FINAL REPORT

Social Media Utility Tool

Submitted by

SHRIYANSHI BARANWAL (181500691)
PRASHANT ASTHANA (181500487)
TANISHQ TRIPATHI (181500751)

In partial fulfillment of the

Requirements

For the degree of

Bachelors of Technology

In

Computer Science

Supervised By

Dr. Manoj Varshney

Department of Training and Development

GLA University, Mathura

TOPIC

SOCIAL MEDIA UTILITY TOOL

CONTENT

DECLARATION

We hereby declare that the work which is being presented in the MINI PROJECT "**Social Media Utility Tool**", will be fulfilled of the requirements for Mini Project viva voce, is an authentic record of my work carried under the supervision of **Dr. Manoj Varshney**.

Course: Bachelor of Technology

Branch: Computer Science

Year: III

Semester: 6th

Name of Candidate (Roll No):

- SHRIYANSHI BARANWAL (181500691)
- PRASHANT ASTHANA (181500487)
- TANISHQ TRIPATHI (181500751)

Signature of Candidates:

ACKNOWLEDGMENT

We express our sincere indebtedness towards our mentor, **Dr. Manoj Varshney**, Computer Science & Engineering. GLA University, Mathura for his invaluable guidance, suggestions, and supervision throughout the work. Without his kind patronage and guidance, the project would not have taken shape. We would also like to express our gratitude and sincere regards for his kind approval of the project, time-to-time counseling, and pieces of advice.

We would also like to thank our HOD **Mr. Anand Singh Jalal** Department of Computer Science & Engineering. GLA University, Mathura for her expert advice and counseling from time to time.

We owe sincere thanks to all the faculty members in the department of Computer Science & Engineering for their kind guidance and encouragement from time to time.

Shriyanshi Baranwal(181500691)

Prashant Asthana(181500487)

Tanishq Tripathi(181500751)

ABSTRACT

In our project we are creating an application to combine all basic applications in a single utility. With this one can easily schedule a WhatsApp message, send email to a particular person directly, search anything on Google or can directly download a YouTube video also. The general problems to be addressed by the Social Media Utility Tool is the following:

a. To provide reliable and easy to use Social Media tools that manage an individual user's need.

While a number of commercially available and public domain products have solved these problems, the Social Media Utility Tool provides a solution with certain functional improvements over existing tools. Specific problem areas are the following:

a. Scheduling message sending on a certain time.
b. Sending email direct through the application.
c. Search anything from google.
d. Search any video on YouTube.

Here we will be using python programming language for frontend and Tkinter frame work for GUI.

PROBLEM STATEMENT

The general problems to be addressed by the Social Media Utility Tool are the following:

a. To provide reliable and easy to use Social Media tools that manage an individual user's need.

While a number of commercially available and public domain products have solved these problems, the Social Media Utility Tool provides a solution with certain functional improvements over existing tools. Specific problem areas are the following:

a. Scheduling message sending on a certain time.

b. Sending email direct through the application.

The Social Media Utility Tool addresses the requirements of a particular customer. One needs a tool with a collection of features not available in any single existing tool. This aspect of the problem is one of combining a set of existing features in a way that the features have yet to be combined in another product.

Here we will be using Python for building our application.

REASON FOR SELECTING THE TOPIC

Basically Python has multiple features: -

1. Python's code is easy to read and understand
2. Python is quick
3. Python is Compatible
4. Facilitates Test-Driven Development
5. Strong Standard Library
6. Python Supports Big Data
7. Strong Supportive Community and Corporate Sponsors

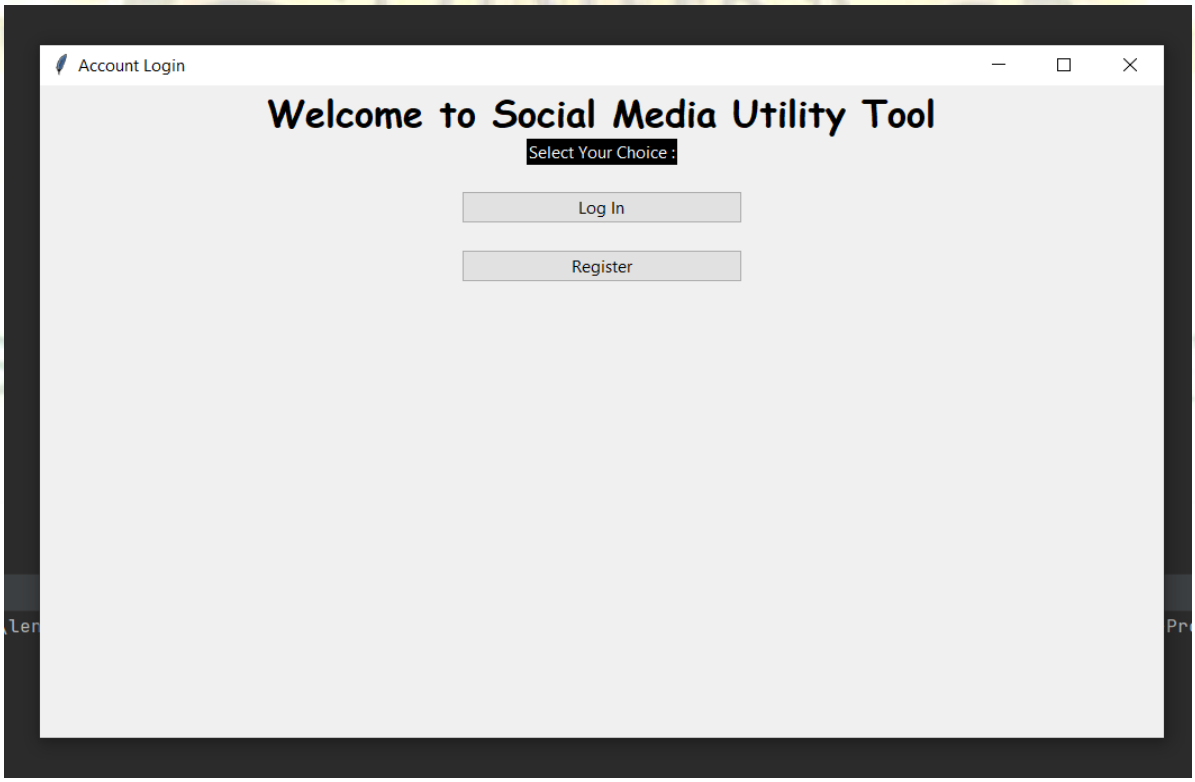
The technology we are going to use is Python and Tkinter. Our main objective for selecting this platform was to learn about it. Python library helps you quickly and easily experiment, build UIs, add features, and fix bugs faster.

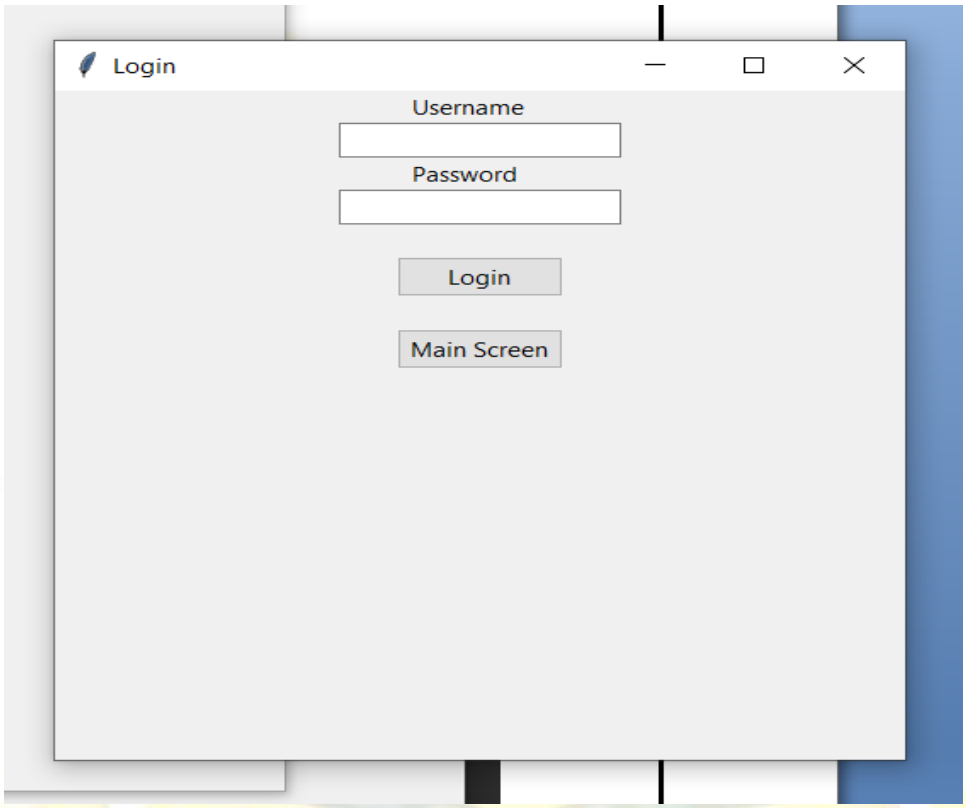
INTRODUCTION

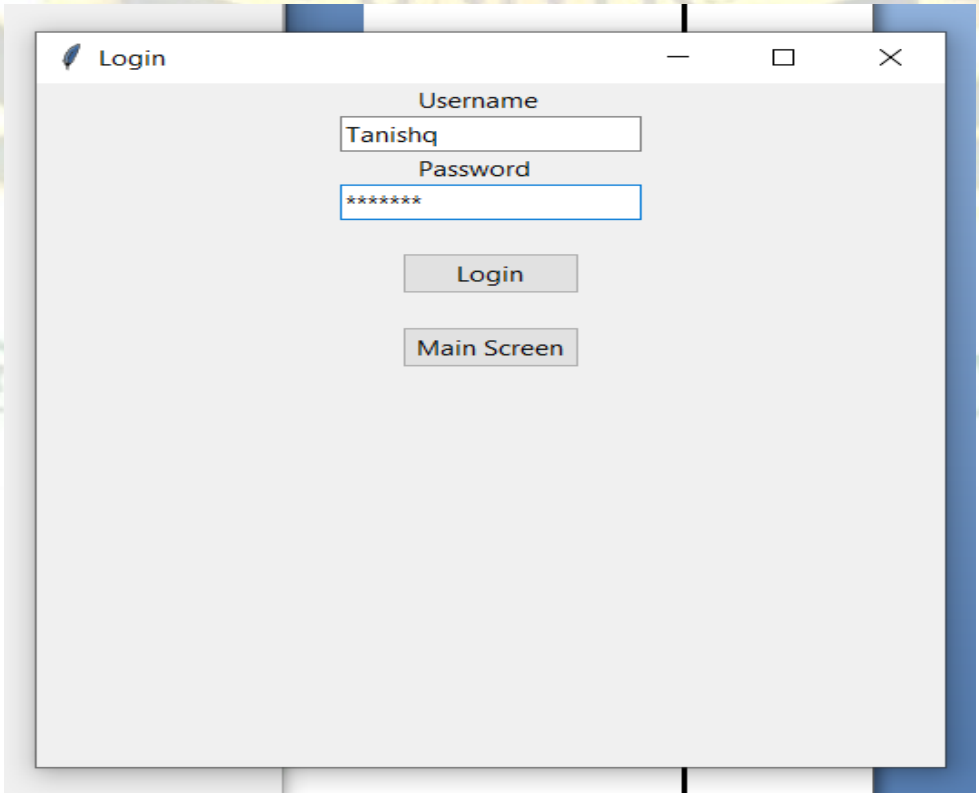
For our project we are creating an application to combine all basic applications in a single utility. Through the application you can easily schedule a WhatsApp message, send email to a particular person, directly search anything on Google or can directly download a you tube video also.

1. **Home Page**-Here we will add 2 options, login and register.

2. **Login Page/Register Page**- By using our credentials i.e., Id & valid password we can login to the welcome page. Here we can also create the new password in case we forgot the old one and can also redirect to the previous page i.e., home page.
3. **Welcome Page**-Here we will and the options for WhatsApp, Gmail, google search, YouTube search.

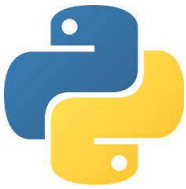



Login
Username
Password
Login
Main Screen


Login
Username
Tanishq
Password
****
Login
Main Screen

LANGUAGE USED

PYTHON:



Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are optional. It has fewer syntactic exceptions and special cases than C or Pascal.

Python is a general-purpose programming language. Hence, we can use the programming language for developing both desktop and web applications. Also, we can use Python for developing complex scientific and numeric applications. Python is designed with features to facilitate data analysis and visualization.

Reason for selecting Python:

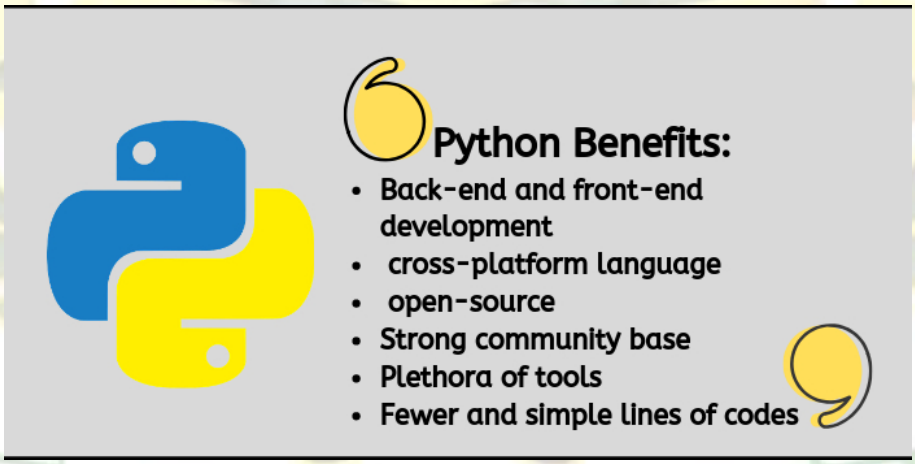
- Readable and Maintainable Code
- Multiple Programming Paradigms
- Compatible with Major Platforms and Systems
- Robust Standard Library
- Many Open-Source Frameworks and Tools
- Simplify Complex Software Development
- Adopt Test Driven Development

Additionally, Python supports the use of modules and packages, which means that programs can be designed in a modular style and code can be reused across a variety of projects. Once you've developed a module or package you need, it can be scaled for use in other projects, and it's easy to import or export these modules.

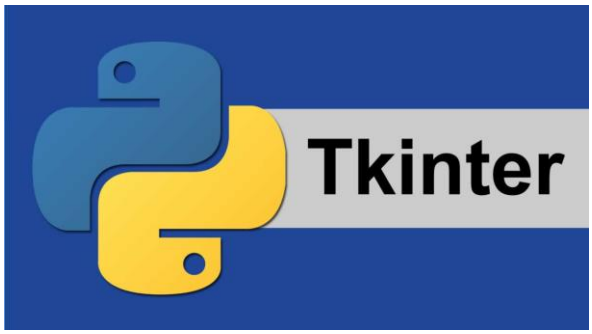
One of the most promising benefits of Python is that both the standard library and the interpreter are available free of charge, in both binary and source form. There is no exclusivity either, as Python and all the necessary tools are available on all major platforms. Therefore, it is an enticing option for developers who don't want to worry about paying high development costs.

If this description of Python over your head, don't worry. You'll understand it soon enough. What you need to take away from this section is that Python is a programming language used to develop software on the web and in app form, including mobile. It's relatively easy to learn, and the necessary tools are available to all free of charge.

That makes Python accessible to almost anyone. If you have the time to learn, you can create some amazing things with the language.



ABOUT TKINTER (LIBRARY IN PYTHON)



Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Tk/Tcl has long been an integral part of Python. It provides a robust and platform independent windowing toolkit, that is available to Python programmers using the tkinter package, and its extension, the tkinter.tix and the tkinter.ttk modules.

The tkinter package is a thin object-oriented layer on top of Tcl/Tk. To use tkinter, you don't need to write Tcl code, but you will need to consult the Tk documentation, and occasionally the Tcl documentation. tkinter is a set of wrappers that implement the Tk widgets as Python classes. In addition, the internal module _tkinter provides a threadsafe mechanism which allows Python and Tcl to interact.

Most of the time, tkinter is all you really need, but a number of additional modules are available as well. The Tk interface is located in a binary module named _tkinter. This module contains the low-level interface to Tk, and should never be used directly by application programmers. It is usually a shared library (or DLL), but might in some cases be statically linked with the Python interpreter.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Example:

```
#!/usr/bin/python
import Tkinter
top = Tkinter.Tk()
# Code to add widgets will go here...
top.mainloop()
```

This would create a following window −

OBJCETIVE & REQUIREMENTS

Objective: -

The main objective of creating such an application is to combine all basic applications in a single utility. Through the application one can easily schedule a WhatsApp message, send email to a particle person, directly search anything on Google or can directly download a YouTube video also. The Social Media Utility Tool addresses the requirements of a particular customer. One needs a tool with a collection of features not available in any single existing tool. This aspect of the problem is one of combining a set of existing features in a way that the features have yet to be combined in another product.

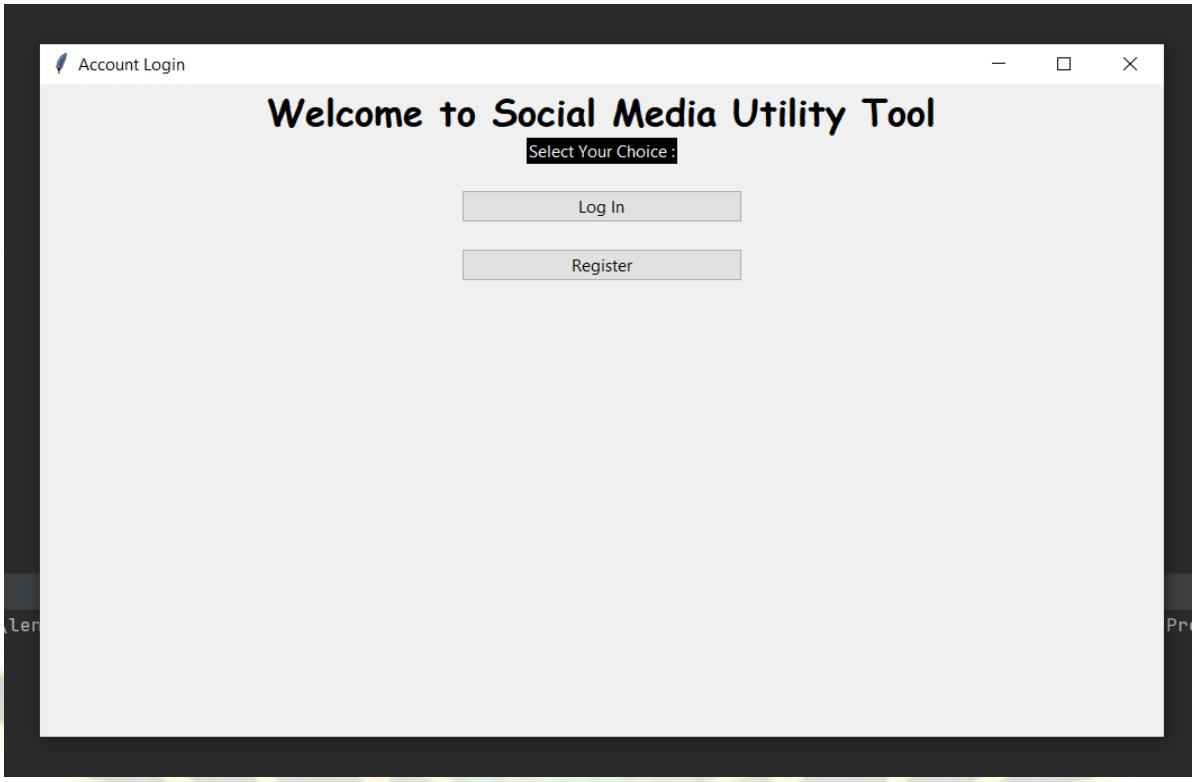
Hardware: -

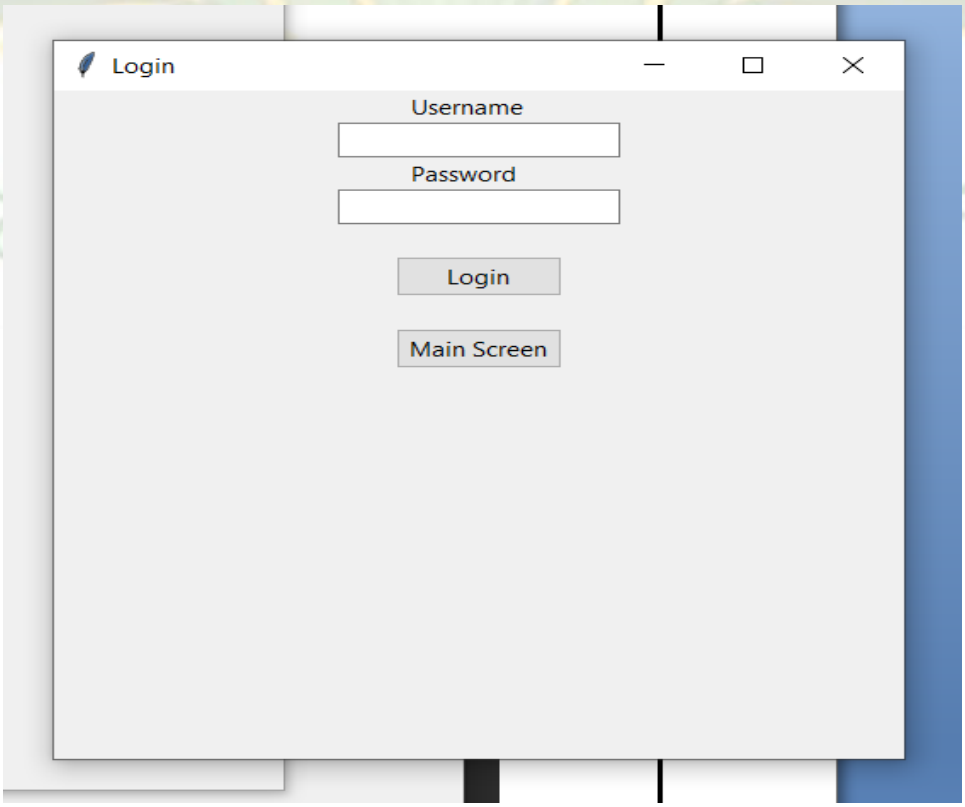
OS: - Mac OSX 10.10+ / Windows 7/8/10
RAM: - 4GiB Recommended
Hard Drive: -1GiB of available disk Space minimum (700MiB for IDE)
Screen Resolution: -1280x800 screen resolution
Processor: -At least i3 8th gen 1.6GHz or Faster Processor

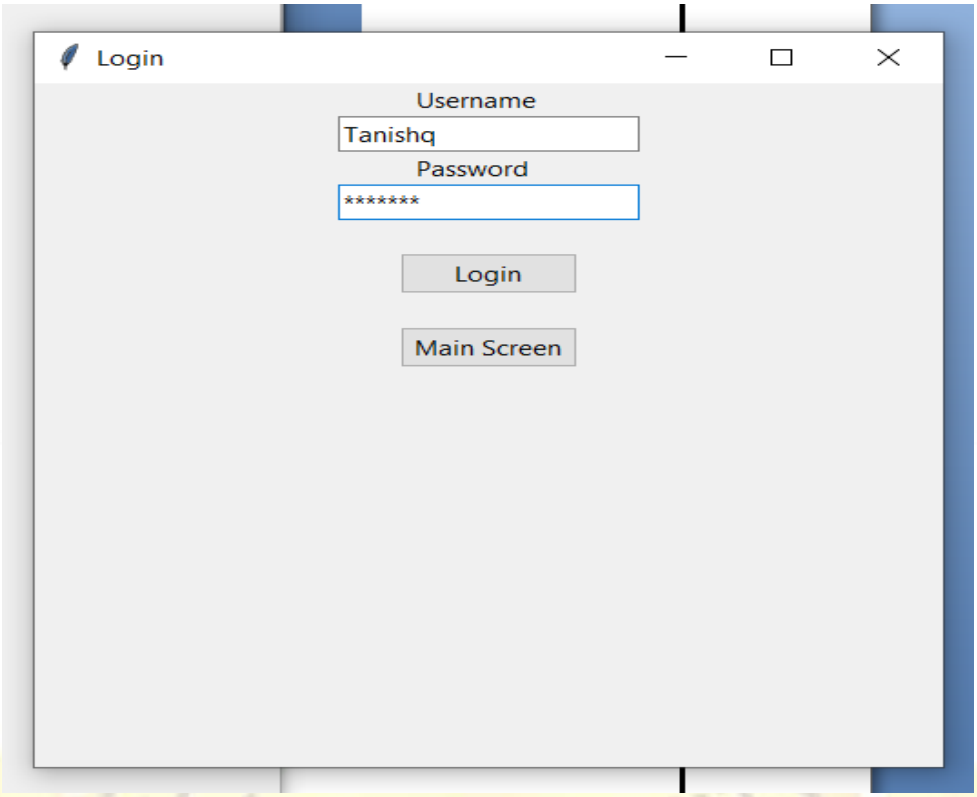
Software: -

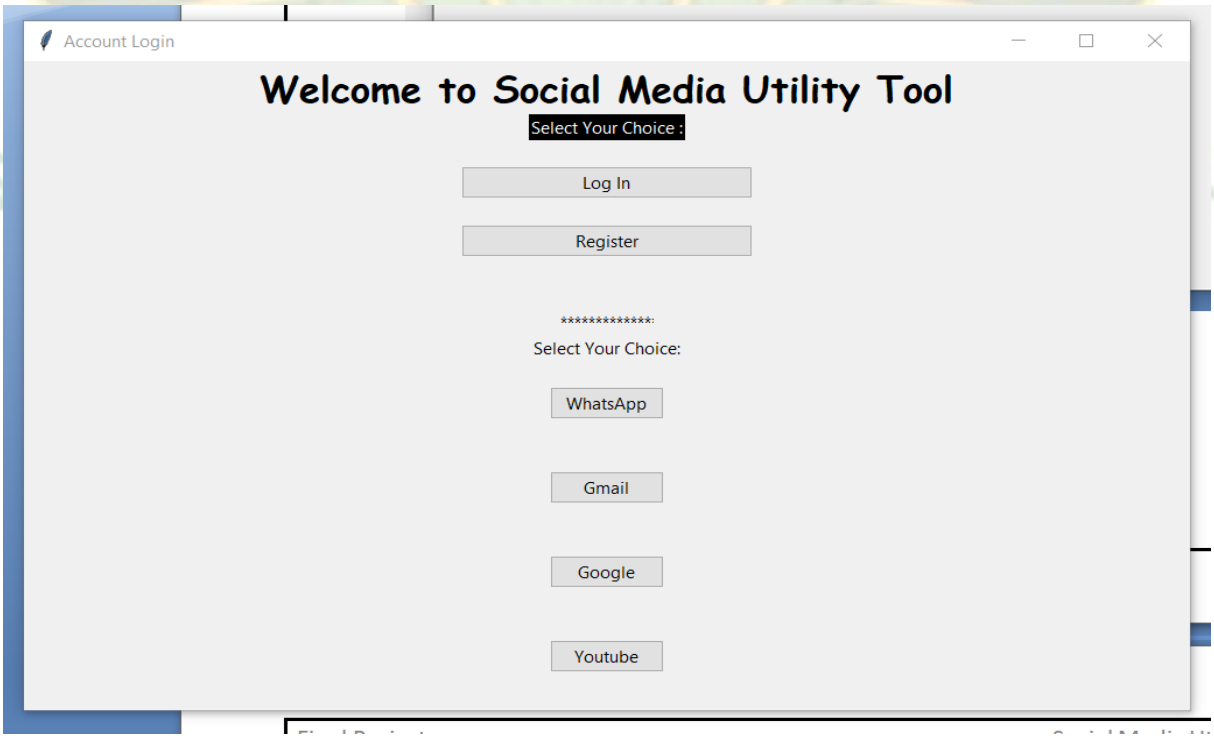
PyCharm CE
Visual Studio Code

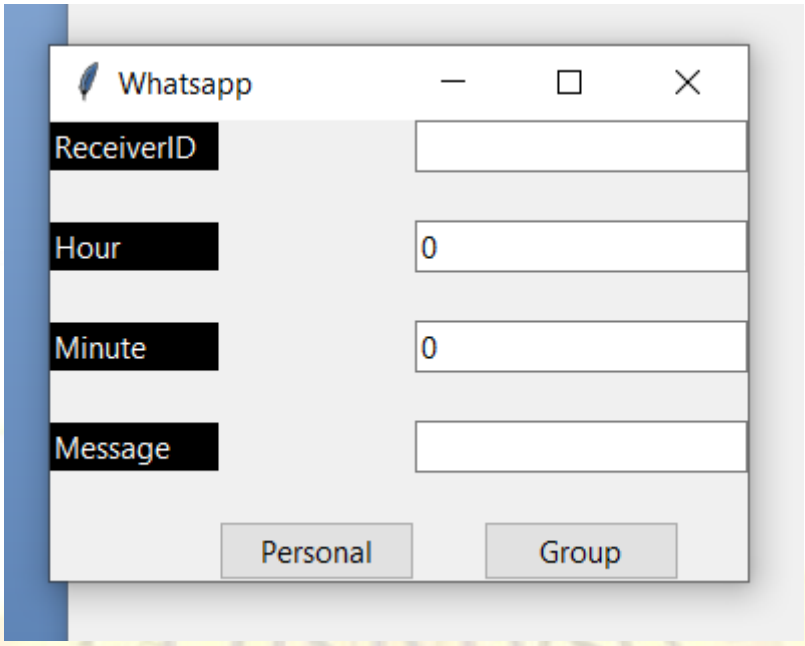
RUNNING PROJECT SCRRENSHOTS

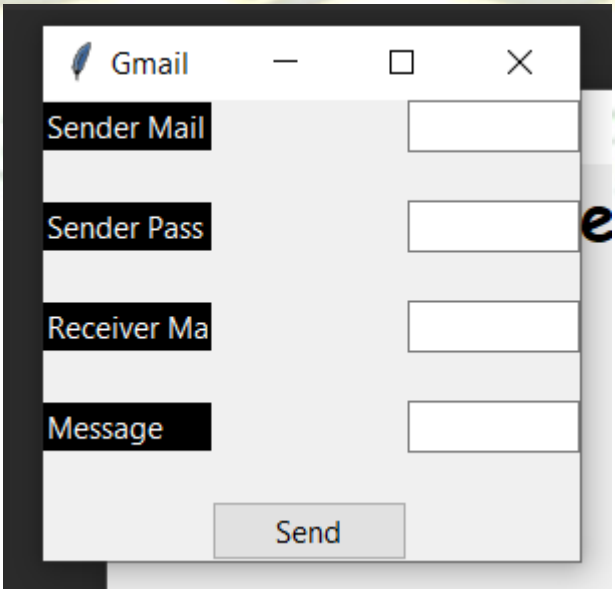


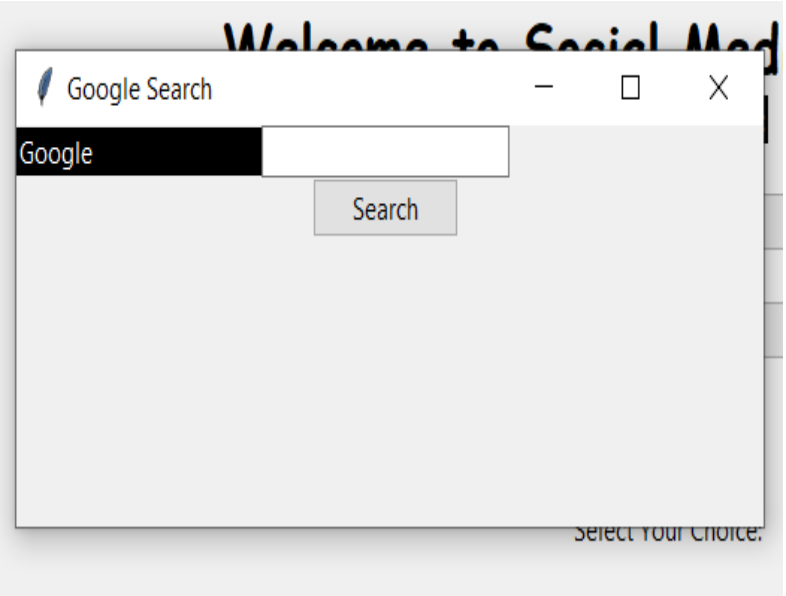


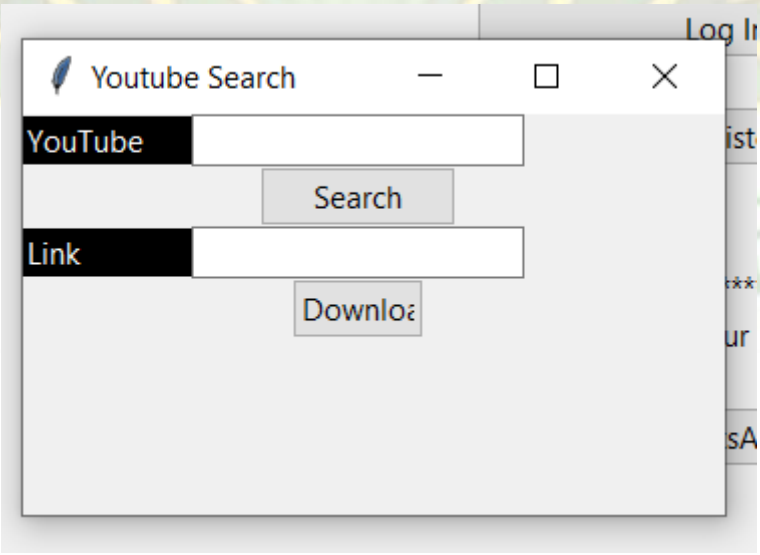

Login
Username
Tanishq
Password
****
Login
Main Screen


Account Login
Welcome to Social Media Utility Tool
Select Your Choice :
Log In
Register
**********:
Select Your Choice:
WhatsApp
Gmail
Google
Youtube


Whatsapp
ReceiverID
Hour
0
Minute
0
Message
Personal
Group

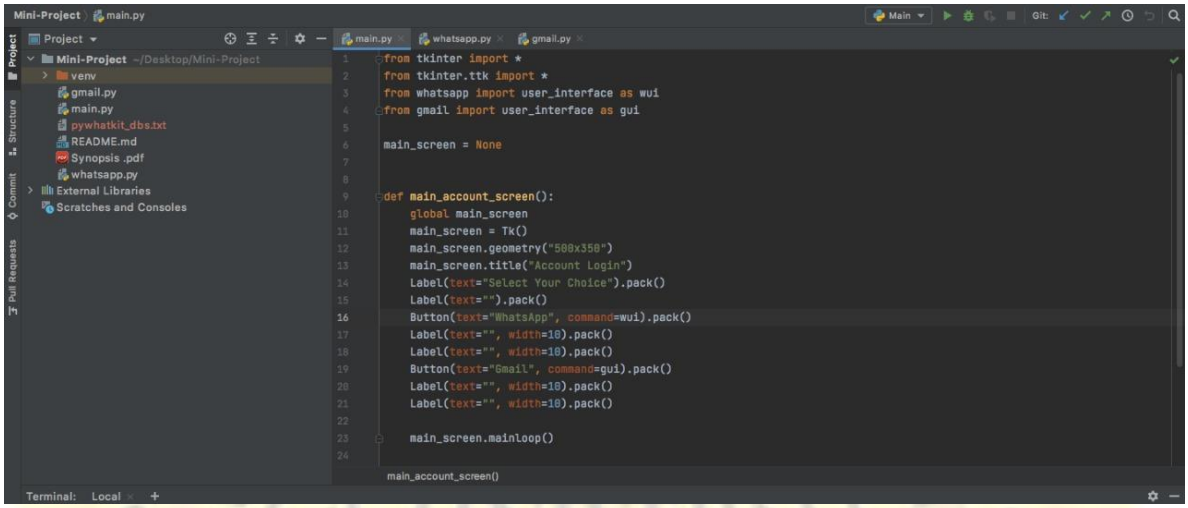

Gmail
Sender Mail
Sender Pass
Receiver Ma
Message
Send


Google Search
Google
Search


Youtube Search
YouTube
Search
Link

CODE SCRRENSHOTS

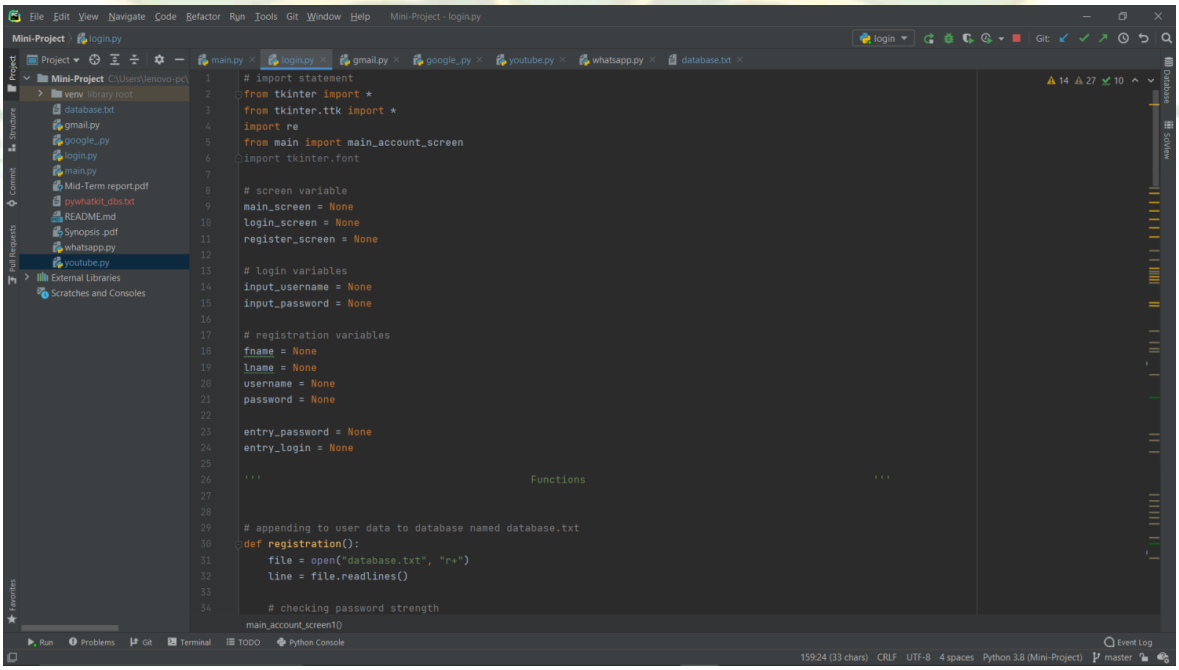
Main.py



```
from tkinter import *
from tkinter.ttk import *
from whatsapp import user_interface as wui
from gmail import user_interface as gui

main_screen = None


def main_account_screen():
    global main_screen
    main_screen = Tk()
    main_screen.geometry("500x350")
    main_screen.title("Account Login")
    Label(text="Select Your Choice").pack()
    Label(text="").pack()
    Button(text="WhatsApp", command=wui).pack()
    Label(text="", width=10).pack()
    Label(text="", width=10).pack()
    Button(text="Gmail", command=gui).pack()
    Label(text="", width=10).pack()
    Label(text="", width=10).pack()

    main_screen.mainloop()

main_account_screen()
```

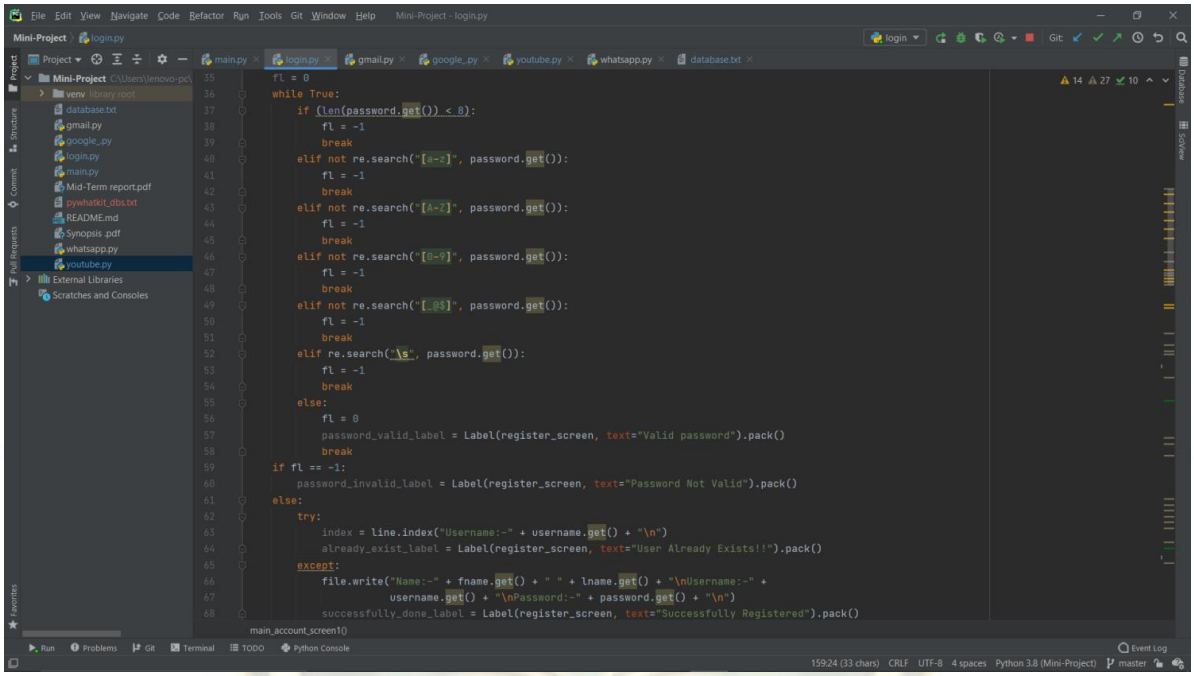
Login.py

1.



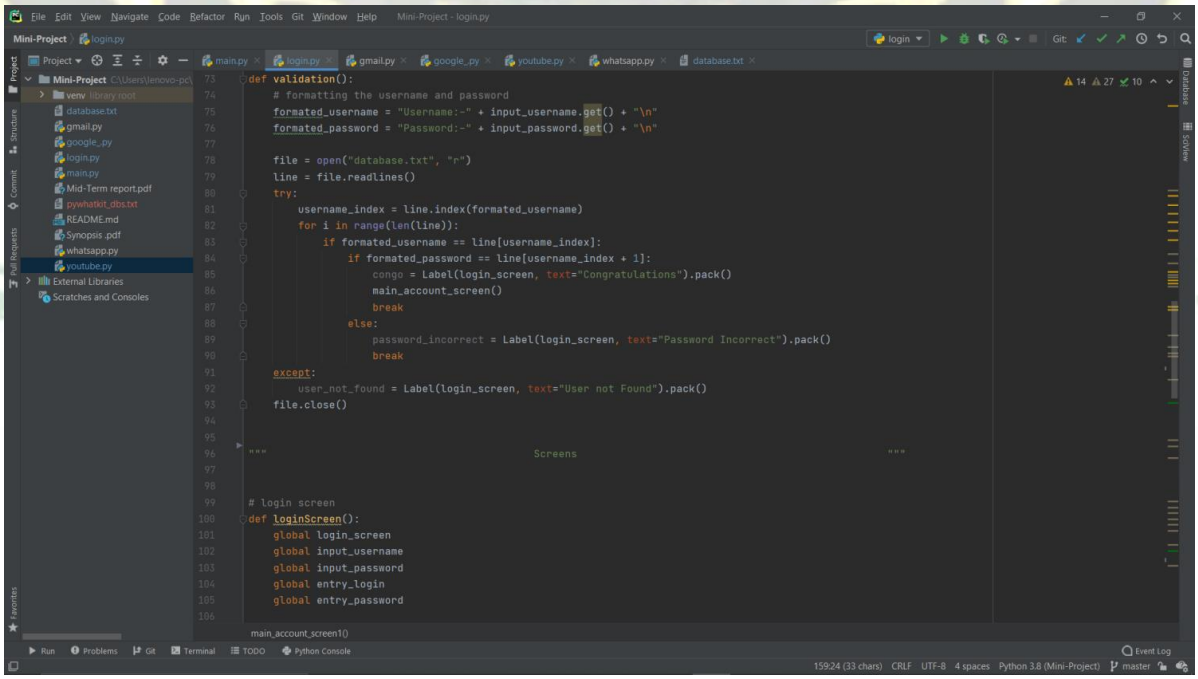
```
# import statement
from tkinter import *
from tkinter.ttk import *
import re
from main import main_account_screen
import tkinter.font

# screen variable
main_screen = None
login_screen = None
register_screen = None

# login variables
input_username = None
input_password = None

# registration variables
fname = None
lname = None
username = None
password = None

entry_password = None
entry_login = None

'''                                    Functions                                    '''


# appending to user data to database named database.txt
def registration():
    file = open("database.txt", "r+")
    line = file.readlines()

    # checking password strength
```

2.



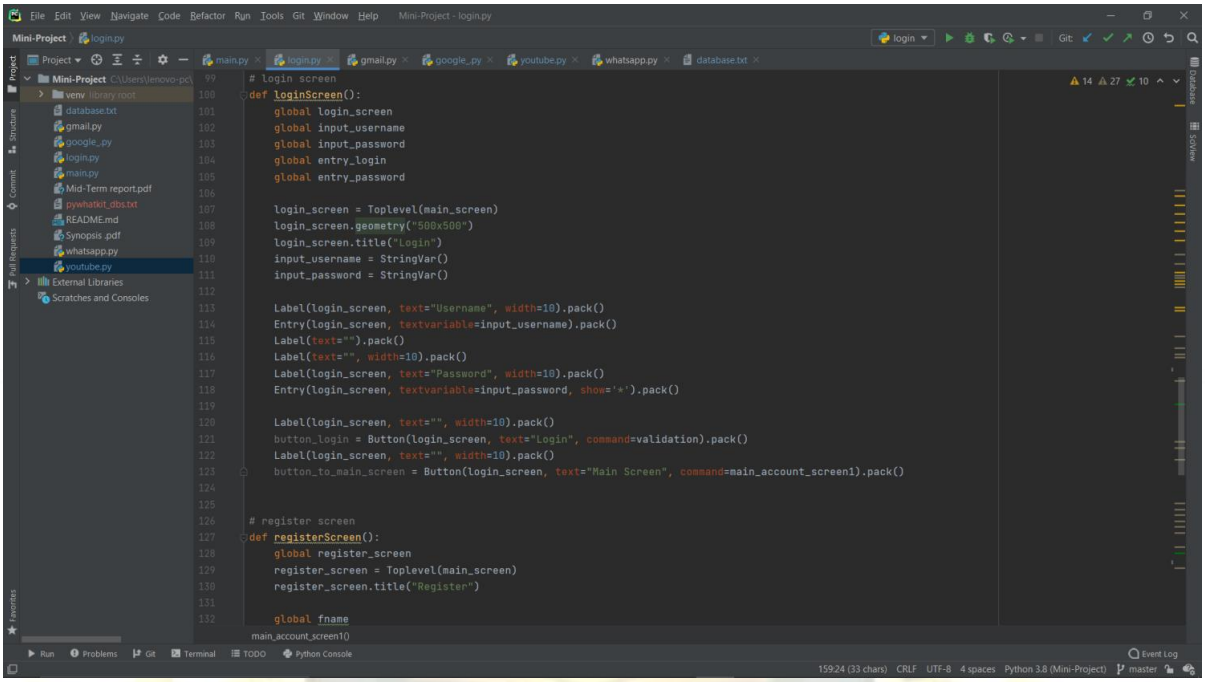
```
    fl = 0
    while True:
        if (len(password.get()) < 8):
            fl = -1
            break
        elif not re.search("[a-z]", password.get()):
            fl = -1
            break
        elif not re.search("[A-Z]", password.get()):
            fl = -1
            break
        elif not re.search("[0-9]", password.get()):
            fl = -1
            break
        elif not re.search("[_@$]", password.get()):
            fl = -1
            break
        elif re.search("\s", password.get()):
            fl = -1
            break
        else:
            fl = 0
            password_valid_label = Label(register_screen, text="Valid password").pack()
            break
    if fl == -1:
        password_invalid_label = Label(register_screen, text="Password Not Valid").pack()
    else:
        try:
            index = line.index("Username:-" + username.get() + "\n")
            already_exist_label = Label(register_screen, text="User Already Exists!!").pack()
        except:
            file.write("Name:-" + fname.get() + " " + lname.get() + "\nUsername:-" +
                       username.get() + "\nPassword:-" + password.get() + "\n")
            successfully_done_label = Label(register_screen, text="Successfully Registered").pack()
```

3.



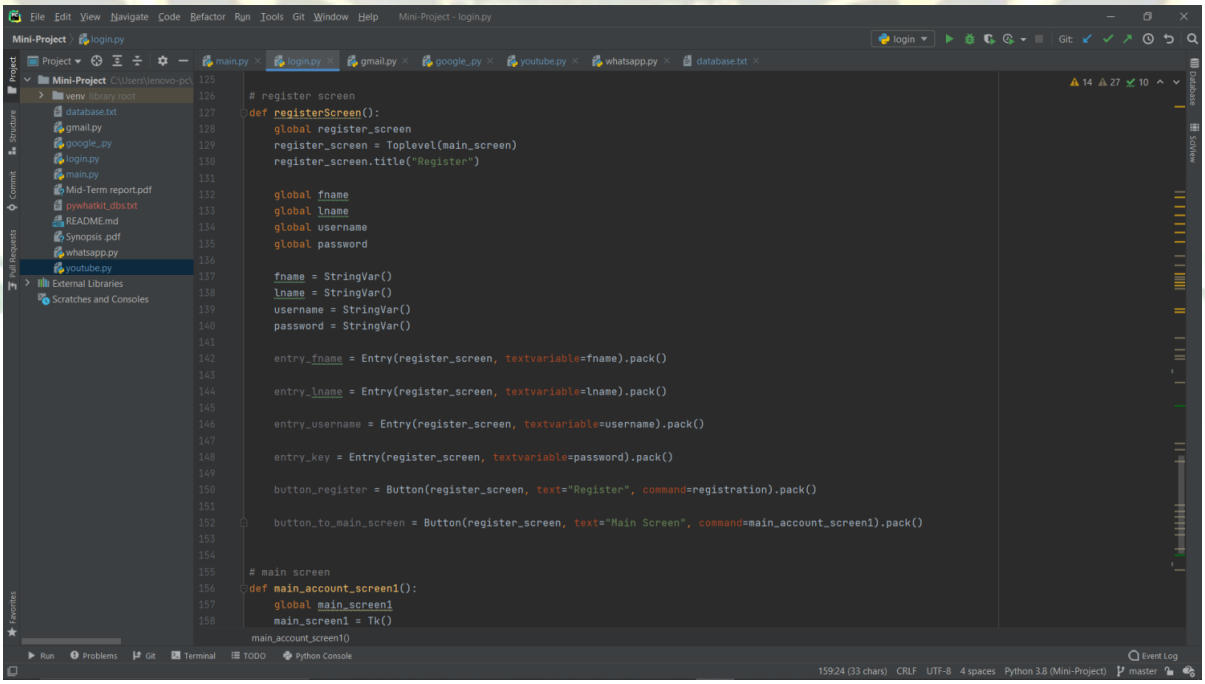
```
def validation():
    # formatting the username and password
    formated_username = "Username:-" + input_username.get() + "\n"
    formated_password = "Password:-" + input_password.get() + "\n"

    file = open("database.txt", "r")
    line = file.readlines()
    try:
        username_index = line.index(formated_username)
        for i in range(len(line)):
            if formated_username == line[username_index]:
                if formated_password == line[username_index + 1]:
                    congo = Label(login_screen, text="Congratulations").pack()
                    main_account_screen()
                    break
                else:
                    password_incorrect = Label(login_screen, text="Password Incorrect").pack()
                    break
    except:
        user_not_found = Label(login_screen, text="User not Found").pack()
    file.close()


""" Screens """


# login screen
def loginScreen():
    global login_screen
    global input_username
    global input_password
    global entry_login
    global entry_password
```

4.



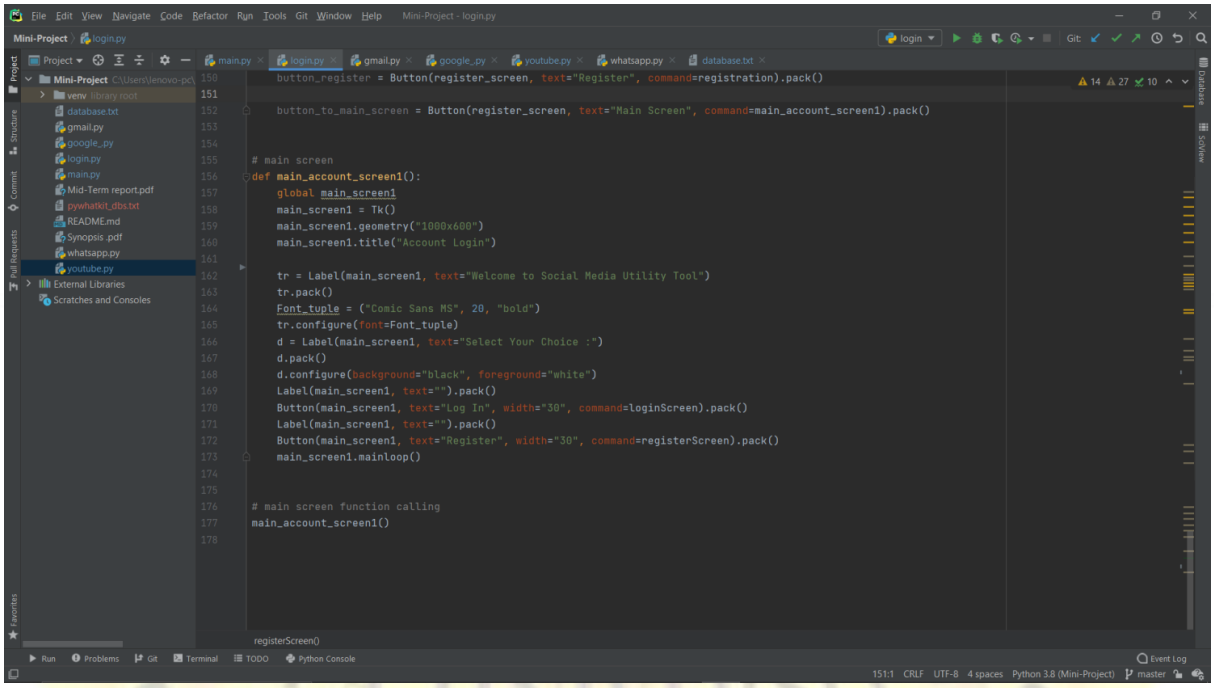
```
# login screen
def loginScreen():
    global login_screen
    global input_username
    global input_password
    global entry_login
    global entry_password

    login_screen = Toplevel(main_screen)
    login_screen.geometry("500x500")
    login_screen.title("Login")
    input_username = StringVar()
    input_password = StringVar()

    Label(login_screen, text="Username", width=10).pack()
    Entry(login_screen, textvariable=input_username).pack()
    Label(text="").pack()
    Label(text="", width=10).pack()
    Label(login_screen, text="Password", width=10).pack()
    Entry(login_screen, textvariable=input_password, show='*').pack()

    Label(login_screen, text="", width=10).pack()
    button_login = Button(login_screen, text="Login", command=validation).pack()
    Label(login_screen, text="", width=10).pack()
    button_to_main_screen = Button(login_screen, text="Main Screen", command=main_account_screen1).pack()


# register screen
def registerScreen():
    global register_screen
    register_screen = Toplevel(main_screen)
    register_screen.title("Register")

    global fname
```

5.



```
# register screen
def registerScreen():
    global register_screen
    register_screen = Toplevel(main_screen)
    register_screen.title("Register")

    global fname
    global lname
    global username
    global password

    fname = StringVar()
    lname = StringVar()
    username = StringVar()
    password = StringVar()

    entry_fname = Entry(register_screen, textvariable=fname).pack()

    entry_lname = Entry(register_screen, textvariable=lname).pack()

    entry_username = Entry(register_screen, textvariable=username).pack()

    entry_key = Entry(register_screen, textvariable=password).pack()

    button_register = Button(register_screen, text="Register", command=registration).pack()

    button_to_main_screen = Button(register_screen, text="Main Screen", command=main_account_screen1).pack()


# main screen
def main_account_screen1():
    global main_screen1
    main_screen1 = Tk()
```

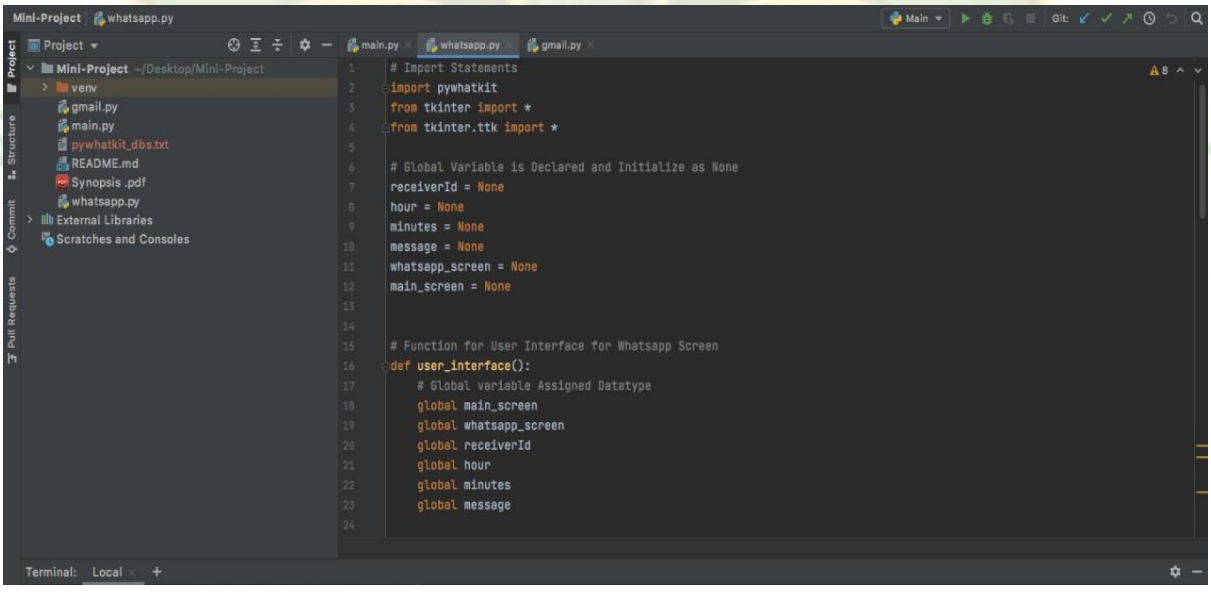
6.



```
    button_register = Button(register_screen, text="Register", command=registration).pack()

    button_to_main_screen = Button(register_screen, text="Main Screen", command=main_account_screen1).pack()


# main screen
def main_account_screen1():
    global main_screen1
    main_screen1 = Tk()
    main_screen1.geometry("1000x600")
    main_screen1.title("Account Login")

    tr = Label(main_screen1, text="Welcome to Social Media Utility Tool")
    tr.pack()
    Font_tuple = ("Comic Sans MS", 20, "bold")
    tr.configure(font=Font_tuple)
    d = Label(main_screen1, text="Select Your Choice :")
    d.pack()
    d.configure(background="black", foreground="white")
    Label(main_screen1, text="").pack()
    Button(main_screen1, text="Log In", width="30", command=loginScreen).pack()
    Label(main_screen1, text="").pack()
    Button(main_screen1, text="Register", width="30", command=registerScreen).pack()
    main_screen1.mainloop()


# main screen function calling
main_account_screen1()
```

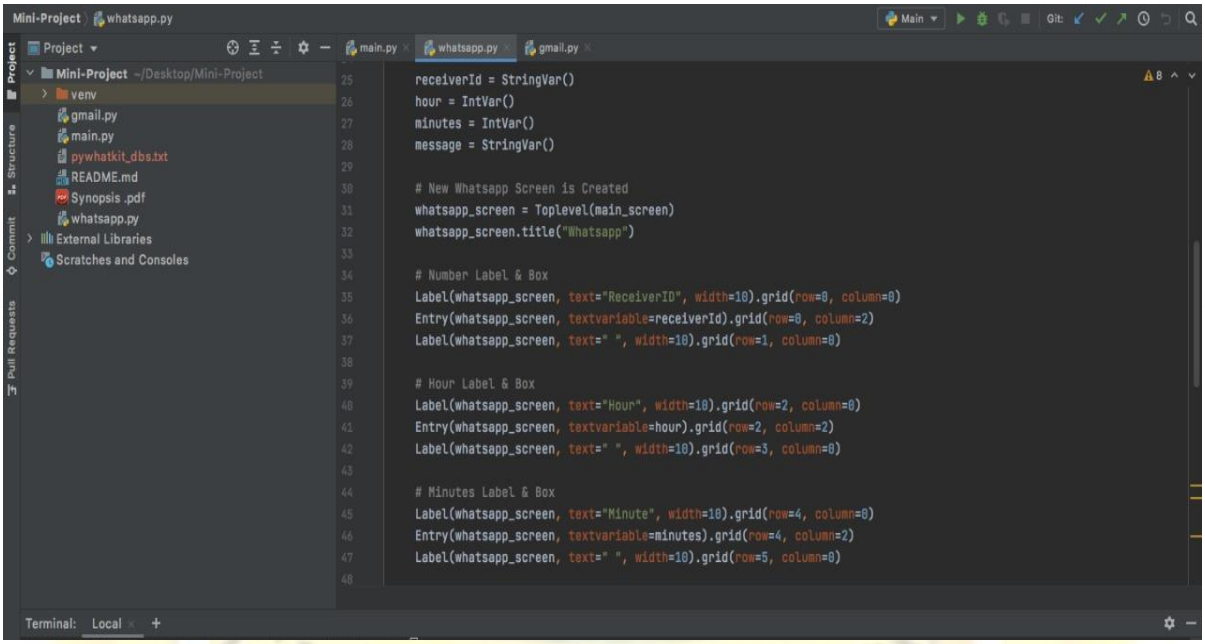
Whatsapp.py

1.

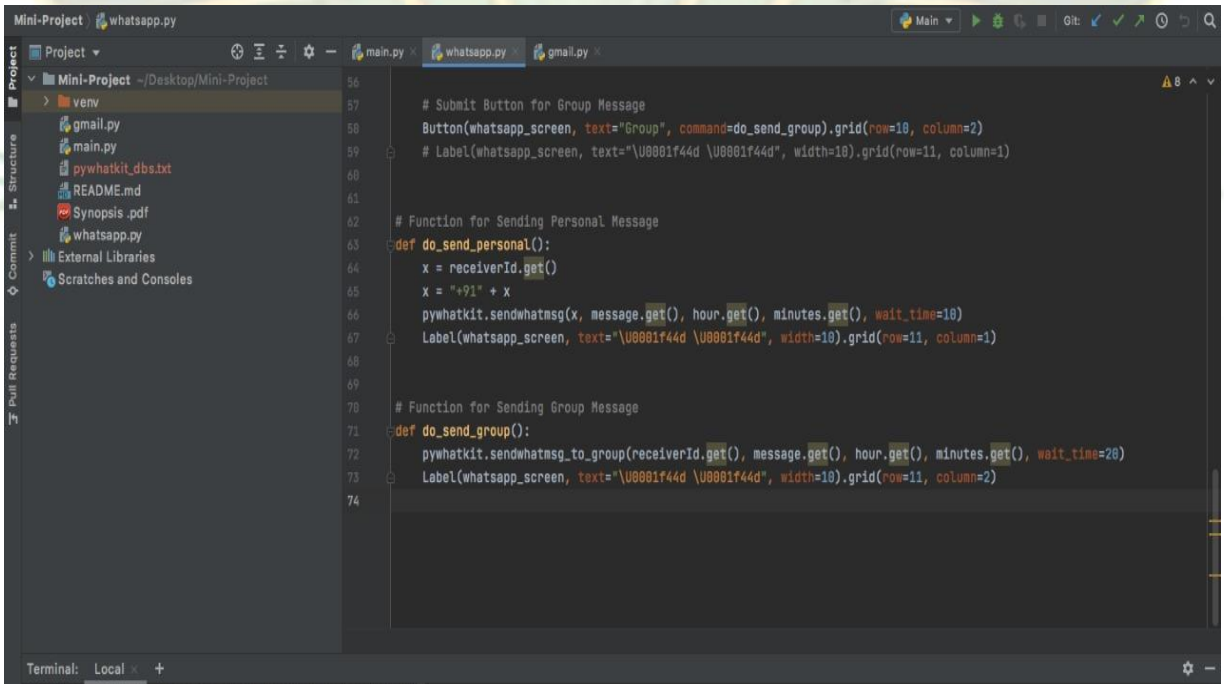


```
# Import Statements
import pywhatkit
from tkinter import *
from tkinter.ttk import *

# Global Variable is Declared and Initialize as None
receiverId = None
hour = None
minutes = None
message = None
whatsapp_screen = None
main_screen = None


# Function for User Interface for Whatsapp Screen
def user_interface():
    # Global variable Assigned Datatype
    global main_screen
    global whatsapp_screen
    global receiverId
    global hour
    global minutes
    global message
```

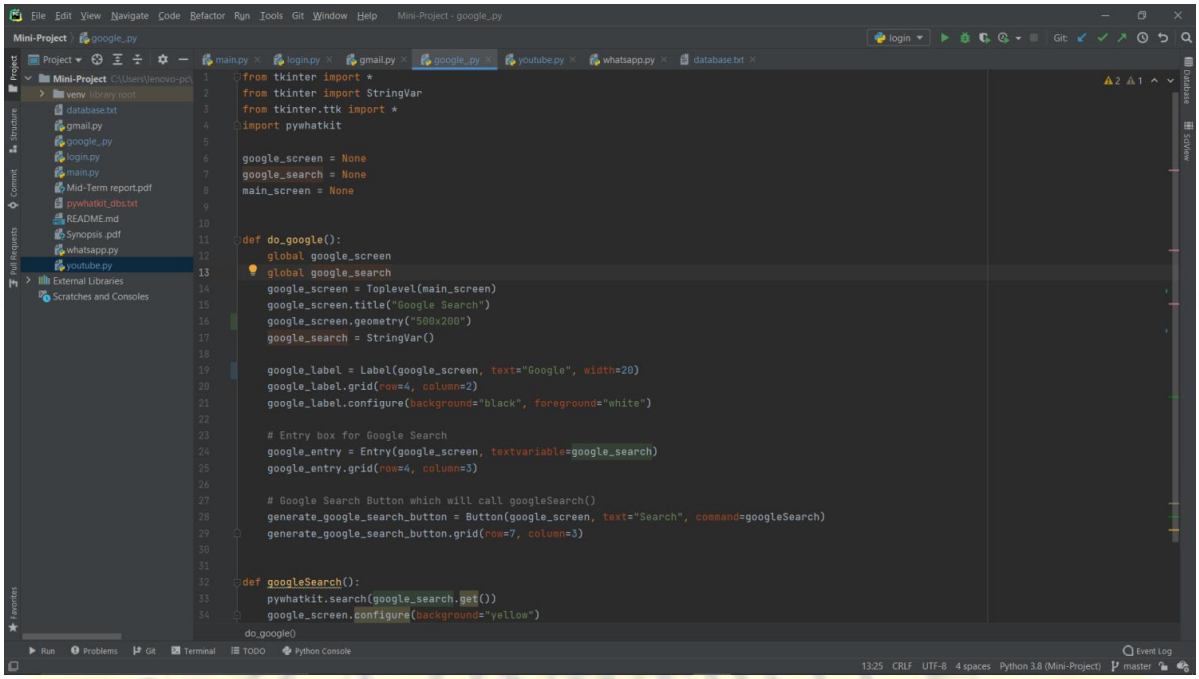
2.


receiverId = StringVar()
hour = IntVar()
minutes = IntVar()
message = StringVar()
# New Whatsapp Screen is Created
whatsapp_screen = Toplevel(main_screen)
whatsapp_screen.title("Whatsapp")
# Number Label & Box
Label(whatsapp_screen, text="ReceiverID", width=10).grid(row=0, column=0)
Entry(whatsapp_screen, textvariable=receiverId).grid(row=0, column=2)
Label(whatsapp_screen, text=" ", width=10).grid(row=1, column=0)
# Hour Label & Box
Label(whatsapp_screen, text="Hour", width=10).grid(row=2, column=0)
Entry(whatsapp_screen, textvariable=hour).grid(row=2, column=2)
Label(whatsapp_screen, text=" ", width=10).grid(row=3, column=0)
# Minutes Label & Box
Label(whatsapp_screen, text="Minute", width=10).grid(row=4, column=0)
Entry(whatsapp_screen, textvariable=minutes).grid(row=4, column=2)
Label(whatsapp_screen, text=" ", width=10).grid(row=5, column=0)

3.


# Submit Button for Group Message
Button(whatsapp_screen, text="Group", command=do_send_group).grid(row=10, column=2)
# Label(whatsapp_screen, text="\U0001f44d \U0001f44d", width=10).grid(row=11, column=1)
# Function for Sending Personal Message
def do_send_personal():
x = receiverId.get()
x = "+91" + x
pywhatkit.sendwhatmsg(x, message.get(), hour.get(), minutes.get(), wait_time=10)
Label(whatsapp_screen, text="\U0001f44d \U0001f44d", width=10).grid(row=11, column=1)
# Function for Sending Group Message
def do_send_group():
pywhatkit.sendwhatmsg_to_group(receiverId.get(), message.get(), hour.get(), minutes.get(), wait_time=20)
Label(whatsapp_screen, text="\U0001f44d \U0001f44d", width=10).grid(row=11, column=2)

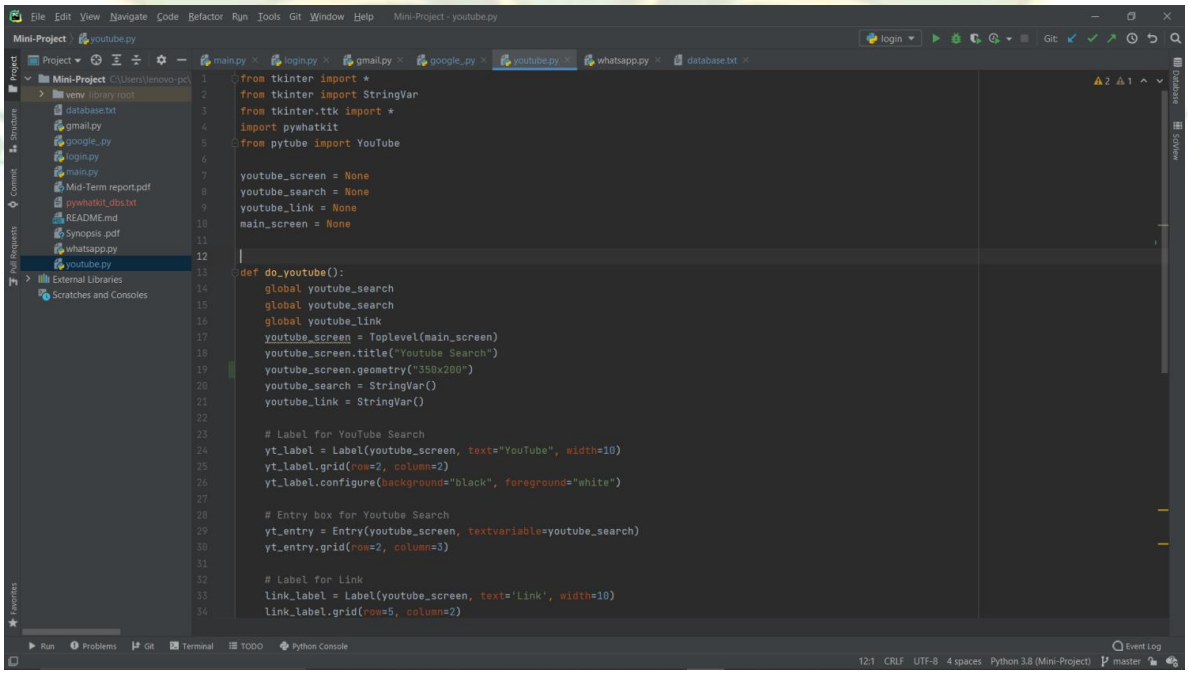
Google.py



```
from tkinter import *
from tkinter import StringVar
from tkinter.ttk import *
import pywhatkit

google_screen = None
google_search = None
main_screen = None


def do_google():
    global google_screen
    global google_search
    google_screen = Toplevel(main_screen)
    google_screen.title("Google Search")
    google_screen.geometry("500x200")
    google_search = StringVar()

    google_label = Label(google_screen, text="Google", width=20)
    google_label.grid(row=4, column=2)
    google_label.configure(background="black", foreground="white")

    # Entry box for Google Search
    google_entry = Entry(google_screen, textvariable=google_search)
    google_entry.grid(row=4, column=3)

    # Google Search Button which will call googleSearch()
    generate_google_search_button = Button(google_screen, text="Search", command=googleSearch)
    generate_google_search_button.grid(row=7, column=3)


def googleSearch():
    pywhatkit.search(google_search.get())
    google_screen.configure(background="yellow")
```

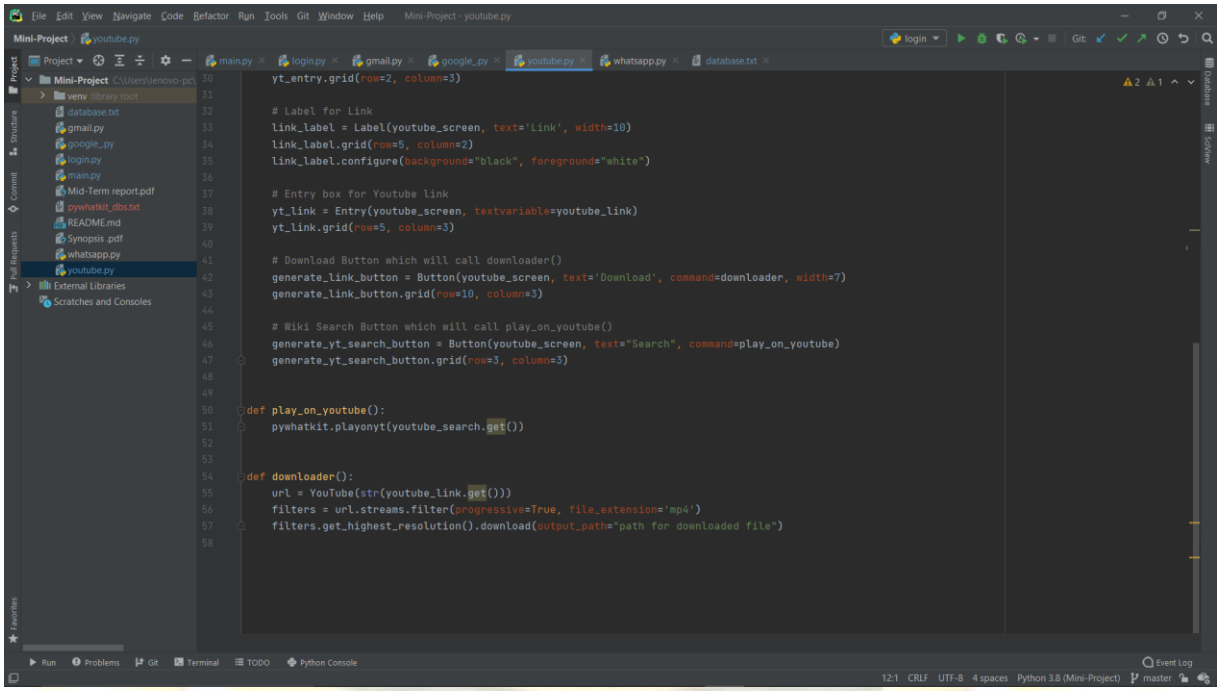
Youtube.py

1.



```
from tkinter import *
from tkinter import StringVar
from tkinter.ttk import *
import pywhatkit
from pytube import YouTube

youtube_screen = None
youtube_search = None
youtube_link = None
main_screen = None


def do_youtube():
    global youtube_search
    global youtube_search
    global youtube_link
    youtube_screen = Toplevel(main_screen)
    youtube_screen.title("Youtube Search")
    youtube_screen.geometry("350x200")
    youtube_search = StringVar()
    youtube_link = StringVar()

    # Label for YouTube Search
    yt_label = Label(youtube_screen, text="YouTube", width=10)
    yt_label.grid(row=2, column=2)
    yt_label.configure(background="black", foreground="white")

    # Entry box for Youtube Search
    yt_entry = Entry(youtube_screen, textvariable=youtube_search)
    yt_entry.grid(row=2, column=3)

    # Label for Link
    link_label = Label(youtube_screen, text='Link', width=10)
    link_label.grid(row=5, column=2)
```

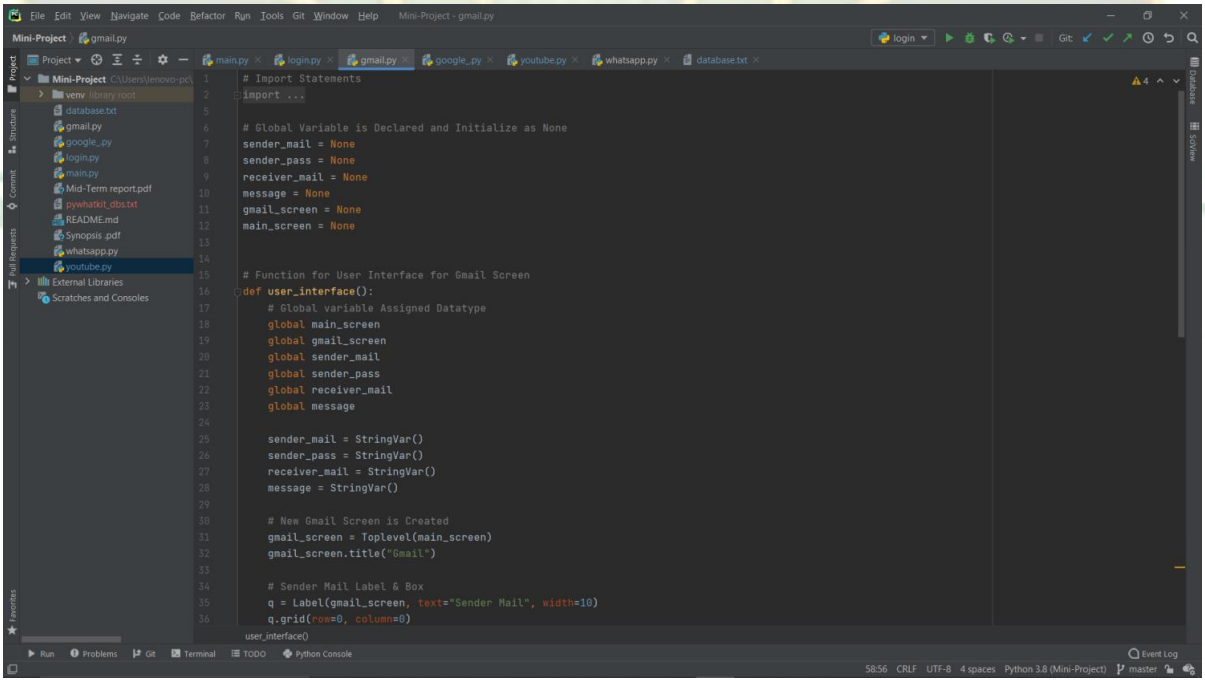
2.



```
    yt_entry.grid(row=2, column=3)

    # Label for Link
    link_label = Label(youtube_screen, text='Link', width=10)
    link_label.grid(row=5, column=2)
    link_label.configure(background="black", foreground="white")

    # Entry box for Youtube link
    yt_link = Entry(youtube_screen, textvariable=youtube_link)
    yt_link.grid(row=5, column=3)

    # Download Button which will call downloader()
    generate_link_button = Button(youtube_screen, text='Download', command=downloader, width=7)
    generate_link_button.grid(row=10, column=3)

    # Wiki Search Button which will call play_on_youtube()
    generate_yt_search_button = Button(youtube_screen, text="Search", command=play_on_youtube)
    generate_yt_search_button.grid(row=3, column=3)


def play_on_youtube():
    pywhatkit.playonyt(youtube_search.get())


def downloader():
    url = YouTube(str(youtube_link.get()))
    filters = url.streams.filter(progressive=True, file_extension='mp4')
    filters.get_highest_resolution().download(output_path="path for downloaded file")
```

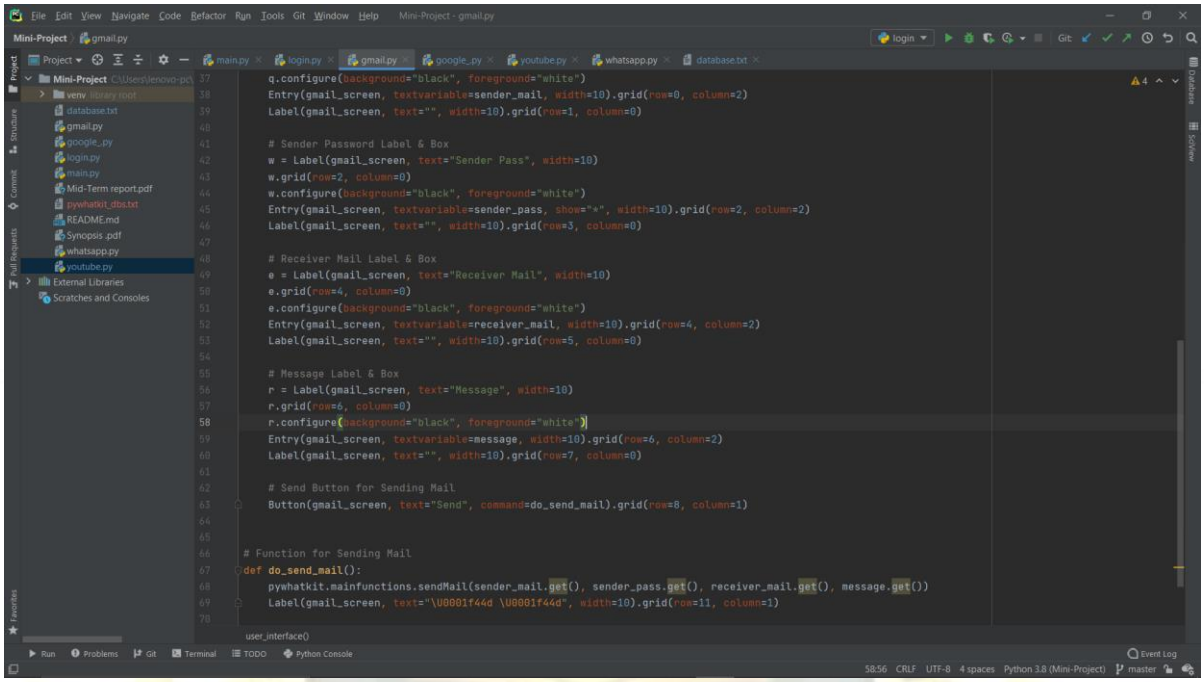
Gmail.py

1.



```
# Import Statements
import ...

# Global Variable is Declared and Initialize as None
sender_mail = None
sender_pass = None
receiver_mail = None
message = None
gmail_screen = None
main_screen = None


# Function for User Interface for Gmail Screen
def user_interface():
    # Global variable Assigned Datatype
    global main_screen
    global gmail_screen
    global sender_mail
    global sender_pass
    global receiver_mail
    global message

    sender_mail = StringVar()
    sender_pass = StringVar()
    receiver_mail = StringVar()
    message = StringVar()

    # New Gmail Screen is Created
    gmail_screen = Toplevel(main_screen)
    gmail_screen.title("Gmail")

    # Sender Mail Label & Box
    q = Label(gmail_screen, text="Sender Mail", width=10)
    q.grid(row=0, column=0)
```

2.



```
    q.configure(background="black", foreground="white")
    Entry(gmail_screen, textvariable=sender_mail, width=10).grid(row=0, column=2)
    Label(gmail_screen, text="", width=10).grid(row=1, column=0)

    # Sender Password Label & Box
    w = Label(gmail_screen, text="Sender Pass", width=10)
    w.grid(row=2, column=0)
    w.configure(background="black", foreground="white")
    Entry(gmail_screen, textvariable=sender_pass, show="*", width=10).grid(row=2, column=2)
    Label(gmail_screen, text="", width=10).grid(row=3, column=0)

    # Receiver Mail Label & Box
    e = Label(gmail_screen, text="Receiver Mail", width=10)
    e.grid(row=4, column=0)
    e.configure(background="black", foreground="white")
    Entry(gmail_screen, textvariable=receiver_mail, width=10).grid(row=4, column=2)
    Label(gmail_screen, text="", width=10).grid(row=5, column=0)

    # Message Label & Box
    r = Label(gmail_screen, text="Message", width=10)
    r.grid(row=6, column=0)
    r.configure(background="black", foreground="white")
    Entry(gmail_screen, textvariable=message, width=10).grid(row=6, column=2)
    Label(gmail_screen, text="", width=10).grid(row=7, column=0)

    # Send Button for Sending Mail
    Button(gmail_screen, text="Send", command=do_send_mail).grid(row=8, column=1)


# Function for Sending Mail
def do_send_mail():
    pywhatkit.mainfunctions.sendMail(sender_mail.get(), sender_pass.get(), receiver_mail.get(), message.get())
    Label(gmail_screen, text="\U0001f44d \U0001f44d", width=10).grid(row=11, column=1)
```

FUTURE SCOPE

We are going to add other social media applications like Instagram, twitter etc., to make our application more versatile and useful. We will try to add a feature in it so that we can send Automated Message to a Group in WhatsApp using Group ID. We are thinking to add a smart assistant button in out tools so that all the work can be done by a voice-controlled system. This assistant can add up reminder of your work and can make a task list on its own as well as set up an alarm for your work. We are thinking to add a button in YouTube Screen to select the resolution of the video that is going to be downloaded. In the near future we are going to add more modules and functionalities to make it a complete package.

TESTING TECHNOLOGY TO BE USED

- At First Stage we are going to test out Application on our personal system.
- At Second Stage we are going to test the Application on our colleague's system as well.
- At Third and Final Stage we are going to test the Application on a group of users to check the feasibility and stability of our application.

THE CONTRIBUTIONS OUR PROJECT IS GOING TO MAKE

We are going to create an application which is a social media utility tool, which is a single application containing

Different type of social media tools like WhatsApp, Instagram so you need not to switch to different applications, you can access all these through a single tool, which is names as Social Media Utility Tool.

SCOPE FOR EXTENSION INTO MAJOR PROJECT

In this project we are going to create a login page and after logging in person can send a message or email or can do a Google search or can watch YouTube videos as well as download the YouTube video. After completion of this project, we are going to extend our project by adding more social media applications like twitter, Instagram, snapchat, Facebook etc.

CONCLUSION

At the very end of this we are having a tool that can: -

1. Automate whatsapp message for a certain time on a certain mobile number.
2. Send e-mail using Gmail's SMTP communication protocol.
3. Download a YouTube video using only its URL (Uniform Resource Locator)
4. Search anything on Google on a single tap.

REFERENCES

- ***https://www.w3schools.com/***
- ***https://stackoverflow.com/***
- ***https://www.tutorialspoint.com/***
- ***https://www.youtube.com/***