Instagram Analyser

By Jean Paul Jimenez Rueda

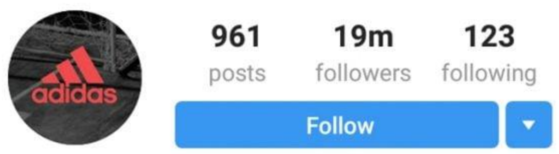
Content:

Analysis

Problem Identification.

Social media is growing; platforms such as Instagram, Facebook and Snapchat attract users every day. It has become a very useful tool to promote any sort of service and products. Currently the increase of social media over the last decade there has been a large raise on the amount of time people spend on these platforms.

Many use Instagram to promote their accounts but they need to make sure their account is targeting the right audience by analysing the gain/loss of followers.



There information given on followers is very basic and it does not give enough information on whether the account is gaining followers or loosing followers.

If a computational approach were applied to this situation, a system could be created where it will periodically log in all this data, analyse it, and display it back in a way the user can see the gain/loss of followers. The user will not waste time scrolling and looking for user who have unfollowed. The whole process will be automated.

Computational methods that the solution employs.

Computers are far more superior than human when processing vast amount of data and it makes it an excellent tool for this project for the following reasons:

- Speed - computers can process huge amounts of raw data much faster than the conscious human brain and can therefore analyse the number of followers in mere seconds.
- Accuracy - Computers do not make mistakes, they do not miscount the number of followers.
- Diligence -Computers can do a task indefinitely, no matter how redundant. No need to eat, sleep, or take breaks, they can monitor changes in the followers such as an increase or decrease.

The computer is suited for this solution because it can compare and calculate the number of followers to following ration in just mere seconds. The computer can monitor any changes in followers such as an increase or decrease and log these changes. It will return the data that has been analysed so that the user can understand. The user does not need to spend too much time and can work on other stuff while the computer automates the whole process. The solution lends itself well to numerous computational methods:

- Decomposition – it will be used to divide the problem's solution into smaller chunks for example, create a sub-procedure that monitors changes in the followers, it could also display the names of those who are not following the user.
- Abstraction – abstract thinking can be used in this solution, as Instagram has many features, the program only needs to focus on the following section and the numbers therefore the program can ignore content such as "users you may know".
- Heuristics – would be needed to know when to send a signal to the user about the followers such as sending a signal when the user has lost X number of followers or gained X number of followers.

Stakeholders.

The clients for my analytic software are users who use Instagram for business or personal purposes, and they will use it to determine whether their account is growing adequately or not. There are many reasons for why users may want to keep track of their followers, so I will narrow it down to users who have a business and want to use Instagram to promote their products.

An alternative client could be an influencer who uses Instagram as a form of social media marketing to make money as an induvial.

Kylie Jenner could be seen as an example of a famous influencer because she uses Instagram as to make money from sponsorships.

Kylie Jenner could use my system to see if her sponsorships are being successful. This will not only help the company she is promoting but also help her to gain more sponsors by showing that previous sponsors were successful and therefore make her a reliable influencer and sponsor. She could also use this information to calculate how much she will charge sponsors based on their success.

I had a conversation with a small business, Cocoatco who uses Instagram to upload pictures of her cat and promote cat products. I will ask her what she thinks will be useful to implement in my system. This will help meet the user's requirements.

The user should have some coding experience to able to able to change some values such as log in if they wish to switch accounts.

Interview with @Cocoatco

What is the problem with current Instagram regarding the growth of your business?

She stated that "Is very difficult to review the account if you are not associated with a large account". This means that my program will be of good use to her as it will expand the information gathered from Instagram.

What features would you want to see in my system?

She stated that "I want able to see the correlation between the followers and my posts". This could be a feature that can be implemented for the purpose of growing the business.

She also said that she would like to see graphs being involved in the system which that are easy to view and allows her to visually understand the numbers and view trends.

She mentioned that a good tip would be to have a home page which shows everything in a simple way because she said that sometimes she will not have enough time to look at everything.

In the interview she said that "I would like to see which posts are more popular, is not necessary going to help the business but it helps with making similar posts". It could be a useful feature to have.

She also said that she would like the program to be light and not use too many resources as she would like to do other stuff in the background, this means the program may need to be adjusted to use as little resources possible.

To summarise the interview and what she wants is:

- Graphs
- A homepage
- Ranking System
- A light program

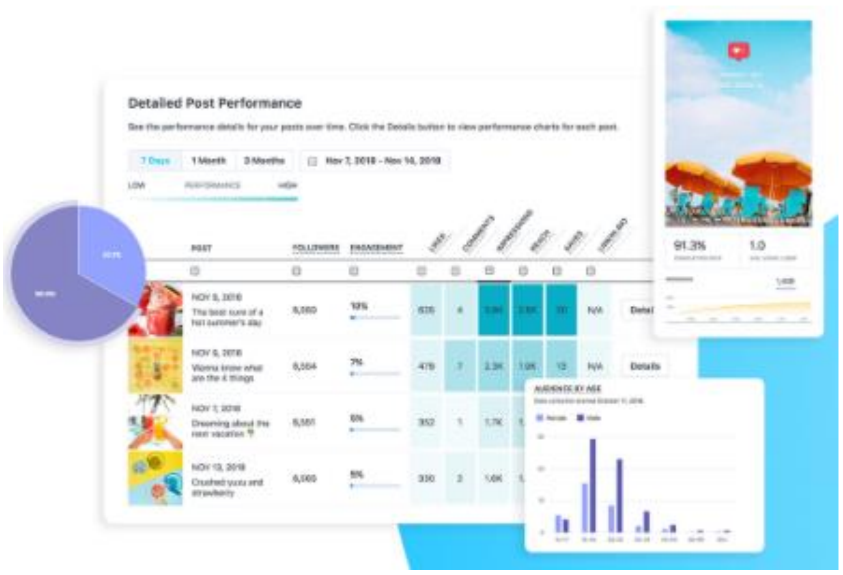
Research the Problem

There are numerous existing solutions which are relevant to the project. I will use these existing programs as inspiration for my project.

An existing program called is called "later" and can be found at www.later.com

This is an analytics webpage for Instagram that is very advanced and detailed.

There are some good features in the system such as:



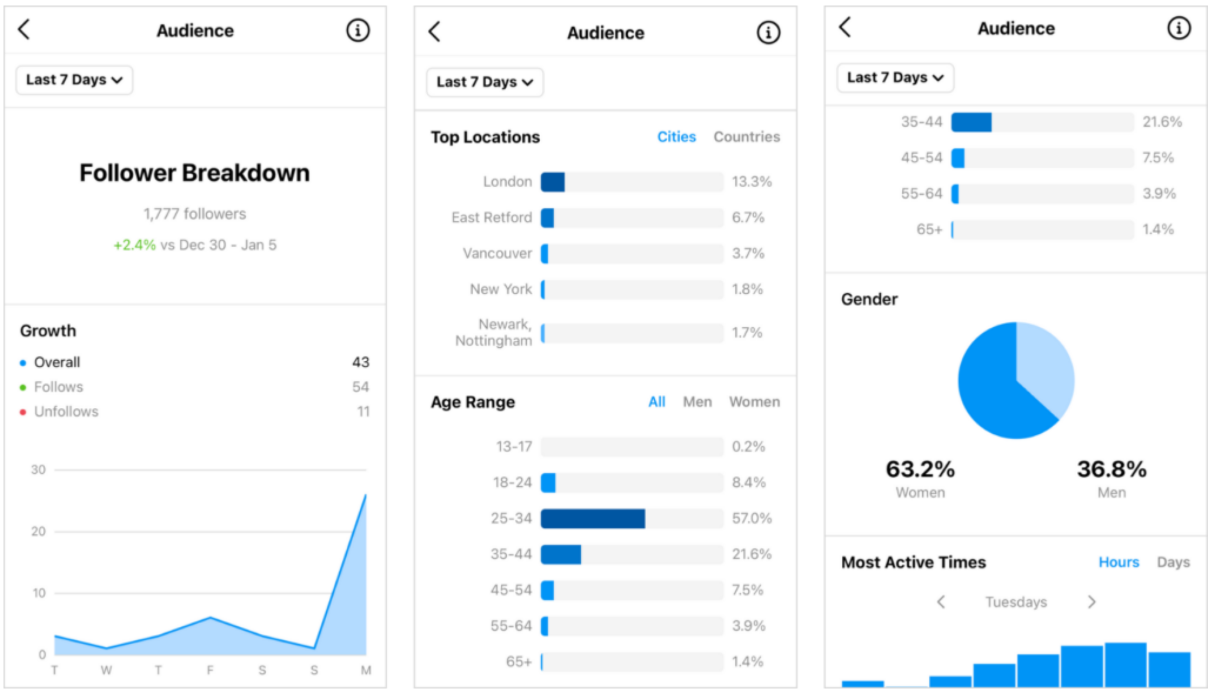
• Having a ranking system of posts which allows the user to see what posts have been more successful.
• Having graphs which allows the user to see a visual representation of followers.

I will take inspiration and have similar features in my program however, I believe there is a lot for the average user, and it comes at an expense therefore, I will use abstraction to only focus on what is really important like making sure the graphs are easier to understand and use colours that can be viewed.

I believe this system is has way too many features to satisfy the needs of my client and therefore I will not include some of the features it has such as:

- Demographic data
- Hashtag analytics
- Optimizing post times

Another existing solution is the built-in analytics feature Instagram has made. The problem is that in order to access this feature the account has to be activated in "business mode" this means that the account loose privacy and some users may want not to want to lose privacy. This is what it looks like:



There are some good features on this program such as:

- Very easy to understand information.
- Good categorisation to keep track of specifics.

This is a very good example of a simple yet effective program that tells you the important information unlike the first example is that is not as "flashy" meaning it does not have graphs and numbers everywhere.

My system will not have information such as "gender" because in order to access that information. The system needs access to the Instagram database which I cannot access.

Proposed Solution

My proposal is a system that can show what users are not following back. The program will also have additional features that were suggested by my client in the interview such as having graphs for visual representation of quantitative data, having a GUI which will allow the user to log in into their account. Instagram has basic information and what my program will do is simply add a bit more detailed information.

All of this will be achieved by using Selenium which is a web atomisation program that is useful for scrapping data and other functions, it will specifically be used to automate the log in process which logs the user into their Instagram account. Using PyQt5 for the GUI which will be simple yet sufficient for the users need.

Essential Features

After reviewing some existing solutions and taking into consideration the interview with Cocoatco I have concluded that in my system will incorporate the following features which will satisfy my client's needs as well as improve existing solutions to create the program. The features are:

- Monitorisation of follower count

This will display the user the names of the users who are not following back.

- Display graphs

Graphs are the first step of analysis work and my system being an analyser is essential to have graphs. This will send back data in a way that the user can visually understand and can help see patterns that are often hard to see.

I will be using matplotlib which is a visualisation library which I will be using to make graphs, it is very popular and easy to get started with.

- Automated log in

The system will have an automated log in meaning that the user will not have to spend instead, they will be promoted with a window which will allow the user to enter the username and password and system will automate the log in process.

Justification

All these features will make the system solve the problem, to add more detailed follower information. The automation will be the core behind the scrapping and log in process. I believe these features will be enough to satisfy the stakeholder.

Limitations

As my system needs to be simple enough to meet the requirements of my stakeholders, compared to the existing solutions, having too many features may be overkill for the use of my stakeholder so I will limit some of the features such the demographic data and other features that were previously mentioned. This will also prevent the project from becoming too complicated and be too hard to program.

The GUI should not be very complex, one of the solutions had too many graphs which could confuse the user and may not be necessary. Using PySimpleGUI I will create a

GUI which will be user friendly. I will keep in touch with my stakeholder and ask on design feedback and if she wishes to have more or less.

Security could be a problem for the user as the user will have to enter the log in details which will be stored within the program. The log in details will be used by selenium to automate the log in process.

Returning the names of the those who do not follow the user could become long process if the user has too many followers as the program will need to first scroll through both following and follower list. With an average of 100 user the system will take 1 minute to run the whole program return the names. If the user has more than 1000 followers, the program will take 10 minutes to load. Not very fast, this means that the program will be slow.

Finally, I will not be able to add multiple accounts to be analysed. The user will have to logout of the current account within the program and log in to the new account that wants to be analysed.

One of my stakeholder's ideas was to add a system which shows the correlation between followers and posts. I do not think I will be able to do this as it falls out of my programming skills and it may be too complicated.

System requirements

The system will not be too complicated for the computer to run. I believe a simple standard computer will be enough to run it.

Hardware:

- Standard Computer/laptop – nothing fancy. Most computers/laptops are powerful enough to be able to run this program.
- Access to WIFI – If there is no connection made to the internet the user will not be able to use this program as it needs to connect to the internet and Instagram.

Software:

- Python interpreter – As the code will be written in python.
- Operating system that can support python – All operating systems support python so this should not be a problem but is worth noting.
- Selenium – Very important for the automation process.
- PyQt5 library – This will allow the graphs and GUI to be visible.
- Matplotlib library – Used to show graphs.
- Chromedriver – Important for the automation to work.

Other requirements:

- An Instagram account – without an Instagram account this program is of no use. The program is created around Instagram.

The minimum computer specifications:

- Minimum processor rate = 1.5Ghz
- Minimum ram = 2GB
- Minimum disk space = 500mgb

For the computer specs I have used my old laptop spec's which I will be running the system in for testing. I have a far more powerful computer, so I have decided to use minimum specs which make refence to my laptop.

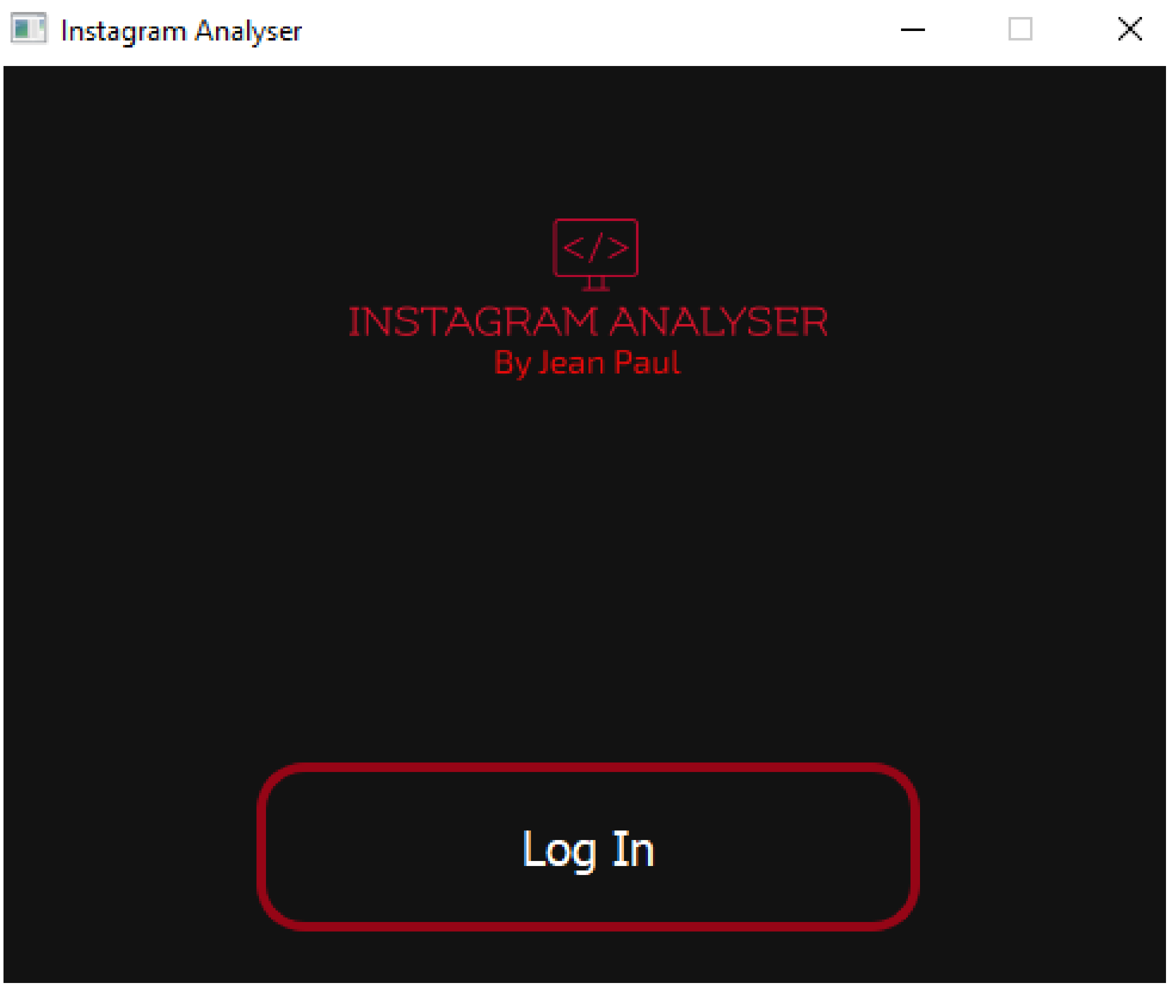
Success criteria

Core:

Criteria	Evidence
Selenium driver to open web browser.	Screenshot from the program code and browser showing that is being run by an automated service.
Selenium driver being able to click and type.	Screenshot from the program source and the outcome.
Automated log in.	Screenshots from the programming editor to the Instagram log in page and screenshot once logged in.
Selenium driver able to scrape data	Screenshot from the programming editor displaying information from the internet.
Simple GUI	Screenshot from the GUI window
Selenium driver able to scrape data and display back to the user every so often.	Screenshot from the code editor showing last time the scrape was done.
Display everything scraped	Screenshot from the code editor

Design

Main Menu Graphical User Interface:



This the main menu for my design which will be displayed when the user runs the code displays a logo and a button. The button allows the user to log in and run the program with one simple click.

Justification:

I made sure the button was big, so it is easy to see and hard to miss. Using a border will help with visualisation as the colour of the button blends with the background. Also, the system only displaying this button will eliminate all sorts of confusion.

How it links to success criteria:

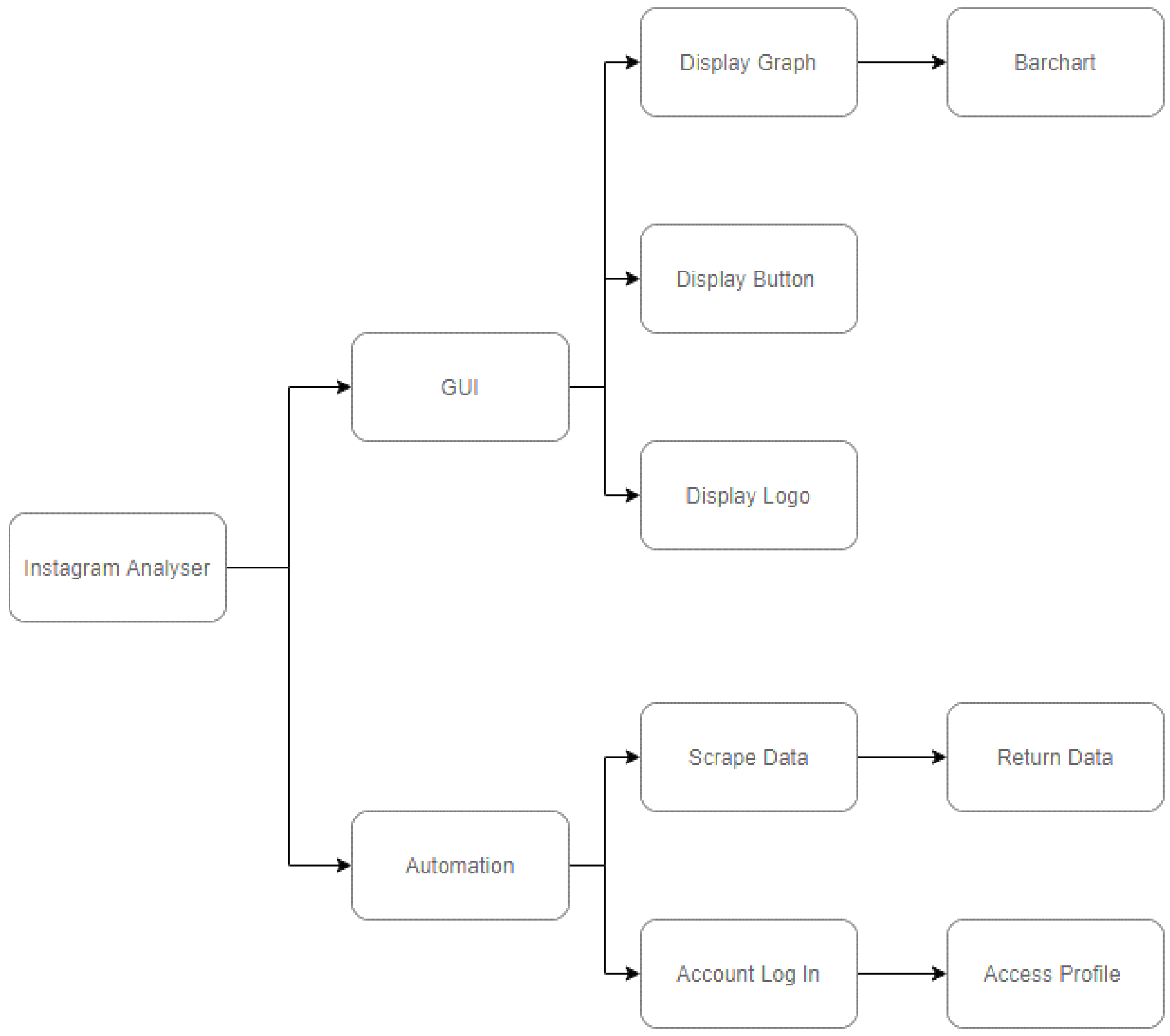
- Simple GUI

The design of this menu is very minimalistic and will get the user logged on with just a single click. With only one button being displayed the user cannot go wrong.

Log in Button:

Once pressed it will be able to call the class which logs the user in and scrape data.

Decomposing the problem into smaller problems:



This diagram illustrates the different processes that occur to make the program work. In reality, the program functions in a linear way meaning that the program will first call one function then that function will call another and so on. However, this diagram show the two main functions and their sub functions.

I will explain these in further detail:

Explanation:

- GUI – Used to display the main log in screen which will be the first thing that runs in the program.
    - The logo and button are displayed and then display charts.

- Automation – Selenium is used to automate. It will automate log in and will also scrape data from the web.
    - First accesses the account then the profile.
    - Scrapes the data and returns the data

Pseudo Code of functions:

Automation:

```
Path.driver = (C:\driver.location.exe) #location in directory for
driver

Class InstagramBot(self):

      Def __innit__(self):

            Self.drive = browser(Path)#Opens web browser

            Self.driver = search(“website.com”) #Searchs web site

            Self.driver = selectFieldByXpath(websiteXpath);

            click()

            Wait (1) #Lets page load

            Log_in() #call function


      Log_in (self):

            Self.driver = selectFieldByXpath(usernameBoxXpath);

            Click() #find username text box

            Self.driver = insertUsernameFromLogging(Username) #types
the username

            Self.driver = selectFieldByXpath(passwordBoxXpath);

            Click() #find password text box

            Self.driver = insertPasswordFromLogging(Password)

            Self.driver = selectFieldByXpath(EnterButtonXpath);

            Click() #enter password

            AccessProfile() #call function


      AccessProfile(self):
            Self.driver = selectFieldByXpath(profileiconXpath);

            Click() #clicks profile icon

            Self.driver = selectFieldByXpath(profileTextXpath);

            Click() #cicks access profile page


      Compare_Followers(self):

            Self.driver = selectFieldByXpath(followingButtonLabel
      Xpath);

            Click() #find following section

            Following = self.Get_Names() #call function

            Self.driver = selectFieldByXpath(followersButtonLabel
      Xpath);

            Followers = self.Get_Names() #find follower section

            Not_Following_Back = users in following if user not in
      followers #stores names in a variable

            Print(Not_Following_Back) #print variable

      Self. Get_followers_Count()


      Get_Names(self):

            Wait(1)

            Scroll_box = findFieldByXpath(scrollboxDimensionsXpath);
#finds the scroll box

            Last_height, height = (0,1) #sets values

            While last_height ! = height: #if not same

                 Last_height = height #new height value is added,
used to determine end of scroll box

                 Wait(1):

                 Height = self.driver.execute.script("scroll_down",
Scroll_box

            Name_datatype = scroll_box.findFieldByTagName('a')

            Names = [name.text for name in name_datatype if name_type
!= '']
```

```
            Return names.

      Get_followers_Count(self):

            Following =
Self.driver.find.element.by.ID(intigerOfFollowingID)

      Followers =
Self.driver.find.element.by.ID(intigerOfFollowersID)
```

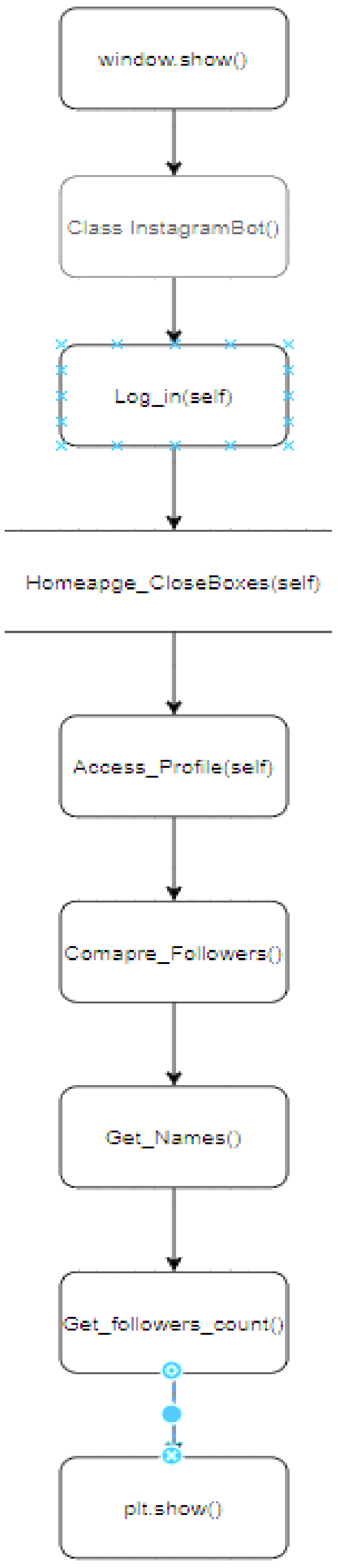
GUI:

```
Import all necessary libraries

App = QApplication(sys.argv) # system information

Window = QWidget() #Sets the window as a widget

Window = windowtitle(Instagram Analyser)

Window = setDimensions (500)

Window = windowStyle (background_color: dark gray)


Grid = QGridLayout() #enables adding widgets to window


Image = QPixmap (“Logo.png”) #sets image as a variable for location
of png

Logo = Qlabel() #sets logo as a label

Logo.setPixmap(image) #makes the logo the image file

Logo.alignmet (center)

Logo.logoStyle(margin-space: 50px)


Button1 = QPushButton(“Log in”) #text in button

Button1.clicked.call(InstagramBot)

Button1.setCursor(PointingHand)

Button1.buttonStyle{

      Border: 4px;
      Color: ‘red’;
      Font.size: 20px;
      Font.color: ‘White’;
      Padding: 20px;
      Margin: 30px 100px;}
      *hover{color: ‘light red’;}
}

Grid.addWidget(logo, 0, 1)
Grid.addWidgetButton(button1, 0, 1)
```

Graphs:

```
Fig = plot.figure(figureSize = x)
Names = [“followers”, “following”}
Follower_count_Value = [x, follower_count] #varaible taken from
extracted data

Following_count_value = [following_count, x] #varaible taken from
extracted data

Position = [0, 1] #positon of first bar
Position2 = [0, 1] #postioon of second bar

Plot.bar = (position, follower_count, width = x, color = x)
Plot.bar = (position2, following_count, width = x, color = x)

Plot.xticks = (position, names)

Plot.show()
```

Flowchart of functions working together:



The structure of the solution:

Input	Process	Output
Display GUI	Load function	Display logo and button
Click Log in	Opens IG website and enters log in credentials	Allows user to enter the system and see data
Access Profile Page	Loads code that selects profile	Shows profile apge
Click Followers Screen	Show following/follwers	Shows the follower screen including all the followers and following
Scrape Data	Return names	Display all names in terminal
Load Graph	Display	Barchart

I will use python for coding, and I will be using selenium which is a suit of tools to automate web browsers. Selenium will allow me to create a code that opens chrome and search for Instagram log in page. In the page the user will enter the credentials in the program and selenium will read this data and enter it on the website. If correct, Instagram will allow access to the account and then the system will begin to scrap data.

USABILITY:

The main screen will be designed in a way that is simple, making it easy for the user to understand. It will have a brief overview of all the data and clear buttons allowing the user to see and press without trouble.

Classes:

Log In:	Data Type	Why?
Username	STR and INT	A username may contain characters and numbers
Password	STR and INT	A password may contain characters and numbers

Describe the approach to testing:

What will I test?	What should happen?
Testing that log in will open Instagram	Instagram should open in the log in page allowing the user to enter their details
Testing that if login is correct the Instagram log in page should close automatically.	Instagram should close for the user but stay open for the system to scrap data from.
Testing that the GUI loads correctly	The GUI should display graphs and display data that was gathered from the IG from the web.
Testing that Selenium scraps data properly	Selenium should gather the appropriate data in the appropriate Data Type, if its followers it should load Integer

Development and Testing

Test Plan.

To begin with, I well take a 'module by module' approach to develop the system. I will be testing my code as I develop at different stages it by performing alpha testing to test each module at a time. In the end every module should be put together and the system is expected to be robust and work.

I will begin with simple prototypes which get the core module working (nothing fancy, just the basics of for example a GUI it will just be a window), then will improve this prototype to get a better working module that will be used in the final code. Having simple prototypes will give me a good confidence boost and give but more importantly give me a base to build up from. I believe having the core code is really important instead of diving head straight into creating the full code.

I will be showing screenshots of blocks of code and errors with some annotation then explaining what that block of code does.

Iteration Number	Test	Test description
1	Log In	Opening Instagram login page using Selenium and entering username and password. Once logged in it will show home screen.

2	Scrape Data from Web to The System	Read the data correct data from web. Send data back to the system. Display correct data to the system
3	Simple GUI Prototype	Create a simple GUI using PyQt5 showing a button
4	Display Charts	Use the library matplotlib.pyplot to create graphs. Use the data from web as input values. Display graph correctly

Iteration 1(Log in).

Logging in is essential, is a very important step that has to be achieved for the system to work.

In this iteration I will focus on developing the Log in module which will consist of:

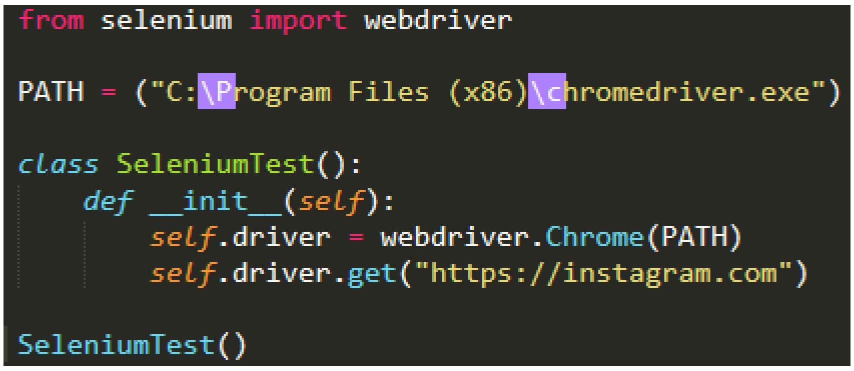
- Opening google chrome.
- Instagram log in website is entered.
- Username and password are entered.
- Instagram home page is shown on screen once successfully logged in.

Opening chrome & Instagram

To tackle this iteration, I will begin by making a prototype with little annotation to make sure the program is working.

Initially I need to set up Selenium with the right drivers and Path to work.

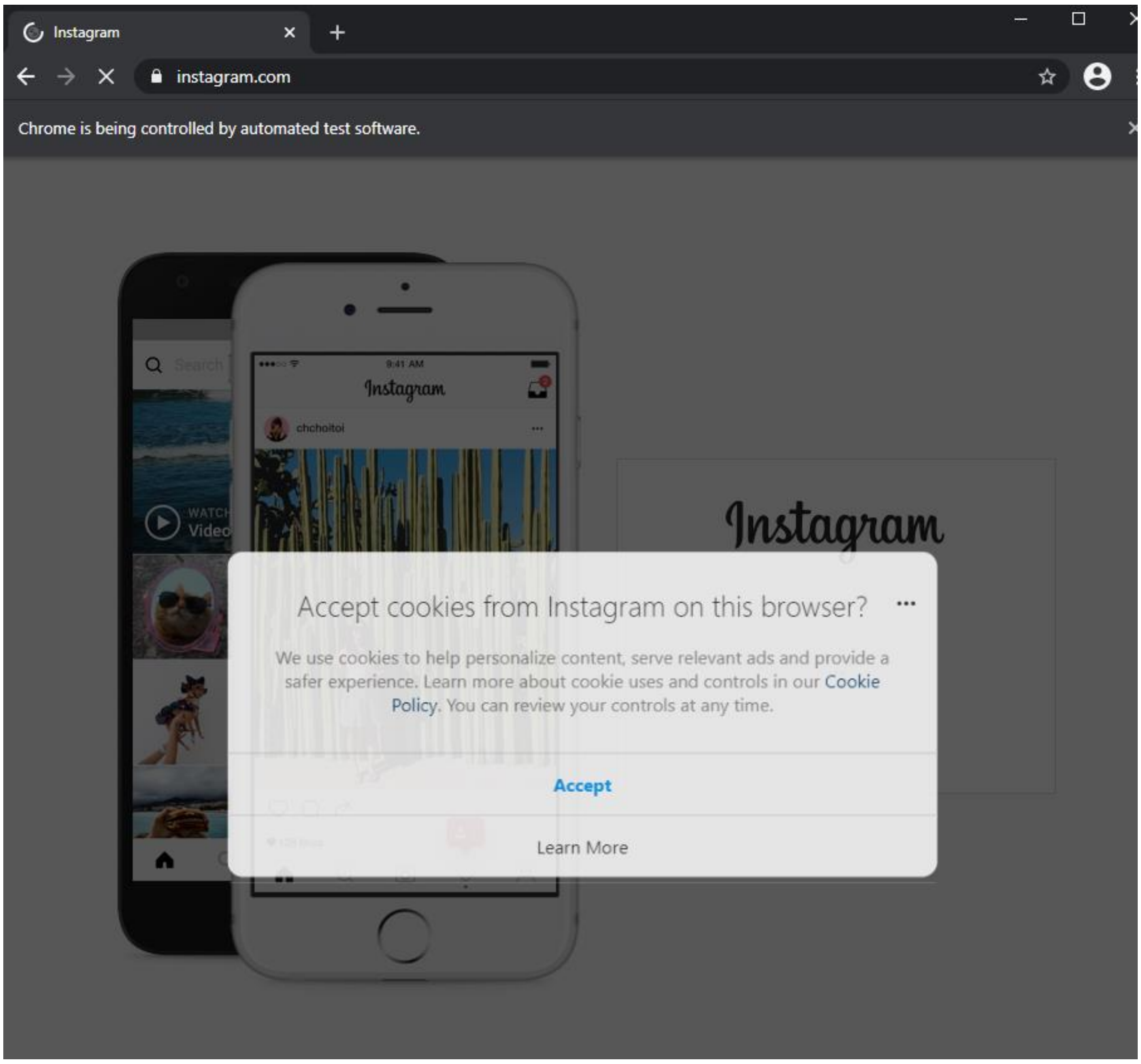
Here is the initial code:



```
from selenium import webdriver

PATH = ("C:\Program Files (x86)\chromedriver.exe")

class SeleniumTest():
    def __init__(self):
        self.driver = webdriver.Chrome(PATH)
        self.driver.get("https://instagram.com")

SeleniumTest()
```

This code is the core of the system, it demonstrates that selenium is working and running. The code opens google chrome through automation straight to the Instagram website however, as soon as the page loads the web browser closes.

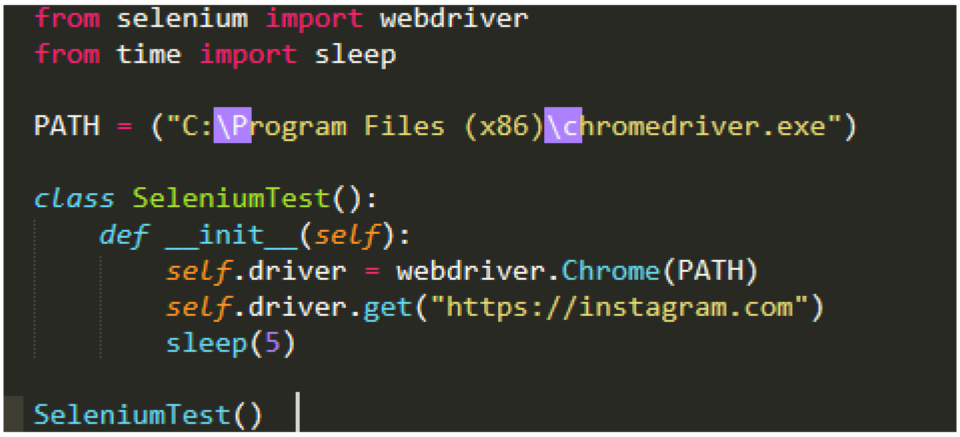
This is what the code displays:



Accept cookies from Instagram on this browser?

We use cookies to help personalize content, serve relevant ads and provide a safer experience. Learn more about cookie uses and controls in our Cookie Policy. You can review your controls at any time.

Currently, this is the screen that is displayed. To overcome the problem of the web browser closing as soon as it is loaded, I will have to import the time library which will give the browser a set amount of time to be displayed before it closes. In order to achieve this, I have made this code:



```
from selenium import webdriver
from time import sleep

PATH = ("C:\Program Files (x86)\chromedriver.exe")

class SeleniumTest():
    def __init__(self):
        self.driver = webdriver.Chrome(PATH)
        self.driver.get("https://instagram.com")
        sleep(5)

SeleniumTest()
```

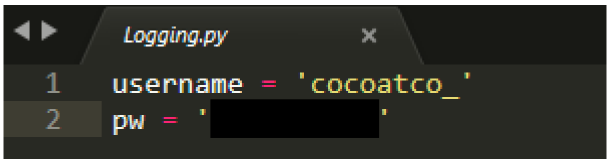
Once tested the web browser is displayed for 5 seconds before closing.

User Log in

The next step is to get the user logged in.

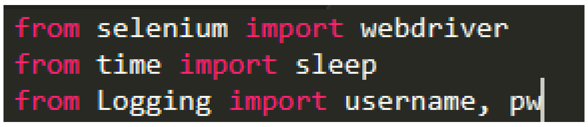
My program is automated, but it does not mean it knows what to do. I need to instruct Selenium on what buttons to press and what/where to enter in the username and password boxes.

The username and password will be stored in a different file which will contain username and password as shown here:



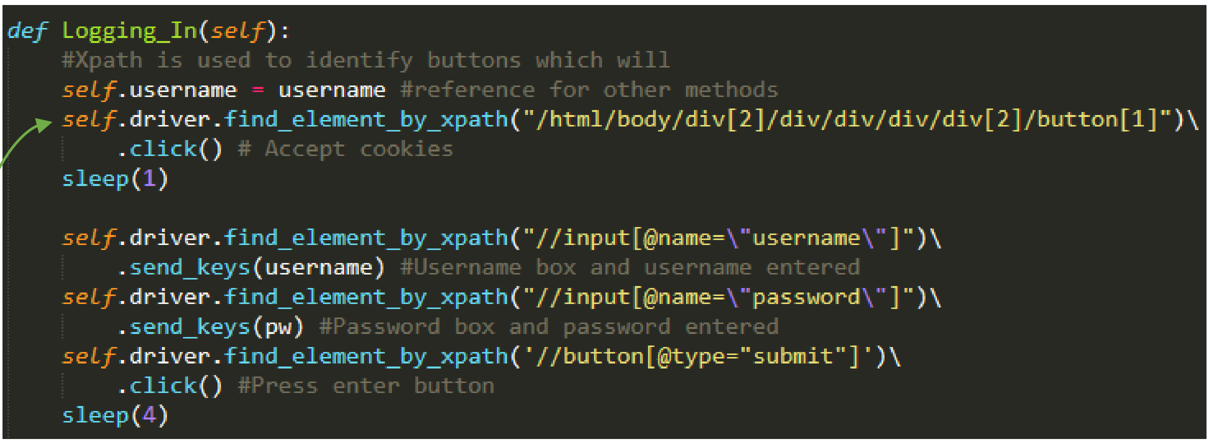
```
Logging.py
username = 'cocoatco_'
pw = '[redacted]'
```

The username and password will now need to be imported to the main file which contains the code:



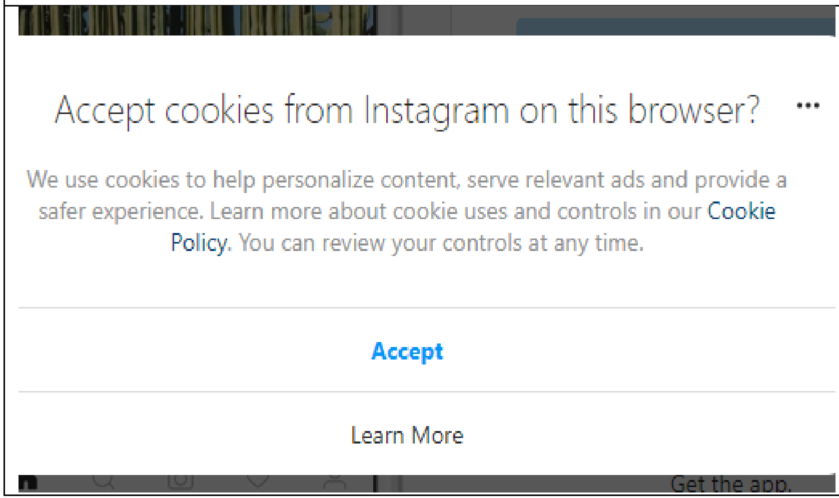
```
from selenium import webdriver
from time import sleep
from Logging import username, pw
```

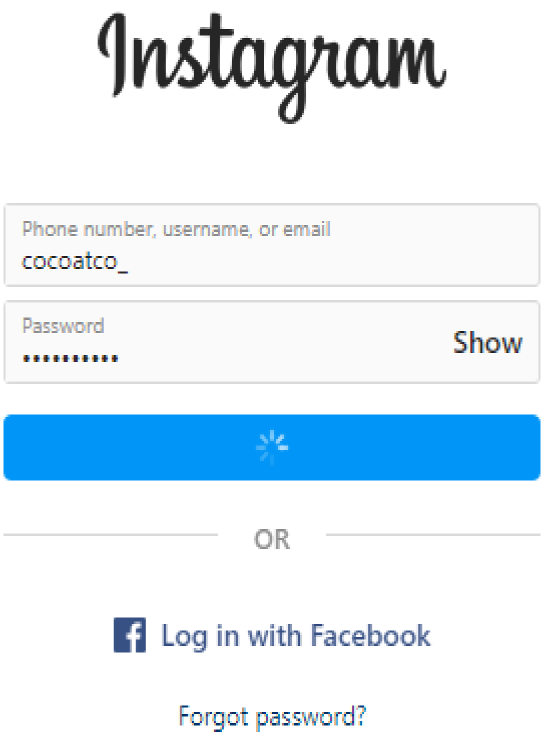
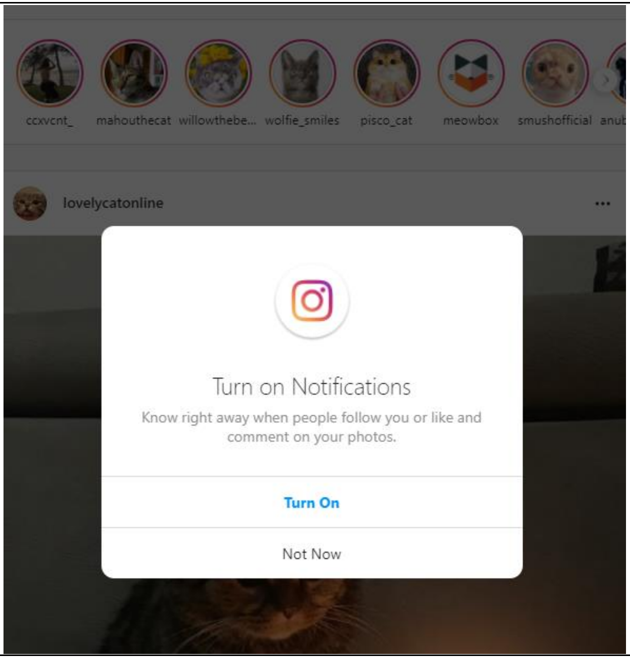
To make the program automate the process of logging in I have created this function in the within the class:



```
def Logging_In(self):
    #Xpath is used to identify buttons which will
    self.username = username #reference for other methods
    self.driver.find_element_by_xpath("/html/body/div[2]/div/div/div/div[2]/button[1]")\
        .click() # Accept cookies
    sleep(1)

    self.driver.find_element_by_xpath("//input[@name=\"username\"]")\
        .send_keys(username) #Username box and username entered
    self.driver.find_element_by_xpath("//input[@name=\"password\"]")\
        .send_keys(pw) #Password box and password entered
    self.driver.find_element_by_xpath('//button[@type="submit"]')\
        .click() #Press enter button
    sleep(4)
```

Before the user can enter its credentials, a box requesting cookies is shown, to overcome this obstacle Selenium finds the "accept" button/element through the Xpath then clicks, it waits 1 second to allow for the username and password boxes to appear. This is what is displayed:

What is displayed from Test 1 when run	Description
 Accept cookies from Instagram on this browser? We use cookies to help personalize content, serve relevant ads and provide a safer experience. Learn more about cookie uses and controls in our Cookie Policy. You can review your controls at any time. Accept Learn More	Cookies permission box is displayed. It has to be accepted for it to disappear and be able to enter username and password

 Phone number, username, or email cocoatco_ Password •••••••••• Show OR Log in with Facebook Forgot password?	The accept button has been clicked. The program has entered the username and password and has clicked the enter button. It takes the username and password from the imported variable "username" and "pw".
 Turn on Notifications Know right away when people follow you or like and comment on your photos. Turn On Not Now	The user has successfully logged in and the home page is being displayed. Turn on notification pop up box is automatically shown when the user logs in.

Test 1 (Log in) has been complete and successful.

<u>What has been done:</u>

I have initially made sure Selenium was working and running by creating a small code which opens the web browser and searches https://instagram.com. Then I proceeded by making a function which instructs selenium to find the username and password boxes and enter the username and password. The username and password are imported variables from another file. It then inputs the username and password, and presses enter. Once Instagram checks if the username and password match with their database the user logs in.

Summary of the system so far:

The program is able to log the user in through an automated process.

Success criteria met:

- Opening google chrome.
- Instagram log in website is entered.
- Username and password are entered.
- Instagram home page is shown on screen once successfully logged in.

All have been met and iteration 1 is working as expected.

Iteration 2 (Web Scrapping)

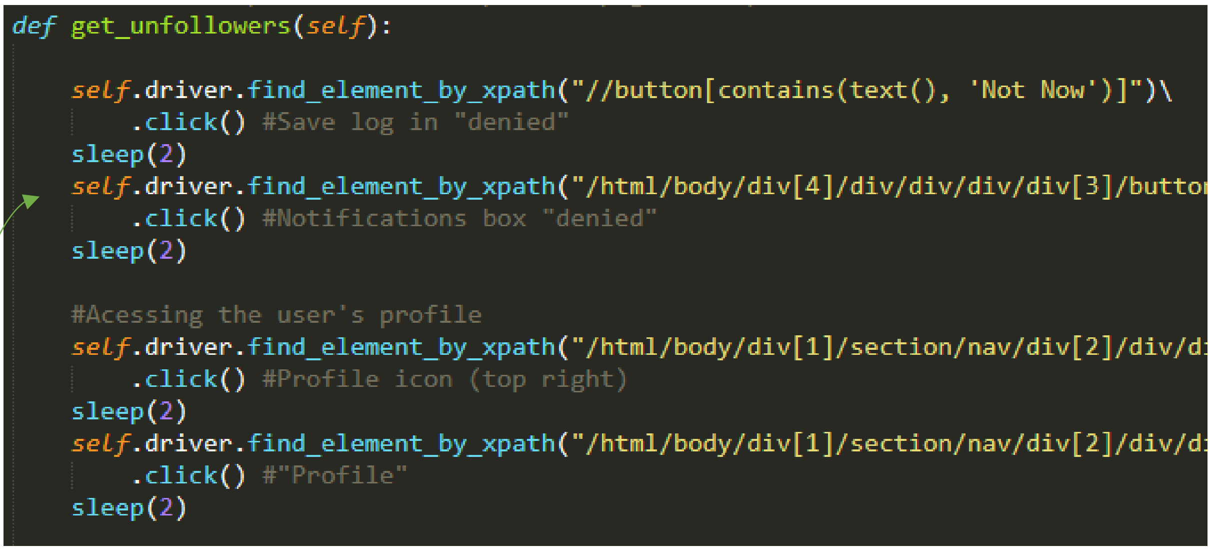
In this iteration I will focus on web scrapping because is important to meet one of my essential features which was to monitor follower count.

In this iteration I will develop the web scraping process module which will consist of:

- Clicking "not now" on the notifications box.
- Accessing the user's profile page.
- Accessing the following and followers
- Comparing following to followers
- Returning the names of those who the user follows but are not following the user back.

Not now boxes

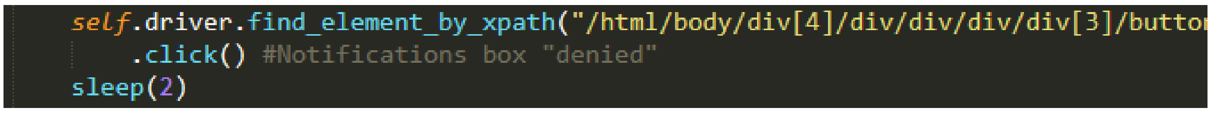
Using the code that has been previously developed to log the user in, I will add a function which will access the user's profile page once the system has logged, here is what the code looks like:



```
def get_unfollowers(self):

    self.driver.find_element_by_xpath("//button[contains(text(), 'Not Now')]")\
        .click() #Save log in "denied"
    sleep(2)
    self.driver.find_element_by_xpath("/html/body/div[4]/div/div/div/div[3]/butto
        .click() #Notifications box "denied"
    sleep(2)

    #Acessing the user's profile
    self.driver.find_element_by_xpath("/html/body/div[1]/section/nav/div[2]/div/d
        .click() #Profile icon (top right)
    sleep(2)
    self.driver.find_element_by_xpath("/html/body/div[1]/section/nav/div[2]/div/d
        .click() #"Profile"
    sleep(2)
```

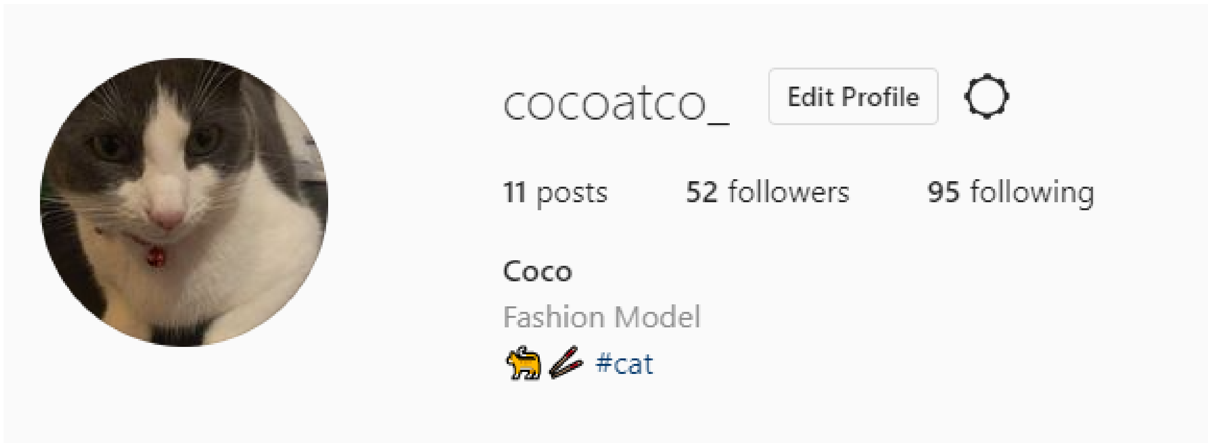
Some of the Xpath code was too long to fit in the screenshot and so it has been cut out to maintain good image quality and nothing important was cut out.

Previously, we had a problem where once the user was logged in (Iteration 1) a notification box would appear requesting access to display notifications on the user's web browser. That was solved by finding the Xpath of the button and clicking No. Here is the code:



```
    self.driver.find_element_by_xpath("/html/body/div[4]/div/div/div/div[3]/butto
        .click() #Notifications box "denied"
    sleep(2)
```

This function clicks the user profile picture which his located on the top right side of the home page and loads the users profile page as shown:



The program can now access the user's profile page which will be used to look at followers and following.

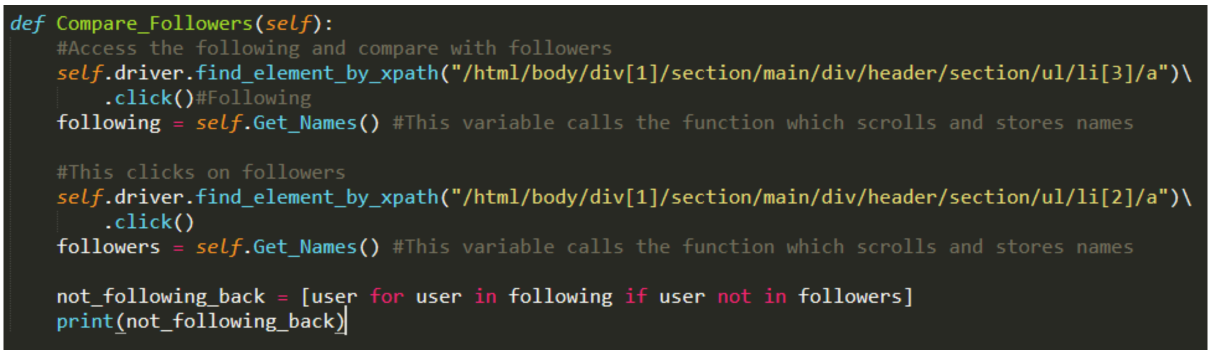
Comparing followers and following

Next step is to make the program compare the users who can be seen in following to followers. The way the program will achieve this is by storing the names whilst is scrolling down “following” the data box. It will then store the names of the users who can be seen in following whilst scrolling down the “followers” data box. Once it has scanned both lists, it will simply compare the names in both lists and return the names of those who were seen in one but not another.

I will create two functions:

- Function 1 - compare the names of users.
- Function 2 – scroll through both followers and following whilst storing names.

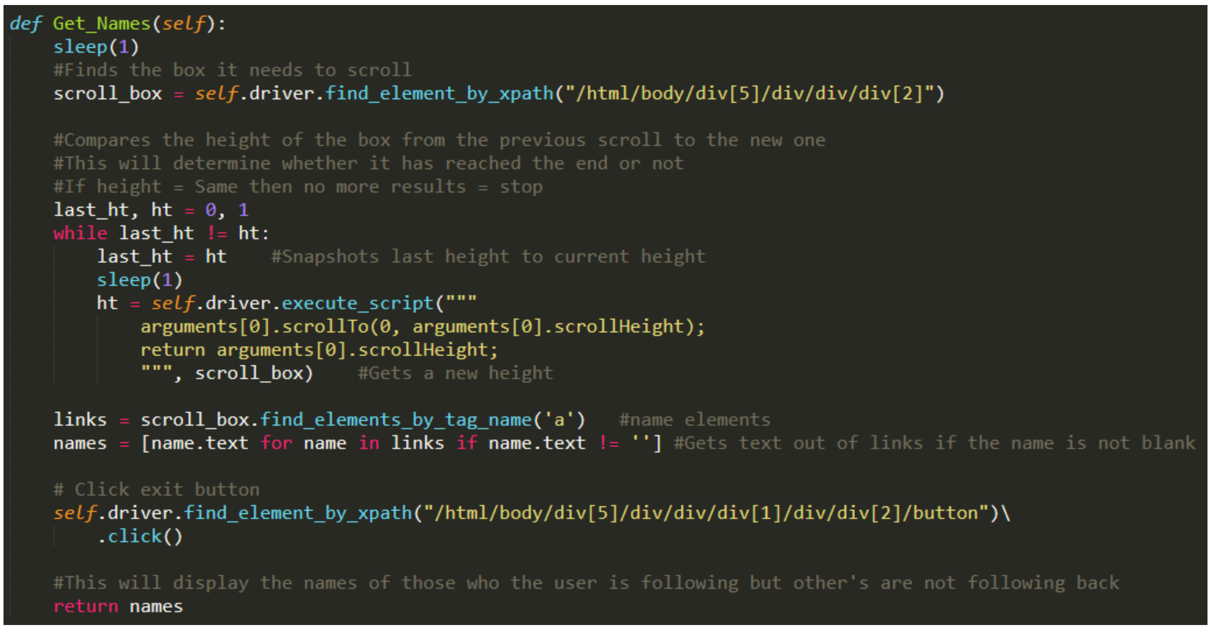
Here is the code for Function 1:



```python
def Compare_Followers(self):
    #Access the following and compare with followers
    self.driver.find_element_by_xpath("/html/body/div[1]/section/main/div/header/section/ul/li[3]/a")\
        .click()#Following
    following = self.Get_Names() #This variable calls the function which scrolls and stores names

    #This clicks on followers
    self.driver.find_element_by_xpath("/html/body/div[1]/section/main/div/header/section/ul/li[2]/a")\
        .click()
    followers = self.Get_Names() #This variable calls the function which scrolls and stores names

    not_following_back = [user for user in following if user not in followers]
    print(not_following_back)
```

The function “Compare_Followers” will click the on the “following” data box and once the data box has been loaded, the function will call Function 2 “Get_Names”. Once “Get_Names” has been completed, the program will have stored all the usernames. It will repeat the process for the “followers”. Finally, it will check for users in following that are not in followers.

Here is the code for Function 2:

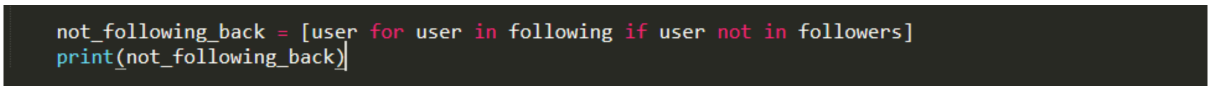


```python
def Get_Names(self):
    sleep(1)
    #Finds the box it needs to scroll
    scroll_box = self.driver.find_element_by_xpath("/html/body/div[5]/div/div[2]")

    #Compares the height of the box from the previous scroll to the new one
    #This will determine whether it has reached the end or not
    #If height = Same then no more results = stop
    last_ht, ht = 0, 1
    while last_ht != ht:
        last_ht = ht    #Snapshots last height to current height
        sleep(1)
        ht = self.driver.execute_script("""
            arguments[0].scrollTo(0, arguments[0].scrollHeight);
            return arguments[0].scrollHeight;
            """, scroll_box)    #Gets a new height

    links = scroll_box.find_elements_by_tag_name('a')    #name elements
    names = [name.text for name in links if name.text != ''] #Gets text out of links if the name is not blank

    # Click exit button
    self.driver.find_element_by_xpath("/html/body/div[5]/div/div[1]/div/div[2]/button")\
        .click()

    #This will display the names of those who the user is following but other's are not following back
    return names
```

The function "Get_names" will first find the data box (it needs to know what scroll down on) and it will initialise two variables: last height and height (ht) to know when to stop scrolling. It first set the value to 0 and as it scrolls down the value will change. The program will know if it has reached the end by comparing the height of the previous scroll to the current scroll. If the heigh of the current scroll is the same to height of the previous scroll it means it has reached the end. Selenium can execute JavaScript without the need of any libraries and that is how it accomplished to scroll.

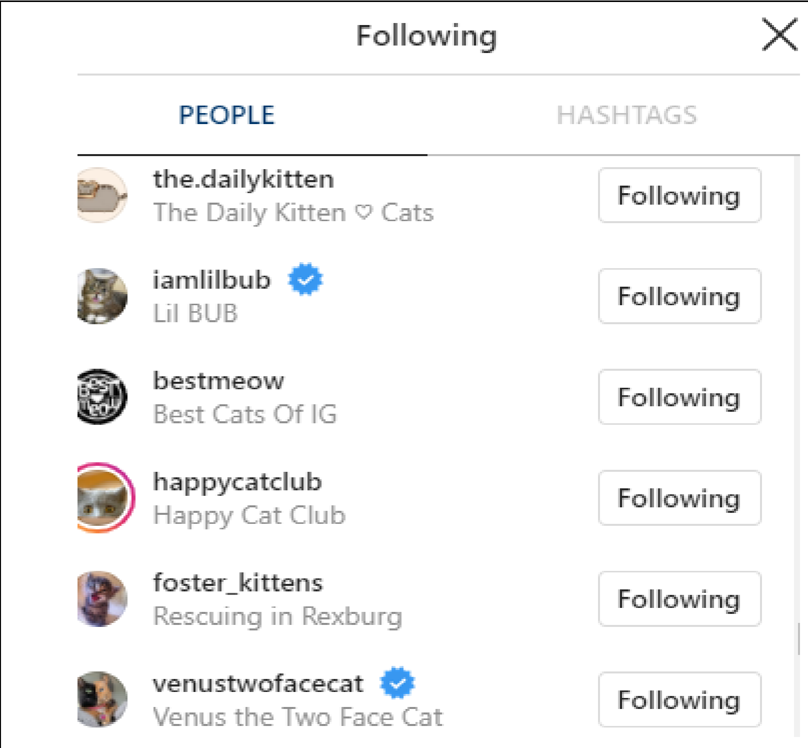
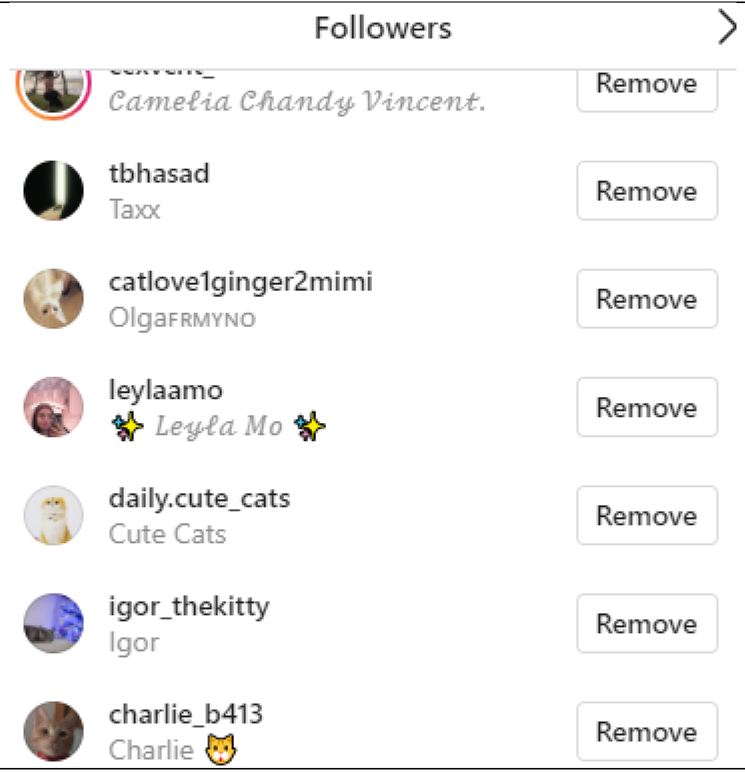
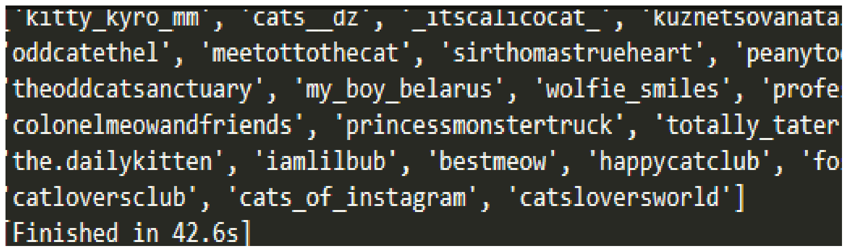
Finally, the variable "links" is used to get the names of the username list by identifying that every name is link. The variable is used to aid the variable "names" to know what it has to store. The variable "names" simply extracts the names which can been seen in the following/follower's box. In the end, the code will click on the exit button and return every single name.

Now to combine the power of Function 1 and Function 2, once function 2 has every single username which can be seen in the data boxes function 1 will only print the names who appear in following but not in followers:


```
not_following_back = [user for user in following if user not in followers]
print(not_following_back)
```

This is the result of the two functions with evidence:

What is displayed in Test 2 after running	Description

 Following PEOPLE HASHTAGS the.dailykitten The Daily Kitten ♡ Cats Following iamlilbub Lil BUB Following bestmeow Best Cats Of IG Following happycatclub Happy Cat Club Following foster_kittens Rescuing in Rexburg Following venustwofacecat Venus the Two Face Cat Following	Once in the user's profile, it will click on following data box and scroll down until it reaches then end. I cannot show the scrolling process due to the evidence being screenshots.
 Followers Camelia Chandy Vincent. Remove tbhasad Taxx Remove catlove1ginger2mimi OlgaFRMYNO Remove leylaamo ✨ Leyla Mo ✨ Remove daily.cute_cats Cute Cats Remove igor_thekitty Igor Remove charlie_b413 Charlie 🐱 Remove	It has now finished with following data box and closes the box. It moves on to follower's data box and scrolls down until it reaches the end. I cannot show the scrolling process due to the evidence being screenshots.
	The names of those who do not follow the user are being displayed/returned. Using function 2 it saves the names of

```
'kitty_kyro_mm', 'cats__dz', '_itscalicocat_', 'kuznetsovanata
'oddcatethel', 'meetottothecat', 'sirthomastrueheart', 'peanyto
'theoddcatsanctuary', 'my_boy_belarus', 'wolfie_smiles', 'profe
'colonelmeowandfriends', 'princessmonstertruck', 'totally_tater
'the.dailykitten', 'iamlilbub', 'bestmeow', 'happycatclub', 'fo
'catloversclub', 'cats_of_instagram', 'catsloversworld']
[Finished in 42.6s]
```

	the users which are displayed in both boxes and using function 1 it compares the usernames of who from following is not in followers and returns the names.

What has been done:

I have been able to make the program take data from the browser and returned in the system by using two different functions which work with one another. One function compares the names that the other function has gathered from the web browser.

Summary of the system so far:

The system is able to log the user in through an automation process and get the names of those who do not follow the user back.

Success criteria met:

- Clicking “not now” on the notifications box.
- Accessing the user’s profile page.
- Accessing the following and followers
- Comparing following to followers
- Returning the names of those who the user follows but are not following the user back.

The first bullet point does not have evidence, but it has been completed. The rest of the bullet points have evidence and have been met. I would say that the success criteria has been met.

Stakeholder feedback:

After showing my client the process so far, she said:

“Is very useful to not have to enter the username and password every time I want to log in, the automation definitely saves time, and I can already see that there’s a lot of people who don’t follow me back”.

“Is very responsive and it all can be done under a minute. So far very useful to see who doesn’t follow me back.”

“Where is the GUI?”

<u>Response to Feedback:</u>

It seems like I am going on right track as she is satisfied with the program. She understands is not yet complete and no other features have been added in. I need to now work on the GUI.

Iteration 3 (Simple GUI Prototype)

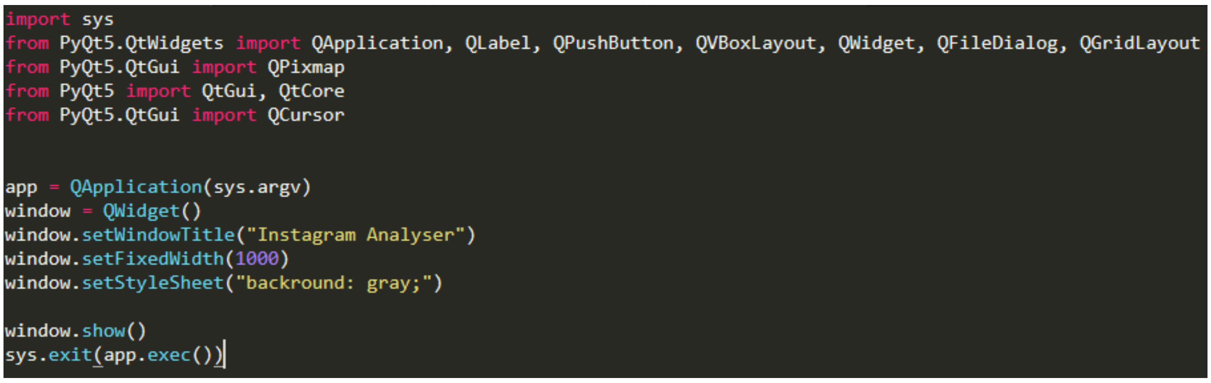
In this iteration the graphical user interface which will contain widgets and buttons. I will use PyQt5 to create the GUI.

In this iteration I will focus developing the GUI module which will consist of:

- Create a window object.
- Create logo widget.
- Create Button Widget.

Window Object

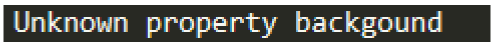
I will first build a small window just to get a base from which to build up on. Here is the code:



```
import sys
from PyQt5.QtWidgets import QApplication, QLabel, QPushButton, QVBoxLayout, QWidget, QFileDialog, QGridLayout
from PyQt5.QtGui import QPixmap
from PyQt5 import QtGui, QtCore
from PyQt5.QtGui import QCursor


app = QApplication(sys.argv)
window = QWidget()
window.setWindowTitle("Instagram Analyser")
window.setFixedWidth(1000)
window.setStyleSheet("backround: gray;")

window.show()
sys.exit(app.exec())
```

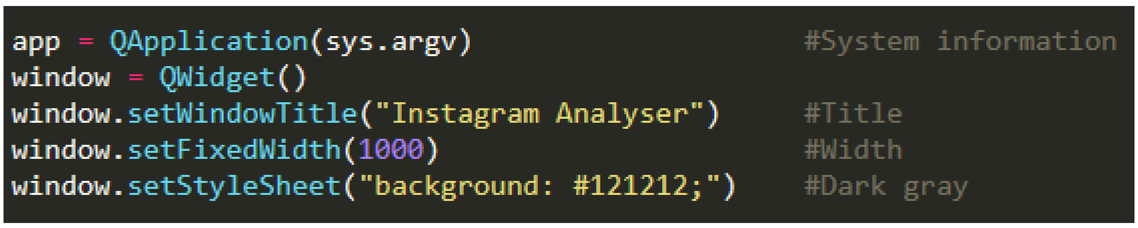
Very simple code which creates a small window with the title being "Instagram Analyzer" and a Gray background.

I encountered this problem and quickly realised I had misspelled background.



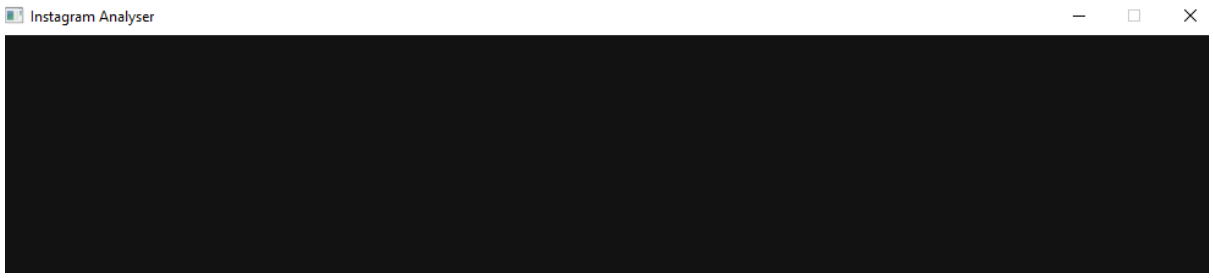
```
Unknown property backgound
```

Once fixed I then decided to add a different background colour. This is the new and working code:



```
app = QApplication(sys.argv)                        #System information
window = QWidget()
window.setWindowTitle("Instagram Analyser")         #Title
window.setFixedWidth(1000)                          #Width
window.setStyleSheet("background: #121212;")        #Dark gray
```

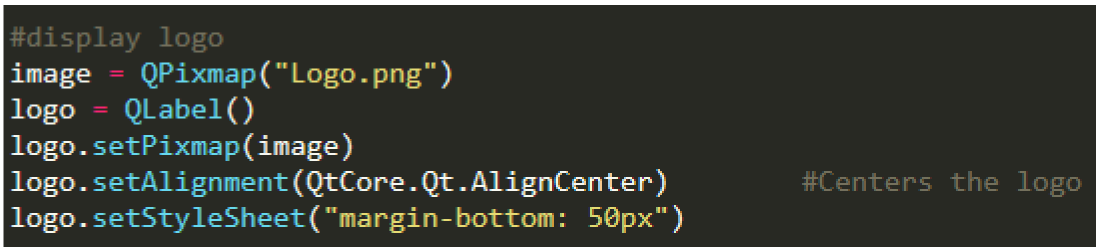
And here is the result:



It currently it only displays the window. The next step will be adding a logo to be displayed in the window.

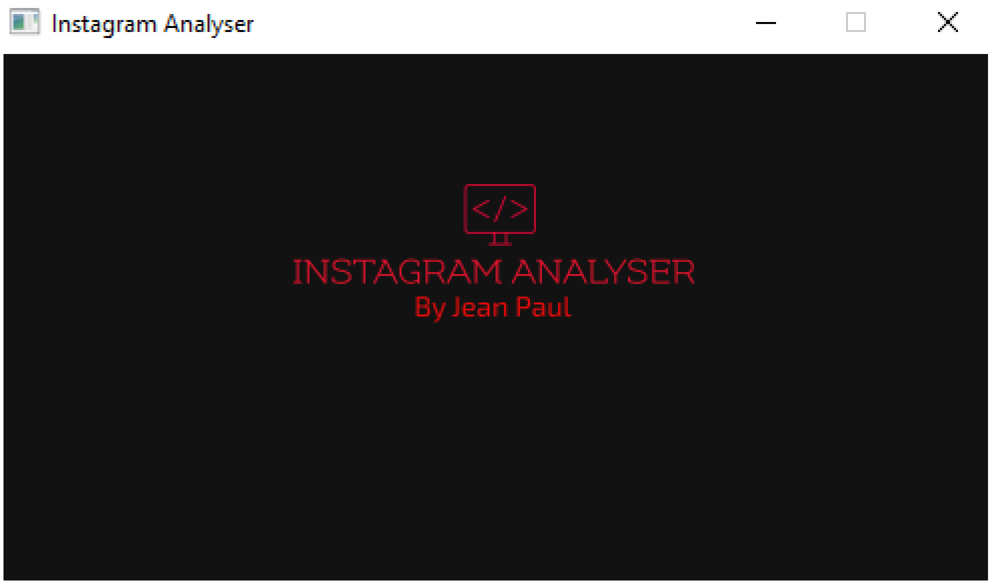
Logo Widget:

I have a simple logo which will be displayed in the main window. Here is the code:



```
#display logo
image = QPixmap("Logo.png")
logo = QLabel()
logo.setPixmap(image)
logo.setAlignment(QtCore.Qt.AlignCenter)        #Centers the logo
logo.setStyleSheet("margin-bottom: 50px")
```

Initially the logo is displayed in the centre of the screen but nothing else is shown on screen, I will then add a button which will be displayed below the logo.

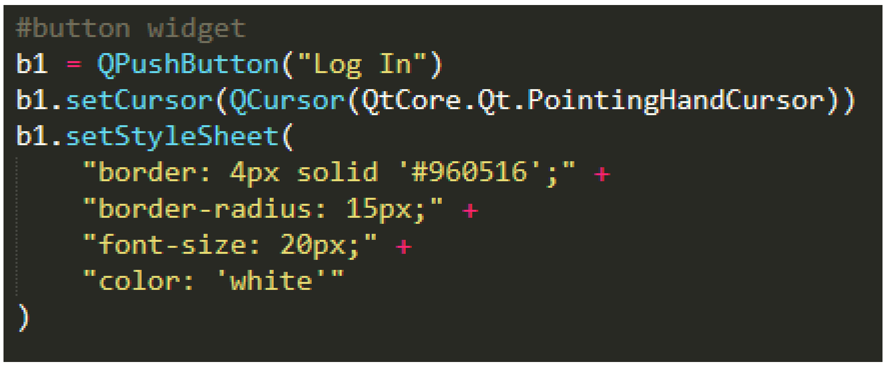


The Logo widget is being displayed and is not exactly in the centre of the window. That will leave space for the button to fit.

Button Widget:

The next step is to add a button which the user can click. Right now, when the button is clicked it does not call any function so is just there for aesthetics, but it will be changed so that when called it runs the log in and web scrapping.

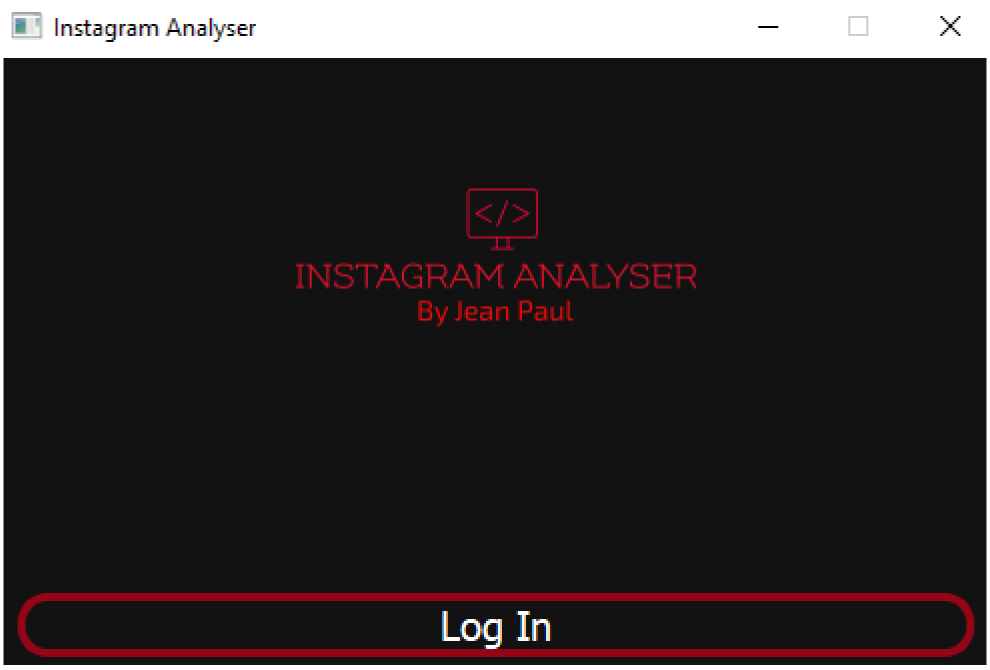
The button code is here:



```
#button widget
b1 = QPushButton("Log In")
b1.setCursor(QCursor(QtCore.Qt.PointingHandCursor))
b1.setStyleSheet(
    "border: 4px solid '#960516';" +
    "border-radius: 15px;" +
    "font-size: 20px;" +
    "color: 'white'"
)
```

I have added a button with “Log in” text. When the mouse hovers over the button it will change from the normal cursor to the hand cursor. I have used some CSS to make the button look better.

This is the result overall result:



<u>What has been done:</u>

I have been able to make a simple GUI with a logo and a button.

<u>Summary of the system so far:</u>

I have made an automated user log in and made Selenium obtain data from the browser, do calculations and display information. I have made a GUI which doesn't do anything so far but will modified so when clicked "Log in" it calls Selenium and performs test 1 and 2.

Success criteria Met:

- Create a window object.
- Create logo widget.
- Create Button Widget.

All have been met.

Improving Iteration 3 (GUI)

I now have a simple GUI working, I have to make the button perform operations to make the GUI relevant .

In this test I will focus on improving the current GUI module which will consist of:

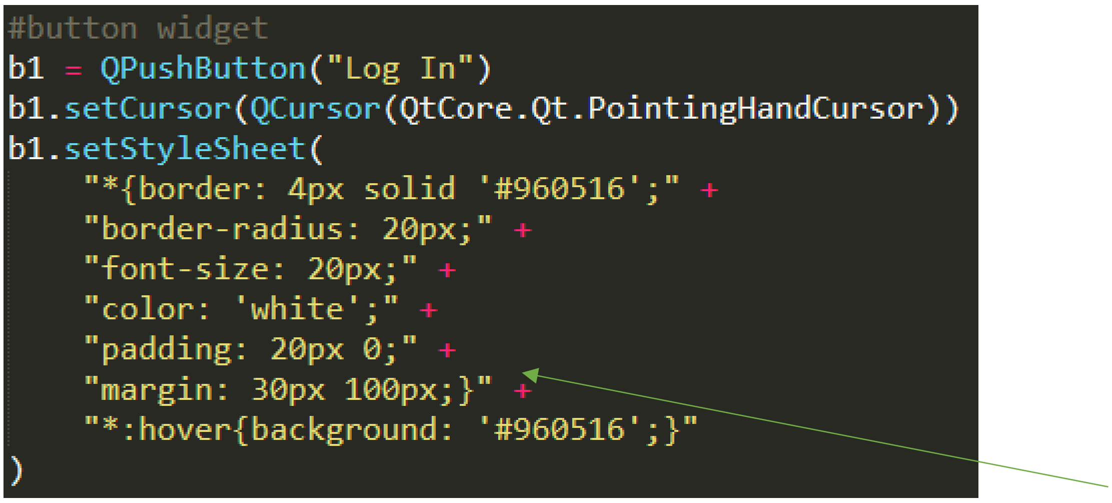
- Improving the Button
  - Add padding.
  - Add hover event.

- When the button is clicked perform an action
  - When clicked make it run test 1 and test 2

There will not be many changes to the GUI itself but the main focus of this test it to make the button work by calling the function to make the system work.

I want to make the button look better by making the button not take the whole width of the window.

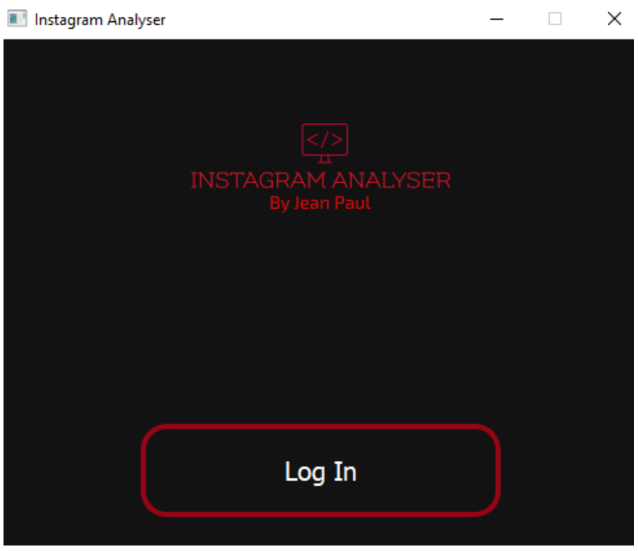
Here is the code:



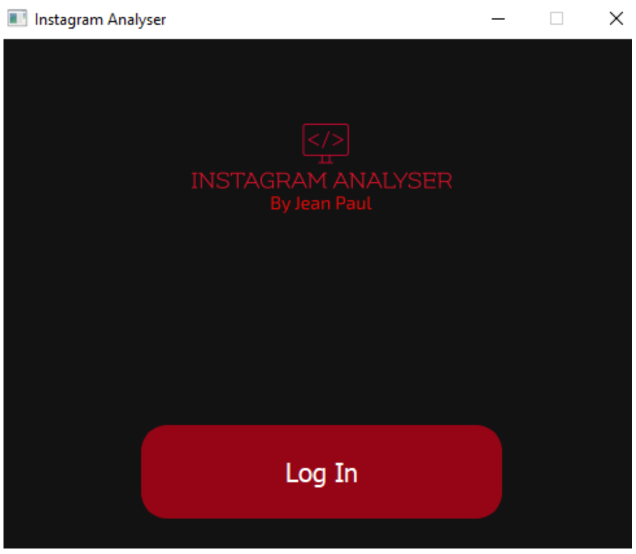
```
#button widget
b1 = QPushButton("Log In")
b1.setCursor(QCursor(QtCore.Qt.PointingHandCursor))
b1.setStyleSheet(
    "*{border: 4px solid '#960516';" +
    "border-radius: 20px;" +
    "font-size: 20px;" +
    "color: 'white';" +
    "padding: 20px 0;" +
    "margin: 30px 100px;}" +
    "*:hover{background: '#960516';}"
)
```

I will add padding and margin to make the button smaller and look professional and cleaner. I have also increased the height of the button to make it more accessible. Not much was changed but it does improve the overall design and accessibility of the button. I've also added a hover event which means the button will become a different colour to show is being hovered over using a lighter red.

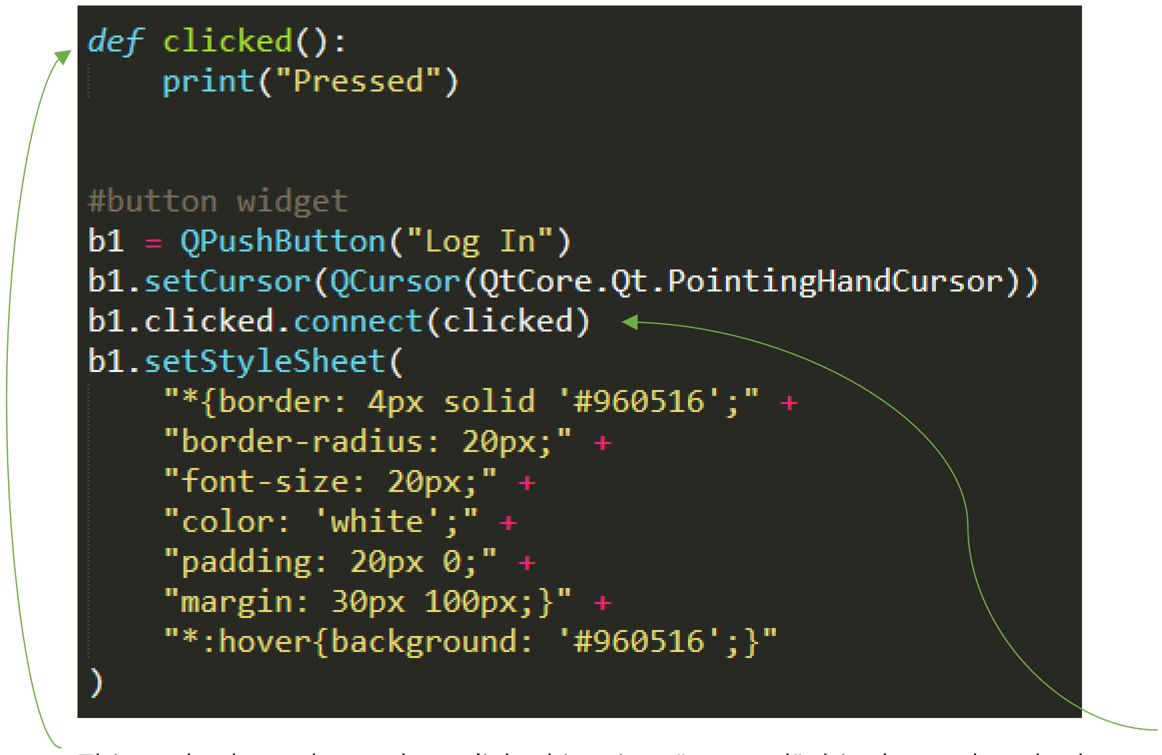
Here are the results:



The button is smaller on width and bigger on height making it more accessible.



When the mouse is hovered over the buttons, we can see it lights up. The cursor is not shown on the screenshot, but it changes to the hand cursor which indicates a clickable button.



```python
def clicked():
    print("Pressed")


#button widget
b1 = QPushButton("Log In")
b1.setCursor(QCursor(QtCore.Qt.PointingHandCursor))
b1.clicked.connect(clicked)
b1.setStyleSheet(
    "*{border: 4px solid '#960516';" +
    "border-radius: 20px;" +
    "font-size: 20px;" +
    "color: 'white';" +
    "padding: 20px 0;" +
    "margin: 30px 100px;}" +
    "*:hover{background: '#960516';}"
)
```

This code shows how when clicked it prints "pressed" this shows that the button now reacts when clicked.

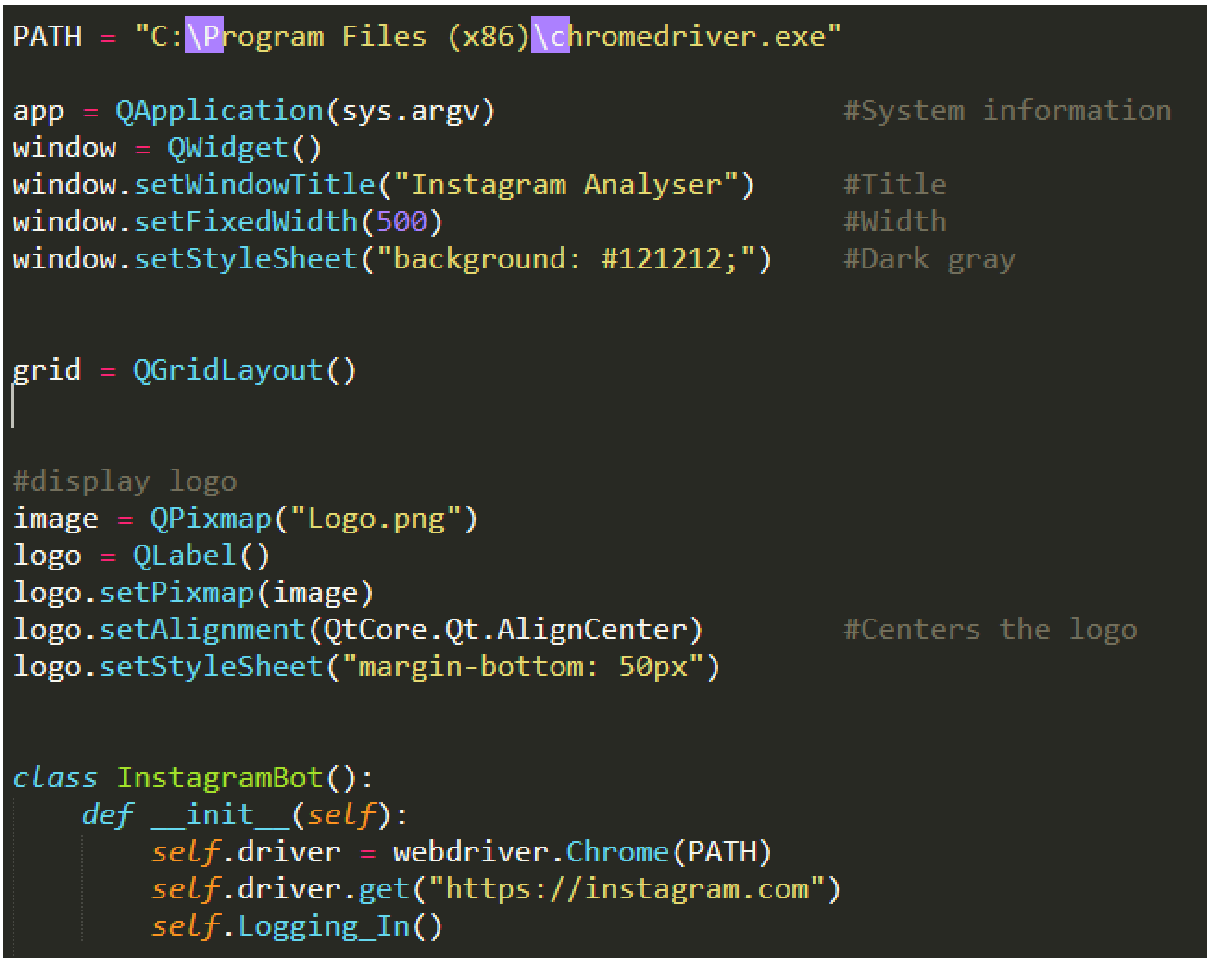
Once tested, when clicked it prints "pressed" in the console.

So far, the button doesn't perform any actions but it will soon change.

Making the button perform an action when pressed:

I copied and pasted the code which contained iteration 1 and iteration 2 which runs the automated log in (Test 1) and the data scrapper (Test 2) into the GUI. When the button is pressed it calls the class and runs as normal.

Here is the code:



```
PATH = "C:\Program Files (x86)\chromedriver.exe"

app = QApplication(sys.argv)                     #System information
window = QWidget()
window.setWindowTitle("Instagram Analyser")      #Title
window.setFixedWidth(500)                        #Width
window.setStyleSheet("background: #121212;")     #Dark gray


grid = QGridLayout()


#display logo
image = QPixmap("Logo.png")
logo = QLabel()
logo.setPixmap(image)
logo.setAlignment(QtCore.Qt.AlignCenter)         #Centers the logo
logo.setStyleSheet("margin-bottom: 50px")


class InstagramBot():
    def __init__(self):
        self.driver = webdriver.Chrome(PATH)
        self.driver.get("https://instagram.com")
        self.Logging_In()
```

We can see that some of the code for the window properties is above the class "InstagramBot()" which contains the code for running iteration 1 and 2 as previously mentioned.

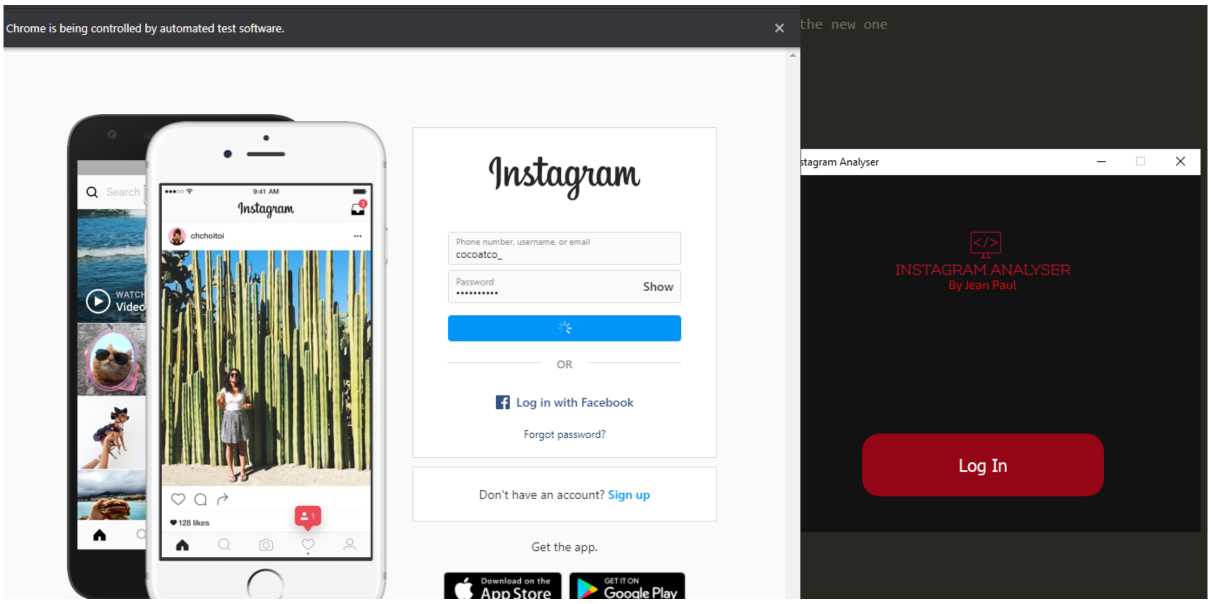
The way the button calls the class once pressed is by using a "connecting" property to the button.

Here is the code:

```
#button widget
b1 = QPushButton("Log In")
b1.setCursor(QCursor(QtCore.Qt.PointingHandCursor))
b1.clicked.connect(InstagramBot)
b1.setStyleSheet(
    "*{border: 4px solid '#960516';" +
    "border-radius: 20px;" +
    "font-size: 20px;" +
    "color: 'white';" +
    "padding: 20px 0;" +
    "margin: 30px 100px;}" +
    "*:hover{background: '#960516';}"
)
```

We can see that when clicked it calls the class.

Here is the evidence:


Chrome is being controlled by automated test software.

<u>What has been done:</u>

I have been able to improve the button by making it look better and performing an action when pressed. When pressed it runs Selenium and will begin the automated log in and web scrapper.

<u>Summary of the system so far:</u>

The system has a GUI which is presented to the user before Selenium runs. Once the user has pressed logged in the button will call the class which contains the automation process and web scraping. So far, the system is running as it should.

Success criteria met:

- Improving the Button.
  - Add padding.
  - Add hover event.
- When the button is clicked perform an action .
- When clicked make it run test 1 and test 2.

The success criteria has been met and everything runs properly.

Stakeholder feedback:

After showing my client the process so far, she said:

"The GUI is looking good and very simple use with one button to press, quickly opens web browser and continues to log in".

"Graphs would be really nice to understand the data".

Response to Feedback:

So far, I am meeting my client's needs by adding a guy and automating web scrapping and automated log in. I will now work on adding graphs to the system.

Iteration 4(Charts)

I will be using bar charts to show the difference in followers to following. Is a nice feature to include and is part of the essential features. It will make the program more accessible because some users may not understand numbers. Having a visual representation and seeing which bar is higher should satisfy their needs. Having a bar chart is also a feature suggested by my client.

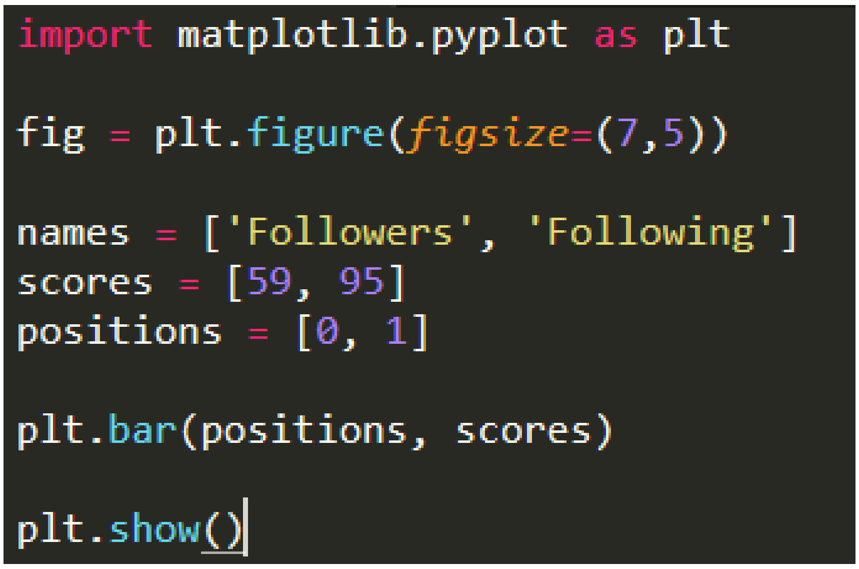
In this iteration I will focus on developing a simple bar chat with made up values which will consist of:

- Creating a simple prototype
  - Adding values .
  - Adding visual bars.
- Improving the simple prototype
  - Adding names in the bar chart.
  - Add names to the X and Y axis.
  - Add different colours for following and followers.
  - Make bar chart thinner.

Creating simple prototype:

I will make a simple code with fake values to make sure the code is working properly.

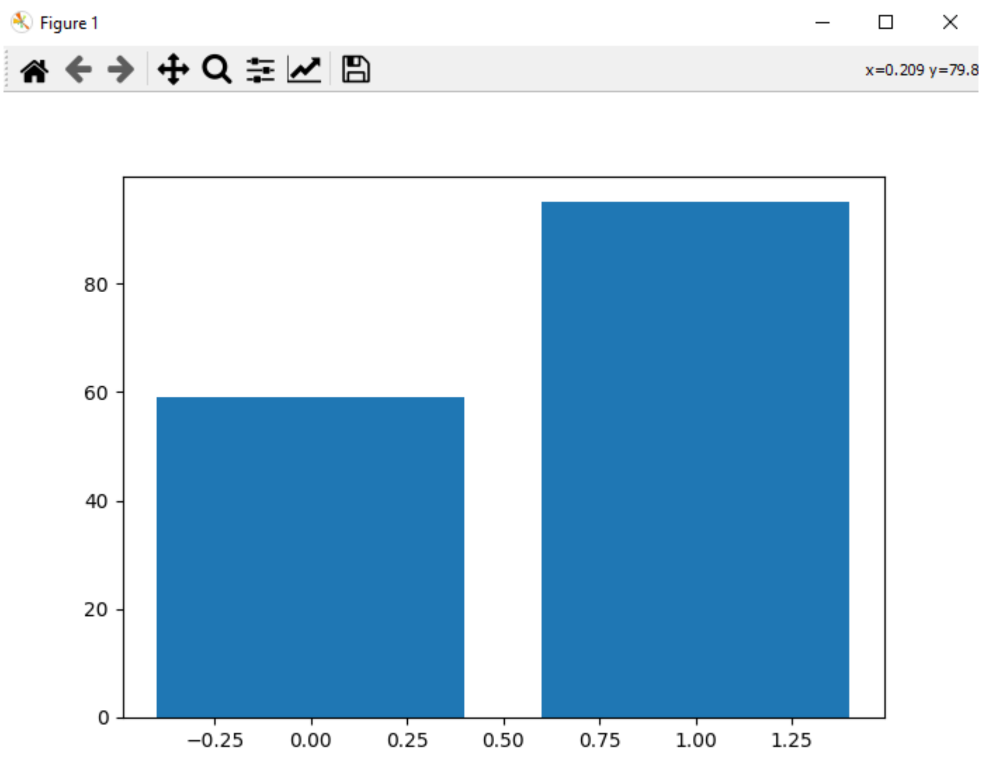
Here is the code:



```python
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(7,5))

names = ['Followers', 'Following']
scores = [59, 95]
positions = [0, 1]

plt.bar(positions, scores)

plt.show()
```

I stored the names for the two different bar charts as “followers” and “following” which make reference to the users followers. The scores is linked to the position of the names meaning that “follower’s value is 59 and following value is 95”. The position variable just declares where each bar goes in the graph.

Here is the result:



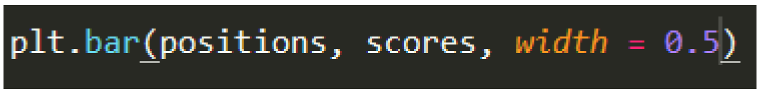
Currently the X axis and Y axis are not specified and are just numbers, also, the bar charts are the same colour which could confuse the user. The bar charts do not have label, so the user does not know which one is which. The bar charts are also to wide.

To summaries the problem with current bar chart:

- X axis and Y axis not label.
- Bar charts same colour.
- Bar charts are not label.
- Bar chart very wide.

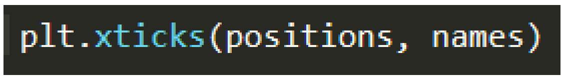
The next step is to fix the current problems.

I have made a code which make the bar chart thinner:



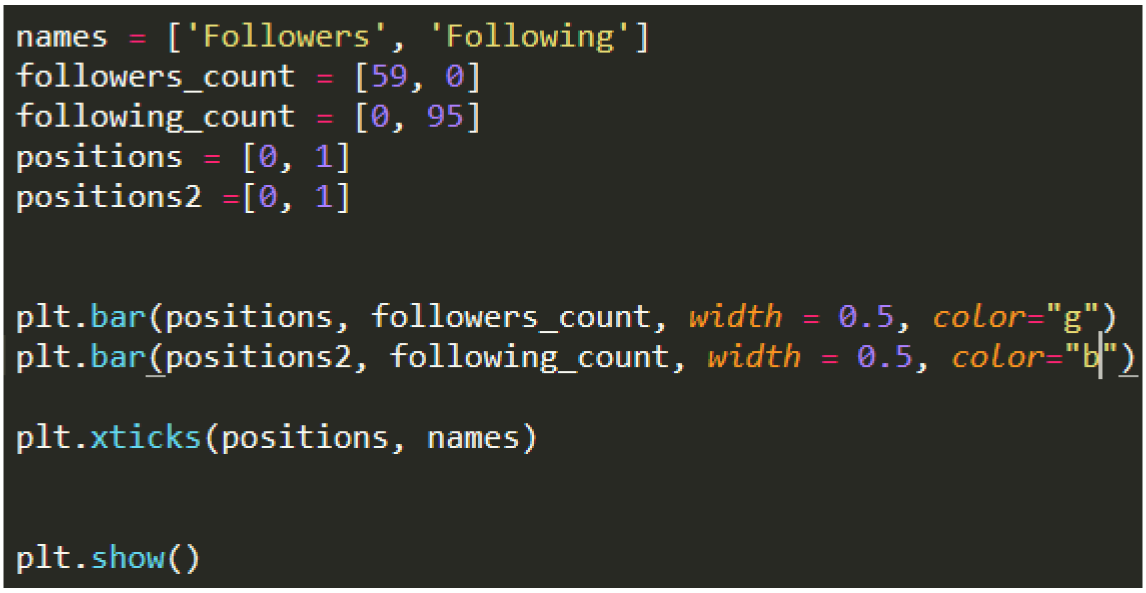
```
plt.bar(positions, scores, width = 0.5)
```

I have made a code which labels the bars:



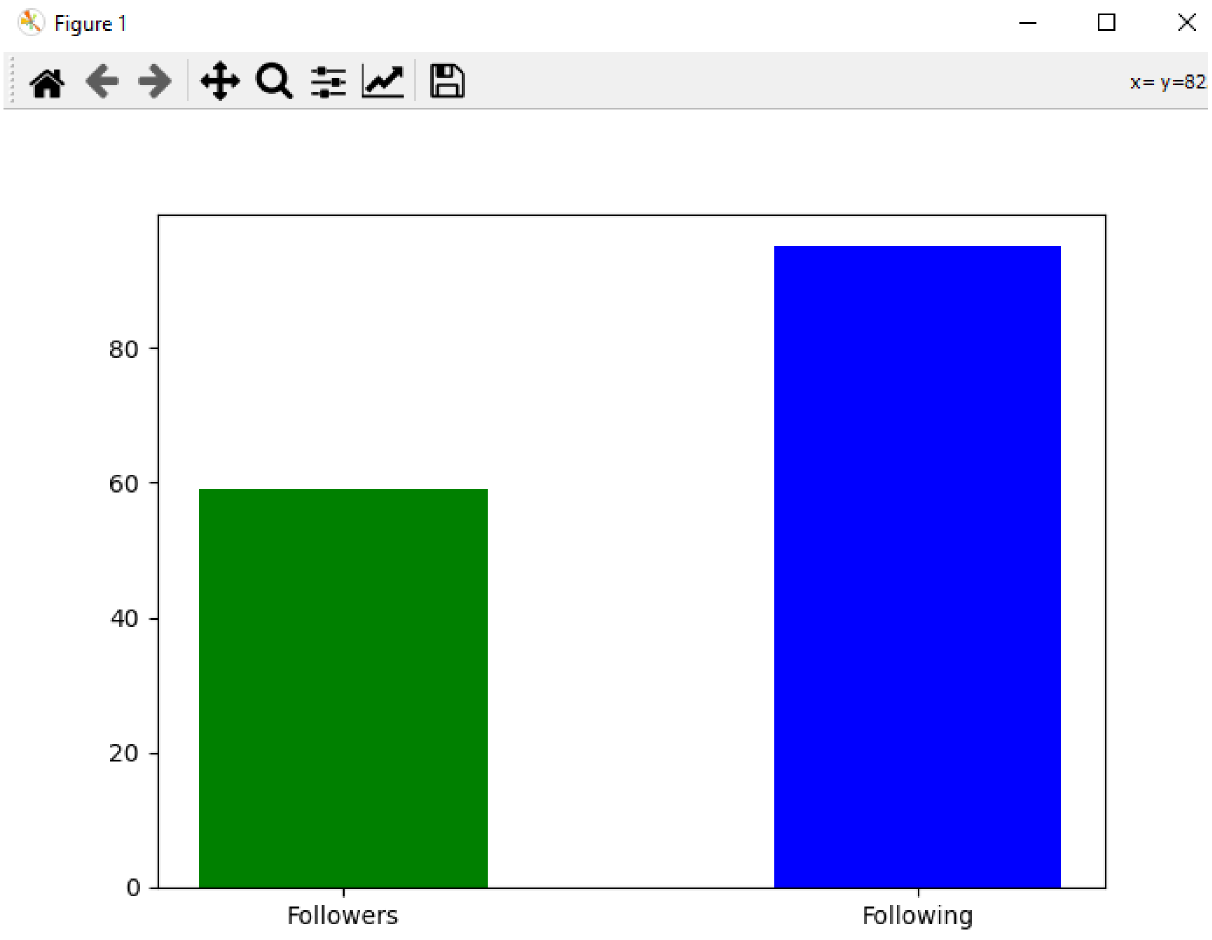
```
plt.xticks(positions, names)
```

I have made a code which changes the colours of the bars:



```
names = ['Followers', 'Following']
followers_count = [59, 0]
following_count = [0, 95]
positions = [0, 1]
positions2 =[0, 1]


plt.bar(positions, followers_count, width = 0.5, color="g")
plt.bar(positions2, following_count, width = 0.5, color="b")

plt.xticks(positions, names)


plt.show()
```

Here is the overall result of this test:



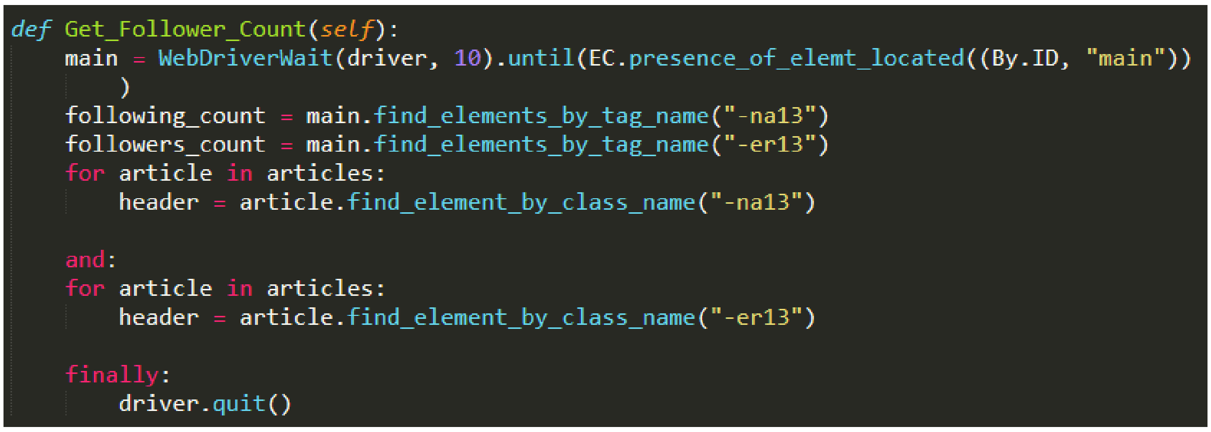
<u>What has been done:</u>

A simple bar chart with two different colours, label x axis and y axis has been created using fake values which shows the following compared to the followers.

Next step is to extract the numbers from the web browser and use them as values.

In order to achieve this, I will develop a code which first tells selenium what to identify then extract the data. The extracted data should be saved as a variable and will be used to the graph.

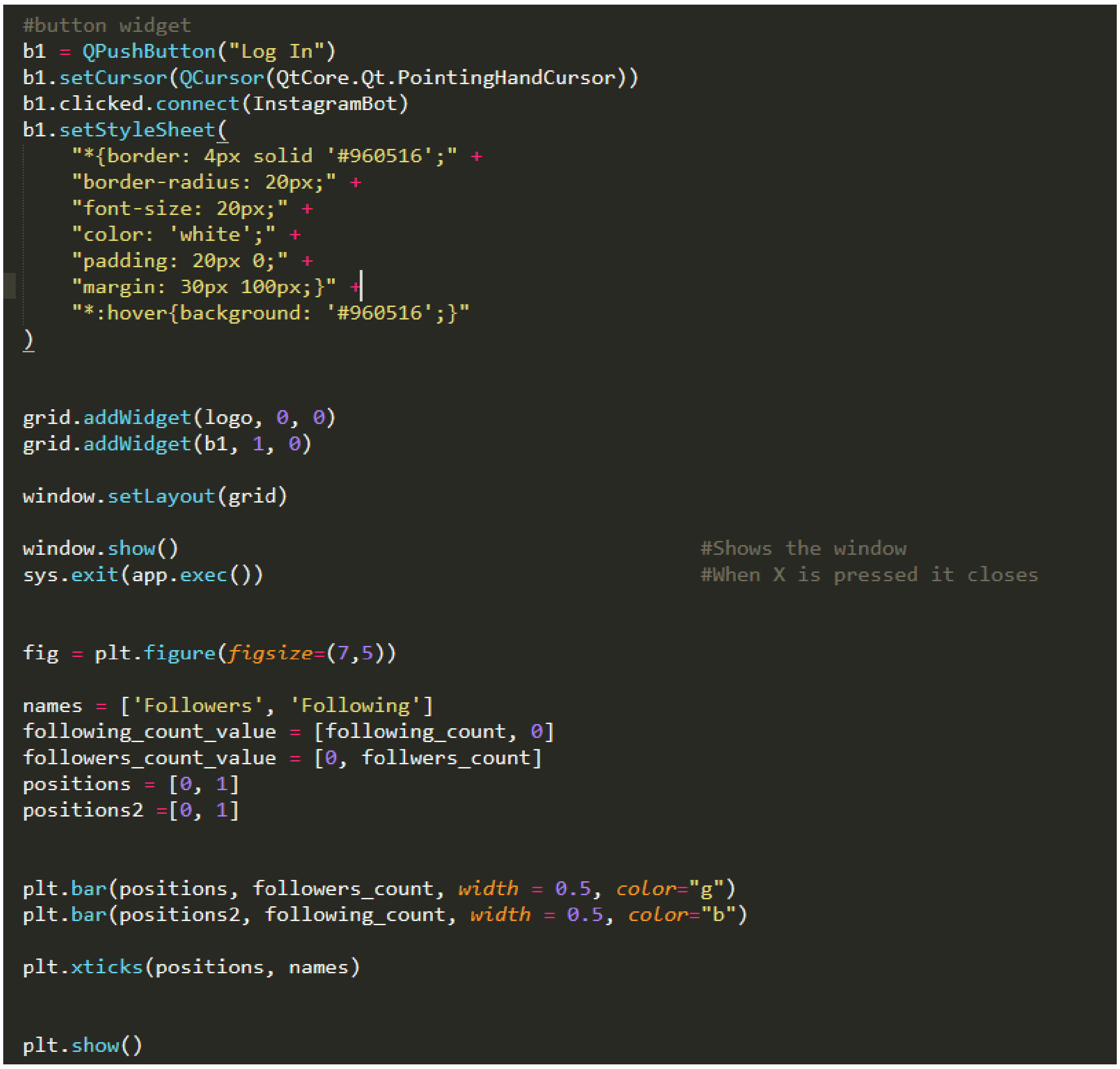
Here is the code to extract data from the web:



```
def Get_Follower_Count(self):
    main = WebDriverWait(driver, 10).until(EC.presence_of_elemt_located((By.ID, "main"))
        )
    following_count = main.find_elements_by_tag_name("-na13")
    followers_count = main.find_elements_by_tag_name("-er13")
    for article in articles:
        header = article.find_element_by_class_name("-na13")

    and:
    for article in articles:
        header = article.find_element_by_class_name("-er13")

    finally:
        driver.quit()
```

It finds the tag name in the main script and then stores the value.

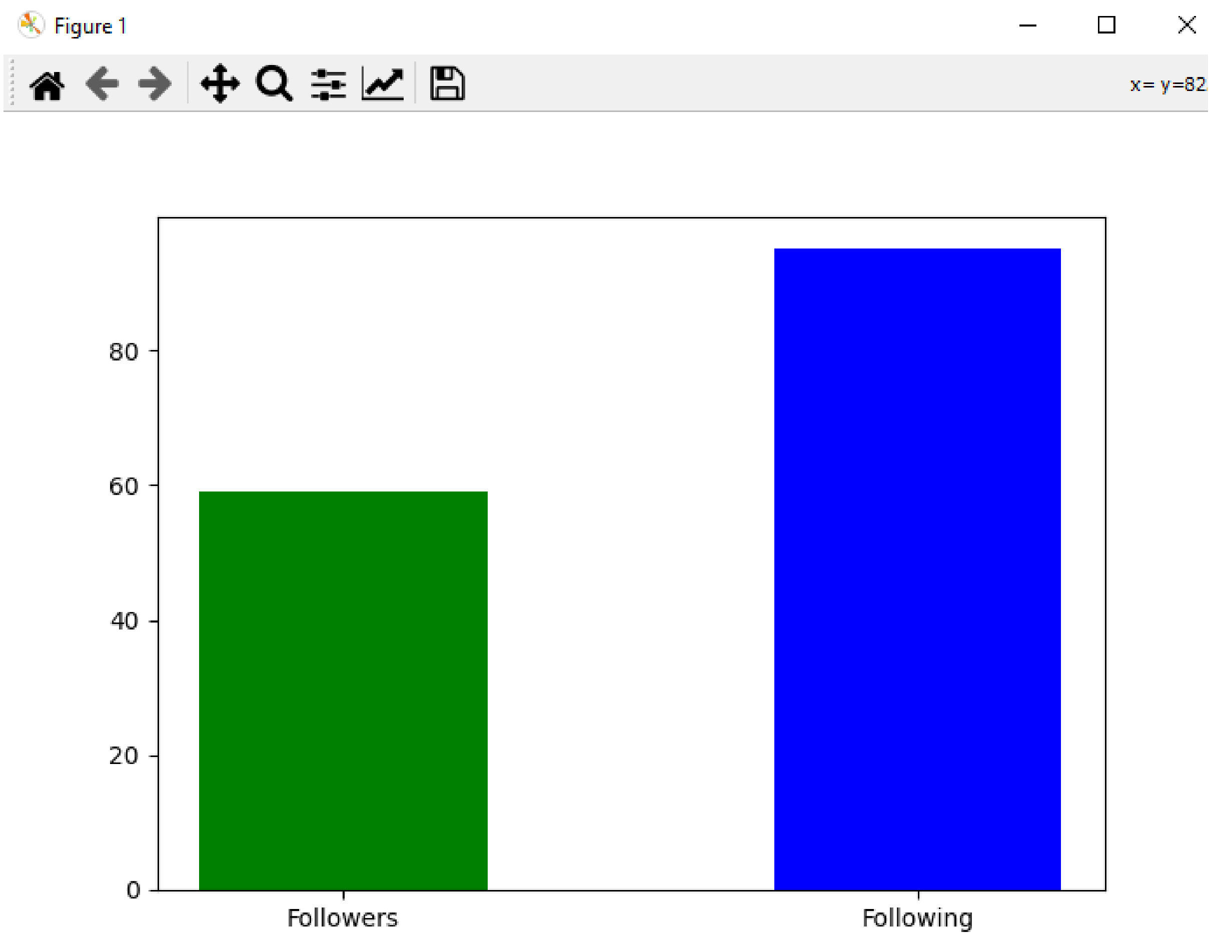
Here is the barchart implemented in the code:



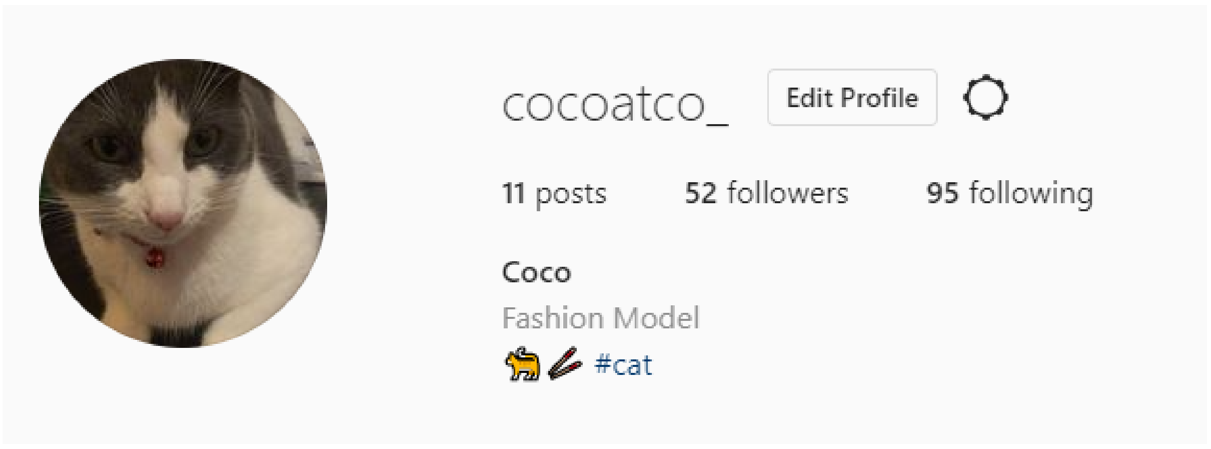
```
#button widget
b1 = QPushButton("Log In")
b1.setCursor(QCursor(QtCore.Qt.PointingHandCursor))
b1.clicked.connect(InstagramBot)
b1.setStyleSheet(
    "*{border: 4px solid '#960516';" +
    "border-radius: 20px;" +
    "font-size: 20px;" +
    "color: 'white';" +
    "padding: 20px 0;" +
    "margin: 30px 100px;}" +
    "*:hover{background: '#960516';}"
)


grid.addWidget(logo, 0, 0)
grid.addWidget(b1, 1, 0)

window.setLayout(grid)

window.show()                                                   #Shows the window
sys.exit(app.exec())                                            #When X is pressed it closes


fig = plt.figure(figsize=(7,5))

names = ['Followers', 'Following']
following_count_value = [following_count, 0]
followers_count_value = [0, follwers_count]
positions = [0, 1]
positions2 =[0, 1]


plt.bar(positions, followers_count, width = 0.5, color="g")
plt.bar(positions2, following_count, width = 0.5, color="b")

plt.xticks(positions, names)


plt.show()
```

What has been done:

The graph is now able to get the values from the extracted data thanks to selenium. Here is the evidence:



We can see that followers is below 60 and following is above 80. I know this is accurate because the actual account has this many followers and following:



cocoatco_ Edit Profile

11 posts 52 followers 95 following

Coco
Fashion Model
🐈✏️ #cat

What has been achieved:

Create a graph showing followers compared to following using values extracted from the web.

Summary of the system so far:

The system is complete and it has now fully working as it should.

Criteria met:

- Creating a simple prototype
    - Adding values .
    - Adding visual bars.
- Improving the simple prototype
    - Adding names in the bar chart.
    - Add names to the X and Y axis.
    - Add different colours for following and followers.
    - Make bar chart thinner.

Not only was it meet but it was also able to get the values from the web.

Evaluation

Final Testing:

Many features have already been tested during the testing phase and evidence shows is working correctly. Everything was tested using a 'module by module' approach and in the end, everything was coded together to make the final code. The final test will ensure everything works as one robust piece of code.

I am going to test functionality to ensure everything works.

How tests where doe in Final Code:

As all the test where previously done, there should be no surprises therefore I will create a simple check list which will make the final judgment on whether the system is complete or not.

Test	Working?
Webdriver Opens google chrome	Yes
Webdriver searches Instagram	Yes
Webdriver clicks on specified items	Yes
Username and password entered	Yes
Log in	Yes
User's profile accessed	Yes
User's followers compared	Yes
Return names of those not following back	Yes
Scrape data from web	Yes
Display GUI Main Window	Yes
Load Logo	Yes
Load Button	Yes
When button pressed perform action	Yes
Display graph	Yes
Display correct amount of followers and following in different colors	yes
Bars labeled	Yes

Stakeholder Testing:

After developing the program, I needed to ensure it meet my stakeholders needs. My stakeholder and I agreed on meeting up to test the code on my laptop.

When we meet I asked the following questions:

- Does the program satisfy your needs?
- Is the program running smooth?
- Is there any problems?
- Any recommendations for further improvement?

Here is what she answered:

"The program satisfies my needs to a basic level, is far more detailed than Instagram however I believe it lacks more features. The program does run smooth and after I click once I do not have to do anything else, I can find out who does not follow me and see the graph which helps with my dyslexia and now I am able to see if I have more followers than following. It is very important for the page to look professional.

There was a problem with time issues, sometimes I get errors because the page doesn't load quick enough for the program to process it but a simple refresh will get it working.

I recommend having a homepage to display all this data"

Limitations:

I was not able to add a live monitoring or a homepage because it was way to advance for my programming knowledge.

If Instagram decides to change their page layout, I do not think my code will work because it finds elements using very specific Xpaths and if Instagram changes does then the program won't know what to look for or where to type. This is a major setback.

The code itself is not very portable as it needs some logo.png and numerous libraries as well as having the web driver on the user's computer. I could however make the game run as an exe which stop the need of having the libraries on the user's computer.

Maintenance:

O Further maintenance could be done by optimizing the code by adding more comments which could help other developers.

Further Improvements:

To further improve the program, I can make the program run quicker by making functions have less waiting time. This, however, could affect the overall performance of the program and it could possibly cause errors and make the program not work.

I could also add a homepage as my stakeholder suggested which would be a great improvement as it will make the program look better and more professional.

Final Code

```
#Importing Libraries

import sys
                #Imports system info

from PyQt5.QtWidgets import *
     #Imports Widgets (Labels, buttons)

from PyQt5.QtGui import *
     #Allows images to show

from PyQt5 import QtGui, QtCore

from PyQt5.QtGui import *
from selenium import webdriver
      #Web automation

from time import sleep
            #Delay module

from Logging import username, pw
      #Username and password

import matplotlib.pyplot as plt


PATH = "C:\Program Files (x86)\chromedriver.exe"


app = QApplication(sys.argv)
      #System information

window = QWidget()

window.setWindowTitle("Instagram Analyser")
      #Title

window.setFixedWidth(500)
      #Width

window.setStyleSheet("background: #121212;")
      #Dark gray



grid = QGridLayout()



#display logo

image = QPixmap("Logo.png")

logo = QLabel()

logo.setPixmap(image)

logo.setAlignment(QtCore.Qt.AlignCenter)
      #Centers the logo

logo.setStyleSheet("margin-bottom: 50px")
```

```
class InstagramBot():

      def __init__(self):

            self.driver = webdriver.Chrome(PATH)

            self.driver.get("https://instagram.com")

            self.Logging_In()


      def Logging_In(self):

            #Xpath is used to identify buttons which will

            self.username = username
      #reference for other methods


      self.driver.find_element_by_xpath("/html/body/div[2]/div/div/d
iv/div[2]/button[1]")\

                  .click()
            # Accept cookies

            sleep(1)



      self.driver.find_element_by_xpath("//input[@name=\"username\"]
")\

                  .send_keys(username)
      #Username box and username entered


      self.driver.find_element_by_xpath("//input[@name=\"password\"]
")\

                  .send_keys(pw)
            #Password box and password entered


      self.driver.find_element_by_xpath('//button[@type="submit"]')\

                  .click()
            #Press enter button

            sleep(4)

            self.Homepage_CloseBoxes()
      def Homepage_CloseBoxes(self):
            #Closes pop up boxes

      self.driver.find_element_by_xpath("//button[contains(text(),
'Not Now')]")\
                  .click()
            #Save log in "denied"
            sleep(2)

      self.driver.find_element_by_xpath("/html/body/div[4]/div/div/d
iv/div[3]/button[2]")\
                  .click()
            #Notifications box "denied"
            sleep(2)
            self.Acess_Profile()


      def Acess_Profile(self):
            #Acessing the user's profile

      self.driver.find_element_by_xpath("/html/body/div[1]/section/n
av/div[2]/div/div/div[3]/div/div[5]/span")\
                  .click()
            #Profile icon (top right)
            sleep(2)

      self.driver.find_element_by_xpath("/html/body/div[1]/section/n
av/div[2]/div/div/div[3]/div/div[5]/div[2]/div[2]/div[2]/a[1]/div/di
v[2]/div/div/div/div")\
                  .click()
            #"Profile"
            sleep(2)
            self.Compare_Followers()


      def Compare_Followers(self):
            #Access the following and compare with followers

      self.driver.find_element_by_xpath("/html/body/div[1]/section/m
ain/div/header/section/ul/li[3]/a")\
                  .click()#Following
            following = self.Get_Names() #This variable calls the
function which scrolls and stores names

            #This clicks on followers

      self.driver.find_element_by_xpath("/html/body/div[1]/section/m
ain/div/header/section/ul/li[2]/a")\
                  .click()
            followers = self.Get_Names() #This variable calls the
function which scrolls and stores names

            not_following_back = [user for user in following if user
not in followers]
            print(not_following_back)
            self.Get_Follower_Count()

      def Get_Names(self):
            sleep(1)
            #Finds the box it needs to scroll
            scroll_box =
self.driver.find_element_by_xpath("/html/body/div[5]/div/div/div[2]"
)

            #Compares the height of the box from the previous scroll
to the new one
            #This will determine whether it has reached the end or
not
            #If height = Same then no more results = stop
            last_ht, ht = 0, 1
            while last_ht != ht:
                  last_ht = ht     #Snapshots last height to current
height
                  sleep(1)
                  ht = self.driver.execute_script("""
                        arguments[0].scrollTo(0,
arguments[0].scrollHeight);
                        return arguments[0].scrollHeight;
                        """, scroll_box) #Gets a new height


            links = scroll_box.find_elements_by_tag_name('a')#name
elements
            names = [name.text for name in links if name.text != '']
#Gets text out of links if the name is not blank


            # Click exit button


      self.driver.find_element_by_xpath("/html/body/div[5]/div/div/d
iv[1]/div/div[2]/button")\
                  .click()


            #This will display the names of those who the user is
following but other's are not following back
            return names


      def Get_Follower_Count(self):
            main = WebDriverWait(driver,
10).until(EC.presence_of_elemt_located((By.ID, "main"))
                  )
            following_count = main.find_elements_by_tag_name("-na13")
            followers_count = main.find_elements_by_tag_name("-er13")
            for article in articles:
                  header = article.find_element_by_class_name("-na13")
            and:
            for article in articles:
                  header = article.find_element_by_class_name("-er13")

            finally:
                  driver.quit()

#button widget
b1 = QPushButton("Log In")
b1.setCursor(QCursor(QtCore.Qt.PointingHandCursor))
b1.clicked.connect(InstagramBot)
b1.setStyleSheet(
      "*{border: 4px solid '#960516';" +
      "border-radius: 20px;" +
      "font-size: 20px;" +
      "color: 'white';" +
      "padding: 20px 0;" +
      "margin: 30px 100px;}" +
      "*:hover{background: '#960516';}"
)


grid.addWidget(logo, 0, 0)
grid.addWidget(b1, 1, 0)

window.setLayout(grid)

window.show()
      #Shows the window
```

```
sys.exit(app.exec())
      #When X is pressed it closes



fig = plt.figure(figsize=(7,5))


names = ['Followers', 'Following']

following_count_value = [following_count, 0]

followers_count_value = [0, follwers_count]

positions = [0, 1]

positions2 =[0, 1]



plt.bar(positions, followers_count, width = 0.5, color="g")

plt.bar(positions2, following_count, width = 0.5, color="b")


plt.xticks(positions, names)



plt.show()
```