
	7.4	Ordering the most common pairs first in descending order	This means when that query word is entered then the most common pair word will be outputted first
	7.5	Suggesting those pairs to the user	These pairs when they match the user query word will be suggested to the user and then this speeds up the overall process as the most likely question words to do with that query are likely to come up first.
8		Outputting data to the user	The user has to see what is going on, therefore an output allows them to see the results of their search
	8.1	Twitter handle of tweeter	The twitter handle provides the user to see if they like the tweet then they can research more about that specific user and potentially find a new friend or customer helping out John my stakeholder

No.		Identify what makes this successful	Justify
	8.2	Location of tweet sent	The location of the tweet helps to find local issues for the user or if there is an event they want to find out about, it is much more personal if it is nearby rather than halfway across the world
	8.3	Date of when tweet was sent	The date indicates how recent it was – e.g. if this was used to find out about the news then the date of tweet in seeing the reliability of that information
	8.4	Text of actual tweet	This is the information of the tweet – the most important for many, this is how the user can see what the actual tweet is about
	8.5	Number of likes on tweet	The number of likes on the tweet indicate popularity. However, as the program sorts out tweets based on location and date, this isn't the most useful feature. Although it can show my user whether the tweet is a bot tweet or not
9		Pasting all this onto an accessible Microsoft Excel file	This pastes all the output onto an accessible Microsoft Excel sheet so the user can look at it more specifically. Furthermore, with every tweet there is a small hyperlink attached to see the whole tweet on the web browser. This feature would be very useful in particular for my stakeholder Victoria, who would be able to access any useful resources linked in the tweets for her teaching

1.5 Features

1.5.1 Essential Features

Essential features	Why is it essential?
Running of MySQL in the background	This is essential since the main program references the database I created and specifically the tables inside it and without MySQL running its console then I cannot access nor store any variables or data inside them (such as the 100 words I ask the user to pinpoint the user's search query)

Essential features	Why is it essential?
User Interface	This is essential since without it, the user can no longer interact with the program actively in the front end. If the user cannot input anything, then the program cannot progress and continue running. As a result, it will be seen as not working. During the design of this project, I have tried to simplify the output to the user so anyone can use my Twitter Search Crawler as from my research I have seen from other crawlers, that it is either too complicated to use without proper guidance or too expensive to use therefore by simplifying it; lack of computer knowledge does not affect the running of the program and it is free currently so no one will be outpriced by it.
Twitter's REST API	Without the Twitter REST API, I cannot find and filter the tweets that I would like to show the user in relation to his search query. This is where the Twitter data arrives from, so without this, there is no connection to Twitter to extract the specific tweets, affecting the running of the program.
Twitter Credentials	Without the Twitter credentials such as the API key, I would not be able to have access to the API therefore unable to run any processes on the REST API and extract tweets from there as I would be denied access in the first place.

1.5.2 Additional Features

Feature I would like to add	Why? Any problems?
Voice recognition	The research required would be substantial and I would need to carry out much more testing as the voice recognition software may not recognise a particular accent the user may have or may have inaccuracies in understanding the words inputted into the voice recognition software. It would increase time taken on the project, and due to time constraints, this seems quite an unlikely prospect of being added to the program.

Feature I would like to add	Why? Any problems?
Usable on other devices (e.g. iPad)	This would require a reshuffle in User Interface design as they need to be created in an app format to be usable on an iPad. Another logistical difficulty is that an Apple uses the IOS operating system which is closed source so is quite difficult to convert when you do not have the source code. There are solutions online but they are not always the safest and may lead to viruses being installed on the computer as a result of trying to convert it. Changing the design for an android device is much simpler since it is open source therefore the source code and design can be changed in whatever way I desire.
Able to input in words from different languages	This would increase the users able to use this program but would require translations in many languages (which I may be able to do for some languages like Mandarin) and would require the API itself to be able to filter in words entered from different languages in tweets. Or another solution is to translate the user input into English, enter it into the API, gain the results and output but that would mean the integration of a translator but there is also the problem of words lost in translation that may not pick out an accurate translation and provide the user the wrong information. There is also the problem of time constraints as this would require heavy amounts of research as I need to look into different languages and make sure the program accepts the different inputs in other languages and also make sure the program can identify when it is not a correct input.

1.5.3 Limitations

Limitations	Why is it a limitation?	Justify who is still ok to be limited by it.
Only able to use on one platform	If the user does not use that specific platform, in my case being run on an Windows OS then they will have difficulty running the program on an IPad for example	Those who already use and are familiar with using Windows. As the program is specialised for Windows OS. Currently, I have not decided to implement on multiple Operating Systems since I want to first create the final product then adapt the program into an app-format.

Limitations	Why is it a limitation?	Justify who is still ok to be limited by it.
Requires installation of libraries to be able to be used	Without the libraries, being downloaded (e.g. Twython), then the code cannot reference the functions in those libraries therefore the code cannot run fully. If the libraries cannot be referenced, then the code will just run errors.	If the libraries are already downloaded, then there is no issue in functionality and the code if everything else is correct can run smoothly. The solution may be to download the libraries, the first time the program is running and show the user that they are downloading something before they use the program. If this was in an app format then in the installation of the app, the downloading of the libraries can be included.
Requires a MySQL database to be set up with all necessary tables	Without MySQL running in the background, data cannot be stored and accessed in/from those tables meaning a lack of accuracy in trying to gain exactly what the user wants to search.	If MySQL is running, then the functionality of the program is not affected, however it does require more memory usage when it runs in the background of the program.
The user has to keep going through all the words to find exactly which word they want	As the words are run in a loop and being outputted to the user, the user has to keep saying “N” for no for the words they do not want, which may be repetitive and irritate the user at the start	If they are willing to go through the laborious process of going through the words, and repeatedly inputting “N” until they find the word they want then they will find the correct search for what they desire. This is only planned to be a problem at the very beginning while the AI is getting accustomed to user responses then the user may not have to go through many words (or every word like before) as the AI will provide more specific suggestions powered by previous user responses.

Limitations	Why is it a limitation?	Justify who is still ok to be limited by it.
Only able to search through one account	If the user would like to search through their own Twitter account then they would be unable to.	If they do not have a Twitter account, then the user can still use this Twitter search crawler as it uses my generated account as its proxy. The reason why a normal user account is able to be used is due to Twitter required a Twitter development account with special features to run these processes. Only issue of using my account is security as this deals with real-time data so if a search request was made but the search was related to violence such as guns then the Twitter authorities may ban my proxy account rendering the use of my account as unusable. As a result, I will try and choose specific users who will act sensibly in their search query and clearly understand the constraints.
Only able to input the English alphabet	This means people abroad cannot input their language of choice into the search function. This may mean the usage of this product may be limited due to its lack of adaptability depending on the language inputted. This would mean an alphabet would have to be inputted for every language to show equality for all users and Twitter itself may not accept some of these options. As the REST API is a 3rd party software, it may also not accept or recognise some of the inputs even if we added a feature to input in different languages.	Those who speak English and input English are fine although, there is some adaptability involved in that, the user can input any location around the world and it will still find tweets about the subject near that area therefore, it can still be used abroad if needed.

1.6 Hardware and Software Requirements Analysis

Requirements	Hardware/ Software	Justify
Desktop Computer/ Laptop	Hardware	Without a computer, the user would be unable to run the program as it has currently not been adapted into an app which would be able to be run on a handheld device Minimum Specification requirements: 16GB of RAM 64-bit Operating System Windows 10 20H2 Intel® Core™ i7-550U CPU 2.4 Ghz speed 447GB PNY CS900 480GB SSD (SATA (SSD)) 2 Cores Requires a screen resolution of at least 1707x960 pixels Monitor Width: 3840mm Monitor Height: 2160mm
Keyboard	Hardware	Without the keyboard, the user cannot make any inputs especially since the program has not been adapted to an app therefore, there is no touch screen input. If the user cannot input anything, they cannot interact with the program so they cannot detail what they want to crawl Twitter for.
MySQL	Software	Needs to run MySQL to have the “TwitterCrawler” Database and the tables inside them which for example store the 100+ query words inside them and without MySQL then the tables cannot be run to store user data. If they cannot store user data then the process of going through hundreds of words each time the program is run will continue instead of the program becoming more efficient by storing responses.
Windows 10	Software	Windows is required as it, as a type of OS, is needed to manage the computer’s memory (allocation) and processes while the program is running. Also, process scheduling may be required to switch from MySQL to Python when running and the OS manages and is responsible for this.

Requirements	Hardware/ Software	Justify
Python	Software	Without the IDE to run the code, the code cannot function. If I release the source code openly, and the user wishes to make modifications then they require Python to edit the code as it is written in the syntax of Python.
Spreadsheet software	Software	The results of the search are stored in a csv file which Windows decides to use a spreadsheet software (e.g. Microsoft Excel) as a way to view the file. This allows the user to see more of the tweet which was previously cut off when run in Python and it also allows them to copy the hyperlink of the tweet they would like to see further of onto their browser.
Web Browser	Software	On the spreadsheet, there is a hyperlink provided with every tweet – if the user would like to find out more about the tweet then the user requires a web browser to post the link onto and access Twitter to find out more.

2 Design

2.1 First Iteration

2.1.1 What is my first iteration?

My first iteration is to get a query from the user and scrape Twitter for results linking to the query from the user and output these results to the user.

2.1.2 Decomposition of the problem

Steps needed:

(i) What is the user's word? How do I get the user's word? How does the user enter their word?
(ii) How do I gain access to Twitter?
(iii) How do I scrape Twitter for tweets?
(iv) How do I store these results?
(v) How do I sort the data? What takes priority when two tweets say the same thing?
(vi) How do I output the tweets? How do I make it easy to read and understand the results?

(i) What is the user's word? How do I get the user's word? How does the user enter their word?

(a) Need to inform the user they need to input something
(b) Need to provide a space for the user to input something
(c) Need to store user's input once inputted

PSEUDOCODE of 1st iteration:

Part 1:

OUTPUT Please input your query word:
queryWord = USERINPUT

(ii) How do I gain access to Twitter?

(a) Need to access Twitter
(b) Need to make a Twitter Development account to gain access to Twitter REST API
(c) Need to get personal credentials from Twitter to allow me to gain access to data
(d) Need to link Python to the Twitter REST API using credentials as login

PSEUDOCODE of 1st iteration:

Part 2:

```
Import TwitterRESTAPI
Credentials = {}
Credentials['Credential1'] = xxxxxxx
Credentials['Credential2'] = yyyyyyy
Login = TwitterRESTAPI(Credentials[Credential1],Credentials[Credential2])
```

(iii) How do I scrape Twitter for tweets?

(a) Need to input user's word into search function

(b) Need to specify number of options to look for

(c) Need to specify language (being English)

PSEUDOCODE:

Part 3:

```
Procedure searchTwitter(query: STRING):
    Search = {}
    Search['userQuery'] = query
    Search['no.of.times'] = 10
    Search['language'] = 'English'
endProcedure
```

(iv) How do I store these results?

(a) Need to separate metadata associated with tweet using a concept similar to a 2d array but with a dictionary instead

(b) Separate twitter handle

(c) Separate date of tweet sent

(d) Separate the content of the tweet

(e) Separate the number of likes

(f) Loop through all the data and separate into individual columns

PSEUDOCODE:

Part 4:

```
Metadata = {'username' = {},'dateTweetSent' = {},'contentOfTweet' = {},'number_of_
likes = {}}
For i in length(Search['no.of.times']):
    Metadata{'username'} = TwitterRESTAPI(USERNAME[i])
    Metadata{'dateTweetSent'} = TwitterRESTAPI(DATE[i])
    Metadata{'contentOfTweet'} = TwitterRESTAPI(TEXT[i])
    Metadata{'number_of_likes'} = TwitterRESTAPI(LIKES[i])
ENDFOR
```

(v) How do I sort the data? What takes priority when two tweets say the same thing?

(a) Sort by number of likes to see how popular the tweet is.

Helps out my stakeholder Albert to see what is popular

PSEUDOCODE:

Part 5:

```
For I in range(length(Metadata)):
    If Metadata{'number_of_likes}[i+1]> Metadata{'number_of_likes'}[i]:
        Metadata{'number_of_likes}[i+1] = Metadata{'number_of_likes'}[i]
    EndIF
ENDFOR
```

(vi) How do I output the tweets? How do I make it easy to read and understand the results?

(a) Output the Metadata dictionary

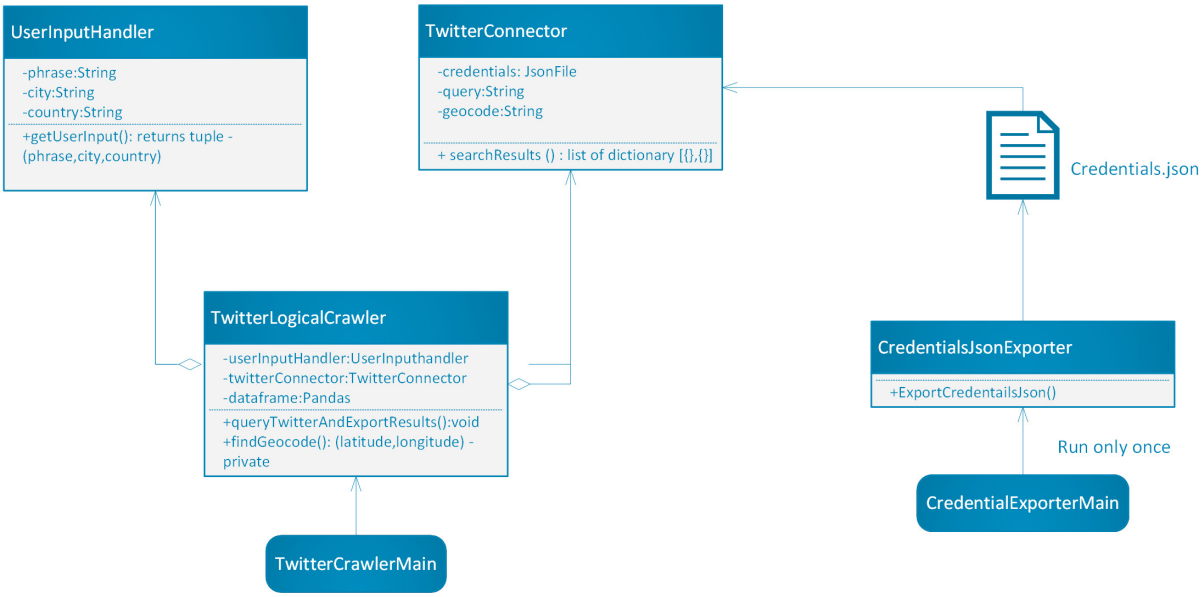
(b) Output onto CSV file

PSEUDOCODE:

Part 6:

```
OUTPUT Metadata
myFile = openWrite(Metadata)
myFile.close()
```

2.1.3 Class Diagram



2.1.4 Implementation of Class Diagram – Primitive

```
from twython import Twython
import pandas as pd

credentials = {}
credentials['CONSUMER_KEY'] = "R8HHEUKe8lWKrziDlZoAJKgs1"
credentials['CONSUMER_SECRET'] = "RBi1sNxWV1tUgGF7YLAN1i3TNUtVzsepSI4aikbHxpmLDA4vxK"
credentials['ACCESS_TOKEN'] = "1439642834425065475-9cMVsrbwCLgbeUWFdmiVY8EuEzBEaL"
credentials['ACCESS_SECRET'] = "kJIYwJf6Z1Xi5Q7OWhLXnd30mjNJGtpWO9VyB3fkQPa4J"

def searchTwitter(interest):
    #with open("twitter_credentials.json", "r") as file:
        #creds = json.load(file)
    python_tweets = Twython(credentials['CONSUMER_KEY'], credentials['CONSUMER_SECRET'])
    query = {'q': interest,
             #'result': 'popular',
            'count': 10,
            'lang': 'en',
            }
    dict_ = {'user': [], 'date': [], 'text': [], 'favorite_count': []}
    for status in python_tweets.search(**query)['statuses']:
        dict_['user'].append(status['user']['screen_name'])
        dict_['date'].append(status['created_at'])
        dict_['text'].append(status['text'])
        dict_['favorite_count'].append(status['favorite_count'])
    df = pd.DataFrame(dict_)
    df.sort_values(by='favorite_count', inplace=True, ascending=False)
    print(df.head(5))
    df.to_csv(r'TwitterSearchResults_A-levelComputerProject1.csv', index = False)

interest = input("Please enter your search word ")
while len(interest)>100 or len(interest)<1:
    interest = input("Please re-enter your search word ")
searchTwitter(interest)
```

2.1.5 Development of Source Code

Source Code

```
from twython import Twython
import pandas as pd
```

Justify

The libraries I used in the first iteration were twython and pandas. Twython is the most widely-used Twitter library for Python and it allows me to connect to the Twitter API and perform searches on real-time data – both of which are integral to my project. I also used pandas which manipulates the data into easily understandable tables to not only help me but the user as well. If I did not use pandas to sort out the output, it would be in a dictionary format which is very difficult to understand.

Source Code

```
#These credentuals are needed to access my Twitter development account
#Specifically the REST API
credentials = {}
credentials['CONSUMER_KEY'] = "R8HHEUKe8lWKrziDlZoAJKgsl"
credentials['CONSUMER_SECRET'] = "RBi1sNxWV1tUgGF7YLAN1i3TNUtVzsepSI4aikbHxpmLDA4vxK"
credentials['ACCESS_TOKEN'] = "1439642834425065475-9cMVsrbwCLgbeUWFdmiVY8EuEzBEaL"
credentials['ACCESS_SECRET'] = "kJIYwJf6Z1Xi5Q7OWhLXnd30mjNJGtpWO9VyB3fkQPa4J"
```

Justify

These credentials are the most important part into letting me use the Twitter API since they are specific to each user and require the owner of these credentials(me) to have been approved by Twitter themselves. They let me through the various security checks and allow me access to the real-time data in Twitter API databases.

Source Code

```
def searchTwitter(interest):
#Using subroutines to better identify my code.
#Easier to understand when each part is decomposed separately.
    query = '' + interest
    python_tweets = Twython(credentials['CONSUMER_KEY'], credentials['CONSUMER_SECRET'])
    #Accesses the dictionary and inputs these into Twython
    query = {'q': query,
             #'result': 'popular',
             #This would've sorted the most popular tweets but that overrides the date function.
             #I want the latest most relevant tweets not the most famous ones
            'count': 10,
            'lang': 'en',
            }
            #Searches for the query
    dict_ = {'user': [], 'date': [], 'text': [], 'favorite_count': []}
    for status in python_tweets.search(**query)['statuses']:
        dict_['user'].append(status['user']['screen_name'])
        dict_['date'].append(status['created_at'])
        dict_['text'].append(status['text'])
        dict_['favorite_count'].append(status['favorite_count'])
    outputLayout = pd.DataFrame(dict_)
    outputLayout.sort_values(by='favorite_count', inplace=True, ascending=False)
    #Sorts the values by how recent they were uploaded then those are sorted into how popular they were
    print(outputLayout.head(5))
    #Only outputs five to not congest the output screen
    outputLayout.to_csv(r'TwitterSearchResults_A-levelComputerProject.csv', index = False)
    #Converts it into a csv file which can be accessed by the user using spreadsheet software
    #For example if they wanted to know more about the user or the comments of the tweet.
```

Justify

This subroutine searches Twitter depending on the user's query.
It gains access to the API using Twython and my credentials for its security checks. The for loop searches the API database and when it matches the query then the separate parts of the twitter data is appended into the separate arrays within the dictionary. I.e. the user's Twitter Handle would be stored in user or the date of when the tweet was first sent out would be stored within date.

Once the for loop has 10 entries it stops and the dictionary is then sorted into a panda data structure to allow for easier viewing as everything is sorted into its separate columns. E.g. User handles all in one, Date all in one, the tweets all in one and the number of likes all in one column. It is then sorted by number of likes but before that it is sorted by how recent it was uploaded. Then currently, I have only set 5 to be outputted since I didn't want to congest the screen and I wanted to make sure those five were outputted correctly. Once that is done, I can change the number outputted to as many as I want. Only problem is, if it is too high, then it may take too long or if there isn't a high enough number of tweets matching the query word then it would be an empty table which would not look good when outputting to the user. Eventually, it is outputted into a spreadsheet file if the user would like to find out more (e.g. Victoria using the hyperlink attached to find more materials).

Source Code

```
interest = input("Please enter your search word ")
searchTwitter(interest)
#Since it was structured in a subroutine, the main program is easy to understand.
#Also quite free, not clogged up with lots of information.
```

Justify

Interest is the user input or the query word which is being searched.
The subroutine is being called into the main program. As I sorted the first iteration, into a subroutine, the main program is much clearer and easier to interpret for anyone looking at my code. More importantly, it is easier for me to make any changes as I know what each subroutine does depending on the subroutine name given. It is also easier for testing as I can test the subroutines separately(in other iterations for example) to better identify any testing errors rather than testing the whole program each time, which takes up more time and energy.

Source Code

```
while len(interest)>100 or len(interest)<1:
    interest = input("Please re-enter your search word ")
```

Justify

Here I have used some validation. I have used a WHILE loop to continuously ask the user to re-enter their user input until they meet the criteria of being 1 character of more and being less than 100 characters. I have also edited the user-input message so the user can understand what they have to do, instead of believing that the program made an error (“re-enter” as opposed to simply “enter”)

2.1.6 Errors in Source Code

I am running this in Jupyter Notebooks IDE.

Error

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-6-f73c9330cfbf> in <module>
     32 while len(interest)>100 or len(interest)<1:
     33     interest = input("Please re-enter your search word ")
---> 34 searchTwitter(interest)

<ipython-input-6-f73c9330cfbf> in searchTwitter(interest)
     19             }
     20     dict_ = {'user': [], 'date': [], 'text': [], 'favorite_count': []}
---> 21     for status in python_tweets.search(query)['statuses']:
     22         dict_['user'].append(status['user']['screen_name'])
     23         dict_['date'].append(status['created_at'])

TypeError: search() takes 1 positional argument but 2 were given
```

Remedial action

After doing some research the solution to this, is to use **query instead since ** allows dictionaries to be unpacked into another dictionary, therefore in this case, by doing **query, the query dictionary can be unpacked by the search function.

Error

```
TwythonError                              Traceback (most recent call last)
<ipython-input-9-63f4a059cbf5> in <module>
     30
     31 interest = input("Please enter your search word ")
---> 32 searchTwitter(interest)
     33 while len(interest)>100 or len(interest)<1:
     34     interest = input("Please re-enter your search word ")
```

Remedial action

This occurred since the subroutine was called before the validation took place so the input can still be changed but it will not have an effect on the program since it is not being applied anywhere.

Solution: Change the order with the validation coming before the subroutine is called.

Error

```
<ipython-input-9-63f4a059cbf5> in searchTwitter(interest)
     19             }
     20     dict_ = {'user': [], 'date': [], 'text': [], 'favorite_count': []}
---> 21     for status in python_tweets.search(**query)['statuses']:
     22         dict_['user'].append(status['user']['screen_name'])
     23         dict_['date'].append(status['created_at'])

~\anaconda3\lib\site-packages\twython\endpoints.py in search(self, **params)
    291
    292         """
--> 293         return self.get('search/tweets', params=params)
    294     search.iter_mode = 'id'
    295     search.iter_key = 'statuses'

~\anaconda3\lib\site-packages\twython\api.py in get(self, endpoint, params, version)
    277     def get(self, endpoint, params=None, version='1.1'):
    278         """Shortcut for GET requests via :class:`request`"""
--> 279         return self.request(endpoint, params=params, version=version)
    280
    281     def post(self, endpoint, params=None, version='1.1', json_encoded=False):

~\anaconda3\lib\site-packages\twython\api.py in request(self, endpoint, method, params, version, json_encoded)
    270             url = '%s/%s.json' % (self.api_url % version, endpoint)
    271
--> 272         content = self._request(url, method=method, params=params,
    273                                 api_call=url, json_encoded=json_encoded)
    274

~\anaconda3\lib\site-packages\twython\api.py in _request(self, url, method, params, api_call, json_encoded)
    200                 ExceptionType = TwythonAuthError
    201
--> 202             raise ExceptionType(
    203                 error_message,
    204                 error_code=response.status_code,

TwythonError: Twitter API returned a 400 (Bad Request), Query parameters are missing.
```

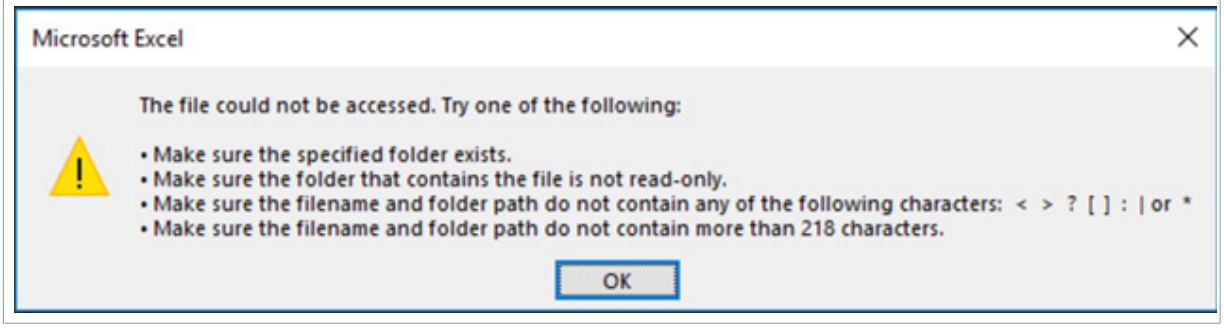
Remedial action

Twython, the library I am using to handle any commands to the Twitter REST API, returned a 400 error meaning that our request could not be understood by the API.

This is due to the fact that Python reads the source code sequentially – this means it reads from the top to the bottom line by line unless any methods are called.

Solution: I need to rearrange my validation and place the validation section before I manually call the subroutine so it can properly recognise the query parameters.

Error



Remedial action
This problem rendered me unable to save the csv file in which the output would be printed to. At first, I changed the file name that I was saving it to making it very short, using Microsoft Word's word count to count the number of characters to make sure in no way the file name exceeds 218 characters. Then I checked the filename to check if it contained any of those forbidden characters. It didn't. The folder was not read-only since I could open other files from the folder and it worked, and allowed me to save it. **Solution:** Then I realised the problem was that the file path was too long with the folder name being way too long. I do this to give me a small brief on what the folder contains without having to check every file but this time, I had to shorten the name. Once I had done that, the file had saved perfectly.

2.1.7 Stakeholder Feedback

John:

- Liked the fact that the twitter crawling search function worked
  - That it could filter out his chosen phrase and find users that had tweeted the phrase he had searched
    - E.g. Network
  - This meant that he can find those who have tweeted for help in network support.
  - As no location was set, he can find tweets from all across the globe asking for network support
    - He wants a location limiter. So, he can find customers in a local area since he found it difficult to help people online on specific networking problems.
      - My plan for this is using longitude and latitude to limit the distance the user can search for.
      - E.g. If the longitude and latitude in a point in London was inputted by the user, then if I set a 50-mile radius – it would only search for tweets within 50 miles of that location in London.
- Likes the fact there is a form of validation such as when inputting to prevent the user from inputting erroneous data stopping the program being run.
- Wants a neater presentation; when it was outputted, some columns were disjointed making it look aesthetically displeasing.

Overall, John is happy with the current prototype but he wants his suggestions implemented as soon as possible so he can properly use the program for his work. Right now, he has to make his phrases a little more specific to try and manually filter out tweets from other countries.

Victoria:

— Liked the fact that the timing of when the tweets were published are shown
  - This is because she can find out if the materials are outdated or not
  - This is very important as the specification for teaching her students changes every year so if the time was not showing, then she may be accidentally teaching her students material that is currently not relevant

— She wants an option where she can search for the tweet directly on her web browser
  - My plan for this is to export all the data onto a csv file and then Twitter(by default) attaches the hyperlink to each tweet so she can copy and paste that onto her web browser.

— Wants to recommend this to her students, but doesn't like the current primitive output which is difficult to read.
  - However, likes the simplicity of just inputting what is necessary such as the actual phrase itself and nothing else.

— Dislikes the fact that when other languages are inputted, it just returns an empty dataset – wants there to be a message indicating her students what is happening if they enter a phrase from a foreign language.

— Wants the program to be personalised to the students – e.g. by asking their name and referring to them by name for each input.
  - To reduce the daunting feel by the program for new students.

Overall, she is alright with the current prototype. She wants more personalisation and more clarity in certain situations as she is unable to discern when the program is running or when it is waiting to be run due to the lack of a proper output when there are no search results. She also wants to be able to find the exact tweet that she is interested in but I believe my CSV file situation will deal with her problem quite easily, however it does require some cooperation from her since she has to save both files in the same exact place.

Albert:

— Likes the fact that date and number of likes are shown on the output so he can the latest and most popular news on his interests.
  - Allowing him to find common interests which he knows others like as well.

— Wants to easily interact with the program
  - Does not want to see the code – finds it overwhelming
  - He believes that he may accidentally delete something and ruin the program.

Albert is happy with the current program. He does want a user interface(that he can easily interact with using buttons and bars to drag along) since he finds the back-end code overwhelming to look at since he doesn't understand much of it. He is worried that if he accidentally deletes something, the whole program will keep returning errors.

2.1.8 Source Code Improvements

```
from twython import Twython
import json
import pandas
class UserInputHandler():
    def __init__(self,phrase):
        self.__phrase = phrase
    def getUserInput(self):
        return self.__phrase
class TwitterConnector():
    # Enter your keys/secrets as strings in the following fields
    credentials = {}
    credentials['CONSUMER_KEY'] = "R8HHEUKe8lWKrziDlZoAJKgsl"
    credentials['CONSUMER_SECRET'] = "RBi1sNxWV1tUgGF7YLAN1i3TNUtVzsepSI4aikbHxpmLDA4vxK"
    credentials['ACCESS_TOKEN'] = "1439642834425065475-9cMVsrbwCLgbeUWFdmiVY8EuEzBEaL"
    credentials['ACCESS_SECRET'] = "kJIYwJf6Z1Xi5Q7OWhLXnd30mjNJGtpWO9VyB3fkQPa4J"

    # Save the credentials object to file
    with open("twitter_credentials_first.json", "w") as file:
        json.dump(credentials, file)

class TwitterLogicalCrawler(UserInputHandler(),TwitterConnector()):
    def __init__(self,phrase):
        self.__phrase = phrase
    def queryTwitterAndExportResults():
        with open("twitter_credentials.json", "r") as file:
            creds = json.load(file)
        python_tweets = Twython(creds['CONSUMER_KEY'], creds['CONSUMER_SECRET'])
        query = {'q': self.__phrase,
             #'result': 'popular',
            'count': 10,
            'lang': 'en',
            }
        dict_ = {'user': [], 'date': [], 'text': [], 'favorite_count': []}
        for status in python_tweets.search(**query)['statuses']:
            dict_['user'].append(status['user']['screen_name'])
            dict_['date'].append(status['created_at'])
            dict_['text'].append(status['text'])
            dict_['favorite_count'].append(status['favorite_count'])
        df = pd.DataFrame(dict_)
        df.sort_values(by='favorite_count', inplace=True, ascending=False)
        print(df.head(5))
    twitterConnector = twitterConnector()
    userInputHandler = UserInputHandler("python")
```

Source Code

```
from twython import Twython
import json
import pandas
```

Justification

In the improved version, I have also chosen to import json in so I can write this onto a file and read and write from it. This is so, if the data is accidentally edited on the program, the credentials will not be affected as they are written onto the file when the program is running.

Twython is used since it is the most widely-used library in Python for the Twitter API and allows me to connect to the Twitter REST API to perform accurate searches on real-time data which is integral to my project.

I decided to use pandas, after seeing the original output which look unorganised and very difficult to understand. Pandas manipulates the data in the dictionary into columns which is sorted in a table. This applies visualisation making it easier to interpret as it is in a visually pleasing structure.

My plan is eventually to have the different classes on different files and import them onto the main file. This is because, it makes it easier to debug as we can see exactly which file has the problem and quicker to fix.

Source Code

```
class UserInputHandler():
    def __init__(self,phrase):
        self.__phrase = phrase
    def getUserInput(self):
        return self.__phrase
```

Justification

This class's only purpose is to deal with the user's input(specifically, phrase). The user inputs it and it is initialised as the variable phrase. The getters provide public access to the private variable. It allows the main program to carry out functions using phrase. Furthermore, I have used decomposition; I have broken up the problem of the first iteration, into different classes and dealt with them individually.

Source Code

```
class TwitterConnector():
    # Enter your keys/secrets as strings in the following fields
    credentials = {}
    credentials['CONSUMER_KEY'] = "R8HHEUKe8lWKrziDlZoAJKgsl"
    credentials['CONSUMER_SECRET'] = "RBi1sNxWV1tUgGF7YLAN1i3TNUtVzsepSI4aikbHxpmLDA4vxK"
    credentials['ACCESS_TOKEN'] = "1439642834425065475-9cMVsrbwCLgbeUWFdmiVY8EuEzBEaL"
    credentials['ACCESS_SECRET'] = "kJIYwJf6Z1Xi5Q7OWhLXnd30mjNJGtpWO9VyB3fkQPa4J"

    # Save the credentials object to file
    with open("twitter_credentials_first.json", "w") as file:
        json.dump(credentials, file)
```

Justification
This is another class. This class is responsible for writing the credentials into a separate text file. This uses decomposition, and here I have broken up the file writing into different stages to easily solve them.

Source Code

```
class TwitterLogicalCrawler(UserInputHandler(),TwitterConnector()):
    def __init__(self,phrase):
        self.__phrase = phrase
    def queryTwitterAndExportResults():
        with open("twitter_credentials.json", "r") as file:
            creds = json.load(file)
        python_tweets = Twython(creds['CONSUMER_KEY'], creds['CONSUMER_SECRET'])
        query = {'q': self.__phrase,
             #'result': 'popular',
            'count': 10,
            'lang': 'en',
            }
        dict_ = {'user': [], 'date': [], 'text': [], 'favorite_count': []}
        for status in python_tweets.search(**query)['statuses']:
            dict_['user'].append(status['user']['screen_name'])
            dict_['date'].append(status['created_at'])
            dict_['text'].append(status['text'])
            dict_['favorite_count'].append(status['favorite_count'])
        df = pd.DataFrame(dict_)
        df.sort_values(by='favorite_count', inplace=True, ascending=False)
        print(df.head(5))
    twitterConnector = twitterConnector()
    userInputHandler = UserInputHandler("python")
```

Justification
This uses aggregation between UserInputHandler and TwitterConnector, meaning that the TwitterLogicalCrawler cannot exist without both UserInputHandler and TwitterConnector but not vice versa. The method queryTwitterAndExportResults, does the searching through the Twitter REST API database for the user's query word and outputs it in a panda dataframe.

2.1.9 Example of Validation

```
Please enter your search word
Please re-enter your search word
Please re-enter your search word oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
Please re-enter your search word good
             user                            date  \
0   katrinachalie  Sun Dec 05 08:48:15 +0000 2021
1      LiatKolink  Sun Dec 05 08:48:15 +0000 2021
2     haaayomidee  Sun Dec 05 08:48:15 +0000 2021
3       macc1_3_7  Sun Dec 05 08:48:15 +0000 2021
4   PrettyFlyyBoss  Sun Dec 05 08:48:15 +0000 2021

                                              text  favorite_count
0  @BoneHoffman Do what makes you happiest, it's ...              0
1  @christi91823779 @CNNPR Okay, the jacking off ...              0
2  RT @mafia10114: I'm really sad people underrat...              0
3  RT @XMaximist: $SYS looks good 🔥 respecting th...              0
4  RT @AriTheDon: When you compliment, kiss, hug,...              0
```

2.1.10 Example of Excel File Output

user	date	text	favorite_count
hwanggeur	Sun Dec 05	@nikzz_08 I've been on it since years and no one gave two shits about it.	0
E15Riot	Sun Dec 05	@Strangerwyd @cllzxicuh @shade1x @vizzion1x_ New chapter new you, you go next.	0
gulf_4ever	Sun Dec 05	RT @Yuuki25442990: My fam here, please stream this Now still 1M, go to 1,1M or 2MðŸ¯,	0
GLOWING_	Sun Dec 05	=C15 GIRL GO SWERVE RN HAHAHA ITS OVER YOU ARE DONE	0
ElaineAizat	Sun Dec 05	RT @ROSEVotingTeam: We are back again on Top 10!	0
moonchild	Sun Dec 05	RT @_jiminbangtan: @btsvotingorg Year 2017 is one of the worst year for us, MAMA deleted 1M+ votes from BTS categories; accused us as a boâ	0
UKUSEDM	Sun Dec 05	RT @AcholaGoodheart: I miss the days i was a go between couples in hi xul,even read love letters before the owner did	0
SaintNi506	Sun Dec 05	@GowTolson Patel Kamlesh should go back to Pakistan and take his racist Pakistani cricketers with him none of themâ€¦ https://t.co/RkLKajQAvn	0
Vinojai6	Sun Dec 05	RT @StrDeejay: #Maanaadu Satelite (@vijaytelevision)and Digital rights (@SonyLIV) sold out for 19Crores	0
Ey7287691	Sun Dec 05	RT @chizaruai: Go to search	0

2.1.11 Test Scenarios

Test Scenario 1: Normal Data

- Switch on your computer
  - o This will load you up to your desktop
  - o Look on the bottom left corner
  - o There is an icon which looks like a folder
  - o Double click that icon
  - o This is how the library icon should look 
  - o Then once it is open, look to the left of your screen
  - o There should be a small folder called downloads with a small blue arrow pointing down
  - o Left click that folder
  - o It should show you lots of files
  - o Find the file called 'TwitterCrawler_FirstIteration_SearchTwitterForQueryWord'
  - o Right click this file
  - o It should pop up with a tiny menu
  - o Move your cursor to Open
- Run the program
  - o If using Jupyter Notebooks, press the button run at the top of the page
  - o If using Python IDE, press Fn(if on a laptop, otherwise no need to press Fn) F5 together to run the program

- Enter your search word in this bar
  - o Enter any word that you like
  - o "Football" is an example of a word you can enter
- Test results will be printed
  - o It should look like this:

```
Please enter your search word test
             user                          date  \
0  UrduWriterDotPK  Sun Dec 05 08:44:42 +0000 2021
1  JennyStrathearn  Sun Dec 05 08:44:42 +0000 2021
2      CallMeKobby  Sun Dec 05 08:44:41 +0000 2021
3  SharadM60272623  Sun Dec 05 08:44:41 +0000 2021
4       MyranYBCMM  Sun Dec 05 08:44:41 +0000 2021

                                                text  favorite_count
0  RT @FarmDiary_: 🏳o😳💻FARM DIARY team has alr...               0
1  RT @belfastcc: In the run-up to Christmas, mak...              0
2  @jjamarboye_GH @QuistDzifa @Nii_Ayitey__ @joe_...              0
3  RT @imVkohli: Back to Test cricket ❤️ https://...              0
4  @syberswan Nah cause Islamic tests are a diffe...              0
```

- Make a note of outputs
  - o Specifically, the actual tweet
  - o Twitter handle and Date of tweet are not required
  - o Any error messages
  - o Any other outputs such as “Empty Dataset”
    - » You can do this by writing it on paper
    - » Another way is to press Fn(if on a laptop, otherwise no need to press Fn) on the bottom left of your keyboard and while holding that press PrtScr/Print Screen on the top right of your keyboard
    - » Then stop holding those two buttons
    - » Now look on the bottom of your screen for an icon that says W and looks like a file
    - » Left click that
    - » Press New
    - » Right click and paste your image
    - » Repeat this for every test run

Test Scenario 2: Erroneous Data

- Switch on your computer
  - o This will load you up to your desktop
  - o Look on the bottom left corner
  - o There is an icon which looks like a folder
  - o Double click that icon
  - o This is how the library icon should look 
  - o Then once it is open, look to the left of your screen
  - o There should be a small folder called downloads with a small blue arrow pointing down
  - o Left click that folder
  - o It should show you lots of files
  - o Find the file called ‘TwitterCrawler_FirstIteration_SearchTwitterForQueryWord’
  - o Right click this file
  - o It should pop up with a tiny menu
  - o Move your cursor to Open

- Run the program
  - o If using Jupyter Notebooks, press the button run at the top of the page
  - o If using Python IDE, press Fn F5 together to run the program

- Enter your search word in this bar
  - o Enter any phrase
  - o E.g. 1234

- Test results will be printed
  - It should look like this

```
Please enter your search word test
             user                         date  \
0  UrduWriterDotPK  Sun Dec 05 08:44:42 +0000 2021
1  JennyStrathearn  Sun Dec 05 08:44:42 +0000 2021
2      CallMeKobby  Sun Dec 05 08:44:41 +0000 2021
3  SharadM60272623  Sun Dec 05 08:44:41 +0000 2021
4       MyranYBCMM  Sun Dec 05 08:44:41 +0000 2021

                                                text  favorite_count
0  RT @FarmDiary_: 🏴‍☠️😳💻FARM DIARY team has alr...               0
1  RT @belfastcc: In the run-up to Christmas, mak...               0
2  @jjamarboye_GH @QuistDzifa @Nii_Ayitey__ @joe_...               0
3  RT @imVkohli: Back to Test cricket ❤️ https://...               0
4  @syberswan Nah cause Islamic tests are a diffe...               0
```

  - Make a note of outputs
    - » Specifically the actual tweet
    - » Twitter handle and Date of tweet are not required
  - Any error messages
    - » E.g. DataError
  - Any other outputs such as “Empty Dataset”
  - You can do this by writing it on paper
  - Another way is to press Fn on the bottom left of your keyboard and while holding that press PrtScr on the top right of your keyboard
  - Then stop holding those two buttons
  - Now look on the bottom of your screen for an icon that says W and looks like a file
  - Left click that
  - Press New
  - Right click and paste your image
  - Repeat this for every test run

Test Scenario 3: Boundary Data

- Switch on your computer
  - This will load you up to your desktop
  - Look on the bottom left corner
  - There is an icon which looks like a folder
  - Double click that icon
  - This is how the library icon should look 
  - Then once it is open, look to the left of your screen
  - There should be a small folder called downloads with a small blue arrow pointing down
  - Left click that folder
  - It should show you lots of files
  - Find the file called ‘TwitterCrawler_FirstIteration_SearchTwitterForQueryWord’
  - Right click this file

- It should pop up with a tiny menu
- Move your cursor to Open

- Run the program
  - If using Jupyter Notebooks, press the button run at the top of the page
  - If using Python IDE, press Fn F5 together to run the program

- Enter your search word in this bar
  - Enter any phrase
  - E.g. “The quick brown fox jumps over the dog who gets to eat treats for dinner which the fox did not like.”
- Test results will be printed
  - It should look like this:

```
Please enter your search word test
             user                          date  \
0  UrduWriterDotPK  Sun Dec 05 08:44:42 +0000 2021
1  JennyStrathearn  Sun Dec 05 08:44:42 +0000 2021
2      CallMeKobby  Sun Dec 05 08:44:41 +0000 2021
3  SharadM60272623  Sun Dec 05 08:44:41 +0000 2021
4       MyranYBCMM  Sun Dec 05 08:44:41 +0000 2021

                                                text  favorite_count
0  RT @FarmDiary_: 🏳️‍🌈😳💻FARM DIARY team has alr...               0
1  RT @belfastcc: In the run-up to Christmas, mak...               0
2  @jjamarboye_GH @QuistDzifa @Nii_Ayitey__ @joe_...               0
3  RT @imVkohli: Back to Test cricket ❤️ https://...               0
4  @syberswan Nah cause Islamic tests are a diffe...               0
```

  - Make a note of outputs
    - » Specifically the actual tweet
    - » Twitter handle and Date of tweet are not required
  - Any error messages
    - » E.g. DataError
  - Any other outputs such as “Empty Dataset”
  - You can do this by writing it on paper
  - Another way is to press Fn on the bottom left of your keyboard and while holding that press PrtScr on the top right of your keyboard
  - Then stop holding those two buttons
  - Now look on the bottom of your screen for an icon that says W and looks like a file
  - Left click that
  - Press New
  - Right click and paste your image
  - Repeat this for every test run

Test Scenario 4: Erroneous Data

- Switch on your computer
  - o This will load you up to your desktop
  - o Look on the bottom left corner
  - o There is an icon which looks like a folder
  - o Double click that icon
  - o This is how the library icon should look
  - o Then once it is open, look to the left of your screen
  - o There should be a small folder called downloads with a small blue arrow pointing down
  - o Left click that folder
  - o It should show you lots of files
  - o Find the file called 'TwitterCrawler_FirstIteration_SearchTwitterForQueryWord'
  - o Right click this file
  - o It should pop up with a tiny menu
  - o Move your cursor to Open

- Run the program
  - o If using Jupyter Notebooks, press the button run at the top of the page
  - o If using Python IDE, press Fn F5 together to run the program

- Enter your search word in this bar
  - o Enter any phrase
  - o E.g. "巧克力"
    - » For this input, you are required to change your keyboard settings to another language.
    - » A faster method is using a translator which is searched for on your web browser and in this case choosing Mandarin and English.
      - * Then put chocolate into English and copy the Mandarin and paste it into your input box.

- Test results will be printed
  - o It should look like this:

```
Please enter your search word test
             user                             date  \
0  UrduWriterDotPK  Sun Dec 05 08:44:42 +0000 2021
1  JennyStrathearn  Sun Dec 05 08:44:42 +0000 2021
2      CallMeKobby  Sun Dec 05 08:44:41 +0000 2021
3  SharadM60272623  Sun Dec 05 08:44:41 +0000 2021
4       MyranYBCMM  Sun Dec 05 08:44:41 +0000 2021

                                                text  favorite_count
0  RT @FarmDiary_: 🏳c👴💻FARM DIARY team has alr...               0
1  RT @belfastcc: In the run-up to Christmas, mak...              0
2  @jjamarboye_GH @QuistDzifa @Nii_Ayitey__ @joe_...              0
3  RT @imVkohli: Back to Test cricket ❤️ https://...              0
4  @syberswan Nah cause Islamic tests are a diffe...              0
```

- Make a note of outputs
  - Specifically the actual tweet
  - Twitter handle and Date of tweet are not required
- Any error messages
  - E.g. DataError
- Any other outputs such as “Empty Dataset”
- You can do this by writing it on paper
- Another way is to press Fn on the bottom left of your keyboard and while holding that press PrtScr on the top right of your keyboard
- Then stop holding those two buttons
- Now look on the bottom of your screen for an icon that says W and looks like a file
- Left click that
- Press New
- Right click and paste your image
- Repeat this for every test run

2.1.12 Test Cases

NOTE: As this scrapes real life data for its tests, the expected results are always given as examples of outputs.

Test plan:

ID	Testing	Type	Data	Expected
1	Phrase variable is searched correctly	Normal	phrase = ‘python’	E.g. “How to learn Python online?” – a random tweet about Python should be outputted
2	Phrase variable is on the boundary- lower limit: 1 character	Boundary	phrase = ‘a’	E.g. “a donkey” – a tweet containing ‘a’ by itself inside it
3	Phrase variable is on the boundary – upper limit: 100 characters	Boundary	Phrase = ‘The quick brown fox jumps over the dog who gets to eat treats for dinner which the fox did not like.’	E.g. “The quick brown fox jumps over the dog who gets to eat treats for dinner which the fox did not like. is a good story” – a tweet containing the whole string inside it

ID	Testing	Type	Data	Expected
4	Phrase variable is invalid - integers	Erroneous	Phrase = '1234'	No output as it is not a string with letters
5	Phrase variable is invalid – different language	Erroneous	Phrase = "巧克力" - which means chocolate in English	Error as the syntax the crawler recognises is in English.

ID	Testing	Type	Data	Expected	Actual
1	Phrase variable is searched correctly	Normal	phrase = 'python'	E.g. "How to learn Python online?" – a random tweet about Python should be outputted	"RT @tymwol: One of my favourite Python programs". – This passes as it is a tweet that contains the user's query word inside **Passes**
2	Phrase variable is on the boundary-lower limit: 1 character	Boundary	phrase = 'a'	E.g. "a donkey" – a tweet containing 'a' by itself inside it	"RT @Harry_Styles: There comes a time when a bl…" – This passes as 'a' is separate in the user's tweet rather than being inside a word as it is a letter **Passes**
3	Phrase variable is on the boundary – upper limit: 100 characters	Boundary	Phrase = 'The quick brown fox jumps over the dog who gets to eat treats for dinner which the fox did not like.'	E.g. "The quick brown fox jumps over the dog who gets to eat treats for dinner which the fox did not like. is a good story" – a tweet containing the whole string inside it	No output – this presumably means there are no tweets containing that string. This fails the test as there is no proper output **Fails** I created a twitter account to manually test it and it had passed the test on the second time. **Remedial action**

ID	Testing	Type	Data	Expected	Actual
4	Phrase variable is invalid - integers	Erroneous	Phrase = '1234'	No output as it is not a string with letters	"RT @wesyang: The truth is that Chappele is mo…" – since the output statement cuts off the tweet, there is no way to check if this passed or failed the test. **Remedial action**
5	Phrase variable is invalid – different language	Erroneous	Phrase = "巧克力" - which means chocolate in English	Error as the syntax the crawler recognises is in English.	"" – no output, empty table as the syntax is not recognised as the crawler was developed to recognise only the English syntax. **Passes**

2.1.13 Evaluation - First Iteration

First of all, I had achieved the goals of my first iteration being: to create a Twitter account and gain access to the Twitter REST API and being able to filter out user input through the Twitter REST API to output tweets.

Using my stakeholders, we had tested different inputs but instead of already setting up the test cases, I had asked my stakeholders to manually enter in the test cases, to increase their familiarity with the program. We tested normal data cases, erroneous data cases and boundary data cases.

I had implemented visualisation into the output to try and let the users see the output in the clearest fashion currently. I did this using a library called pandas which sorts out the raw dictionary into a more understandable format. First, before I imported pandas, the data was outputted in a hard-to-understand format. However, in terms of formatting the work is not finished since it is not lined up which to some users as John pointed out, looks aesthetically displeasing.

In terms of my success criteria, I had achieved number 1 – the program loads without error, 2.1 – there is an input box for the user to enter in their personalised input,4 – I am able to use my account with authorisation from Twitter itself, 4.1 – I can connect to the Twitter REST API, 4.2 – I can connect the Twitter REST API to Python, allowing me to do searches using the user's input, 6.2 – importing Twython which allows me to do Twitter related tasks on Python, 6.4 – importing Pandas to make the output interpretable, 8.1,8.3,8.4 and 8.5 – these all contribute to the output(such as the date, Twitter handle, number of likes and most importantly the actual tweet).

What is required of me next, is to provide validation to prevent users causing the program to crash and provide them the opportunity to reinput a word if they enter a wrong input. Also to maintain security, I should have a list of banned words since this is all going through the account I created so I need to make sure that the users do not misuse that privilege since Twitter may ban the use of that account if a dangerous input was given(such as guns). I also need to provide an option for the user to enter their longitude and latitude or make the program automatically track it so then the user can find relevant tweets in a specified area, meeting John's requirement. The reason for using longitude and latitude is that it is more specific than if the location name was asked by the user and there are multiple places with the same name than the program may be confused and may have to resort to backtracking until it finds the correct location. Only problem with that is it uses more CPU resources and takes up more memory which may affect the user's experience with any other programs being fun. Not only that, it also increases the time taken for the output to be shown to the user, especially if it was a location with many possibilities like San Jose which has approximately 1,716 different places with the same name.

2.2 Second Iteration

2.2.1 What is my second iteration?

I want to create a usable AI system and link it to my main program.

Why?

The AI would benefit the user by providing them the most optimal verbs for the word they input. Thus, decreasing time spent for the correct verb in their own context.

2.2.2 Decomposition of the problem

Steps needed:

Step 1

- How to create a database?
- How do I know what tables to put in the database?
- What are the primary keys?
- What are the secondary keys?
- What are the tables called?

Justification:
I am using a database so I can store vast amounts of records from the user's inputs very easily and efficiently. Furthermore, they are designed to facilitate the modification and retrieval of data so I can change the number of times each word is chosen every time a user inputs it.

Step 2

- How do I create an AI system?
- Why am I trying to create an AI system?

Justification:
This is to improve **usability** for the user specifically it provides each user a customised interface in which is created from their previous inputs. If two users enter the same exact information in terms of the word they enter then the AI will provide a **consistent** interface in terms of the questions it asks – it may also improve it depending on other users. So the AI, while being consistent, it gives a unique feedback to every user who uses it, having improved from the previous user.

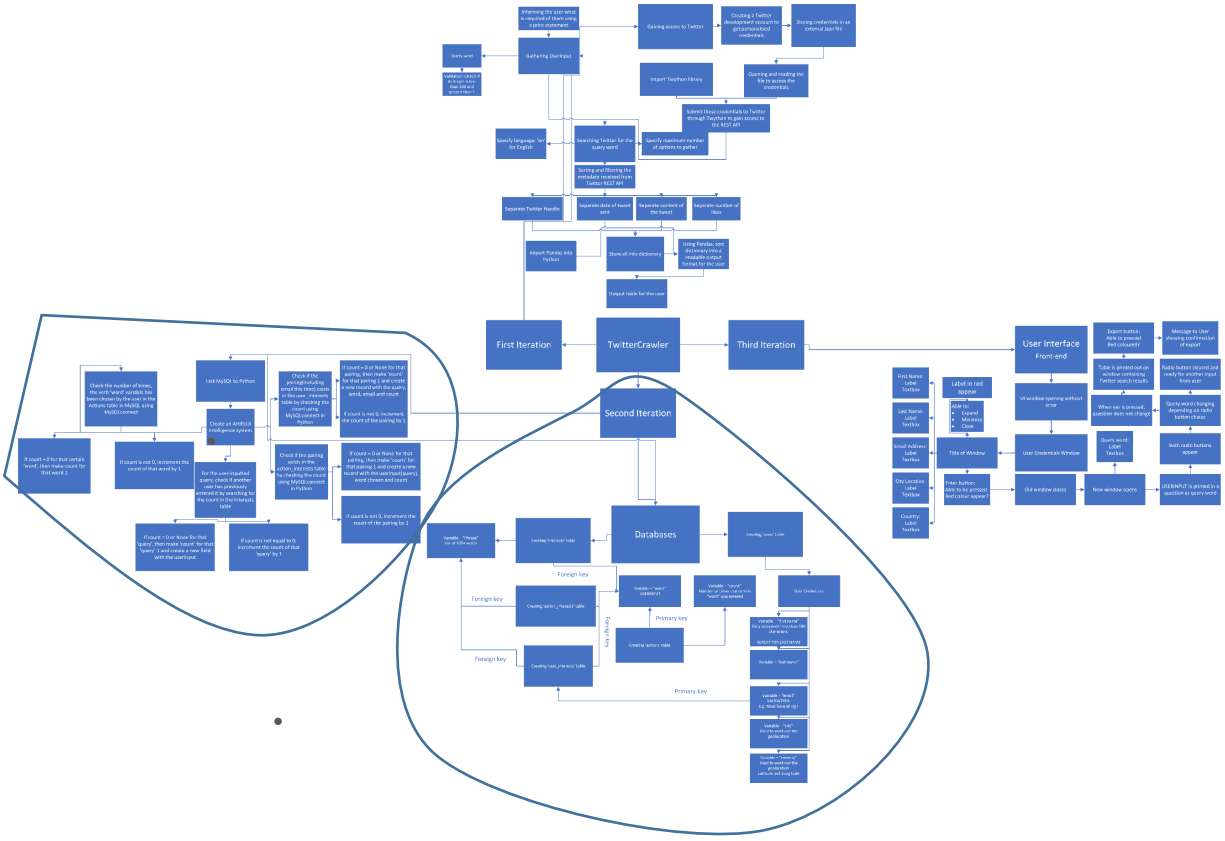
Step 3

- How do I make my Python inputs link to the inputs within the tables?
- How do I link MySQL to Python?

Justification:
By connecting MySQL to Python, I have a direct link with the user's inputs and the database so the user's inputs can now actually be stored within the table. This means the user's actions now, can directly affect the records within the table unlike before which helps the Artificial Intelligence area too. The reason, I am linking it after I have created the artificial intelligence is since I am applying computational thinking in the fact that I am using a scheduling algorithm called Longest Remaining Time First (LRTF) in my thinking. It is also easier to test each individual part rather than if they are linked while the AI is being created so each error may directly affect the table, and lead to more unwanted errors.

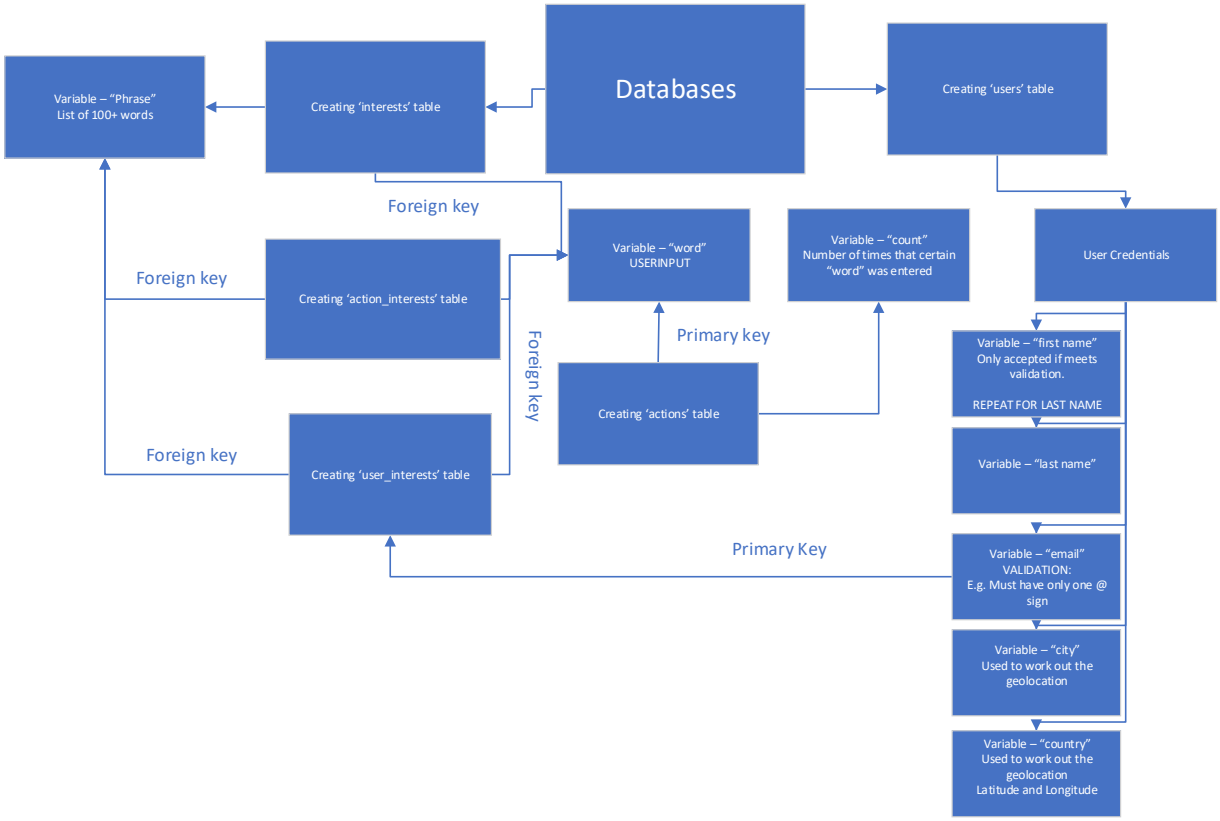
2.2.3 Inheritance Diagram

Whole Program:

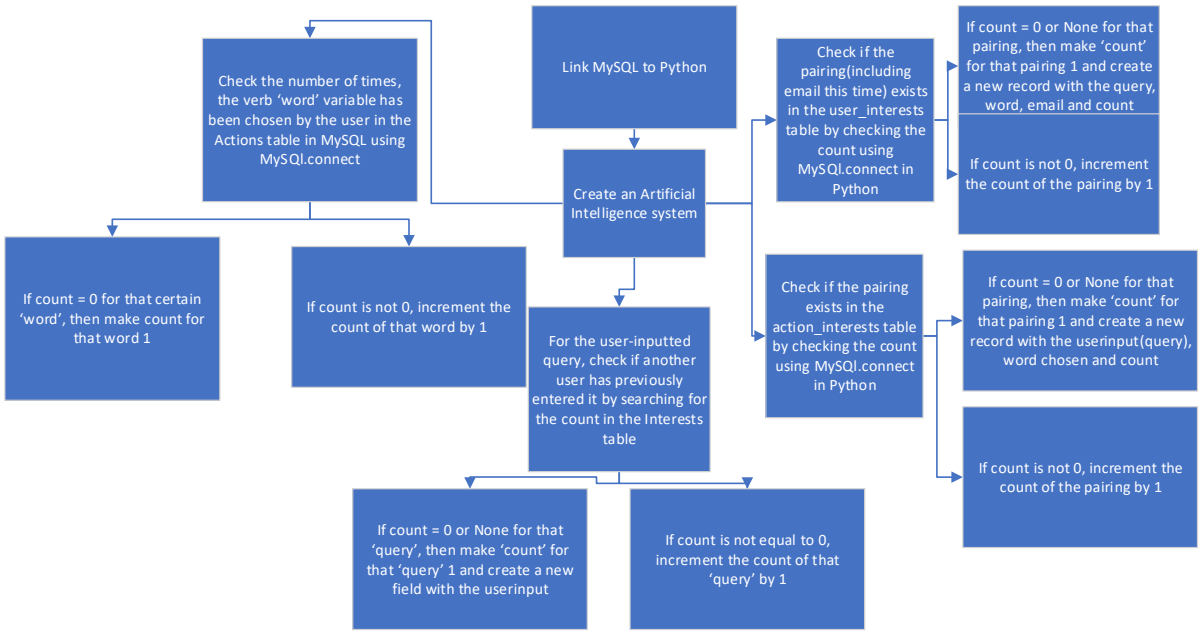


Second Iteration:

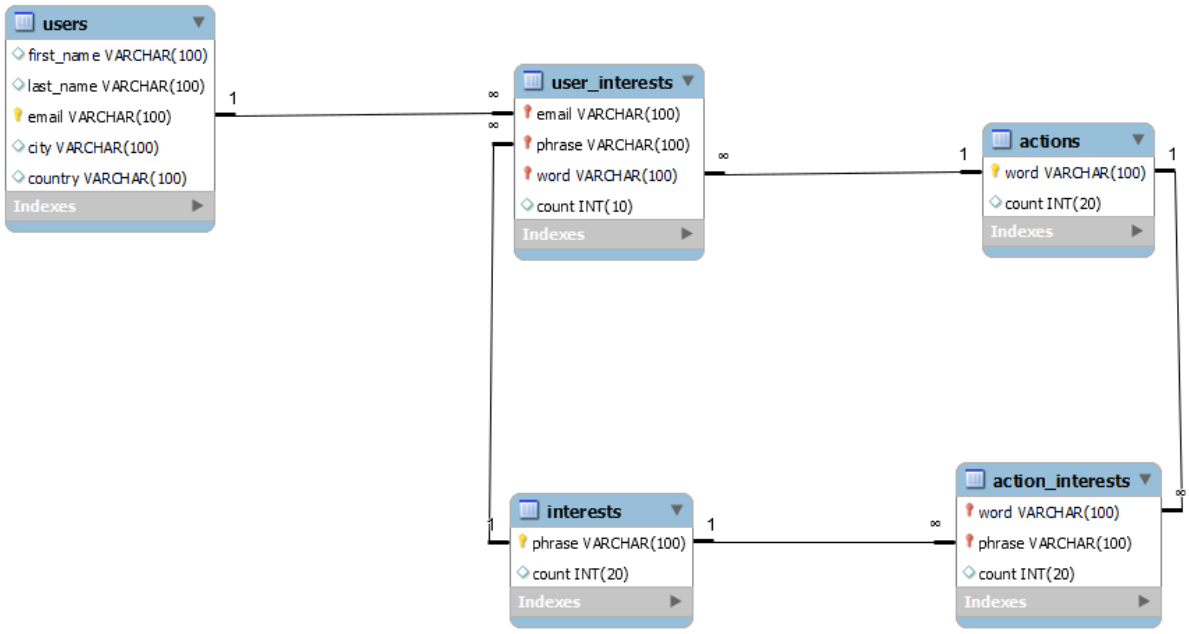
Databases:



Creating AI:



2.2.4 Entity Relationship Diagram



Steps in detail

Step 1

(a) Create an Entity-relationship diagram

(b) The tables I need:

- » One to store user's details like their name and email
- » One to store their userinput
- » One to store which word they choose and how many times that word has been chosen overall by all users
- » One to match the user to the word they entered in
- » One to match the pairing of the word they entered and the verb they chose

(c) Create a database – name it Twittercrawler to make it distinguishable

(d) Create the tables while making sure to clearly define the primary keys and headers in each table

(e) Input the verbs into the 'words' table

PSEUDOCODE: (in MySQL)

```
create database TwitterCrawler;

create table users(first_name varchar(100), last_name varchar(100), email
varchar(100), city varchar(100), country varchar(100), primary key (email));

create table actions(word varchar(100), count int(20));

create table interests(phrase varchar(100), count int(20));
```

create table action_interests(word varchar(100), phrase varchar(100), count int(20), constraint composite_word primary key(word, phrase), foreign key(word) references actions(word), foreign key(phrase) references interests(phrase));

create table user_interests(email varchar(100), phrase varchar(100), constraint email_phrase primary key(email, phrase), foreign key(email) references users(email), foreign key(phrase) references interests(phrase));

Step 2

(a) Create baseline Artificial Intelligence
- When a word is chosen
- Increment count by 1 in the relevant table(the one containing all the verbs)
- Pair it with the user input
- Check if this pairing exists in the table
- If word is entered again, highest count word is outputted first or if no pairing exists, create a new record in the table
  - Use a few sample words in an array at first
    - For testing to create a provisional design

PSEUDOCODE:

```
ArrayOfVerbs = [‘eat’,’go’,’buy’,’play’]
ArrayOfUserChoices = [[],[],[],[]]
Choice = USERINPUT
Count = 0
For i in range(length(ArrayOfVerbs)):
    OUTPUT(“Do you want to “,ArrayOfVerbs[i],choice)
    yesOrNo = USERINPUT
    if yesOrNo == ‘Y’:
            count = count + 1
            ArrayOfUserChoices.append(ArrayOfVerbs[i],choice,count)
    ENDIF
ENDFOR
For i in range(length(ArrayOfUserChoices):
    If choice == ArrayOfUserChoices[i][1]:
            OUTPUT ArrayOfUserChoices[i][0], choice
    ENDIF
ENDFOR
```

Step 3

(a) Link MySQL to Python using the .connect method in python
(b) Store user data in the users table
(c) Output all words, until user says yes
(d) Pair words with the most commonly chosen verb. It can be any word
(e) Pair user with the paired word and verb

PSEUDOCODE:

```
Twitter = Twittercrawler.connect()
firstName = USERINPUT
lastName = USERINPUT
email = USERINPUT
city = USERINPUT
country = USERINPUT
countEmail = 0
While countEmail == 0:
  Twitter.insert into Twittercrawler.users(firstName,lastName,email,city,country)
countEmail = countEmail + 1
yesOrNo = USERINPUT
words = [100 verbs…]
wordPlace = 0
while yesOrNo == “N”:
  wordPlace = wordPlace + 1
pairArray = []
individualUserArray = []
individualUserArray.append(words[wordPlace],choice)
pairArray.append(individualUserArray)
```

2.2.5 MySQL Justification

MySQL

```
mysql> insert into actions values("teach",0);
Query OK, 1 row affected (0.03 sec)

mysql> insert into actions values("learn",0);
Query OK, 1 row affected (0.00 sec)

mysql> insert into actions values("study",0);
Query OK, 1 row affected (0.00 sec)
```

Justification

I am adding the 100 verbs into the actions table. As I cannot just append an array into the table, I must, for each word, insert it manually into the table. Each time the “verb” would change but the count would remain the same.

The reason every term is added individually is because, each line is a separate record in the table.

Within MySQL, one thing I noticed is that copying and pasting does not conventionally work so the only way is to type it out for each one. Unlike Python, MySQL requires a semicolon at the end of each line so if this is not added at the end, it returns an error and the line must be retyped.

MySQL

```
mysql> select * from actions;
+--------------+-------+
| word         | count |
+--------------+-------+
| teach        |     0 |
| learn        |     0 |
| study        |     0 |
| play         |     0 |
| donate       |     0 |
| watch        |     0 |
| help         |     0 |
| availability |     0 |
| how          |     0 |
| free         |     0 |
| be           |     0 |
| have         |     0 |
| do           |     0 |
| say          |     0 |
| go           |     0 |
| get          |     0 |
| make         |     0 |
| know         |     0 |
| think        |     0 |
| take         |     0 |
| see          |     0 |
| come         |     0 |
| want         |     0 |
| look         |     0 |
| use          |     0 |
| find         |     0 |
| give         |     0 |
| tell         |     0 |
| work         |     0 |
| try          |     0 |
| ask          |     0 |
| need         |     0 |
| feel         |     0 |
| become       |     0 |
| leave        |     0 |
| put          |     0 |
| mean         |     0 |
| keep         |     0 |
```

Justification

"Select * from actions" gathers all the data inputted into the table of actions and outputs it. This is the end product after adding 100 different 'actions' into the table with all count of zero. This uses visualisation since it is easier to understand the scope of the words I have added with it being sorted in a table.

MySQL

```
mysql> use twittercrawler;
Database changed
```

Justification

This is the database we are using. This contains all the tables. I have called it Twittercrawler so if anyone is looking over, then they can understand exactly what the purpose of this database is.

MySQL

```
mysql> create table Actions(word varchar(100), count int(20));
Query OK, 0 rows affected (0.06 sec)
```

Justification

Here I have created the table Actions where the words all go in and here the variable word which contains all the actions and the count is set to zero is created. In the Actions table, I have set the character limit for each word that the user enters to 100 and the count to a 20-digit number. The 20-digit number may be excessive but it does not use up 20 digit number of bits in memory, but only up to the count variable so in terms of memory this is not an issue.

MySQL

```
mysql> show tables;
+--------------------------+
| Tables_in_twittercrawler |
+--------------------------+
| actions                  |
+--------------------------+
1 row in set (0.00 sec)
```

Justification

This shows the table has been created within another table. The outer table is the actual database since all the tables are stored within the database called Twittercrawler.

This shows the benefits of a database, where it shows the name of the database as the header name with all the tables inside it eventually which uses **visualisation** – which makes it not only easier to interpret but also understand instead of using multi-dimensional arrays in Python which increase time taken when running which is a **usability** issue, and leads to a negative user experience. Databases are far more efficient at storing records which means the program will be run more efficiently in regards to data handling and storage, hence why SQL is being used.

MySQL

```
mysql> create table Interets(phrase varchar(100), count int(20));
Query OK, 0 rows affected (0.06 sec)
```

Justification

Here I tried to create the table interests with the user's input and the variable count as the headers but I misspelt the title of the table. This was a human **error**. The solution is to completely delete the table and create a new table with a correctly spelt title.

MySQL

```
mysql> drop table Interets;
Query OK, 0 rows affected (0.00 sec)
```

Justification

"drop" deletes everything including the table itself. This means I can start anew with the Interests table.

MySQL

```
mysql> create table action_interests(word varchar(100), phrase varchar(100), count int(20), constraint composite_word primary key(word, phrase),
foreign key (word) references actions(word), foreign key (phrase) references Interests(phrase));
Query OK, 0 rows affected (0.04 sec)
```

Justification

Here I am creating a new table called action_interests which combines both tables: actions and interests. This is shown by phrase referencing the phrase in the table Interests and the foreign key 'word' referencing word from the table Actions.

MySQL

```
mysql> create table users(first_name varchar(100), last_name varchar(100), email varchar(100), city varchar(100), country varchar(100), primary key(email));
Query OK, 0 rows affected (0.07 sec)

mysql> create table user_interests(email varchar(100), phrase varchar(100),constraint email_phrase primary key(email, phrase),
foreign key (email) references users(email), foreign key (phrase) references interests(phrase));
Query OK, 0 rows affected (0.03 sec)
```

Justification

Here I have created both the table users and user_interests. The table users uses email as the primary key since many can have the same forename or lastname but email is unique to each person. User_interests combined both the users table and the interests table referencing the variable email from users and phrase from interests.

MySQL

```
mysql> show tables;
+--------------------------+
| Tables_in_twittercrawler |
+--------------------------+
| action_interests         |
| actions                  |
| interests                |
| user_interests           |
| users                    |
+--------------------------+
5 rows in set (0.00 sec)
```

Justification

This shows all the tables being created within the database.

This helps me **visualise** the structure of the database more easily and a database is optimised for data storage and handling such as searching for certain queries and storing them.

Justification: This helps create a more efficient running of the program, **optimising** storage and access time, forming a complete solution to the problem of storing the user's inputs in an optimal fashion.

2.2.6 Development of Source Code

Source Code

```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = ""
)
```

Justification

This connects MySQL to python. Host, user and passwd are all variables which are assigned the login details so Python has access to my MySQL server.

Structurally, this provides the basis for the link between the databases stored in MySQL and Python.

Source Code

```
cursor = db.cursor()
```

Justification

The .cursor() creates an instance of the cursor class which allows me to execute MySQL statements in Python, e.g. Select *.

Source Code

```
def insertUser(cursor,firstName,lastName,email,city,country):
    cursor.execute("select first_name from twittercrawler.users where email = %s",(email,))
    #Cannot increment as no count variable
    valueExists = cursor.fetchone()
    if valueExists is None:
        cursor.execute("insert into twittercrawler.users(first_name,last_name,email,city,country) values(%s,%s,%s,%s,%s)",
                       (firstName,lastName,email,city,country,))
    else:
        cursor.execute("update twittercrawler.users set first_name = %s , last_name = %s , city = %s , country = %s where email=%s",
                       (firstName,lastName,city,country,email,))
```

Justification

Email is the primary key in the User's table. It uses the email and the first name to check if the user already exists, and if so, updates their details in the user table. This makes sure the user's details are always up to date and that they can do multiple searches such as if they want to change the city and country to different locations. If the user does not exist, a new user is created with the details the user inputted. Therefore, a user profile is created within the database.

Source Code

```
def getAllWords(cursor,phrase):
    words = []
    wordsSet = set()
    cursor.execute("select word from twittercrawler.action_interests where phrase = %s order by count desc",(phrase,))
    actionInterestsCursor = cursor.fetchall()
    for dt in actionInterestsCursor:
        wordsSet.add(dt[0])
        words.append(dt[0])
     ## executing the statement using 'execute()' method
    cursor.execute("select * from twittercrawler.actions order by count desc")

    ## 'fetchall()' method fetches all the rows from the last executed statement
    data = cursor.fetchall()

    ## showing one by one database
    for dt in data:
        if dt[0] not in wordsSet:
            words.append(dt[0])
    return words
```

Justification

Here I have an empty array called words and an empty set called wordsSet. Using the user's phrase that they inputted, I check if there is any matching pair with the verbs within the action_interests table. fetchAll makes sure to get every single item that meets the criteria of having the same phrase paired up to it and then those verbs are added to both the set and the array. I then fetchAll again, fetching all the 100 verbs in the actions table in order of highest count. If this is not within the set already, it is appended to the array and the array is returned.

Here I am using enumeration checking if the word already exists within the set and after enough trials, it will be an exhaustive search of 100 words repeatedly each time to check if the verb is already present. Furthermore, with enough inputs, there may be a potential for 1000s of pairing with one verb so this will increase the run time for the user.

Source Code

```
def incrementWordFrequency(word,cursor):
    cursor.execute("select count from twittercrawler.actions where word = %s",(word,))
    count = cursor.fetchone()
    print(count)
    count = count[0] + 1
    cursor.execute("update twittercrawler.actions set count = %s where word = %s",(count,word,))
```

Justification

This exports the count of how many time that word has been chosen and adds one each time. Unlike fetchAll, this only selects one variable which is count in this case where the word = 'word chosen'. The new value is then updated to the table in MySQL.

Source Code

```
def incrementPhraseFrequency(phrase,cursor):
    cursor.execute("select count from twittercrawler.interests where phrase = %s",(phrase,))
    valueExists = cursor.fetchone()
    if valueExists is not None:
        count = valueExists[0]+1
        cursor.execute("update twittercrawler.interests set count = %s where phrase = %s",(count,phrase,))
    else:
        cursor.execute("insert into twittercrawler.interests(phrase,count) values(%s,1)",(phrase,))
```

Justification

This checks if the phrase exists by checking if there is a count for the phrase the user inputted. If there is no such phrase, then it is inserted into the table and given a count of 1. If it does exist, then the value of count is incremented by 1.

Source Code

```
def incrementPhraseWordFrequency(word,phrase,cursor):
    cursor.execute("select count from twittercrawler.action_interests where phrase = %s and word = %s",(phrase,word,))
    valueExists = cursor.fetchone()
    if valueExists is not None:
        count = valueExists[0]+1
        cursor.execute("update twittercrawler.action_interests set count = %s where phrase = %s and word = %s",(count,phrase,word,))
    else:
        cursor.execute("insert into twittercrawler.action_interests(word,phrase,count) values(%s,%s,1)",(word,phrase,))
```

Justification

This checks the actions_interests table. This table is basically where the pairing of the user's phrase and the verb is assigned. So I check if a count exists for this pairing, if it does, I increment by 1(add one), if not, I create a new pairing and assign it with a count of 1.

Source Code

```
def incrementPhraseWordEmailFrequency(word,phrase,email,cursor):
    cursor.execute("select count from twittercrawler.user_interests where email = %s and phrase = %s and word = %s",(email,phrase,word,))
    valueExists = cursor.fetchone()
    if valueExists is not None:
        count = valueExists[0]+1
        cursor.execute("update twittercrawler.user_interests set count = %s where email = %s and phrase = %s and word = %s",(count,email,phrase,word,)
    else:
        cursor.execute("insert into twittercrawler.user_interests(email,phrase,word,count) values(%s,%s,%s,1)",(email,phrase,word,))
```

Justification

This checks the user_interests table. This is where the specific user is assigned to their input and the verb they chose. I use the **primary key** email to distinguish each user.
I check if there is a count where the email, phrase and the verb match. If it does, the count variable is incremented by 1, if it does not exist, a new entry is created, with a count of 1.

Source Code

```
def captureMetricsToImproveAI(word,phrase,cursor,email):
    # step 1 - increment the count of the word by 1 in actions table - update query
    incrementWordFrequency(word,cursor)
    # step 2 - add the phrase in interests table and mark count = count +1 - insert/update  query
    incrementPhraseFrequency(phrase,cursor)
    # step 3 - add the combination of word and phrase to action_interests - insert/update
    incrementPhraseWordFrequency(word,phrase,cursor)
    # step 4 - add the combination of word and phrase and email to user_interests - insert/update
    incrementPhraseWordEmailFrequency(word,phrase,email,cursor)
```

Justification

This is the amalgamation of all the previous methods to create an AI. Each method is called within this subroutine to carry out their purpose and these are combined in sequential order. I have repeatedly used a count in each method, since it is easy, not only for me, but for anyone else reading my tables to visualise exactly how many times each part has been chosen.

Source Code

```
#Main part of the program
firstName = input("Please enter your first name: ")
lastName = input("Please enter your last name: ")
email = input("Please input your email address: ")
city = input("Please input your city that you would like to search: ")
country = input("Please input the country in which you would like to search: ")
insertUser(cursor,firstName,lastName,email,city,country)
#insert into twittercrawler.interests(phrase,count) values(%s,1)",(phrase,)
user_interest = input("Please enter your search: ")

words = getAllWords(cursor,user_interest)
```

Justification

This is the main part of the program for the 2nd iteration. This has all the user inputs I require to make these subroutines function accurately and correctly. For example, firstName, lastName, email, city and country are all parameters for the insertUser subroutine and user_interest in the parameter for the getAllWords subroutine.

2.2.7 Errors in Source Code

I am running this in Jupyter Notebooks IDE.

Error

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-5-efcfb2f08671> in <module>
    163         i=i-1
    164     elif user == "Y":
--> 165         queryAndReturnTwitterResult(words[i],user_interest,location)
    166         captureMetricsToImproveAI(words[i],user_interest,cursor,email)
    167         break;

<ipython-input-5-efcfb2f08671> in queryAndReturnTwitterResult(phrase, interest, location)
     18
     19     # Instantiate an object
---> 20     python_tweets = Twython(creds['CONSUMER_KEY'], creds['CONSUMER_SECRET'])
     21
     22     query = phrase + ' '+interest

NameError: name 'creds' is not defined
```

Remedial Action

This was due to the json file not being recognised since it was not imported correctly.

Remedial Solution:

The solution was to rename creds as credentials due to the name of the dictionary with all the keys to logging in being contained within there.

Error

```
    if valueExists is None:
    ^
SyntaxError: invalid syntax
```

Remedial Action

This error was due to a missing bracket. A tip I have learned is that python will indicate a syntax error on the lines below the actual error therefore, to find the error you have to look above the line highlighted by Python and I was correct.

Remedial Solution: Fill in the missing bracket on the line above for valueExists = cursor.fetchOne()

Error

```
mysql> select word from action_interests where phrase = python order by count;
ERROR 1054 (42S22): Unknown column 'python' in 'where clause'
```

Remedial Action

Here, the key terms were all correct but again another Syntax error with Python not having any speech marks to recognise it as a string(value).

Remedial Solution:

The solution was to re-enter the whole line but this time put “python” in speech marks.

Error

```
Please enter your first name: TestError1NoUI
Please enter your last name: TestError1NoUI
Please input your email address: TestError1NoUI@TestError1NoUI.com
Please input your city that you would like to search: London
Please input the country in which you would like to search: UK
Please enter your search: christmas
Do you want to eat christmas? Type Y or N
N
Do you want to learn christmas? Type Y or N
N
Do you want to play christmas? Type Y or N
N
Do you want to donate christmas? Type Y or N
N
Do you want to have christmas? Type Y or N
Y
```

Remedial Action

This is not a visible error but the values being inputted are not being stored within the tables. This is a problem since the user cannot be paired with their inputs.

Remedial Solution:

The solution was to change the == to is not None since 0 would mean there is already a value for count whereas None means there is nothing. No value at all. Furthermore, it was to keep the SQL server actively on while the program is running since it also does not register while the program is running in the background.

Error

```
mysql> drop table users;show tables;
ERROR 1051 (42S02): Unknown table 'twittercrawler.users'
```

Remedial Action

This was due to two instructions being run at the same time and drop table users being run twice, so it does not exist in the database but was being asked to be dropped once more.

Remedial solution:

Use drop table users once

Then on a separate line type show tables; then the error has gone.

2.2.8 Improvements in MySQL and Source Code

Improvements

```
mysql> create table Interests(phrase varchar(100), count int(10));
Query OK, 0 rows affected (0.00 sec)
```

Justification

The improvement here is shortening the limit of “count” to 10 digits instead of the 20 digits previously. This was changed to improve the **usability** of the program by making the program run more efficiently by reducing the number of empty bits.

Improvements

```
mysql> create table users(first_name varchar(100), last_name varchar(100), email varchar(100), postal_town varchar(100), country varchar(75), primary key(email));
Query OK, 0 rows affected (0.01 sec)

mysql> drop table users;
Query OK, 0 rows affected (0.00 sec)
```

Justification

The improvement here is that the header “city” has been replaced by “postal town” – this is because there was a flaw in the planned user interaction where some may feel as they are not being represented since they do not live in a city. (They may live in a town or a hamlet for example, so this has been adapted to include nearly all users).

However, this still had a problem. The character limit for the “postal town” was still too excessively large with the longest postal town only being 85 characters.

The **remedial solution** to this problem was to drop this whole table and create a new table.

Improvements

```
mysql> create table users(first_name varchar(100), last_name varchar(100), email varchar(100), postal_town varchar(90), country varchar(75), primary key(email));
Query OK, 0 rows affected (0.03 sec)

mysql> create table user_interests(email varchar(100), phrase varchar(100), constraint email_phrase primary key(email, phrase),
foreign key(email) references users(email), foreign key(phrase) references interests(phrase));
Query OK, 0 rows affected (0.03 sec)
```

Justification

The improvement here was the “postal town” was correctly reduced to a reasonable amount without containing too many empty bits.

Also, like previously discussed, this deals with a **usability** issue which was lack of representation in their input and this change leads to increased efficiency and increased subjective satisfaction since they now feel part of the program.

Improvements

```
postal_town = input("Please input your postal town that you would like to search: ")
country = input("Please input the country in which you would like to search: ")
while len(postal_town) <1 or len(postal_town)>90:
    postal_town = input("Please re-enter your postal town")
insertUser(cursor,firstName,lastName,email,postal_town,country)
#insert into twittercrawler.interests(phrase,count) values(%s,1)",(phrase,)
user_interest = input("Please enter your search: ")
while len(user_interest)>100 or len(user_interest)<1:
    interest = input("Please re-enter your search word ")
```

Justification

The improvement was implementing the change in header name in the actual program too and also the extra **validation**.

There is **validation** for postal town where it checks for anything within the limits of 1-90 since the shortest town name is 1 character long and the 90 character limit and will keep asking the user until they enter a string which fits the criteria. This, not only acts as **validation** but also aids in **usability** testing since it prevents more errors from being returned. By having this **validation**, it leads to increased efficiency and keeps a more fluid running of the whole program.

2.2.9 Source Code

```
#Second iteration start
import mysql.connector as mysql #2nd
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = ""
)
#Without .connect, none of the program could store data in the database
cursor = db.cursor()
## creating an instance of 'cursor' class which is used to execute the 'SQL' statements in Python

#These methods all check if a value exists, if not creates a new value with count 0 or update current value with
#incremented count
def insertUser(cursor,firstName,lastName,email,postal_town,country):
    cursor.execute("select first_name from twittercrawler.users where email = %s",(email,))
    #Cannot increment as no count variable
    valueExists = cursor.fetchone()
    if valueExists is not None:
        cursor.execute("insert into twittercrawler.users(first_name,last_name,email,city,country) values(%s,%s,%s,%s,%s)
                    (firstName,lastName,email,postal_town,country,))
    #Userinput stored into tables
    else:
        cursor.execute("update twittercrawler.users set first_name = %s , last_name = %s , city = %s ,
                    country = %s where email=%s",
                    (firstName,lastName,postal_town,country,email,))

def getAllWords(cursor,phrase):
    words = []
    wordsSet = set()
    cursor.execute("select word from twittercrawler.action_interests where phrase = %s order by count desc",(phrase,))
    actionInterestsCursor = cursor.fetchall()
    for dt in actionInterestsCursor:
        wordsSet.add(dt[0])
        words.append(dt[0])
     ## executing the statement using 'execute()' method
    cursor.execute("select * from twittercrawler.actions order by count desc")

    ## 'fetchall()' method fetches all the rows from the last executed statement
    data = cursor.fetchall()

    ## showing one by one database
    for dt in data:
        if dt[0] not in wordsSet:
            words.append(dt[0])
    return words
def incrementWordFrequency(word,cursor):
    cursor.execute("select count from twittercrawler.actions where word = %s",(word,))
    count = cursor.fetchone()
    print(count)
    count = count[0] + 1
    cursor.execute("update twittercrawler.actions set count = %s where word = %s",(count,word,))
```

```
def incrementPhraseWordFrequency(word,phrase,cursor):
    cursor.execute("select count from twittercrawler.action_interests where phrase = %s and word = %s",(phrase,word,))
    valueExists = cursor.fetchone()
    if valueExists is not None:
        #Checks if not empty
        count = valueExists[0]+1
        #Increments current count
        cursor.execute("update twittercrawler.action_interests set count = %s where phrase = %s and word = %s",
                    (count,phrase,word,))
    else:
        cursor.execute("insert into twittercrawler.action_interests(word,phrase,count) values(%s,%s,1)",(word,phrase,))


def incrementPhraseWordEmailFrequency(word,phrase,email,cursor):
    cursor.execute("select count from twittercrawler.user_interests where email = %s and phrase = %s and word = %s",
                (email,phrase,word,))
    valueExists = cursor.fetchone()
    if valueExists is not None:
        #Checks if not empty
        count = valueExists[0]+1
        #Increments count
        cursor.execute("update twittercrawler.user_interests set count = %s where email = %s and phrase = %s"
                    "and word = %s",(count,email,phrase,word,))
        #Updates count
    else:
        #Creates a new entry if it doesn't exist already
        cursor.execute("insert into twittercrawler.user_interests(email,phrase,word,count) values(%s,%s,%s,1)",
                    (email,phrase,word,))

def captureMetricsToImproveAI(word,phrase,cursor,email):
    # step 1 - increment the count of the word by 1 in actions table - update query
    incrementWordFrequency(word,cursor)
    # step 2 - add the phrase in interests table and mark count = count +1 - insert/update  query
    incrementPhraseFrequency(phrase,cursor)
    # step 3 - add the combination of word and phrase to action_interests - insert/update
    incrementPhraseWordFrequency(word,phrase,cursor)
    # step 4 - add the combination of word and phrase and email to user_interests - insert/update
    incrementPhraseWordEmailFrequency(word,phrase,email,cursor)
```

2.2.10 Output Screen Example

```
Please enter your first name: Karrison
Please enter your last name: T
Please input your email address: KarrisonT@mail.com
Please input your postal town that you would like to search: London
Please input the country in which you would like to search: UK
Please enter your search: chocolate
Do you want to eat chocolate? Type Y or N
Y
```

Above, it is printed one line at a time and once the user enters in the question asked on each line, and when enter is pressed, the next line pops up and then that will be filled by the user and this continues until the choice of “Yes” and “No” and this time due to the artificial intelligence, “eat chocolate” – the choice I wanted was chosen first.

```
| Karrison  | T                    | KarrisonT@mail.com                | London    | UK            |
```

Here, this shows a snippet of the users’ table where this user’s details have been cropped out to show the connection from the inputs in Python to the inputs into the tables in MySQL.

```
eat     | chocolate      |     1
```

Here, this shows the pairing of “eat” and “chocolate” with the amount of times, that specific pairing has been chosen by users which is once, therefore meaning this input cycle was the first time this pairing was chosen.

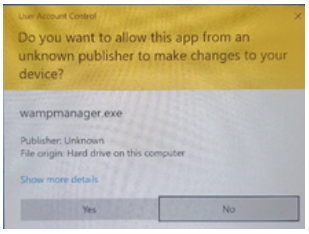
2.2.11 Test Scenarios

Test Scenario 1: Normal Data

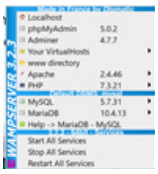
- Switch on your computer
  - o This will load you up to your desktop
  - o Look on the bottom left corner
  - o There is an icon which looks like a folder
  - o Double click that icon
  - o This is how the library icon should look 
  - o Then once it is open, look to the left of your screen
  - o There should be a small folder called downloads with a small blue arrow pointing down
  - o Left click that folder
  - o It should show you lots of files
  - o Find the file called
  - o ‘TwitterCrawler_SecondIteration_CreateAI’
  - o Right click this file
  - o It should pop up with a tiny menu
  - o Move your cursor to Open

- Now before running the program, we need to open mySQL
  - So press the taskbar on the bottom

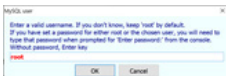
  - Press the outermost key so in this case it is the WINDOWS key
  - Then press the keys on your keyboard to type Wampserver64
  - It will then pop up with a window like this:



  - Then click yes. This will start up the virtual server 
  - On the right hand side of your taskbar, this should pop up in green, if it is another colour, please wait a little while longer until it turns green
  - Left-click this icon



  - This window will pop up, carry your cursor to MySQL then to MySQL console



  - This will pop up, press ok
  - The final step is to type in
    - » use Twittercrawler;
    - » Then press enter
    - » It should reply with a message saying database changed
    - » Now we are finished with MySQL
    - » Keep it open and go back to the original file

- Now we have to run the program
  - If using Jupyter Notebooks, press the button run at the top of the page
  - If using Python IDE, press Fn(if on a laptop, otherwise no need to press Fn) F5 together to run the program

```
Please enter your first name:
```

- Enter in your first name then press enter
- It will ask for your last name then press enter
- Enter in your email next
- Then the city you would like to search in
  - E.g. London
- Then the country

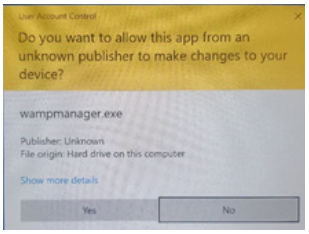
- If I did London, then the country would be the UK
- Now enter in your search word and just do a regular search
  - E.g. enter play as your word
  - Keep typing N until ‘play football’ is suggested
  - Then wait for the results
- Now go back to the MySQL console and type in
  - Select * from actions where word = ‘play’
  - Take a note of the result
    - » You can do this by writing it on paper
    - » Another way is to press Fn(if on a laptop, otherwise no need to press Fn) on the bottom left of your keyboard and while holding that press PrtScr/Print Screen on the top right of your keyboard
      - * Then stop holding those two buttons
      - * Now look on the bottom of your screen for an icon that says W and looks like a file
      - * Left click that
      - * Press New
      - * Right click and paste your image
      - * Repeat this for every test run

Test Scenario 2: Erroneous Data

- Switch on your computer
  - This will load you up to your desktop
  - Look on the bottom left corner
  - There is an icon which looks like a folder
  - Double click that icon
  - This is how the library icon should look 
  - Then once it is open, look to the left of your screen
  - There should be a small folder called downloads with a small blue arrow pointing down
  - Left click that folder
  - It should show you lots of files
  - Find the file called ‘TwitterCrawler_SecondIteration_CreateAI’
  - Right click this file
  - It should pop up with a tiny menu
  - Move your cursor to Open

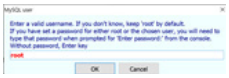
- Now before running the program, we need to open mySQL
  - So press the taskbar on the bottom

- Press the outermost key so in this case it is the WINDOWS key
- Then press the keys on your keyboard to type Wampserver64
- It will then pop up with a window like this:



- Then click yes. This will start up the virtual server 
- On the right hand side of your taskbar, this should pop up in green, if it is another colour, please wait a little while longer until it turns green
- Left-click this icon

- This window will pop up, carry your cursor to MySQL then to MySQL console



- This will pop up, press ok
- The final step is to type in
  - » use Twittercrawler;
  - » Then press enter
  - » It should reply with a message saying database changed
  - » Now we are finished with MySQL
  - » Keep it open and go back to the original file

- Now we have to run the program
  - If using Jupyter Notebooks, press the button run at the top of the page
  - If using Python IDE, press Fn(if on a laptop, otherwise no need to press Fn) F5 together to run the program

```
Please enter your first name:
```

- Enter in your first name then press enter
- It will ask for your last name then press enter
- Enter in your email next
- Then the city you would like to search in
  - E.g. London
- Then the country
  - If I did London, then the country would be the UK
- Now enter in your search word and just do a regular search
  - E.g. enter football as your word
  - Keep typing YN together

- Then wait for the results
- Now go back to the MySQL console and type in
  - Select * from actions where word = ‘play’
  - Take a note of the result
    - You can do this by writing it on paper
    - Another way is to press Fn(if on a laptop, otherwise no need to press Fn) on the bottom left of your keyboard and while holding that press PrtScr/Print Screen on the top right of your keyboard
      - Then stop holding those two buttons
      - Now look on the bottom of your screen for an icon that says W and looks like a file
      - Left click that
      - Press New
      - Right click and paste your image
- Repeat this for every test run

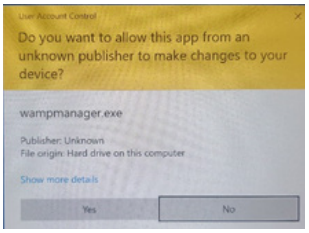
Test Scenario 3: Boundary Data

- Switch on your computer
  - This will load you up to your desktop
  - Look on the bottom left corner
  - There is an icon which looks like a folder
  - Double click that icon
  - This is how the library icon should look 
  - Then once it is open, look to the left of your screen
  - There should be a small folder called downloads with a small blue arrow pointing down
  - Left click that folder
  - It should show you lots of files
  - Find the file called ‘TwitterCrawler_SecondIteration_CreateAI’
  - Right click this file
  - It should pop up with a tiny menu
  - Move your cursor to Open

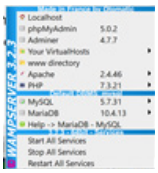
- Now before running the program, we need to open mySQL
  - So press the taskbar on the bottom

  - Press the outermost key so in this case it is the WINDOWS key
  - Then press the keys on your keyboard to type Wampserver64

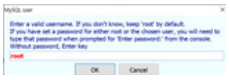
- It will then pop up with a window like this:



- Then click yes. This will start up the virtual server 
- On the right hand side of your taskbar, this should pop up in green, if it is another colour, please wait a little while longer until it turns green
- Left-click this icon



- This window will pop up, carry your cursor to MySQL then to MySQL console



- This will pop up, press ok
- The final step is to type in
  - » use Twittercrawler;
  - » Then press enter
  - » It should reply with a message saying database changed
  - » Now we are finished with MySQL
  - » Keep it open and go back to the original file

- Now we have to run the program
  - If using Jupyter Notebooks, press the button run at the top of the page
  - If using Python IDE, press Fn(if on a laptop, otherwise no need to press Fn) F5 together to run the program

```
Please enter your first name:
```

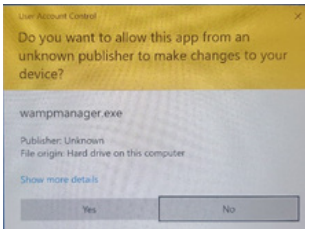
- Enter in your first name then press enter
- It will ask for your last name then press enter
- Enter in your email next
- Then the city you would like to search in
  - E.g. London
- Then the country
  - If I did London, then the country would be the UK
- Now enter in your search word and just do a regular search
  - E.g. enter football as your word
  - Keep typing N every single time
  - Then wait for the results

- Now go back to the MySQL console and type in
  - Select * from user_interests where email = “(your email that you entered)”;
  - Take a note of the result
    - » You can do this by writing it on paper
    - » Another way is to press Fn(if on a laptop, otherwise no need to press Fn) on the bottom left of your keyboard and while holding that press PrtScr/Print Screen on the top right of your keyboard
      - * Then stop holding those two buttons
      - * Now look on the bottom of your screen for an icon that says W and looks like a file
      - * Left click that
      - * Press New
      - * Right click and paste your image
- Repeat this for every test run

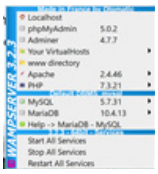
Test Scenario 4: Erroneous Data

- Switch on your computer
  - This will load you up to your desktop
  - Look on the bottom left corner
  - There is an icon which looks like a folder
  - Double click that icon
  - This is how the library icon should look 
  - Then once it is open, look to the left of your screen
  - There should be a small folder called downloads with a small blue arrow pointing down
  - Left click that folder
  - It should show you lots of files
  - Find the file called ‘TwitterCrawler_SecondIteration_CreateAI’
  - Right click this file
  - It should pop up with a tiny menu
  - Move your cursor to Open
- Now before running the program, we need to open mySQL
  - So press the taskbar on the bottom 
  - Press the outermost key so in this case it is the WINDOWS key
  - Then press the keys on your keyboard to type Wampserver64

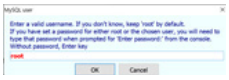
- It will then pop up with a window like this:



- Then click yes. This will start up the virtual server 
- On the right hand side of your taskbar, this should pop up in green, if it is another colour, please wait a little while longer until it turns green
- Left-click this icon



- This window will pop up, carry your cursor to MySQL then to MySQL console



- This will pop up, press ok
- The final step is to type in
  - » use Twittercrawler;
  - » Then press enter
  - » It should reply with a message saying database changed
  - » Now we are finished with MySQL
  - » Keep it open and go back to the original file

- Now we have to run the program
  - If using Jupyter Notebooks, press the button run at the top of the page
  - If using Python IDE, press Fn(if on a laptop, otherwise no need to press Fn) F5 together to run the program

```
Please enter your first name:
```

- Enter in a first name of 101 characters then press enter
- It will ask for your last name then press enter
- Enter in your email next
- Then the city you would like to search in
  - E.g. London
- Then the country
  - If I did London, then the country would be the UK
- Now enter in your search word and just do a regular search
  - E.g. enter football as your word
  - Keep typing YN together
  - Then wait for the results

- Now go back to the MySQL console and type in
  - Select * from users where firstName = '101 character name'
  - Take a note of the result
    - » You can do this by writing it on paper
    - » Another way is to press Fn(if on a laptop, otherwise no need to press Fn) on the bottom left of your keyboard and while holding that press PrtScr/Print Screen on the top right of your keyboard
      - * Then stop holding those two buttons
      - * Now look on the bottom of your screen for an icon that says W and looks like a file
      - * Left click that
      - * Press New
      - * Right click and paste your image
- Repeat this for every test run

2.2.12 Test Cases

ID	Testing	Type	Data	Expected
1	firstName is stored correctly	Normal	firstName = "Karrison"	E.g. when in MySQL, select firstName from users where firstName = "Karrison", should output a table with "Karrison" as the output
2	firstName is stored on the boundary – lower limit: 1 character	Boundary	firstName = "K"	E.g. when in MySQL, select firstName from users where firstName = "K", should output a table with "K" as the output
3	firstName is stored on the boundary – upper limit: 100 characters	Boundary	firstName = "KKKKKKKKKKKKK KKKKKKKKKKKKKKKKKK KKKKKKKKKKKKKKKKKK KKKKKKKKKKKKKKKKKK KKKKKKKKKKKKKK"	E.g. when in MySQL, select firstName from users where firstName = "KKKKKKKKKKKKK KKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK KKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK KKKKKKKKK", should output a table with "KKKKKKKKK KKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK KKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK KKKKKKKKKKKKKK" as the output
4	firstName is invalid – 101 characters	Erroneous	firstName = "KKKKKKKKKKKKK KKKKKKKKKKKKKKKKKK KKKKKKKKKKKKKKKKKK KKKKKKKKKKKKKKKKKK KKKKKKKKKKKKKKKKKK KKKKKKKKKKKKKK1"	E.g. when in MySQL, select firstName from users where firstName = "KKKKKKKKKKKKK KKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK KKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK KKKKKKKKK1", should not be registered in the table, therefore there should be no output since it is greater than the scope of firstName previously defined.
5	lastName is stored correctly	Normal	lastName = "Smith"	E.g. when in MySQL, select lastName from users where lastName = "Smith", should output a table with "Smith" as the output
6	lastName is stored on the boundary – lower limit: 1 character	Boundary	lastName = "S"	E.g. when in MySQL, select lastName from users where lastName = "S", should output a table with "S" as the output
7	lastName is stored on the boundary – upper limit: 100 characters	Boundary	lastName = "SSSSSSSSS SSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSS SSSSS"	E.g. when in MySQL, select lastName from users where lastName = "SSSSSSSSSSSSSSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSSSSSSSSSSSSSSS", should output a table with "SSSSSSSSSSSSSSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSSSSSSSSSSSSSSS" as the output

ID	Testing	Type	Data	Expected
8	lastName is invalid – 101 characters	Erroneous	lastName = SSSSSSSSS SSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSS SSSS1”	E.g. when in MySQL, select lastName from users where lastName = “SSSSSSSSSSSSSSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSSSSSSSSSSSSSS1”, should not be registered in the table, therefore there should be no output since it is greater than the scope of lastName previously defined.
9	Postal town and country are stored correctly	Normal	Postal town = “London” Country = “UK”	When in MySQL, select postal_town,country from users where postal_town = “London”, will return a table with “London” and “UK” as the outputs.
10	Postal town is on the boundary – 90 characters	Boundary	City = “SSSSSSSSSSSSS SSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSS SSSSS” Country = “UK”	When in MySQL, select postal_town,country from users where country = “UK”, will return a table with “SSSSSSS SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS SSSS” and “UK” as the outputs.

ID	Testing	Type	Data	Expected	Actual
1	firstName is stored correctly	Normal	firstName = “Karrison”	E.g. when in MySQL, select firstName from users where firstName = “Karrison”, should output a table with “Karrison” as the output	A table was returned with “Karrison” being the only output inside it **Passes**
2	firstName is stored on the boundary – lower limit: 1 character	Boundary	firstName = “K”	E.g. when in MySQL, select firstName from users where firstName = “K”, should output a table with “K” as the output	A table was returned with “K” being the only output inside it **Passes**
3	firstName is stored on the boundary – upper limit: 100 characters	Boundary	firstName = KKKKKKKKKKKKK KKKKKKKKKKKKK KKKKKKKKKKKKK KKKKKKKKKKKKK KKKKKKKKKKKKK KKKKKKKKKKKKK KKKKKKKK”	E.g. when in MySQL, select firstName from users where firstName = “KKKKKKKKKKKKK KKKKKKKKKKKKKKKKKKKKKKK KKKKKKKKKKKKKKKKKKKKKKK KKKKKKKKKKKKKKKKKKKKKKK KKKKKKKKK KKKKKKKKKK”, should output a table with “KKKKKKKKKKKKK KKKKKKKKKKKKKKKKKKKKKKK KKKKKKKKKKKKKKKKKKKKKKK KKKKKKKKKKKKKKKKKKKKKKK KKKKKKKKK KKKKKKKKKKKK” as the output	A table was returned with the whole of the firstName inside it, albeit on multiple lines since it did not fit completely horizontally in the space assigned by MySQL for the output **Passes**
4	firstName is invalid – 101 characters	Erroneous	firstName = KKKKKKKKKKKKK KKKKKKKKKKKKK KKKKKKKKKKKKK KKKKKKKKKKKKK KKKKKKKKKKKKK KKKKKKKKKKKKK KKKKKKKK1”	E.g. when in MySQL, select firstName from users where firstName = “KKKKKKKKKKKKK KKKKKKKKKKKKKKKKKKKKKKK KKKKKKKKKKKKKKKKKKKKKKK KKKKKKKKKKKKKKKKKKKKKKK KKKKKKKKK KKKKKKKKKK1”, should not be registered in the table, therefore there should be no output since it is greater than the scope of firstName previously defined.	Nothing was outputted, since no criteria was met with the value not being stored in the first place. There could be an option to have a longer length for a name, however, nearly all the users do not even touch the boundary therefore, it would increase the run time for no apparent reason due to the large amount of empty space **Passes**
5	lastName is stored correctly	Normal	lastName = “Smith”	E.g. when in MySQL, select lastName from users where lastName = “Smith”, should output a table with “Smith” as the output	A table was returned with “Smith” being the only output inside it **Passes**
6	lastName is stored on the boundary – lower limit: 1 character	Boundary	lastName = “S”	E.g. when in MySQL, select lastName from users where lastName = “S”, should output a table with “S” as the output	A table was returned with “S” being the only output inside it **Passes**

ID	Testing	Type	Data	Expected	Actual
7	lastName is stored on the boundary – upper limit: 100 characters	Boundary	lastName = "SSSS SSSSSSSSSSSS SSSSSSSSSSSS SSSSSSSSSSSS SSSSSSSSSSSS SSSSSSSSSSSS SSSSSSSSSSSS SSSSS"	E.g. when in MySQL, select lastName from users where lastName = "SSSSSSSSSSSS SSSSSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSSSSS", should output a table with "SSSS SSSSSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSSSSS SSSSSSS" as the output	A table was returned with "SS SSSSSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSSSSS SSSSSSSSSSSSS" being the only output inside it **Passes**
8	lastName is invalid – 101 characters	Erroneous	lastName = SSSSS SSSSSSSSSSSS SSSSSSSSSSSS SSSSSSSSSSSS SSSSSSSSSSSS SSSSSSSSSSSS SSSSSSSSSSSS SSSS1"	E.g. when in MySQL, select lastName from users where lastName = "SSSSSSSSSSSSS SSSSSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSSSS1", should not be registered in the table, therefore there should be no output since it is greater than the scope of lastName previously defined.	Nothing was outputted, since no criteria was met with the value not being stored in the first place. This was expected therefore it **passes**.
9	Postal town and country are stored correctly	Normal	Postal town= "London" Country = "UK"	When in MySQL, select postal_ town, country from users where city = "London", will return a table with "London" and "UK" as the outputs.	A table was returned with "London" and the "UK being the only outputs inside it
10	Postal town is on the boundary – 90 characters	Boundary	City = "SSSSSSSS SSSSSSSSSSSS SSSSSSSSSSSS SSSSSSSSSSSS SSSSSSSSSSSS SSSSSSSSSSSS SSSSSSSSSSSS SSSS" Country = "UK"	When in MySQL, select postal_ town,country from users where country = "UK", will return a table with "SSSSSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSSSSSSS SSSSSSSSSSSSSSSSSSSSSS SS" and "UK" as the outputs.	This **fails** since it did not get stored in the table even though it was within the criteria. This is because, that postal town does not exist so the latitude and longitude does not exist to find a location **Fails** The **remedial action** was to hardcode the longitude and latitude of that location as the same as London then test the postal town was now stored in the users table. It did get stored; therefore, it passes the test. **Remedial action**

2.2.13 Stakeholder Feedback

John:

- Happy about the fact that instead of searching for the word 'help' on the list of 100 verbs by continuously saying No, it is now customised towards him.
  - E.g. If he enters the word "Network", the number one verb is now "help".
  - This means he can do what he wants, even faster, due to the artificial intelligence identifying exactly what he requires.
- John feels the output section looks a little primitive, looks undesirable and although better than the unsorted dictionary format, it does still not look professional in his eyes

Victoria:

- Can search up a variety of inputs and due to her and previous users having already searched up and optimised those inputs, the time spent on finding the correct verb is reduced, therefore she can find the tweets she needs to find more easily and spend more time on researching rather than going through the list of 100 verbs each time.
  - E.g. English resources, one of the top verbs is “find”
  - She can find other teacher resources by using the output in the excel file for the hyperlink.
- Wants a version more interactive so her students can also use it, worried that her students may accidentally misread one of the options and input the wrong letter and mess up their output. Thinks it is a little bland currently to look at, wants it to have colour so her students can be interested by it and she can recommend it to her students.
- Wants more verification such as on the email address so students cannot mess up their details. She wants to know if her students have been messing about so the email address assigned by the school is a good way of logging the activities of each student.

Albert:

- Happy he doesn’t have to keep searching for the verbs that match with his interests
  - According to him – “after a few tries, it got much faster”
  - Intrigued about the artificial intelligence aspect
- Would like a feature where he does not have to repeatedly press CAPS LOCK and press N to find his verb, found that a little irritating.
  - My solution, is to substitute typing with buttons so the user only has to click the button.
- Wants more realistic test cases – he thinks no one has a surname primarily of 100 Ss

2.2.14 Evaluation - Second Iteration

I accomplished my goal for the second iteration, being to create a usable AI system and link to my main program. Although in my plan, I implied that the AI would be done in a separate file, I decided to keep it all in one file. This is because my stakeholders are not the most technologically advanced and making them open multiple files and making sure they save all the different files in the same location would not only be quite difficult on them, it would also increase the time spent on setting up the program instead of keeping it in one file, so the user has to only focus on setting up the necessary libraries on that specific file. I used subroutines to do this, which makes the main part of the program less chunky and it makes the program easier to read and understand to anyone else trying to interpret my code.

My stakeholders and I discussed the test plans and they tested out different cases on the program for me and reported their results back to me. Albert, one of my stakeholders said that my test cases were not the most realistic such as who would have a surname containing only Ss and 100 of them. Therefore, even though I had met the criteria of 100 characters, the stakeholders, specifically Albert would like more realistic example of test cases to try and mock real-life situations.

Once again, I had used **visualisation** for all the tables in MySQL so it is easier to visualise, not only for me but for my stakeholders who had to take note of the outputs when specific statements are run in the test cases we concocted together.

Within the Artificial Intelligence section of the program, I am using **concurrent processing** to carry out my procedures. When one table is incremented, it then passes onto the next called method and that is incremented but due to the fact there is no output when those methods are running, it gives the illusion that they are all carried out simultaneously. This works perfectly for now but in the future, in terms of maintenance, some methods may take too much processor time depending on the number of inputs given such as the variable phrase (the word the user inputs), as the number of users increase, the number of different phrases increase, therefore increasing the size of the table. Then having to check if this phrase already exists and if it already has a pairing will take a large amount of time since it has to loop through all the existing phrases in the table then either add it or if not, it has to loop again to find its order of pairing in descending order, applying **enumeration**. This means the processor time is at least 2n in the future.

In terms of the success criteria, I have met quite a few in the second iteration:

Number		Identify what makes this successful	Justify	Complete
1		Program loads without error	It won't work if it doesn't load.	First Iteration
2		Design form automatically appears when program is run	The user is able to see that the program is now available to work with.	
	2.1	User can enter in some inputs into the text boxes	This is their search word – the word they want to enquire using my crawler. Without this input, the user cannot search for their chosen topic.	First Iteration

Number		Identify what makes this successful	Justify	Complete
	2.2	Submit button can be clicked	For the program to progress, the user has to be able to click the button, otherwise the program will not know when the user has finished their input.	
	2.3	New window is created	This allows for a cleaner user interface instead of adding all commands onto one window	
	2.4	Old window closes as soon as new window is created	This means there is less clutter on the screen for the user contributing to less confusion as it is easier to interpret and uses up less memory as one window is now already closed.	
	2.5	Validation	If the user input does not meet the criteria for a correct input, then it must be told to the user and they should be able to try input again until they get a permitted input. The criteria the query input has to fulfil: — Not null — Query word is less than 100 characters in length — English syntax (English characters only, no special characters) — Greater than 2 characters — No abbreviations	First Iteration
	2.6	Question statement appears	The user can now review the question they generated– to see if it links to what they want	
	2.7	Radio-button appears	This provides an opportunity for the user to respond to the question	
	2.8	User can select either of the radio-buttons	A radio-button eliminates the element of doubt since the user is only limited to a selection of ‘yes’ or ‘no’	
	2.9a	The question does not change if the user clicked ‘Yes’	The user has found the question for what they are specifically trying to find. Therefore, it requires no more searching as the user’s search has finished.	

Number		Identify what makes this successful	Justify	Complete
	2.9b	The question now changes if the user clicked 'No'	The user has still not found what they are looking for, so as long as the user keeps selecting 'no', the question should keep changing until the user selects 'yes'	
3		User inputting their city and country they want to search	This is important since it gives us the input to know where to run location searches on tweets.	First Iteration
4		Twitter development account has been approved by Twitter	Without Twitter approving the development account, I am unable to access Twitter's databases so I would be unable to search for tweets without it being approved.	First Iteration
	4.1	Connect to the Twitter REST API	This is the basic part of running processes using Twitter's API. If I can connect to Twitter then I can find tweets relating to the user's search query. Now I have gained access to Twitter's databases.	First Iteration
	4.2	Connecting Twitter to Python	Using the credentials as a login, does Twitter allow access to Python to run my code using my created Twitter Development Account as a proxy	First Iteration
	4.2a	Writing the Twitter credentials in a file	A subclass should write the Twitter credentials into a file so it can be accessed by the main program. It is in a file for security purposes so someone cannot gain access to my Twitter credentials without them also having access to the file.	First Iteration
	4.2b	Reading the Twitter credentials from a file	The main program should be able to read and open the file written by the previous subclass and extract the necessary information from there like the consumer key for example.	First Iteration
	4.2c	Loading the Twitter credentials file	Once the file is read then it should be loaded up and the information requested by the code should be made usable in programs to carry out functions with that information.	First Iteration

Number		Identify what makes this successful	Justify	Complete
5		Run and load up MySQL	Am I able to connect to MySQL without any problems? This is where I am going to create a database. MySQL loads up with no errors.	Second Iteration
	5.1	Creating tables in MySQL	When I create tables then it provides the data structure to store any data that I would get from the data processing	Second Iteration
	5.2	Entering in the 100+ question words as the variable 'words' into the action table	This is what is used to specify the user's search query. These words are 100 of the most common English words therefore the user's search query is likely to be involved using one of these 100 words or any words to that effect.	Second Iteration
	5.3	Creating the structure of the other tables	This should when connected to Python, mean that the user input should slot in the variable 'phrase' and be stored within the tables	Second Iteration
	5.4	Integrating MySQL into Python	This allows us to connect user input and place that into the table and for us to use any tables previously created in MySQL in Python. This is useful for both storing and getting data.	Second Iteration
	5.5	Load up and store data in the database	Run MySQL instructions in Python using cursor – which is a variable that allows me to carry out MySQL instructions in Python such as SELECT, which means I can specifically call data from certain tables.	Second Iteration
6		Importing libraries	I need to use externally created libraries for some parts of the project instead of spending time manually crafting out already made functions	Second Iteration
	6.1	Importing mysql. connector	This is a library that allows us to connect to MySQL and without it I cannot connect Python to MySQL. This is similar to 3.3 however that is more specified towards carrying out functions with the data from MySQL compared to this, which is specifically to providing the bridge between MySQL and Python	Second Iteration

Number		Identify what makes this successful	Justify	Complete
	6.2	Importing Twython	This is a Python library which has been created to provide a way to access Twitter data. This is important for us to use as I need to access Twitter lists of Tweets being sent out and data on those tweets	First Iteration
	6.3	Importing JSON	JSON helps us to store the Twitter Credentials like the API key in a human-readable format but able to also be translated and understood by the main program.	First Iteration
	6.4	Importing Pandas	Pandas allows us to sort through all the data that I have gathered from Tweets quickly and efficiently as it is a data frame as well as a library to allow for analysis and manipulation of data.	First Iteration
	6.5	Importing Nominatim	I use this library to find the user entered location specifically the longitude and latitude and then use that to sort through the tweets filtering by distance from user entered area.	
	6.6	Importing Tkinter	Tkinter is a library specifically designed for aiding Graphical User Interface(GUI) design and this will help me in creating a User Interface to help the user visualise better what the program can do and what they, the user can do in progressing the program	
	6.7	Importing time	Instead of rapid movement of text, time allows me to slow it down such as for the updating questions when “No” is pressed, time will help to slow it down by a few seconds so the user can see it has changed.	
7		Incorporating Artificial Intelligence	By incorporating some sort of AI, this will allow us to become much more efficient providing the user a much more flawless experience as time passes.	Second Iteration

Number		Identify what makes this successful	Justify	Complete
	7.1	Storing count of number of times each word was selected	Whenever the user clicks “Yes” to a question, the count for that ‘word’ increments by one, this will then be useful for finding the most common query words.	Second Iteration
	7.2	User words being stored with the question word chosen with it	The query words will then be stored with the question words that was chosen. The less common words (e.g. if hospital and play were never paired together) will be then eventually be sorted out of the list, to reduce time.	Second Iteration
	7.3	Pairing common query words and question words together	This will then create a pair and the more pairs that are created then the more likely the AI can find the most common question words	Second Iteration
	7.4	Ordering the most common pairs first in descending order.	This means when that query word is entered then the most common pair word will be outputted first	Second Iteration
	7.5	Suggesting those pairs to the user	These pairs when they match the user query word will be suggested to the user and then this speeds up the overall process as the most likely question words to do with that query are likely to come up first.	Second Iteration
8		Outputting data to the user	The user has to see what is going on, therefore an output allows them to see the results of their search	Second Iteration
	8.1	Twitter handle of tweeter	The Twitter handle provides the user to see if they like the tweet then they can research more about that specific user and potentially find a new friend or customer helping out John my stakeholder.	First Iteration
	8.2	Location of tweet sent	The location of the tweet helps to find local issues for the user or if there is an event they want to find out about, it is much more personal if it is nearby rather than halfway across the world.	First Iteration
	8.3	Date of when tweet was sent	The date indicates how recent it was – e.g. if this was used to find out about the news then the date of tweet in seeing the reliability of that information	First Iteration

Number		Identify what makes this successful	Justify	Complete
	8.4	Text of actual tweet	This is the information of the tweet – the most important for many, this is how the user can see what the actual tweet is about.	First Iteration
	8.5	Number of likes on tweet	The number of likes on the tweet indicate popularity. However, as the program sorts out tweets based on location and date, this isn't the most useful feature. Although it can show my user whether the tweet is from a bot tweeting or not.	First Iteration
9		Pasting all this onto an accessible Microsoft Excel file	This pastes all the output onto an accessible Microsoft Excel sheet so the user can look at it more specifically. Furthermore, with every tweet there is a small hyperlink attached to see the whole tweet on the web browser. This feature would be very useful in particular for my stakeholder Victoria, who would be able to access any useful resources linked in the tweets for her teaching.	First Iteration

For example, 5.4("Integrating MySQL into Python") and 5.5("Load up and store data in the database") is proven in the test data since the user inputs entered in by the stakeholders in Python are registered in the tables. When the stakeholder entered in "Smith" as their last name; in the table Users, when a search for the last name "Smith" was completed, it returned the user's details. Furthermore, this is not possible without the infrastructure of the tables being created which meets 5.1("Creating tables in MySQL") and 5.3("Creating the structure of the other tables"). The connection from MySQL to Python is provided by the mysql.connector library which allows me to carry out MySQL instructions in Python such as searching for the first name "Karrison" which had passed the expected output test.

Unmet criteria are 2.9a("The question does not change if the user clicked 'Yes'") and 2.9b("The question now changes if the user clicked 'No'") since the users currently have to type "Y" for Yes and "N" for No and this is very inefficient. I can tackle this by providing a user interface so the users can interact with the program more easily and I can place buttons so instead of the user typing, they can click which is much faster and easier and

it meets Albert's suggestion of changing the typing function since it is very difficult on the user. Here I am applying **optimisation**, since this way I can reduce the time the user needs to interact with the program to get their desired result through introducing the idea of buttons. I also need to add validation for email addresses since no email address can only be one character long and every email address has an @ symbol within it. Email addresses allow me to track the user, in case they have committed any violation such as with a inappropriate input and if this loophole exists then, they can bypass my security function and disturb the program for other users.

2.3 Third Iteration

2.3.1 What is my third iteration?

I want to create a User Interface to help my users interact with the program more easily and then connect this to my main program so any actions that take place in the interface, lead to an action taking place in the database.

Why?

This will increase the range of users, since some may be intimidated by the primitive user-input currently. This will allow them to feel comfortable with the program. It will also increase the interactive nature of the user with the program therefore Victoria's students can use the program more engagingly and will be more encouraged to use the program at a future date.

2.3.2 Decomposition of the problem

Steps needed:

2.3.2.1 How do I create an external window?
Justification: Without an external window, there is no user interface for the user to see. It would just be printed in the console whether there are limited interactions so for a better user experience, an external window has to be developed

2.3.2.2 How do I create buttons?
Justification: This increases speed of user interaction. It is a very effective **usability** feature when implemented correctly.

2.3.2.3 How do the buttons go on the window
Justification: To let the user even interact with them, they require the user to see them, hence why they must be implemented on the external window which is created.

2.3.2.4 How does the user enter their input?
Justification: Without a space for the user to enter in the word/phrase they want to crawl Twitter for, then the program is useless since the whole purpose of the program is to crawl Twitter for the user's suggestion.

2.3.2.5 How do I get the input into my program from the interface?
Justification: To let the program carry out properly, there has to be a link between the back-end and front-end of the program, since the front-end (the user interface) is only the body but it cannot move without the brain (the back-end).

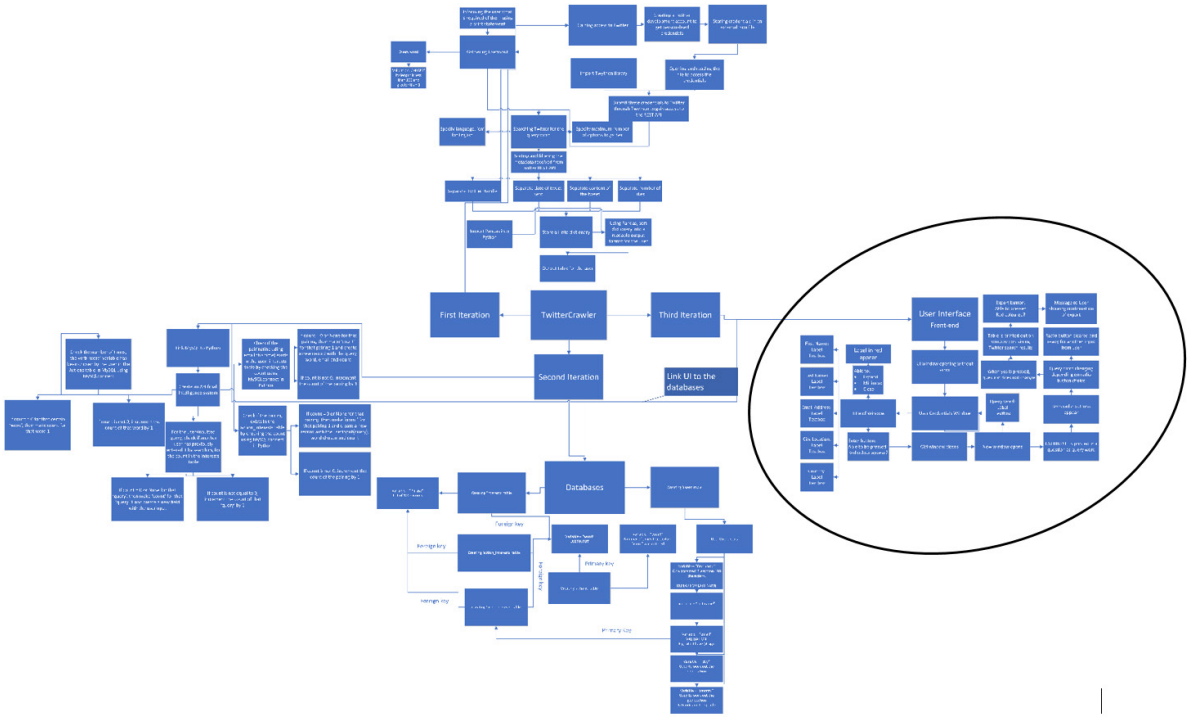
2.3.2.6 How do I create multiple windows?
Justification: Otherwise, one window containing everything will look too squashed and unaesthetically pleasing for the user and cause discomfort and potentially a negative user experience if it is too confusing to keep track of everything due to bad user interface design.

2.3.2.7 How does the user interact with the program in terms of buttons?
Justification: There is no point of having buttons, if the user cannot interact with them. For it to serve as a **usability** feature, it must work and that means they must act when the user clicks on them for example. The user interacts with them, through their mouse, clicking on the respective button in accordance with their decision of Yes and No in the case of choosing the correct action for the user input.

2.3.2.8 How do I display the output onto the interface so the user can see it?
Justification: The user interface must contain the whole program, including the output. Therefore, the table of results must be outputted onto the table so the user can see and read it, and then decide what they want to do with that data being sorted and supplied to them.

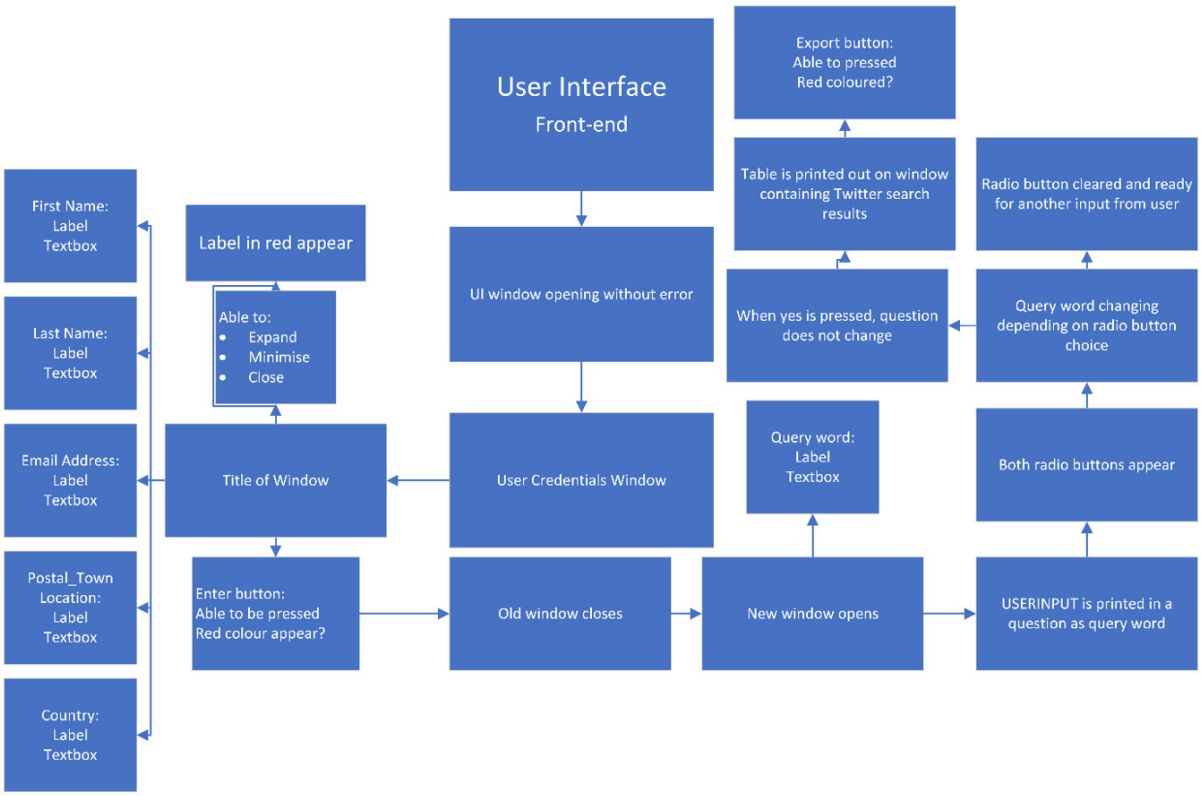
2.3.2.9 How do I close windows?
Justification: To prevent the desktop being cluttered with too many screens of all the different processes, the windows must close to make sure the user is focused on one window. Therefore, a **usability** feature that must be provided to the user, is the ability to close, minimise or expand windows as well as automatically closing them when they are not required to help those not completely adept with using the computer or my program.

2.3.3 Inheritance Diagram



Third Iteration:

User Interface:



Steps in detail

Step 1

(a) Import the Tkinter library – this library is the one suggested by many, to help create separate windows
(b) Put random dimensions for the window
(c) Create the window
(d) Keep adjusting the window size until I have the right size.

Step 2

(a) Use the button method() to create a button

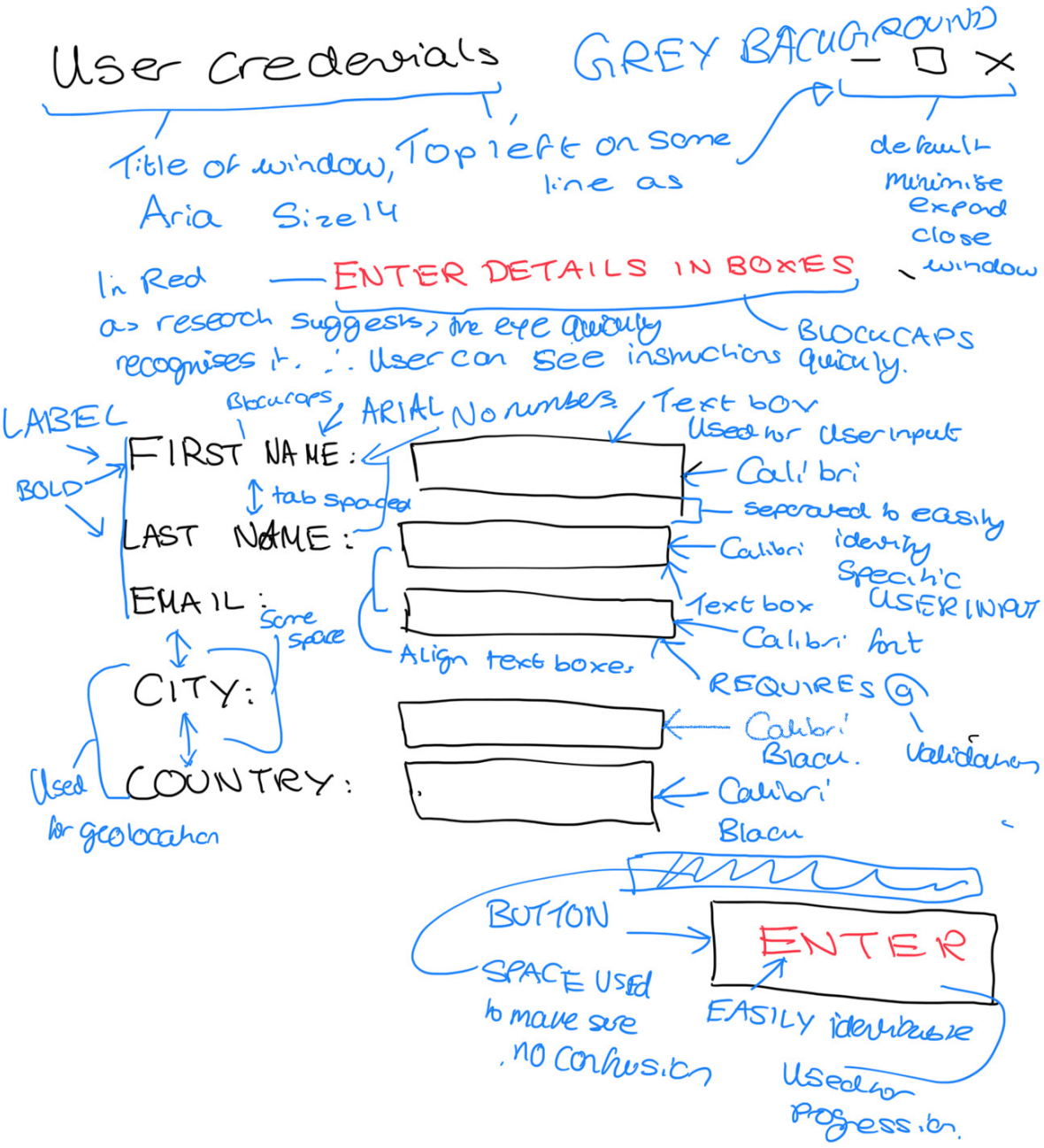
Step 3

(a) Keep outputting the window and button
(b) Keep adjusting the dimensions of the button
(c) Keep adjusting the placement of the button until it is in the right place
(d) Add the text to the button – what it says inside of the button area

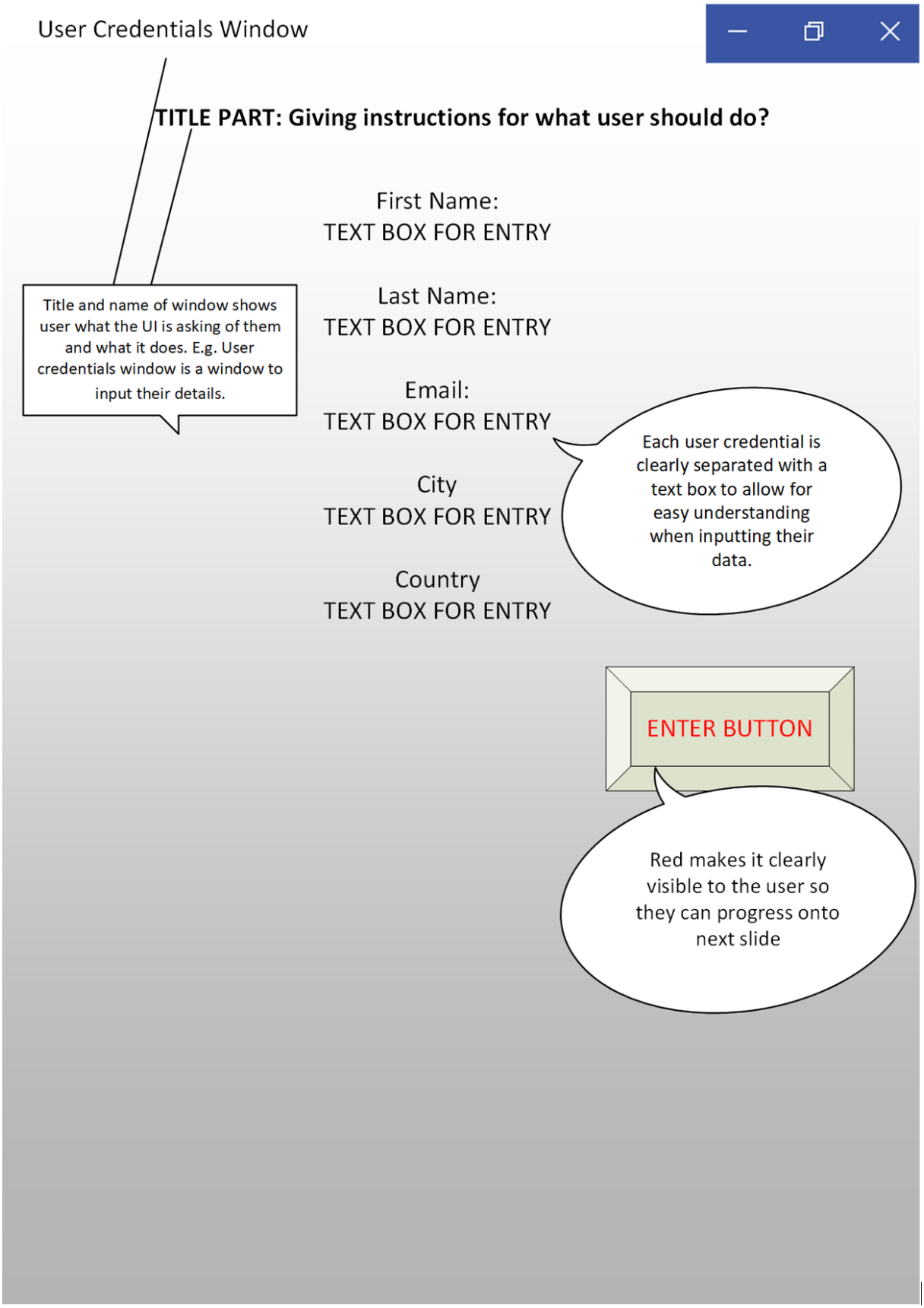
Step 4

(a) Create a message box – this will tell the user what to do
(b) Create a text box
- One for First name
- One for Last name
- One for email
- One for postal town
- One for country

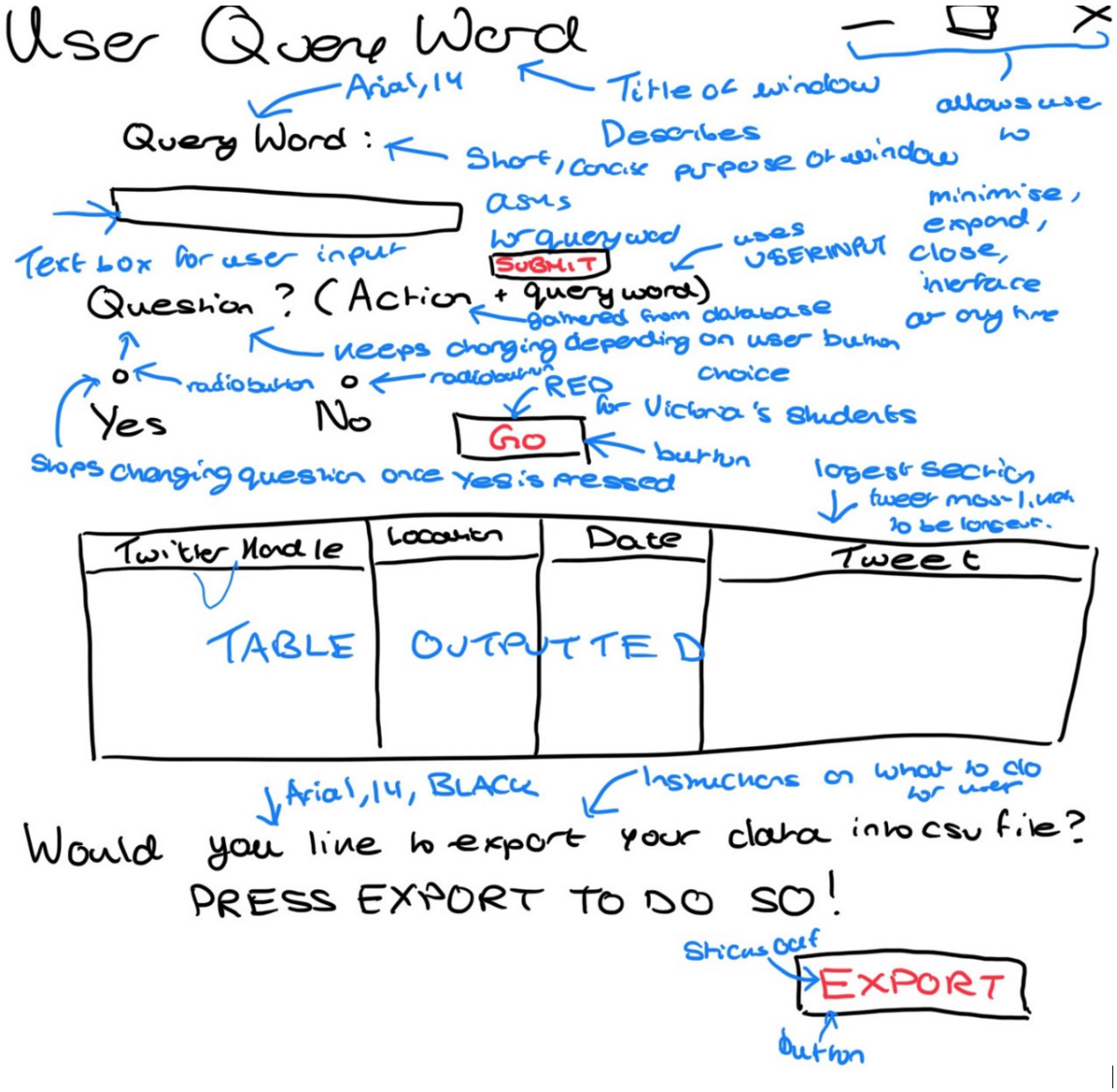
2.3.4 Plan of how the User Interface should look



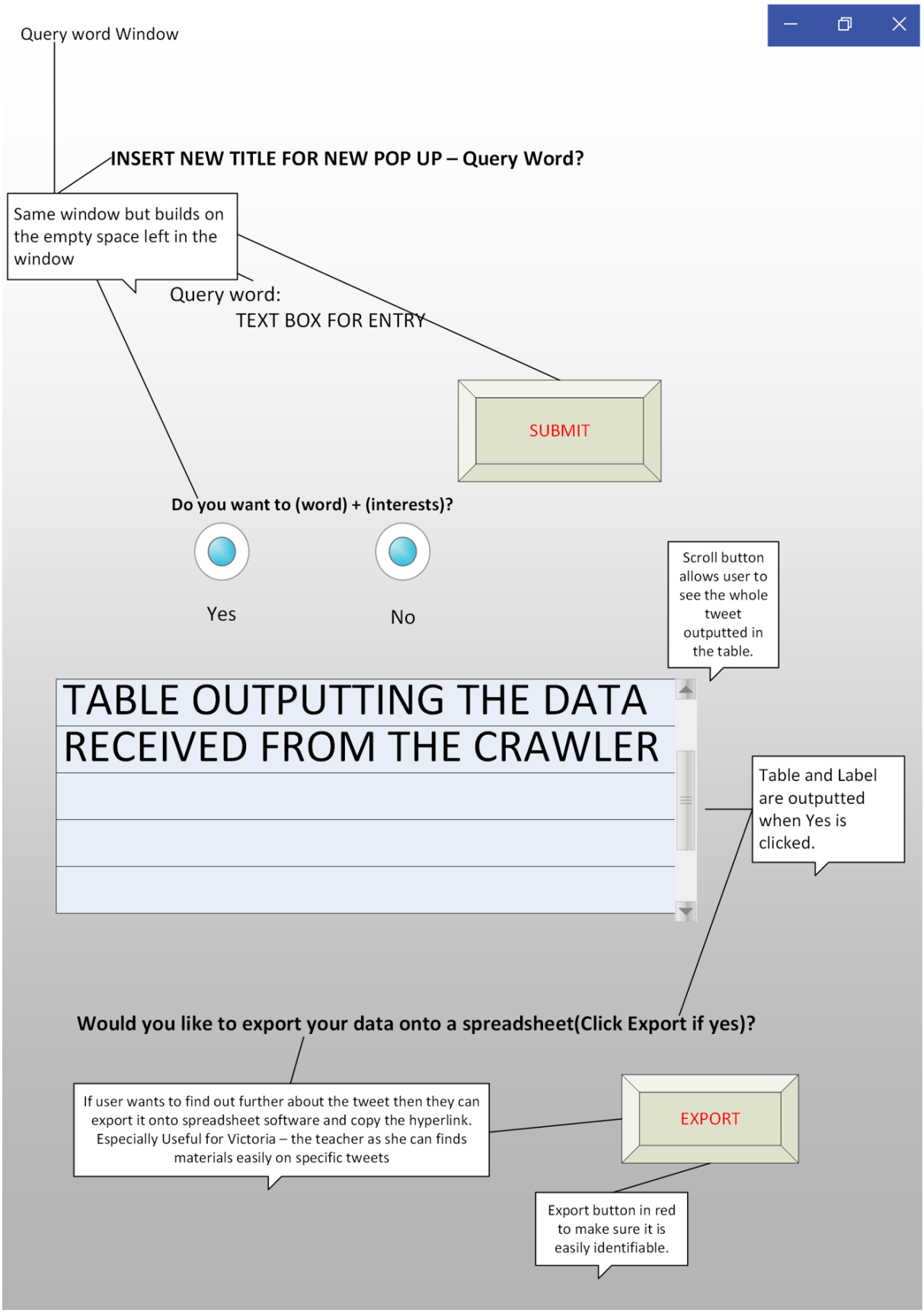
Computer generated on Visio



Drawing


User Query Word
Arial, 14
Title of window
allows use to
Query Word :
Describes
Short, concise purpose of window
asks for query word
minimise, expand, close, interface at any time
Text box for user input
SUBMIT
uses USERINPUT
Question ? (Action + query word)
gathered from database
keeps changing depending on user button choice
radiobutton
radiobutton
Yes
No
RED
for Victoria's students
GO
button
stops changing question once Yes is pressed
longest section tweet most likely to be longest.
Twitter Handle
Location
Date
Tweet
TABLE OUTPUTTED
Arial, 14, BLACK
Instructions on what to do for user
Would you like to export your data into csv file?
PRESS EXPORT TO DO SO!
Stick out
EXPORT
button

Computer generated on Visio



2.3.5 Development of Source Code

Source Code

```
import tkinter as tk
from tkinter import ttk
import tkinter.messagebox
import textwrap
import time
```

Justification

These are all the libraries used in the making of the User interface. Tkinter is the main library used in Python to create graphical user interface elements due to widgets available in its toolkit. Textwrap makes the text look orderly if a piece of text is longer than the specified dimensions given by me in the creation of certain elements. E.g. if the user enters in a very long phrase, text wrap will help me by spreading it on multiple lines instead of one long stretched out input, which makes it look more appealing overall. Time is used in the creation of the excel file to timestamp when each file was created.

Source Code

```
windowUserCreds = tk.Tk()
#creates a new window
windowUserCreds.title("User Credentials")
#The new window created has been given the Title:
#User Credentials
windowUserCreds.geometry("500x400")
#Size of window in pixels
```

Justification

This is where the main window is created. This is the master window in Tkinter standards, every other window from here would be a sub-window or secondary window. I have called it User Credentials since this is the window where I plan to get the user's details like first name and last name. Tkinter works in pixel sizes, rather than centimetres or metres so the 500 x 400 indicates 500 pixels by 400 pixels for the size of the master window. Using **iteration** in my thought process, I gathered this size after running it multiple times, adjusting the size each time until it was just right.

Source Code

```
def openQueryWindow(firstname,lastname,email,city,country):
    windowUserCreds.destroy()
    #Closes the original window
    newQueryWindow = tk.Tk()
    #Creates new window
    #newQueryWindow = Toplevel(windowUserCreds)
    newQueryWindow.title("User Query Word")
    #Gives title User Query Word to new window
    newQueryWindow.geometry("1200x1200")
    #Size of window
    queryLbl = tk.Label(newQueryWindow, text='Query word:')
    #Creates a Label and the text in the table is inputted
    queryTxt = tk.Entry(newQueryWindow)
    #Creates Text Box for user to input
```

Justification

This is the sub-window. It uses the parameters which are defined in the main program but the syntax of Python means that the main part of the program is usually defined last and the subroutines are first, so the order isn't the most accurate. Here, .destroy() allows me to get rid of the previous window instead of creating a new window each time, cluttering up the screen, potentially confusing the user. Then a new window is created, called User Query Word – as the title suggests, this is where I will ask the user for their phrase. I have added a text box and a label(message) which tells them exactly what to do.

Source Code

```
#Stores Userinput
queryLbl.place(x=100,y=50)
#Where label is placed
queryTxt.place(x=100,y=100)
#Where text box is placed
questionLbl = tk.Label(newQueryWindow, text = "")
questionLbl.place(x=100,y=200)
submitButtonUser = tk.Button(newQueryWindow, text = "Submit",fg="red",bg = "white",
                             command = lambda: displayLabel(questionLbl,queryTxt,newQueryWindow,firstname,
                                                            lastname,email,postal_town,country))
#lambda pass the functions as arguments instead of invoking the function
submitButtonUser.place(x=225,y=150)
#questionWordsQueryWindow()
```

Justification

This is a continuation of the openQueryWindow subroutine. Here the label and text box is placed on the window. As you can see, it uses the x and y axis and I want the label and text box to be aligned hence why they have the same x-axis but the y-axis is different since the textbox is underneath the label. The question label is what will be outputted as the question and currently the text is an empty string, this is because it should in the end using the verb from the actions table in MySQL and the phrase the user entered to create a question but to make it easier to understand, I created another subroutine where this action will take place. However, I have set up an activation sequence where it will only happen if the submit button is pressed by the user. To do this, I had to use a feature called lambda, which passes the function as an argument rather than immediately invoking the function allowing me to continue with my plan of having a sequence of; only if this condition is met (submit button pressed), then the subroutine will be called.

Source Code

```
def displayLabel(questionLbl,queryTxt,newQueryWindow,firstname,lastname,email,postal_town,country)
        global count
        #This means count can be called anywhere in the program
        count = 0
        phrase = queryTxt.get()
        #This is the word that has been gathered from the userinput
        choice = tk.IntVar()
        words = getAllWords(cursor,phrase)
        #This gathers all words from the table Actions in the database
        #print("phrase ",phrase) - this was a test to see if it did take in the userinput
        choice.set(None)
        questionLbl.config(text='Do you want to '+words[count]+' '+phrase)
        #Here I am creating the modified question which changes depending on the userinput
        radioYes = tk.Radiobutton(newQueryWindow,text = 'Yes', variable=choice,value=1)
        radioNo = tk.Radiobutton(newQueryWindow, text = 'No',variable=choice,value=0)
        #to group the buttons, use same variable as here variable = choice
        #Value is required to be able to only select either Yes or No
        radioYes.place(x=120,y=250)
        radioNo.place(x=180,y=250)
        #This is just where the radiobuttons are being placed
        goButtonCreds = tk.Button(newQueryWindow, text = "Go",fg="red",bg = "white",
                                  command =lambda: goButtonPress(questionLbl,queryTxt,
                                                                 newQueryWindow,choice,
                                                                 words,firstname,lastname,email,
                                                                 postal_town,country))
        #lambda pass the functions as arguments instead of invoking the function
        goButtonCreds.place(x=250,y=250)
```

Justification

This is the method where it changes the question, depending on the user's choice. It uses count to go through the table of actions, eventually when I connect it but now during the testing phase, I am going through an array I created. The two radio buttons are placed and created to clear every time they are pressed. This is a **usability** function since it makes it easier for the user to either choose yes or no, rather than typing it. The go button uses the lambda function to make sure goButtonPress is only called when the button is pressed. Also, the code here is annotated heavily to allow for future **maintenance** of the radio buttons

Source Code

```
def goButtonPress(questionLbl,queryTxt,newQueryWindow,choice,words,firstname,lastname,email,postal_town,country)
    print(choice)
    phrase = queryTxt.get()
    #This is the word that has been gathered from the userinput
    if choice.get()==0:
        #This checks if the user had chosen No on the radio button
        global count
        count=count+1
        #Increments count
        questionLbl.config(text='Do you want to '+words[count]+' '+phrase)
        #Changes the question, each time No is pressed
    if choice.get()==1:
        createTable(newQueryWindow,words[count],phrase,firstname,lastname,email,postal_town,country)
        #This calls the method CreateTable and pulls in the Twitter results and puts them into a table
        #Only when yes is pressed
    #time.sleep(1)
    #^No longer applicable due to creation of GO button so the user consciously changing question
    choice.set(None) # None should clear the radio button so no dot remains.
```

Justification

goButtonPress like the name suggests is the method containing the instructions of what should occur when the go button is pressed by the user. Throughout this program, I am applying a **sequential** way of thinking, using **selection** in the way that, if this happens then this method should run with the help of lambda. Here I use the global variable count to keep incrementing, changing the question if the user chooses "No" as the radio button. If "Yes" is pressed then it calls the CreateTable method(). Choice.set(None) was needed since previously, the input by the user for the radiobuttons would not be cleared even with a new question so the addition of this line clears the buttons to allow for a new input to take place. Previously, I had thought of using time.sleep to count 1 second but with the go button, this became redundant. This was also heavily annotated since this was quite a difficult part due to the **visualisation** of yes and no being seen numerically by the algorithm so it can process it.

Source Code

```
def wrapText(value,length = 90):
    return '\n'.join(textwrap.wrap(value,length))
    #This is used for any long outputs from Twitter
    #This places it in separate lines
    #Make sure table is not stretched unnecessarily
```

Justification

The wrapText method was created to solve the issue of a very long output from Twitter specifically the tweet. I noticed the **problem** of the output causing the table to stretch so with wrapText, it prevents the table being stretched by placing it on a new line if it goes beyond the boundary. Therefore I added this as a **creative solution** to the unique problem I was facing at first.

Source Code

```
def createTable(newQueryWindow,word,interest,firstname,lastname,email,city,country):
    # Add a Treeview widget
    tree = ttk.Treeview(newQueryWindow, column=("User", "Location", "Date","Text"), show='headings', height=5)
    #Treeview is part of Tkinter and it aids me in creating the infrastructure for a table
    tree.place(x=100,y=300)
    verticalScrollbar = ttk.Scrollbar(newQueryWindow,
                            orient ="vertical",
                            command = tree.yview)
    verticalScrollbar.place(x=1150,y=300, height=330)
    #This is if the output is longer than expected, then the user can scroll
    #It is also a feature needed if the user wants to change the number of outputs
    #Usability feature
    tree.configure(yscrollcommand = verticalScrollbar.set)
    s = ttk.Style()
    s.configure('Treeview',rowheight=60)
    tree.column("# 1", width=100, anchor=tk.CENTER)
    tree.heading("# 1", text="User")
    tree.column("# 2", width=100,anchor=tk.CENTER)
    tree.heading("# 2", text="Location")
    tree.column("# 3", width=200, anchor=tk.CENTER)
    tree.heading("# 3", text="Date")
    tree.column("# 4", width=650, anchor=tk.W)
    tree.heading("# 4", text="Text")
    #Name and placement of headers
```

Justification

This is the method which is called once the user presses the “Yes” radiobutton. I noticed the **problem** highlighted by my stakeholder of it not looking professional, therefore, I use the Treeview widget from the library from Tkinter since this creates the infrastructure for creating a table. By using Treeview, I can meet my **stakeholder**, John’s, view of making it look more professional. I also insert the headings which are the same as the excel spreadsheet and the table using pandas. Another **creative solution** I added was a scrollbar as a **usability** feature in case the output goes on multiple lines and does not fit in the original table – the scrollbar is a solution to this by allowing the user to scroll down if required. Then, I defined where each column is and the width of each since the width does not have to be equal. For example, the tweet would take up more space than the date on average.

Source Code

```
# Insert the data in Treeview widget
tree.insert('', 'end', text="1", values=('Amit', 'Kumar', '17701',wrapText('LongggggggggggggggggggTextLong
tree.insert('', 'end', text="1", values=('Ankush', 'Mathur', '17702','asdfasdf'))
tree.insert('', 'end', text="1", values=('Manisha', 'Joshi', '17703','asdfadf'))
tree.insert('', 'end', text="1", values=('Shivam', 'Mehrotra', '17704','asdfasdf'))
tree.insert('', 'end', text="1", values=('Does', 'it', 'scroll?','asdf'))
tree.insert('', 'end', text="1", values=('Doess', 'ist', 'scrolls?','qhdfg'))
#Here was some examples of Test inputs before it was connected to the backend and to the Twitter REST API
```

Justification

Here, for testing purposes, I am applying **problem decomposition**. Instead of adding the back-end all together and testing on those outputs, I inserted some of my own outputs to test out if it all works. E.g. I use wrapText on “Longggg…Text” to check if the wrapText method works in combination with the Treeview widget from Tkinter

Source Code

```
exportButtonLabel = tk.Label(newQueryWindow, text = "Your Top 5 Twitter Search Results Above
                            [Click Export button if you want to export full results in CSV format: ")
#Label created to tell ths user what will happen
exportButtonCreds = tk.Button(newQueryWindow, text = "Export",fg="red",bg = "white",
                              command =lambda: exportButtonPress(df))
#lambda pass the functions as arguments instead of invoking the function
exportButtonLabel.place(x=100,y=650)
exportButtonCreds.place(x=650,y=700)
#Placement of the button
print("done")
#Prints in the console separately to show that it works - testing
```

Justification

This label has the text exactly telling the user what the button will do. It is placed in the same window as the query word and the table proven by the location saying newQueryWindow. The button itself uses lambda like many of the buttons previously to make sure the method is only called when the button is pressed rather than straight away. I have also made it print done in the console to make sure the button works but this was in the early developmental stage of the button. I later came up with a better solution which I will reference later on which is called by the exportButtonPress.

Source Code

```
def exportButtonPress(df):
    timestamp = datetime.now()
    #catches the current time
    timestampStr = timestamp.strftime("%d-%b-%Y_(%H_%M)")
    #Sorts time and date into a string to be added into the filename
    extension = ".csv"
    fileName = "TwitterSearchResults_"+timestampStr+extension
    #Concatenates all features of the filename

    df.to_csv(fileName, index = False)
    #This converts our filename into Excel so now the timestamp will be visible
    tk.messagebox.showinfo(title="Information",message="Export Successful!")
    #MessageBox creates a new miniature window, only to just say the message
    #Only has message and an ok button to acknowledge it was read by user
```

Justification

This using the time library captures the exact date and time of when the program is run, specifically when the button is pressed and then converts into a string with day, month, year, hour and the minute which is concatenated all together with the prefix and the file extension to be the name of the csv file containing the output of the program at that time. Now, previously, I mentioned a better **solution** to the **problem** of how does the user know when the output has been successfully exported to a csv file? The **creative solution** that I concocted was to use Tkinter's widget messagebox to create a small miniature window with its only purpose being to tell the user that it has been successfully outputted and with a small ok button so the user can acknowledge this.

Source Code

```
#main
labelUserCreds = tk.Label(windowUserCreds,text = "Please enter user credentials below: (press enter when finished)",fg = "red",
                          font = ("Arial",11))
labelUserCreds.place(x=35,y=10)


firstnameLbl = tk.Label(windowUserCreds, text='First Name:')
firstnameTxt = tk.Entry()
firstnameLbl.place(x=100,y=50)
firstnameTxt.place(x=200,y=50)

lastnameLbl = tk.Label(windowUserCreds, text='Last Name:')
lastnameTxt = tk.Entry()
lastnameLbl.place(x=100,y=100)
lastnameTxt.place(x=200,y=100)

emailLbl = tk.Label(windowUserCreds, text='Email address:')
emailTxt = tk.Entry()
emailLbl.place(x=100,y=150)
emailTxt.place(x=200,y=150)

postal_townLbl = tk.Label(windowUserCreds, text='Postal Town:')
postal_townTxt = tk.Entry()
postal_townLbl.place(x=100,y=200)
postal_townTxt.place(x=200,y=200)

countryLbl = tk.Label(windowUserCreds, text='Country:')
countryTxt = tk.Entry()
countryLbl.place(x=100,y=250)
countryTxt.place(x=200,y=250)
#These are all the user details - label, textbox, placement
enterButtonCreds = tk.Button(windowUserCreds, text = "Enter",fg="red",bg = "white",
                             command = lambda:[insertUser(cursor,firstnameTxt.get(),lastnameTxt.get(),
                                                          emailTxt.get(),postal_townTxt.get(),countryTxt.get()),
                                               openQueryWindow(firstnameTxt.get(),lastnameTxt.get(),emailTxt.get(),
                                                               postal_townTxt.get(),countryTxt.get())])
enterButtonCreds.place(x=400,y=275)
#Enter button details

tk.mainloop()
```

Justification

This is the main part of the program for the user interface. It contains all the inputs required in the main window by the user. In this case it is, first name, last name, email, postal town and country and they all have been given text boxes to enter in their details which would eventually be stored in the users table in the database. It also has the placement of each label and text box as well as an additional button called the enter button, which when pressed will call the openQueryWindow at the very top of this program then the program after this point is carried out **sequentially.**

Source Code

```
#result = queryAndReturnTwitterResult(word,interest,findGeoCode(postal_town,country))
df = queryAndReturnTwitterResult(word,interest,findGeoCode(postal_town,country))
#Calls the connect and search to Twitter method

result = df.values.tolist()
#Converts the values into a list

for res in result[:5]:
        tree.insert('', 'end', text="1", values=(res[0], res[1],res[2],wrapText(res[3])))
        #Wraps the text output from Twitter
```

Justification

The variable ‘df’ calls the method queryAndReturnTwitterResult and this means this variable contains the Twitter results. Then these results are sorted into a list and the actual Tweet is wrapped. This is done through iteration, I go through each tweet and wrap each one in a loop and then each result is inserted into the table all ready for the user to see it when outputted. The date and username for example, are likely not needed to be looped so the subroutine wrapText is not called when they are referenced.

2.3.6 Errors and Remedial Actions in the Source Code

I am running this in Python IDE

Error

```
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Users\Karrison\AppData\Local\Programs\Python\Python39\lib\tkinter\__init__.py
__call__
    return self.func(*args)
  File "C:\Users\Karrison\Desktop\Desktop\America - programming\A-level Computer Science
estingofTKinterLibrary_NOTACTUALTEST_A-levelComputerScienceProject_091021.py", line 294,
    exportButtonCreds = tk.Button(newQueryWindow, text = "Export",fg="red",bg = "white",
exportButtonPress(df))
  File "C:\Users\Karrison\Desktop\Desktop\America - programming\A-level Computer Science
estingofTKinterLibrary_NOTACTUALTEST_A-levelComputerScienceProject_091021.py", line 299,
ress
    df.to_csv(r'TwitterSearchResults_A-levelComputerProject.csv', index = False)
  File "C:\Users\Karrison\AppData\Local\Programs\Python\Python39\lib\site-packages\panda
", line 3466, in to_csv
    return DataFrameRenderer(formatter).to_csv(
  File "C:\Users\Karrison\AppData\Local\Programs\Python\Python39\lib\site-packages\panda
at.py", line 1105, in to_csv
    csv_formatter.save()
  File "C:\Users\Karrison\AppData\Local\Programs\Python\Python39\lib\site-packages\panda
.py", line 237, in save
    with get_handle(
  File "C:\Users\Karrison\AppData\Local\Programs\Python\Python39\lib\site-packages\panda
line 702, in get_handle
    handle = open(
PermissionError: [Errno 13] Permission denied: 'TwitterSearchResults_A-levelComputerProj
```

Remedial Action

Here the error was due to me running the program a second time while keeping the linked excel file open.

This led to the operating system denying my program access to editing the file, rendering it unable to import the twitter data into the excel file.

My **creative solution** to this **problem** was to create copies of the excel file table and use timestamping to differentiate them so different excel files are created each time, stopping this error from occurring again.

Error

```
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Users\Karrison\AppData\Local\Programs\Python\Python39\lib\tkinter\__init__.py", line 1884, in
__call__
    return self.func(*args)
  File "C:\Users\Karrison\Desktop\Desktop\America - programming\A-level Computer Science Project\GeneralT
estingofTKinterLibrary_NOTACTUALTEST_A-levelComputerScienceProject_091021.py", line 293, in <lambda>
    exportButtonCreds = tk.Button(newQueryWindow, text = "Export",fg="red",bg = "white",command =lambda:
exportButtonPress(df))
NameError: name 'df' is not defined
```

Remedial Action

This error was due to a change in structure in my program. After the method wrapText was introduced, I had to create a slight change by wrapping each word in a list. ‘df’ was created as a solution to this problem. The **remedial action** to this problem was to define ‘df’ as a public variable so it can be used in any subroutine rather than a private variable like I had previously done before this action. ‘df’ was now going to be used to call the Twitter results and store them, then be used to convert the results into a list which is then wrapped in case the specific tweet was too long. ‘df’ was created as a tool to reduce **usability** and **maintenance** issues within the tables – this will reduce the likelihood of errors the user will experience

Error

```
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Users\Karrison\AppData\Local\Programs\Python\Python39\lib\tkinter\__init__.py", line 1884, in
__call__
    return self.func(*args)
  File "C:\Users\Karrison\Desktop\Desktop\America - programming\A-level Computer Science Project\GeneralT
estingofTKinterLibrary_NOTACTUALTEST_A-levelComputerScienceProject_091021.py", line 293, in <lambda>
    exportButtonCreds = tk.Button(newQueryWindow, text = "Export",fg="red",bg = "white",command =lambda:
exportButtonPress(df))
  File "C:\Users\Karrison\Desktop\Desktop\America - programming\A-level Computer Science Project\GeneralT
estingofTKinterLibrary_NOTACTUALTEST_A-levelComputerScienceProject_091021.py", line 299, in exportButtonP
ress
    messagebox.showinfo("Export Successful!")
NameError: name 'messagebox' is not defined
```

Remedial Action

Until this error, I did not realise that messageBox was not a integral part of the Tkinter widgets. The **remedial action** for this error, was to define import messageBox and define messageBox as tk.messageBox() since it was a secondary widget of the Tkinter library. After this action was taken, the error did not return therefore it had been solved.

Error

TwitterSearchResults_A-levelComputerProject.csv22-Dec-2021 (16

TwitterSearchResults22-Dec-2021_(16

Remedial Action

Here, although it wasn't explicitly returned as an error, it was still an error since it didn't do what I expected. The problem in the first one was the fact; the extension came before the time and the minute did not appear at all. The second one, the minute nor the extension appeared at all. This was due to the spaces in the string and the fact that a colon should not appear in the filename E.g. 16:22 is incorrect. Therefore, the **remedial action** was to replace all colons with an underscore instead and replace any spaces with an underscore. Furthermore, rearrange the string so the file extension appears last by separating it from the filename and then concatenating it once the date and time have been added.

Error

```
Traceback (most recent call last):
  File "C:\Users\Karrison\AppData\Local\Programs\Python\Python39\lib\tkinter\__init__.py", line 1884, in
__call__
    return self.func(*args)
  File "C:\Users\Karrison\Desktop\Desktop\America - programming\A-level Computer Science Project\GeneralT
estingofTKinterLibrary_NOTACTUALTEST_A-levelComputerScienceProject_091021.py", line 219, in <lambda>
    goButtonCreds = tk.Button(newQueryWindow, text = "Go",fg="red",bg = "white",command =lambda: goButton
Press(questionLbl,queryTxt,newQueryWindow,choice,words,firstname,lastname,email,city,country) )
  File "C:\Users\Karrison\Desktop\Desktop\America - programming\A-level Computer Science Project\GeneralT
estingofTKinterLibrary_NOTACTUALTEST_A-levelComputerScienceProject_091021.py", line 230, in goButtonPress
    createTable(newQueryWindow,words[count],phrase,firstname,lastname,email,city,country)
  File "C:\Users\Karrison\Desktop\Desktop\America - programming\A-level Computer Science Project\GeneralT
estingofTKinterLibrary_NOTACTUALTEST_A-levelComputerScienceProject_091021.py", line 259, in createTable
    result = queryAndReturnTwitterResult(word,interest,findGeoCode(city,country))
  File "C:\Users\Karrison\Desktop\Desktop\America - programming\A-level Computer Science Project\GeneralT
estingofTKinterLibrary_NOTACTUALTEST_A-levelComputerScienceProject_091021.py", line 141, in queryAndRetur
nTwitterResult
    python_tweets = Twython(creds['CONSUMER_KEY'], creds['CONSUMER_SECRET'])
NameError: name 'Twython' is not defined
```

Remedial Action

This is because the library was not correctly imported. Also, another error – although not pointed out here – was that there was a lack of clarity between the use of creds and credentials in referencing the dictionary. The **remedial action** in this case was to correctly import the library Twython so it is recognised by the Python IDE and use creds as the official dictionary name.

Error

```
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Users\Karrison\AppData\Local\Programs\Python\Python39\lib\tkinter\__init__.py", line 1884, in
__call__
    return self.func(*args)
  File "C:\Users\Karrison\Desktop\Desktop\America - programming\A-level Computer Science Project\GeneralT
estingofTKinterLibrary_NOTACTUALTEST_A-levelComputerScienceProject_091021.py", line 223, in <lambda>
    goButtonCreds = tk.Button(newQueryWindow, text = "Go",fg="red",bg = "white",command =lambda: goButton
Press(questionLbl,queryTxt,newQueryWindow,choice,words,firstname,lastname,email,city,country) )
  File "C:\Users\Karrison\Desktop\Desktop\America - programming\A-level Computer Science Project\GeneralT
estingofTKinterLibrary_NOTACTUALTEST_A-levelComputerScienceProject_091021.py", line 234, in goButtonPress
    createTable(newQueryWindow,words[count],phrase,firstname,lastname,email,city,country)
  File "C:\Users\Karrison\Desktop\Desktop\America - programming\A-level Computer Science Project\GeneralT
estingofTKinterLibrary_NOTACTUALTEST_A-levelComputerScienceProject_091021.py", line 284, in createTable
    captureMetricsToImproveAI(words,interest,cursor,email)
NameError: name 'words' is not defined
```

Remedial Action

This error was due to the fact that the console of MySQL was not running so the words could not be accessed to be appended into the array so the **remedial action** to this error, is to activate the console and keep it running in the background while the program is running.

2.3.7 Third Iteration – Source Code

```
import tkinter as tk
from tkinter import ttk
import tkinter.messagebox
import textwrap
import time
from geopy.geocoders import Nominatim
import json
from twython import Twython
import pandas as pd
from datetime import datetime
#Had to use CMD to download library: py -m pip install numpy

# imports Tkinter library which is a library specifically for User Interface design
import mysql.connector as mysql


#words = ['play','teach','learn']
count=0

windowUserCreds = tk.Tk()
#creates a new window
windowUserCreds.title("User Credentials")
#The new window created has been given the Title - User Credentials
windowUserCreds.geometry("500x400")
#Size of window in pixels


#Connect Python to MySQL database
# main part of the program
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = ""
)
cursor = db.cursor()
    #Use words and input into a sentence + query word
    #Add radio button Yes or No
    #If no, go onto next word


#sql functions backend code beginnning.

def insertUser(cursor,firstName,lastName,email,postal_town,country):
    cursor.execute("select first_name from twittercrawler.users where email = %s",(email,))
    #Cannot increment as no count variable
    valueExists = cursor.fetchone()
    if valueExists is None:
        cursor.execute("insert into twittercrawler.users(first_name,last_name,email,city,country) values(%s,%s,%s,%s,%s)",
                       (firstName,lastName,email,postal_town,country,))
    else:
        cursor.execute("update twittercrawler.users set first_name = %s , last_name = %s , city = %s , country = %s where email=%s",
                       (firstName,lastName,postal_town,country,email,))


#Database Word:
def getAllWords(cursor,phrase):
    words = []
    wordsSet = set()
    cursor.execute("select word from twittercrawler.action_interests where phrase = %s order by count desc",(phrase,))
    actionInterestsCursor = cursor.fetchall()
    #Gathers all actions from the action_interests table where the userinput is paired with it
    for dt in actionInterestsCursor:
        wordsSet.add(dt[0])
        #No duplicates added here^
        words.append(dt[0])

    #if word is not already in set, add to words

    #
    ## executing the statement using 'execute()' method
    cursor.execute("select * from twittercrawler.actions order by count desc")

    ## 'fetchall()' method fetches all the rows from the last executed statement
    data = cursor.fetchall() ## it returns a list of all databases present


    ## showing one by one database
    for dt in data:
        if dt[0] not in wordsSet:
            words.append(dt[0])
    return words
```

```
def findGeoCode(postal_town,country):
    app = Nominatim(user_agent="tutorial")
    # get location raw data
    location = postal_town + ", " + country
    location = app.geocode(location).raw

    return (location['lat'],location['lon'])
    #Longitude and latitude of location


def incrementWordFrequency(word,cursor):
    cursor.execute("select count from twittercrawler.actions where word = %s",(word,))
    count = cursor.fetchone()
    #Only fetches Count
    print(count)
    count = count[0] + 1
    #Increments it
    cursor.execute("update twittercrawler.actions set count = %s where word = %s",(count,word,))
    db.commit()

    '''cursor.execute("select count from twittercrawler.actions where word = %s",(word,))
    test = cursor.fetchone()
    print(test)'''
```

```
def incrementPhraseFrequency(phrase,cursor):
    cursor.execute("select count from twittercrawler.interests where phrase = %s",(phrase,))
    valueExists = cursor.fetchone()
    if valueExists is not None:
        count = valueExists[0]+1
        #increments current count
        cursor.execute("update twittercrawler.interests set count = %s where phrase = %s",(count,phrase,))
        #updates count in mySQL
    else:
        cursor.execute("insert into twittercrawler.interests(phrase,count) values(%s,1)",(phrase,))
        #creates new count and auto adjusts to one
    db.commit()

def incrementPhraseWordFrequency(word,phrase,cursor):
    cursor.execute("select count from twittercrawler.action_interests where phrase = %s and word = %s",(phrase,word,))
    valueExists = cursor.fetchone()
    if valueExists is not None:
        count = valueExists[0]+1
        cursor.execute("update twittercrawler.action_interests set count = %s where phrase = %s and word = %s",(count,phrase,word,))
        #updates existing row and increments count by 1
    else:
        cursor.execute("insert into twittercrawler.action_interests(word,phrase,count) values(%s,%s,1)",(word,phrase,))
        #creates new row - inserts all and makes count = 1
    db.commit()

def incrementPhraseWordEmailFrequency(word,phrase,email,cursor):
    cursor.execute("select count from twittercrawler.user_interests where email = %s and phrase = %s and word = %s",(email,phrase,word,))
    valueExists = cursor.fetchone()
    if valueExists is not None:
        count = valueExists[0]+1
        #increments by 1
        cursor.execute("update twittercrawler.user_interests set count = %s where email = %s and phrase = %s and word = %s",(count,email,phrase,word,))
        #executes this instruction in MySQL
    else:
        cursor.execute("insert into twittercrawler.user_interests(email,phrase,word,count) values(%s,%s,%s,1)",(email,phrase,word,))
        #executes this instruction in MySQL - insert user details into user table
    db.commit()
```

```
def captureMetricsToImproveAI(word,phrase,cursor,email):
    # step 1 - increment the count of the word by 1 in actions table - update query
    incrementWordFrequency(word,cursor)
    # step 2 - add the phrase in interests table and mark count = count +1 - insert/update  query
    incrementPhraseFrequency(phrase,cursor)
    # step 3 - add the combination of word and phrase to action_interests - insert/update
    incrementPhraseWordFrequency(word,phrase,cursor)
    # step 4 - add the combination of word and phrase and email to user_interests - insert/update
    incrementPhraseWordEmailFrequency(word,phrase,email,cursor)
```

```
#call twitter
def queryAndReturnTwitterResult(phrase,interest,location):
    #
    # Load credentials from json file
    with open("twitter_credentials.json", "r") as file:
        creds = json.load(file)

    # Instantiate an object
    python_tweets = Twython(creds['CONSUMER_KEY'], creds['CONSUMER_SECRET'])

    query = phrase + ' '+interest
    #combines them
    geocode = location[0]+','+location[1]+',50mi'
    #combines lat and long
    #print(geocode)
    #to double check location - test

    # Create our query
    query = {'q': query,
            #'result_type': 'popular', - encountered an error, fixed by commenting 'popular'
            #out since it was not a predefined term
            'count': 10,
            'geocode':geocode,
            'lang': 'en',
            }

        # Search tweets
    dict_ = {'user': [], 'location':[], 'date': [], 'text': [], 'favorite_count': []}
   # print(python_tweets.search(**query)['statuses'][1])

    for status in python_tweets.search(**query)['statuses']:
        dict_['user'].append(status['user']['screen_name'])
        dict_['location'].append(status['user']['location'])
        dict_['date'].append(status['created_at'])
        dict_['text'].append(status['text'])
        dict_['favorite_count'].append(status['favorite_count'])
    #Separates the data gathered from the Twitter REST API
    #Separated into different categories

    # Structure data in a pandas DataFrame for easier manipulation
    df = pd.DataFrame(dict_)
    df.sort_values(by='favorite_count', inplace=True, ascending=False)
    #Sorted by no.of likes

    return df


#python frontend code beginning.
def openQueryWindow(firstname,lastname,email,postal_town,country):
    windowUserCreds.destroy()
    #Closes the original window
    newQueryWindow = tk.Tk()
    #Creates new window
    #newQueryWindow = Toplevel(windowUserCreds)
    newQueryWindow.title("User Query Word")
    #Gives title User Query Word to new window
    newQueryWindow.geometry("1200x1200")
    #Size of window
    queryLbl = tk.Label(newQueryWindow, text='Query word:')
    #Creates a Label and the text in the table is inputted
    queryTxt = tk.Entry(newQueryWindow)
    #Creates Text Box for user to input

    #Stores Userinput
    queryLbl.place(x=100,y=50)
    #Where label is placed
    queryTxt.place(x=100,y=100)
    #Where text box is placed
    questionLbl = tk.Label(newQueryWindow, text = "")
    questionLbl.place(x=100,y=200)
    submitButtonUser = tk.Button(newQueryWindow, text = "Submit",fg="red",bg = "white",
                        command = lambda: displayLabel(questionLbl,queryTxt,newQueryWindow,firstname,
                                                       lastname,email,postal_town,country))
    #lambda pass the functions as arguments instead of invoking the function
    submitButtonUser.place(x=225,y=150)
    #questionWordsQueryWindow()
```

```
def displayLabel(questionLbl,queryTxt,newQueryWindow,firstname,lastname,email,postal_town,country):
        global count
        #This means count can be called anywhere in the program
        count = 0
        phrase = queryTxt.get()
        #This is the word that has been gathered from the userinput
        choice = tk.IntVar()
        words = getAllWords(cursor,phrase)
        #This gathers all words from the table Actions in the database
        #print("phrase ",phrase) - this was a test to see if it did take in the userinput
        choice.set(None)
        questionLbl.config(text='Do you want to '+words[count]+' '+phrase)
        #Here I am creating the modified question which changes depending on the userinput
        radioYes = tk.Radiobutton(newQueryWindow,text = 'Yes', variable=choice,value=1)
        radioNo = tk.Radiobutton(newQueryWindow, text = 'No',variable=choice,value=0)
        #to group the buttons, use same variable as here variable = choice
        #Value is required to be able to only select either Yes or No
        radioYes.place(x=120,y=250)
        radioNo.place(x=180,y=250)
        #This is just where the radiobuttons are being placed
        goButtonCreds = tk.Button(newQueryWindow, text = "Go",fg="red",bg = "white",
                                  command =lambda: goButtonPress(questionLbl,queryTxt,
                                                                 newQueryWindow,choice,words,
                                                                 firstname,lastname,email,postal_town,country))
        #lambda pass the functions as arguments instead of invoking the function
        goButtonCreds.place(x=250,y=250)



def goButtonPress(questionLbl,queryTxt,newQueryWindow,choice,words,firstname,lastname,email,postal_town,country):
    print(choice)
    phrase = queryTxt.get()
    #This is the word that has been gathered from the userinput
    if choice.get()==0:
        #This checks if the user had chosen No on the radio button
        global count
        count=count+1
        #Increments count
        questionLbl.config(text='Do you want to '+words[count]+' '+phrase)
        #Changes the question, each time No is pressed
    if choice.get()==1:
        createTable(newQueryWindow,words[count],phrase,firstname,lastname,email,postal_town,country)
        #This calls the method CreateTable and pulls in the Twitter results and puts them into a table
        #Only when yes is pressed

    #time.sleep(1)
    #^No longer applicable due to creation of GO button so the user consciously changing question
    choice.set(None) # None should clear the radio button so no dot remains.
def wrapText(value,length = 90):
    return '\n'.join(textwrap.wrap(value,length))
    #This is used for any long outputs from Twitter
    #This places it in separate lines
    #Make sure table is not stretched unnecessarily



def createTable(newQueryWindow,word,interest,firstname,lastname,email,postal_town,country):
    # Add a Treeview widget
    tree = ttk.Treeview(newQueryWindow, column=("User", "Location", "Date","Text"), show='headings', height=5)
    #Treeview is part of Tkinter and it aids me in creating the infrastructure for a table
    tree.place(x=100,y=300)
    verticalScrollbar = ttk.Scrollbar(newQueryWindow,
                           orient ="vertical",
                           command = tree.yview)
    verticalScrollbar.place(x=1150,y=300, height=330)
    #This is if the output is longer than expected, then the user can scroll
    #It is also a feature needed if the user wants to change the number of outputs
    #Usability feature
    tree.configure(yscrollcommand = verticalScrollbar.set)
    s = ttk.Style()
    s.configure('Treeview',rowheight=60)
    tree.column("# 1", width=100, anchor=tk.CENTER)
    tree.heading("# 1", text="User")
    tree.column("# 2", width=100,anchor=tk.CENTER)
    tree.heading("# 2", text="Location")
    tree.column("# 3", width=200, anchor=tk.CENTER)
    tree.heading("# 3", text="Date")
    tree.column("# 4", width=650, anchor=tk.W)
    tree.heading("# 4", text="Text")
    #Name and placement of headers

    #result = queryAndReturnTwitterResult(word,interest,findGeoCode(postal_town,country))
    df = queryAndReturnTwitterResult(word,interest,findGeoCode(postal_town,country))
    #Calls the connect and search to Twitter method

    result = df.values.tolist()
    #Converts the values into a list

    for res in result[:5]:
           tree.insert('', 'end', text="1", values=(res[0], res[1],res[2],wrapText(res[3])))
           #Wraps the text output from Twitter


    # Insert the data in Treeview widget
    '''tree.insert('', 'end', text="1", values=('Amit', 'Kumar', '17701',wrapText('LongggggggggggggggggggText
    tree.insert('', 'end', text="1", values=('Ankush', 'Mathur', '17702','asdfasdf'))
    tree.insert('', 'end', text="1", values=('Manisha', 'Joshi', '17703','asdfadf'))
    tree.insert('', 'end', text="1", values=('Shivam', 'Mehrotra', '17704','asdfasdf'))
    tree.insert('', 'end', text="1", values=('Does', 'it', 'scroll?','asdf'))
    tree.insert('', 'end', text="1", values=('Doess', 'ist', 'scrolls?','ghdfg'))'''
    #Here was some examples of Test inputs before it was connected to the backend and to the Twitter REST API
```

```
    captureMetricsToImproveAI(word,interest,cursor,email)

    exportButtonLabel = tk.Label(newQueryWindow,
    text = "Your Top 5 Twitter Search Results Above [Click Export button if you want to export full results in CSV format: ")
    #Label created to tell ths user what will happen
    exportButtonCreds = tk.Button(newQueryWindow, text = "Export",fg="red",bg = "white",command =lambda: exportButtonPress(df))
    #lambda pass the functions as arguments instead of invoking the function
    exportButtonLabel.place(x=100,y=650)
    exportButtonCreds.place(x=650,y=700)
    #Placement of the button
    print("done")
    #Prints in the console separately to show that it works - testing
def exportButtonPress(df):
    timestamp = datetime.now()
    #catches the current time
    timestampStr = timestamp.strftime("%d-%b-%Y_(%H_%M)")
    #Sorts time and date into a string to be added into the filename
    extension = ".csv"
    fileName = "TwitterSearchResults_"+timestampStr+extension
    #Concatenates all features of the filename

    df.to_csv(fileName, index = False)
    #This converts our filename into Excel so now the timestamp will be visible
    tk.messagebox.showinfo(title="Information",message="Export Successful!")
    #MessageBox creates a new miniature window, only to just say the message
    #Only has message and an ok button to acknowledge it was read by user



#main
labelUserCreds = tk.Label(windowUserCreds,text = "Please enter user credentials below: (press enter when finished)",fg = "red",
                          font = ("Arial",11))
labelUserCreds.place(x=35,y=10)


firstnameLbl = tk.Label(windowUserCreds, text='First Name:')
firstnameTxt = tk.Entry()
firstnameLbl.place(x=100,y=50)
firstnameTxt.place(x=200,y=50)

lastnameLbl = tk.Label(windowUserCreds, text='Last Name:')
lastnameTxt = tk.Entry()
lastnameLbl.place(x=100,y=100)
lastnameTxt.place(x=200,y=100)

emailLbl = tk.Label(windowUserCreds, text='Email address:')
emailTxt = tk.Entry()
emailLbl.place(x=100,y=150)
emailTxt.place(x=200,y=150)

postal_townLbl = tk.Label(windowUserCreds, text='Postal Town:')
postal_townTxt = tk.Entry()
postal_townLbl.place(x=100,y=200)
postal_townTxt.place(x=200,y=200)

countryLbl = tk.Label(windowUserCreds, text='Country:')
countryTxt = tk.Entry()
countryLbl.place(x=100,y=250)
countryTxt.place(x=200,y=250)
#These are all the user details - label, textbox, placement
enterButtonCreds = tk.Button(windowUserCreds, text = "Enter",fg="red",bg = "white",
                             command = lambda:[insertUser(cursor,firstnameTxt.get(),lastnameTxt.get(),
                                                          emailTxt.get(),postal_townTxt.get(),countryTxt.get()),
                                               openQueryWindow(firstnameTxt.get(),lastnameTxt.get(),emailTxt.get(),
                                                               postal_townTxt.get(),countryTxt.get())])
enterButtonCreds.place(x=400,y=275)
#Enter button details

tk.mainloop()
```

2.3.8 Stakeholder Feedback

John:

- Loves the fact that it looks professional to all users
  - Does want to see more real-life data (in addition with the current tests) outputted when testing in the tables
    - This can be tackled in the post-development testing
- The table makes it look much less primitive than before. The option of the scroll button for long tweets is a very good idea
- Likes the fact that the artificial intelligence is implemented fully into the user interface.
- Likes that there is enough validation
    - With name, postal town, country and email
  - However, he believes it is linked to censorship – moral aspect to tackle?
    - Controlling what users can search and enter

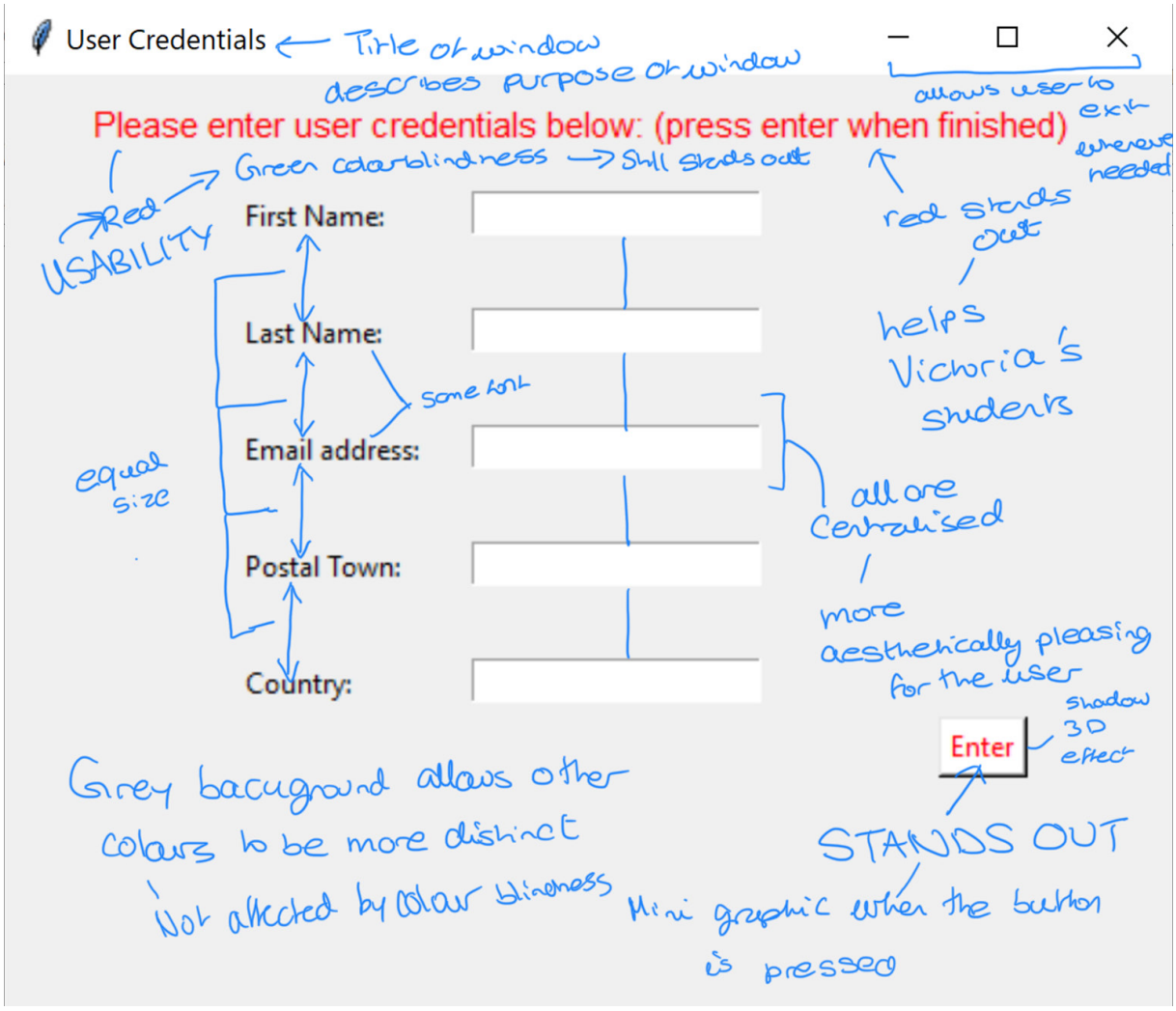
Victoria:

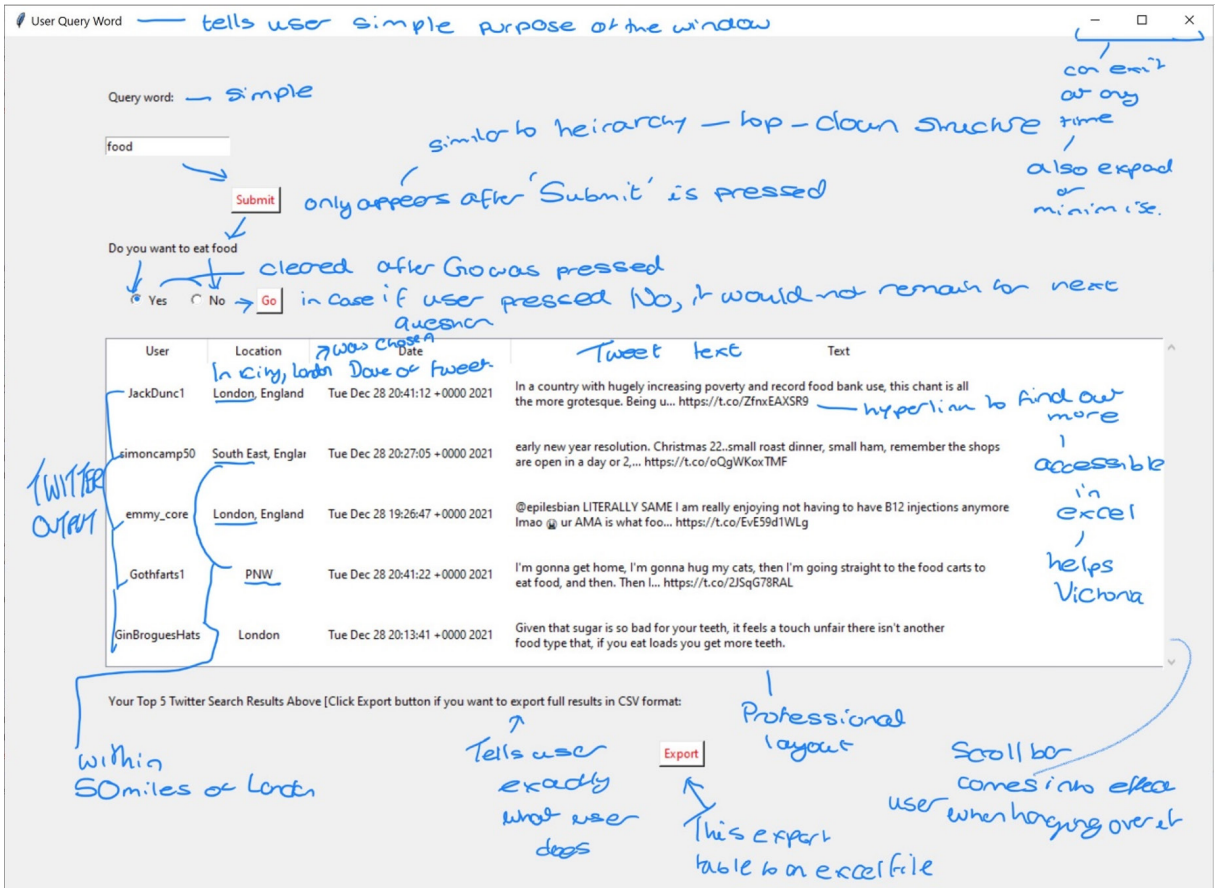
- The user interface means the students can interact with the program more easily.
  - Not frightened by seeing the code
  - Buttons to click increases engagement
  - More likely to use this as a tool of learning
- The time-stamped excel file, being a **usability** feature, means she knows exactly when the search was done so it is easier for her to have multiple tabs comparing resources found on the Twitter crawler
- Loves the consideration of colour-blind children, when choosing colours for the UI design as a **usability** feature
  - Allows for more inclusion from all children.

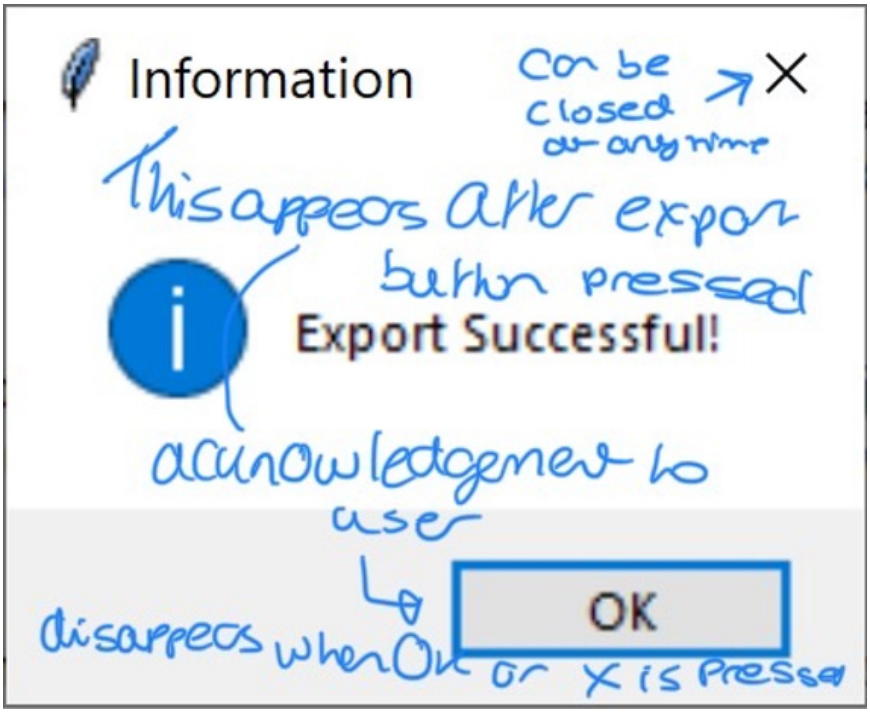
Albert:

- Loves the fact that he has to type less and click more.
  - He has to only press the No button to progress rather than typing N each time
- In the future, he wants a text-to-speech tool and a speech-to-text tool which means he has to no longer use his hands at all, only his mouth to control the program.

2.3.9 Output Screens – Annotated






Information
Can be closed at any time
This appears after export button pressed
Export Successful!
acknowledgement to user
OK
disappears when OK or X is pressed


type of file
What program opens it.
filename describes purpose
TwitterSearchRes
ults_28-Dec-202
1_(20_52).csv
date of creation
time of creation
csv excel
time

2.3.10 Evaluation - Third Iteration

For the third and final iteration, I have accomplished my goal which was to create a user interface and connect to my main program. As a result, any input the user enters in the interface, is stored in the database and I have used their inputs as parameters for my numerous methods, creating a slick running of my program, outputting the data they require.

I have met every single criterion created in the Analysis to define a successful project, I have also added in extra usability features (which will also serve as a useful feature when carrying out maintenance) such as timestamping the excel file, when it has been created by the user so multiple users can use the program simultaneously without there being a clash. Another feature, is using specific colours like red, which would stick out as pink for example for a child with Tritanopia, which accomplishes its goal by being different to the grey background and black writing. This was achieved through a multitude of research on the different types of colour-blindness children face every day, to allow for a program that includes all children no matter their condition.

Another point of discussion with my stakeholders, specifically Albert, was the introduction of a speech-to-text tool. This is a wonderful idea to increase the range of people who can use the program such as those unable to type, however, existing products, which can be implemented into my program, require payment into a subscription service so this means my product may also require a payment to offset that monthly cost which is not a viable option as of this moment. A creative solution to this problem, may be to implement a camera and the user nods and shakes their head, depending on what letter of the alphabet they would like. This may be time-consuming but it is an effective alternative to those who cannot type, only requiring their head for input.

For proof of criteria being met using test data:

- Some are met with test case 1 where I enter the first name as input as it is gathered since:
  - The program runs the user interface, meaning there is no error
    - This is only able to be done since Tkinter is imported.
  - I am able to input the first name; therefore, the design form appears and the text box appears.
  - I am able to gather the input, which I am only able to do, since the submit button is clicked.
- Some are met with test case 10 when I enter the city into the user interface since:
  - User is able to enter the postal town into the text box
  - The limit of 100 characters is supplied by MySQL when the tables were created
  - This was then integrated into Python using mysql.connector

— All the other criteria are met on the next window, where the user enters which question they agree with, proven with test case 9 and 11 since:
  - o As soon as the submit button is pressed, the original window disappears as the new window asking for the query word appears
  - o After a query word is inputted, then the question appears as a combination of the action + the query word
  - o The radio buttons for Yes and No are underneath the question
    - » Leading to the question changing if No is pressed
    - » Going through the 100 verbs if necessary from the database in MySQL
  - o For the output, the data is gathered by the search function linked to the Twitter REST API, which is only possible if Twitter has approved me access to carry out commands from Python.
  - o The credentials are safely stored in an external file and read from there to gain access to the API
  - o Then the results are filtered depending on location using the Nominatim library for geocode filtering
  - o Then these inputs are stored into the database and various methods are run using these parameters are inputs
    - » This creates my Artificial Intelligence as in short, it basically defines the most recommended pairing to the user first
  - o After the search function has gathered all the data, it is outputted in the form of a table containing:
    - » Twitter handle
    - » Location of tweet
    - » Date
    - » Text
    - » Number of likes
  - o There is also an export button which would provide all of this information into an accessible Twitter file for the user.

Number		Identify what makes this successful	Justify	Complete
1		Program loads without error	It won't work if it doesn't load.	First Iteration
2		Design form automatically appears when program is run	The user is able to see that the program is now available to work with.	Third Iteration
	2.1	User can enter in some inputs into the text boxes	This is their search word – the word they want to enquire using my crawler. Without this input, the user cannot search for their chosen topic.	First Iteration

Number		Identify what makes this successful	Justify	Complete
	2.2	Submit button can be clicked	For the program to progress, the user has to be able to click the button, otherwise the program will not know when the user has finished their input.	Third Iteration
	2.3	New window is created	This allows for a cleaner user interface instead of adding all commands onto one window	Third Iteration
	2.4	Old window closes as soon as new window is created	This means there is less clutter on the screen for the user contributing to less confusion as it is easier to interpret and uses up less memory as one window is now already closed.	Third Iteration
	2.5	Validation	If the user input does not meet the criteria for a correct input, then it must be told to the user and they should be able to try input again until they get a permitted input. The criteria the query input has to fulfil: — Not null — Query word is less than 100 characters in length — English syntax (English characters only, no special characters) — Greater than 2 characters — No abbreviations	First Iteration
	2.6	Question statement appears	The user can now review the question they generated– to see if it links to what they want	Third Iteration
	2.7	Radio-button appears	This provides an opportunity for the user to respond to the question	Third Iteration
	2.8	User can select either of the radio-buttons	A radio-button eliminates the element of doubt since the user is only limited to a selection of ‘yes’ or ‘no’	Third Iteration
	2.9a	The question does not change if the user clicked ‘Yes’	The user has found the question for what they are specifically trying to find. Therefore, it requires no more searching as the user’s search has finished.	Third Iteration

Number		Identify what makes this successful	Justify	Complete
	2.9b	The question now changes if the user clicked 'No'	The user has still not found what they are looking for, so as long as the user keeps selecting 'no', the question should keep changing until the user selects 'yes'	Third Iteration
3		User inputting their postal town and country they want to search	This is important since it gives us the input to know where to run location searches on tweets.	First Iteration
4		Twitter development account has been approved by Twitter	Without Twitter approving the development account, I am unable to access Twitter's databases so I would be unable to search for tweets without it being approved.	First Iteration
	4.1	Connect to the Twitter REST API	This is the basic part of running processes using Twitter's API. If I can connect to Twitter then I can find tweets relating to the user's search query. Now I have gained access to Twitter's databases.	First Iteration
	4.2	Connecting Twitter to Python	Using the credentials as a login, does Twitter allow access to Python to run my code using my created Twitter Development Account as a proxy	First Iteration
	4.2a	Writing the Twitter credentials in a file	A subclass should write the Twitter credentials into a file so it can be accessed by the main program. It is in a file for security purposes so someone cannot gain access to my Twitter credentials without them also having access to the file.	First Iteration
	4.2b	Reading the Twitter credentials from a file	The main program should be able to read and open the file written by the previous subclass and extract the necessary information from there like the consumer key for example.	First Iteration
	4.2c	Loading the Twitter credentials file	Once the file is read then it should be loaded up and the information requested by the code should be made usable in programs to carry out functions with that information.	First Iteration

Number		Identify what makes this successful	Justify	Complete
5		Run and load up MySQL	Am I able to connect to MySQL without any problems? This is where I am going to create a database. MySQL loads up with no errors.	Second Iteration
	5.1	Creating tables in MySQL	When I create tables then it provides the data structure to store any data that I would get from the data processing	Second Iteration
	5.2	Entering in the 100+ question words as the variable 'words' into the action table	This is what is used to specify the user's search query. These words are 100 of the most common English words therefore the user's search query is likely to be involved using one of these 100 words or any words to that effect.	Second Iteration
	5.3	Creating the structure of the other tables	This should when connected to Python, mean that the user input should slot in the variable 'phrase' and be stored within the tables	Second Iteration
	5.4	Integrating MySQL into Python	This allows us to connect user input and place that into the table and for us to use any tables previously created in MySQL in Python. This is useful for both storing and getting data.	Second Iteration
	5.5	Load up and store data in the database	Run MySQL instructions in Python using cursor – which is a variable that allows me to carry out MySQL instructions in Python such as SELECT, which means I can specifically call data from certain tables.	Second Iteration
6		Importing libraries	I need to use externally created libraries for some parts of the project instead of spending time manually crafting out already made functions	Second Iteration
	6.1	Importing mysql. connector	This is a library that allows us to connect to MySQL and without it I cannot connect Python to MySQL. This is similar to 3.3 however that is more specified towards carrying out functions with the data from MySQL compared to this, which is specifically to providing the bridge between MySQL and Python	Second Iteration

Number		Identify what makes this successful	Justify	Complete
	6.2	Importing Twython	This is a Python library which has been created to provide a way to access Twitter data. This is important for us to use as I need to access Twitter lists of Tweets being sent out and data on those tweets	First Iteration
	6.3	Importing JSON	JSON helps us to store the Twitter Credentials like the API key in a human-readable format but able to also be translated and understood by the main program.	First Iteration
	6.4	Importing Pandas	Pandas allows us to sort through all the data that I have gathered from Tweets quickly and efficiently as it is a data frame as well as a library to allow for analysis and manipulation of data.	First Iteration
	6.5	Importing Nominatim	I use this library to find the user entered location specifically the longitude and latitude and then use that to sort through the tweets filtering by distance from user entered area.	Third Iteration
	6.6	Importing Tkinter	Tkinter is a library specifically designed for aiding Graphical User Interface (GUI) design and this will help me in creating a User Interface to help the user visualise better what the program can do and what they, the user can do in progressing the program	Third Iteration
	6.7	Importing time	Instead of rapid movement of text, time allows me to slow it down such as for the updating questions when “No” is pressed, time will help to slow it down by a few seconds so the user can see it has changed.	Third Iteration
7		Incorporating Artificial Intelligence	By incorporating some sort of AI, this will allow us to become much more efficient providing the user a much more flawless experience as time passes.	Second Iteration

Number		Identify what makes this successful	Justify	Complete
	7.1	Storing count of number of times each word was selected	Whenever the user clicks “Yes” to a question, the count for that ‘word’ increments by one, this will then be useful for finding the most common query words.	Second Iteration
	7.2	User words being stored with the question word chosen with it	The query words will then be stored with the question words that was chosen. The less common words (e.g. if hospital and play were never paired together) will be then eventually be sorted out of the list, to reduce time.	Second Iteration
	7.3	Pairing common query words and question words together	This will then create a pair and the more pairs that are created then the more likely the AI can find the most common question words	Second Iteration
	7.4	Ordering the most common pairs first in descending order.	This means when that query word is entered then the most common pair word will be outputted first	Second Iteration
	7.5	Suggesting those pairs to the user	These pairs when they match the user query word will be suggested to the user and then this speeds up the overall process as the most likely question words to do with that query are likely to come up first.	Second Iteration
8		Outputting data to the user	The user has to see what is going on, therefore an output allows them to see the results of their search	Second Iteration
	8.1	Twitter handle of tweeter	The Twitter handle provides the user to see if they like the tweet then they can research more about that specific user and potentially find a new friend or customer helping out John my stakeholder.	First Iteration
	8.2	Location of tweet sent	The location of the tweet helps to find local issues for the user or if there is an event they want to find out about, it is much more personal if it is nearby rather than halfway across the world.	First Iteration
	8.3	Date of when tweet was sent	The date indicates how recent it was – e.g. if this was used to find out about the news then the date of tweet in seeing the reliability of that information	First Iteration

Number		Identify what makes this successful	Justify	Complete
	8.4	Text of actual tweet	This is the information of the tweet – the most important for many, this is how the user can see what the actual tweet is about.	First Iteration
	8.5	Number of likes on tweet	The number of likes on the tweet indicate popularity. However, as the program sorts out tweets based on location and date, this isn't the most useful feature. Although it can show my user whether the tweet is from a bot tweeting or not.	First Iteration
9		Pasting all this onto an accessible Microsoft Excel file	This pastes all the output onto an accessible Microsoft Excel sheet so the user can look at it more specifically. Furthermore, with every tweet there is a small hyperlink attached to see the whole tweet on the web browser. This feature would be very useful in particular for my stakeholder Victoria, who would be able to access any useful resources linked in the tweets for her teaching.	First Iteration

3 Evaluation

3.1 Post Development

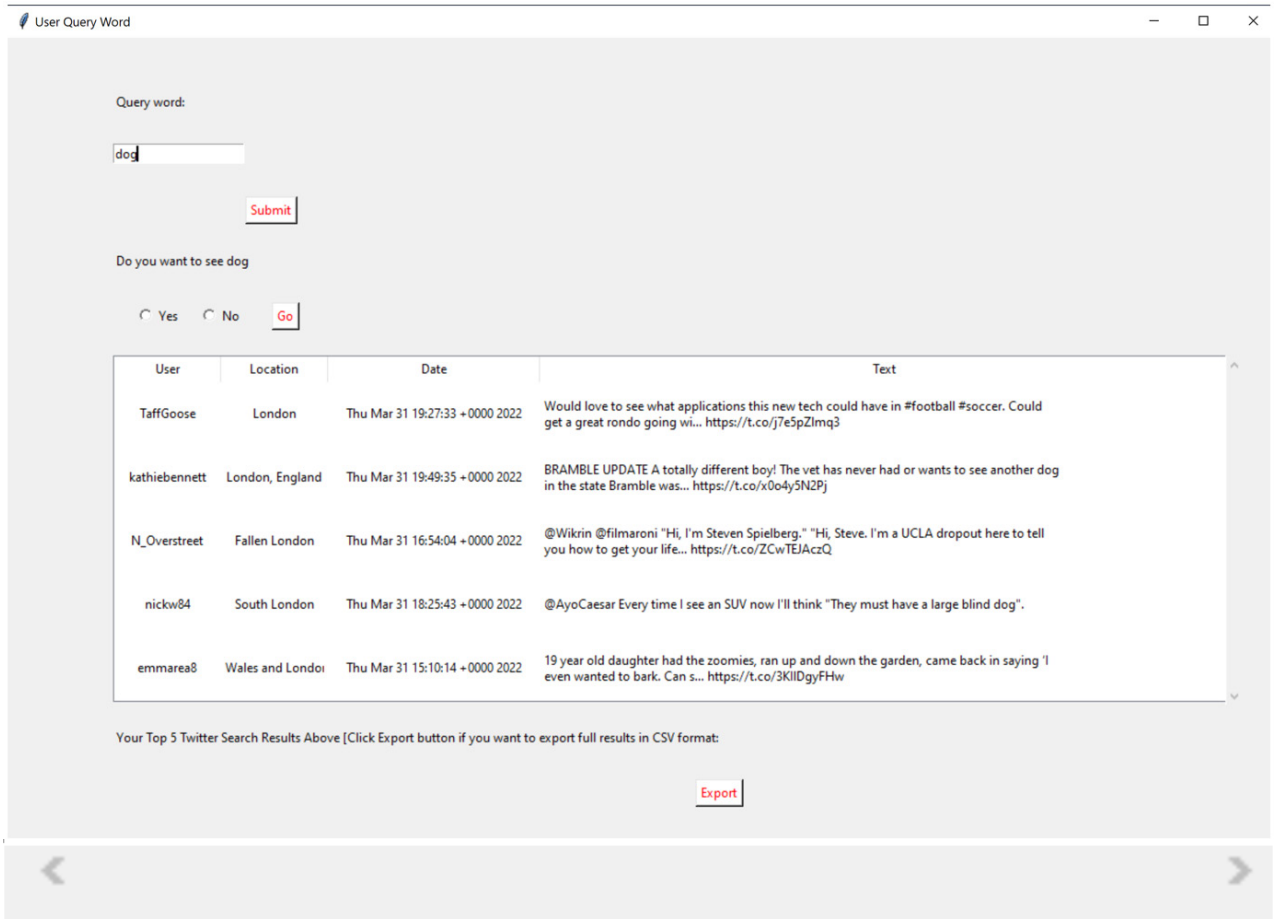
After my third iteration, I have accomplished my goal which was to create a user interface and connect to my main program. As a result, any input the user enters in the interface, is stored in the database and I have used their inputs as parameters for my numerous methods, creating a slick running of my program, outputting the data they require.

After my third iteration, I have accomplished every single point on the checklist, therefore indicating my project is a success from the initial criteria. The usability features such as the scroll button for example has appeared on the user interface, hence this was successful. This has allowed the user to scroll down to check any tweets that may be bigger than the boundary stated in the design of the user input.

In the post-development phase, I have added email validation, so it will not let the user progress until they put a proper email address in. One thing I noticed, in my research, is that in an email address, the domain name can change, the ending can change but the ever-present feature in every single email address is an @ symbol. So, I check if this @ symbol is present in my validation check.

This means Victoria's students can easily be identified as every child in the secondary school education system is given a unique email address and this can be used to identify them specifically, in case of any wrong doings.

Although, I have the power to limit certain searches on words such as those depicting violence. My stakeholder, John and I discussed the moral implications of limiting words that users can enter, since that is a form of censorship, we decided to allow the users to continue with their free right of speech.



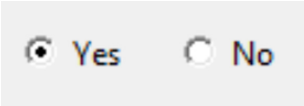
The scroll bar is a useful usability feature I have added to allow the user to scroll down the table when the text results exceed the dimensions assigned on the user interface. This means no information is hidden from them so they can read most of the tweet on the tables and if they want further details, they can use the other usability feature being the 'Export Button'. This provides convenience for the user since this allows them, if they want to find more about a specific tweet, they can click that and all the data in the table will automatically transfer over to the excel file for them to open and interact with.

Example of the Export Button:

There are also the radio buttons I have implemented successfully as a usability feature. These have the ability of creating a more efficient and quicker user interaction with the

interface while having the least mathematical number of inputs by the user. Therefore, reducing run time and increasing user satisfaction.

Example of Radio buttons:



What I would like to improve in further development was discussed with my shareholders, specifically Albert; it was the introduction of a speech-to-text tool. This is a wonderful idea to increase the range of people who can use the program such as those unable to type, however, existing products, which can be implemented into our program, require payment into a subscription service so this means my product may also require a payment to offset that monthly cost which is not a viable option as of this moment. A creative solution to this problem, may be to implement a camera and the user nods and shakes their head, depending on what letter of the alphabet they would like. This may be time-consuming but it is an effective alternative to those who cannot type, only requiring their head for input. Another usability feature I would like to improve on for the future is the load-up time for the program since currently it is very inconsistent and, on some occasions, can take over 10 seconds to load. Furthermore, I could use recursive programming to improve the program, however there is no need currently since the current structure of the program is already quite efficient but this can be considered for the future.

I would also like to implement changes such as having different difficulties based on the user's experience with programs, such as if there is a user who is not necessarily technologically astute then they can customise to have a simplified version of the program which only has basic features for ease of use – this would be perfect for Albert so I purposely made limitations on the user for simplicity. However, if there is a user who is more experienced and comfortable with dealing with more complex programs – like John – then I can add more customisation such as being able to change the tweet search radius from the default 50 miles.

Maintenance issues that could potentially arise is if the database is corrupted since the database itself has no security. It has been backed up, but even that backup is vulnerable to a virus. Therefore, in the future, I want to implement security protocols for the database itself to prevent it from maliciously accessed or corrupted.

Another issue I had to consider is the language I used to design the Twitter Crawler. I chose Python for its user-friendly language syntax and interface to provide convenience for not only me but future maintenance check by others, if I ever choose to outsource. To do this, I had to predict trends in users of Python, to see if it would increase or decrease in the future. After my research, I can say that Python user numbers are likely to increase in the future, therefore it will be much easier to find a steady influx of new users to maintain the program.

A limitation I had to deal with is the lack of options when developing a user interface using Python such as the libraries on offer were limited, with Tkinter being the only viable option due to its vast array of users. However, it does tend to have blocky features in its layout which is sometimes not aesthetically pleasing to look at.

3.2 Post-Developmental Testing

ID	Testing	Type	Data	Expected	Actual
1	firstName is gathered from the user interface correctly	Normal	firstName = “Karrison” print(firstName.get())	“Karrison” is printed in the console	“Karrison” is printed in the console **Passes**
2	firstName is gathered on the boundary – lower limit: 1 character	Boundary	firstName = “A” print(firstName.get())	“A” is printed in the console	“A” is printed in the console **Passes**
3	firstName is gathered on the boundary – upper limit: 100 characters	Boundary	firstName = “Adolph Blaine Charles David Earl Frederick Gerald Hubert Irvin John Kenneth Lloyd Martin Nero Oliver Quincy Randolph“ print(firstName.get())	“Adolph Blaine Charles David Earl Frederick Gerald Hubert Irvin John Kenneth Lloyd Martin Nero Oliver Quincy Randolph” is printed in the console	“Adolph Blaine Charles David Earl Frederick Gerald Hubert Irvin John Kenneth Lloyd Martin Nero Oliver Quincy Randolph” is printed in the console **Passes**
4	firstName is invalid – 173 characters	Erroneous	“Adolph Blaine Charles David Earl Frederick Gerald Hubert Irvin John Kenneth Lloyd Martin Nero Oliver Paul Quincy Randolph Sherman Thomas Uncas Victor William Xerxes Yancy Zeus” print(firstName.get())	“Adolph Blaine Charles David Earl Frederick Gerald Hubert Irvin John Kenneth Lloyd Martin Nero Oliver Paul Quincy Randolph Sherman Thomas Uncas Victor William Xerxes Yancy Zeus” is not printed in the console	“Adolph Blaine Charles David Earl Frederick Gerald Hubert Irvin John Kenneth Lloyd Martin Nero Oliver Paul Quincy Randolph Sherman Thomas Uncas Victor William Xerxes Yancy Zeus” is not printed in the console. For these users, they have an assigned shorter name so this could be used instead for when entering their details. **Passes**
5	firstName is invalid	Erroneous	“”	“” is not printed in the console	“” is not printed in the console **Passes** I have added verification where a minimum of one character is required

ID	Testing	Type	Data	Expected	Actual
6	lastName is gathered correctly	Normal	lastName = “Smith” print(lastName.get())	“Smith” is printed in the console	“Smith” is printed in the console
7	lastName is gathered on the boundary – lower limit: 1 character	Boundary	lastName = “E” print(lastName.get())	“E” is printed in the console	“E” is printed in the console **Passes**
8	lastName is gathered on the boundary – upper limit: 100 characters	Boundary	lastName = “Wolfeschlegel-steinhausen-bergerdorffvor-alternwaren-gewissenhaft-schaferswessen-schaftswarenwohl-gef” print(lastName.get())	“Wolfeschlegel-steinhausen-bergerdorffvoralternwaren-gewissenhaftschaferswessenschaftswarenwohlgef” is printed in the console	“Wolfeschlegel-steinhausenbergerdorff-voralternwaren-gewissenhaftschafers-wessenschaftswaren-wohlgef” is printed in the console **Passes**
9	lastName is invalid – 173 characters	Erroneous	lastName = “Wolfeschlegel-steinhausen-bergerdorffvor-alternwaren-gewissenhaft-schaferswessen-schaftswarenwohl-gefutternund-sorgfaltigkeit-beschutzenvor-angreifendurchihr-raubgierigfiends” print(lastName.get())	“Wolfeschlegel-steinhausen-bergerdorffvoralternwaren-gewissenhaftschaferswessenschaftswarenwohlgefutternund-sorgfaltigkeitbeschutzenvorangreifendurchihrraubgierigfiends” is not printed in the console	“Wolfeschlegel-steinhausenbergerdorff-voralternwaren-gewissenhaftschafers-wessenschaftswaren-wohlgefutternund-sorgfaltigkeitbeschutzen-vorangreifendurchihr-raubgierigfiends” is not printed in the console. This is the longest surname in history provided by the Guinness World Records but he also used a shorter surname, which I tested previously and that fit inside the boundary so there is no need for exception handling here. **Passes**
10	Postal Town and country are gathered correctly	Normal	postalTown = “London” country = “UK” Print(postalTown.get()) Print(country.get())	“London” and the “UK” are printed on the console on separate lines	“London” and the “UK” are printed on the console **Passes**

ID	Testing	Type	Data	Expected	Actual
11	Postal Town is on the boundary – 100 characters	Boundary	PostalTown = "Taumatawhakatangihangakouauotmateaturipukakapikikikikikikikikikikikikikikikikikikikikikikikikikik" print (postal_town.get())	"Taumatawhakatangihangakou Auotamateaturipukakapikikikikikikikikikikikikikikikikikikikikikikikikikikikikikik" is printed in the console When using the geolocation tool, this will return an error since this does not exist.	"Taumatawhakatangihangakou Auotamateaturipukakapikikikikikikikikikikikikikikikikikikikikikikikikikikikikikik" is printed in the console The longest postal town name in the world is only 58 characters long, therefore the boundary will not be a problem. For the sake of testing I have created a fake postal town to only test the boundary **Passes**
12	User details are stored in the Users table in the database	Normal	firstName = "Karrison" lastName = "Smith" email = "KarrisonSmith@mail.com" postalTown = "London" country = "UK"	I expect each of those variables stored under the appropriate headers in the Users table.	All those values were stored under the appropriate headers and were seen in multiple tables. Therefore, I now used the instruction select * from users where email = KarrisonSmith@mail.com and I only received these entries **Passes**

4 Bibliography

Anon., 2021. 100 Most Common English Verbs. [Online]
Available at: https://www.linguasorb.com/english/verbs/most-common-verbs/#:~:text=100%20Most%20Common%20English%20Verbs%20List%20%20,%20%20said%20%2021%20more%20rows%20

Anon., 2021. Accessing the Twitter API with Python. [Online]
Available at: https://stackabuse.com/accessing-the-twitter-api-with-python/

Anon., 2021. Exporting Dataframe to CSV. [Online]
Available at: https://datatofish.com/export-dataframe-to-csv/#:~:text=How%20to%20Export%20Pandas%20DataFrame%20to%20a%20CSV,within%20the%20code%20itself.%20...%203%20Conclusion.%20

Anon., 2021. Geocode Website. [Online]
Available at: https://geocode.xyz/SE18?region=UK

Anon., 2021. Geolocation Using Tywthon. [Online]
Available at: https://stackoverflow.com/questions/27567233/get-tweets-by-geolocation-using-twython

Anon., 2021. MySQL Foreign Key. [Online]
Available at: https://www.w3schools.com/mysql/mysql_foreignkey.asp

Anon., 2021. Searching Twitter by location. [Online]
Available at: https://thoughtfaucet.com/search-twitter-by-location/make-a-geocode-for-twitter-location-search/

Anon., 2021. The Python Code. [Online]
Available at: https://www.thepythoncode.com/article/get-geolocation-in-python [Accessed 02 10 2021].

Anon., n.d. Create GUI using Tkinter Python. [Online]
Available at: https://www.tutorialsteacher.com/python/create-gui-using-tkinter-python [Accessed 09 10 2021].

Anon., n.d. Python GUI examples TKinter Tutorial. [Online]
Available at: https://likegeeks.com/python-gui-examples-tkinter-tutorial/ [Accessed 09 10 2021].

Anon., n.d. W3 Resources. [Online]
Available at: https://www.w3resource.com/python-exercises/geopy/python-geopy-nominatim_api-exercise-2.php [Accessed 02 10 2021].

Awareness, C. B., 2022. Colour Blindness. [Online]
Available at: https://www.colourblindawareness.org/colour-blindness/

Staats, R., n.d. Designing UI with color blind users in mind. [Online]
Available at: https://www.secretstache.com/blog/designing-for-color-blind-users/ [Accessed 2022].