**PROJECT REPORT**

ON

**WHATSNAP**

Submitted In Partial Fulfillment of the Requirements

For the Degree of

**Bachelor of Technology**

In

INFORMATION TECHNOLOGY

**Submitted By:**

Versha (1217648)

BATCH: 2017-2021

**Content:**

**Chapter page no**

Chapter 1: Introduction-----------------------------------------------6-12

Chapter 2: tools used-------------------------------------------------13

Chapter 3: Implementation------------------------------------------14-36

Chapter 4: output -----------------------------------------------------37-47

Chapter 5: conclusion------------------------------------------------48

**Tables Index**:

**Tables page no**

Table 1-------------------------------------------------------------------10-12

**CHAPTER 1 - INTRODUCTION**

**PYTHON**

**Python Language Introduction**

[Python](https://www.geeksforgeeks.org/python-programming-language/) is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**History of Python**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

**Python Features**

Python's features include −

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**Python graphical user interfaces (GUIs)**

* **Tkinter** − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter.
* **wxPython** − This is an open-source Python interface for wxWindows [http://wxpython.org](http://wxpython.org/).
* **JPython** − JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine [http://www.jython.org](http://www.jython.org/).

There are many other interfaces available, which you can find them on the net.

**PYTHON TKINTER GUI**

Tkinter Programming



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.

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

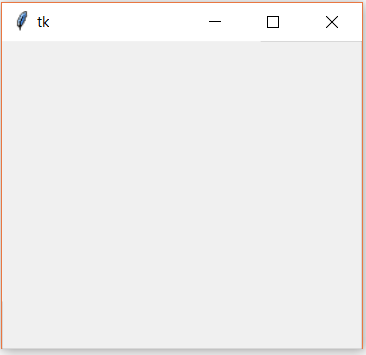
import tkinter

top = tkinter.Tk()

# Code to add widgets will go here...

top.mainloop()

This would create a following window −



**Tkinter Widgets**

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

There are currently 15 types of widgets in Tkinter. We present these widgets as well as a brief description in the following table

|  |  |
| --- | --- |
| **Sr.No.** | **Operator & Description** |
| 1 | **[Button](https://www.tutorialspoint.com/python/tk_button.htm)**  The Button widget is used to display buttons in your application. |
| 2 | **[Canvas](https://www.tutorialspoint.com/python/tk_canvas.htm)**  The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application. |
| 3 | **[Checkbutton](https://www.tutorialspoint.com/python/tk_checkbutton.htm)**  The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time. |
| 4 | **[Entry](https://www.tutorialspoint.com/python/tk_entry.htm)**  The Entry widget is used to display a single-line text field for accepting values from a user. |
| 5 | **[Frame](https://www.tutorialspoint.com/python/tk_frame.htm)**  The Frame widget is used as a container widget to organize other widgets. |
| 6 | **[Label](https://www.tutorialspoint.com/python/tk_label.htm)**  The Label widget is used to provide a single-line caption for other widgets. It can also contain images. |
| 7 | **[Listbox](https://www.tutorialspoint.com/python/tk_listbox.htm)**  The Listbox widget is used to provide a list of options to a user. |
| 8 | **[Menubutton](https://www.tutorialspoint.com/python/tk_menubutton.htm)**  The Menubutton widget is used to display menus in your application. |
| 9 | **[Menu](https://www.tutorialspoint.com/python/tk_menu.htm)**  The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton. |
| 10 | **[Message](https://www.tutorialspoint.com/python/tk_message.htm)**  The Message widget is used to display multiline text fields for accepting values from a user. |
| 11 | **[Radiobutton](https://www.tutorialspoint.com/python/tk_radiobutton.htm)**  The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time. |
| 12 | **[Scale](https://www.tutorialspoint.com/python/tk_scale.htm)**  The Scale widget is used to provide a slider widget. |
| 13 | **[Scrollbar](https://www.tutorialspoint.com/python/tk_scrollbar.htm)**  The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes. |
| 14 | **[Text](https://www.tutorialspoint.com/python/tk_text.htm)**  The Text widget is used to display text in multiple lines. |
| 15 | **[Toplevel](https://www.tutorialspoint.com/python/tk_toplevel.htm)**  The Toplevel widget is used to provide a separate window container. |
| 16 | **[Spinbox](https://www.tutorialspoint.com/python/tk_spinbox.htm)**  The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values. |
| 17 | **[PanedWindow](https://www.tutorialspoint.com/python/tk_panedwindow.htm)**  A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically. |
| 18 | **[LabelFrame](https://www.tutorialspoint.com/python/tk_labelframe.htm)**  A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts. |
| 19 | **[tkMessageBox](https://www.tutorialspoint.com/python/tk_messagebox.htm)**  This module is used to display message boxes in your applications. |

**Geometry Management**

All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: pack, grid, and place.

* **[The](https://www.tutorialspoint.com/python/tk_pack.htm)*[pack()](https://www.tutorialspoint.com/python/tk_pack.htm)*[Method](https://www.tutorialspoint.com/python/tk_pack.htm)** − This geometry manager organizes widgets in blocks before placing them in the parent widget.
* **[The](https://www.tutorialspoint.com/python/tk_grid.htm)*[grid()](https://www.tutorialspoint.com/python/tk_grid.htm)*[Method](https://www.tutorialspoint.com/python/tk_grid.htm)** − This geometry manager organizes widgets in a table-like structure in the parent widget.
* **[The](https://www.tutorialspoint.com/python/tk_place.htm)*[place()](https://www.tutorialspoint.com/python/tk_place.htm)*[Method](https://www.tutorialspoint.com/python/tk_place.htm)** − This geometry manager organizes widgets by placing them in a specific position in the parent widget.

**Chapter -2**

**PLATFORMS/ TOOLS USED**

**Hardware Platform:**

Processor: i3 or above

Processor speed: 2.00GHz CPU

RAM: 4GB or above

Hard disk: 50GB or above

**Software Platform:**

Framework: Pycharm

Front End: tkinter

Back End: Python

Operation System: Windows or any equivalent

**CHAPTER -3**

**IMPLEMENTATION**

**SERVER**

import getpass  
import os  
import time  
from tkinter import \*  
from threading import Thread  
from tkinter import messagebox  
from PIL import ImageTk, Image  
import tkinter as tk  
import socket  
from plyer import notification  
from tkinter import filedialog  
import os  
import ctypes  
import sqlite3  
import wikipedia  
import cv2  
  
kernel32 = ctypes.WinDLL('kernel32')  
user32 = ctypes.WinDLL('user32')  
SW\_HIDE = 0  
hWnd = kernel32.GetConsoleWindow()  
user32.ShowWindow(hWnd, SW\_HIDE)  
  
root = Tk()  
root.title("H-PINGER")  
root.geometry("350x200")  
root.resizable(width=False, height=False)  
  
from itertools import count  
  
  
class ImageLabel(tk.Label):  
  
 def load(self, im):  
 if isinstance(im, str):  
 im = Image.open(im)  
 self.loc = 0  
 self.frames = []  
  
 try:  
 for i in count(1):  
 self.frames.append(ImageTk.PhotoImage(im.copy()))  
 im.seek(i)  
 except EOFError:  
 pass  
  
 try:  
 self.delay = im.info['duration']  
 except:  
 self.delay = 100  
  
 if len(self.frames) == 1:  
 self.config(image=self.frames[0])  
 else:  
 self.next\_frame()  
  
 def unload(self):  
 self.config(image=None)  
 self.frames = None  
  
 def next\_frame(self):  
 if self.frames:  
 self.loc += 1  
 self.loc %= len(self.frames)  
 self.config(image=self.frames[self.loc])  
 self.after(self.delay, self.next\_frame)  
  
  
def center(toplevel):  
 toplevel.update\_idletasks()  
 screen\_width = toplevel.winfo\_screenwidth()  
 screen\_height = toplevel.winfo\_screenheight()  
 size = tuple(int(\_) for \_ in toplevel.geometry().split('+')[0].split('x'))  
 x = screen\_width / 2.5 - size[0] / 3  
 y = screen\_height / 2.5 - size[1] / 3  
 toplevel.geometry("+%d+%d" % (x, y))  
  
  
def handle\_client(c, addr):  
 def close\_after\_2s():  
 root.iconify()  
  
 root.after(4000, close\_after\_2s)  
 client\_username = c.recv(4141)  
 client\_username = client\_username.decode('ascii')  
  
 top = Toplevel()  
 top.title('H-PINGER')  
 top.geometry("400x500")  
 top.attributes('-topmost', 1)  
 top.attributes('-topmost', 0)  
 center(top)  
 top.resizable(width=FALSE, height=FALSE)  
 inputentry = Text(top, bd=0, bg="white", width="29", height="5", font=("Arial", 12))  
 inputentry.configure(highlightbackground='lightgrey', highlightthickness=1)  
 inputentry.bind('<Return>', (lambda event: send()))  
 img = ImageTk.PhotoImage(Image.open("user4.png"))  
 panel = Label(top, image=img)  
 inputentry.place(x=128, y=401, height=90, width=265)  
 img = ImageTk.PhotoImage(Image.open("user4.png"))  
 panel = Label(top, image=img)  
 prompt = client\_username  
 user\_lb = Label(top, text=prompt, width=len(prompt), font=(12,))  
 prompt = addr[0]  
 ip\_lb = Label(top, text=prompt, width=len(prompt), font=("Arial", 8))  
 outputtext = Text(top, bd=0, bg="white", height="8", width="50", font=("Arial", 12))  
 outputtext.configure(highlightbackground='lightgrey', highlightthickness=1)  
 scrollbar = Scrollbar(top, command=outputtext.yview)  
 outputtext['yscrollcommand'] = scrollbar.set  
 SendButton = Button(top, font=30, text="Send", width="12", height=5, bd=0, command=(lambda: send()))  
 send\_img = PhotoImage(file="e2.png") # make sure to add "/" not "\"  
 SendButton.config(image=send\_img)  
 up = Button(top, text="Attach", font=30, width="18", height=3, bd=0, command=(lambda: FileTransfer()))  
 upbt\_img = PhotoImage(file="at3.png") # make sure to add "/" not "\"  
 up.config(image=upbt\_img)  
 up1 = Button(top, text="money transfer ", font=30, width="18", height=3, bd=0, command=(lambda: bank()))  
 abt\_img = PhotoImage(file="money transfer.png")  
 up1.config(image=abt\_img)  
 ggt = Button(top, text="google", font=30, width="18", height=3, bd=0, command=(lambda: google()))  
 rst\_img = PhotoImage(file="google.png")  
 ggt.config(image=rst\_img)  
 hst = Button(top, text="filters", font=30, width="18", height=3, bd=0, command=(lambda: filters()))  
 tit\_img = PhotoImage(file="glass.png")  
 hst.config(image=tit\_img)  
  
 def disable\_event():  
 global root  
 root.quit()  
  
 def send():  
  
 varContent = inputentry.get("1.0", END)  
 varContent = varContent.strip()  
  
 if varContent and (not varContent.isspace()):  
 message = varContent.encode("ascii")  
 c.send(message)  
 message = "\n" + varContent + "\n\n"  
 outputtext.tag\_config('user\_message', justify='right', wrap='word')  
 outputtext.insert(tk.END, message, 'user\_message')  
 outputtext.see(tk.END)  
 inputentry.delete('1.0', END)  
  
 def recv():  
 while True:  
 reply = c.recv(4141)  
 reply = reply.decode('ascii')  
  
 chek = 'START\_TRANSFER\_FILE\_NAME#3@41$\*='  
 if chek in reply:  
 # print(reply)  
 file\_name = reply.split("=", 1)[1]  
 scc = socket.socket()  
 port = 7676  
 host = addr[0]  
 scc.connect((host, port))  
 received\_path = os.path.expanduser('~\\Downloads\\')  
 with open(received\_path + file\_name, 'wb') as f:  
 while True:  
  
 data = scc.recv(1024)  
  
 f.write(data)  
  
 if not data:  
 break  
  
 fmessage = "\n" + file\_name + "\n\n"  
 outputtext.tag\_config('r', background="lightsteelblue", foreground="royalblue")  
 outputtext.insert(tk.END, fmessage, 'r')  
 f.close()  
 scc.close()  
 popup2 = Toplevel()  
 popup2.title('File Received')  
 popup2.geometry('250x100')  
 popup2.attributes('-topmost', 1)  
 popup2.attributes('-topmost', 0)  
 popup2.resizable(width=False, height=False)  
 prompt = file\_name + " received"  
 label1 = Label(popup2, text=prompt, width=len(prompt), font=("Arial", 10))  
 label1.place(x=45, y=32, height=39, width=200)  
 imgn = ImageTk.PhotoImage(Image.open("tick.png"))  
 paneln = Label(popup2, image=imgn)  
 paneln.place(x=14, y=30, height=39, width=30)  
  
 def close\_after\_2s():  
 popup2.destroy()  
  
 popup2.after(4000, close\_after\_2s)  
  
 else:  
  
 reply = "\n" + reply + "\n\n"  
 outputtext.tag\_config('reply', background="lightsteelblue", foreground="black", wrap='word')  
 outputtext.insert(tk.END, reply, 'reply')  
  
 outputtext.see(tk.END)  
 import winsound  
 winsound.PlaySound("notif.wav", winsound.SND\_ALIAS)  
  
 if 'normal' != top.state():  
 popup3 = Toplevel()  
 popup3.title('Notification')  
 popup3.geometry('250x100')  
 popup3.resizable(width=False, height=False)  
 popup3.attributes('-topmost', 1)  
 popup3.attributes('-topmost', 0)  
 prompt = "Message received from \n" + client\_username + ""  
 label3 = Label(popup3, text=prompt, width=len(prompt), font=("Arial", 10))  
 label3.place(x=45, y=32, height=39, width=200)  
 imgn = ImageTk.PhotoImage(Image.open("tick.png"))  
 paneln3 = Label(popup3, image=imgn)  
 paneln3.place(x=14, y=30, height=39, width=30)  
  
 def close\_after\_2s():  
 popup3.destroy()  
  
 popup3.after(6000, close\_after\_2s)  
  
 notification.notify(  
 title='New message received',  
 message="Message received from\n " + client\_username + "",  
 app\_name='H-PINGER',  
 timeout=20,  
 app\_icon='3.ico')  
  
 def FileTransfer():  
 File\_path = filedialog.askopenfilename(title='Choose file to send')  
 if File\_path:  
  
 File\_name = os.path.basename(File\_path)  
 trnsfr\_st = 'START\_TRANSFER\_FILE\_NAME#3@41$\*=' + File\_name  
 message = trnsfr\_st.encode("ascii")  
 c.send(message)  
  
 port = 6767  
 ss = socket.socket()  
 host = "0.0.0.0"  
 ss.bind((host, port))  
 ss.listen(5)  
 conns, addr = ss.accept()  
 while True:  
  
 b = os.path.getsize(File\_path)  
 f = open(File\_path, 'rb')  
 l = f.read(b)  
  
 while (l):  
 conns.send(l)  
  
 l = f.read(b)  
 f.close()  
 break  
  
 ftmessage = "\n" + File\_name + "\n\n"  
 outputtext.tag\_config('u', justify='right', foreground="royalblue")  
 outputtext.insert(tk.END, ftmessage, 'u')  
 conns.close()  
  
 ss.close()  
  
 popup4 = Toplevel()  
 popup4.title('Success')  
 popup4.geometry('250x100')  
 popup4.attributes('-topmost', 1)  
 popup4.attributes('-topmost', 0)  
 prompt = File\_name + " Sent"  
 label4 = Label(popup4, text=prompt, width=len(prompt), font=("Arial", 10))  
 label4.place(x=45, y=32, height=39, width=200)  
 img4 = ImageTk.PhotoImage(Image.open("ms1.png"))  
 panel4 = Label(popup4, image=img4)  
 panel4.place(x=14, y=30, height=39, width=39)  
  
 def close\_after\_2s():  
 popup4.destroy()  
  
 popup4.after(4000, close\_after\_2s)  
  
 def google():  
 def get\_me():  
 entry\_value = entry.get()  
 answer.delete(1.0, END)  
 try:  
 answer\_value = wikipedia.summary(entry\_value)  
 answer.insert(INSERT, answer\_value)  
 except:  
 answer.insert(INSERT, "please check you input or internet connection")  
  
 root = Tk()  
  
 topframe = Frame(root)  
 entry = Entry(topframe)  
 entry.pack()  
 button = Button(topframe, text="search", command=get\_me)  
 button.pack()  
 topframe.pack(side=TOP)  
  
 bottomframe = Frame(root)  
 scroll = Scrollbar(bottomframe)  
 scroll.pack(side=RIGHT, fill=Y)  
 answer = Text(bottomframe, width=70, height=20, yscrollcommand=scroll.set, wrap=WORD, bg="aqua")  
 scroll.config(command=answer.yview)  
 answer.pack()  
 bottomframe.pack()  
  
 root.mainloop()  
  
 def filters():  
 face\_cascade = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')  
  
 specs\_ori = cv2.imread('glass.png', -1)  
 cigar\_ori = cv2.imread('cigar.png', -1)  
 mus\_ori = cv2.imread('mustache.png', -1)  
  
 # Camera Init  
 cap = cv2.VideoCapture(0)  
 cap.set(cv2.CAP\_PROP\_FPS, 30)  
  
 def transparentOverlay(src, overlay, pos=(0, 0), scale=1):  
 overlay = cv2.resize(overlay, (0, 0), fx=scale, fy=scale)  
 h, w, \_ = overlay.shape # Size of foreground  
 rows, cols, \_ = src.shape # Size of background Image  
 y, x = pos[0], pos[1] # Position of foreground/overlay image  
  
 for i in range(h):  
 for j in range(w):  
 if x + i >= rows or y + j >= cols:  
 continue  
 alpha = float(overlay[i][j][3] / 255.0) # read the alpha channel  
 src[x + i][y + j] = alpha \* overlay[i][j][:3] + (1 - alpha) \* src[x + i][y + j]  
 return src  
  
 while 1:  
 ret, img = cap.read()  
 gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
 faces = face\_cascade.detectMultiScale(img, 1.2, 5, 0, (120, 120), (350, 350))  
  
 for (x, y, w, h) in faces:  
 if h > 0 and w > 0:  
 glass\_symin = int(y + 1.5 \* h / 5)  
 glass\_symax = int(y + 2.5 \* h / 5)  
 sh\_glass = glass\_symax - glass\_symin  
  
 cigar\_symin = int(y + 4 \* h / 6)  
 cigar\_symax = int(y + 5.5 \* h / 6)  
 sh\_cigar = cigar\_symax - cigar\_symin  
  
 mus\_symin = int(y + 3.5 \* h / 6)  
 mus\_symax = int(y + 5 \* h / 6)  
 sh\_mus = mus\_symax - mus\_symin  
  
 face\_glass\_roi\_color = img[glass\_symin:glass\_symax, x:x + w]  
 face\_cigar\_roi\_color = img[cigar\_symin:cigar\_symax, x:x + w]  
 face\_mus\_roi\_color = img[mus\_symin:mus\_symax, x:x + w]  
  
 specs = cv2.resize(specs\_ori, (w, sh\_glass), interpolation=cv2.INTER\_CUBIC)  
 cigar = cv2.resize(cigar\_ori, (w, sh\_cigar), interpolation=cv2.INTER\_CUBIC)  
 mustache = cv2.resize(mus\_ori, (w, sh\_mus), interpolation=cv2.INTER\_CUBIC)  
  
 transparentOverlay(face\_glass\_roi\_color, specs)  
 # transparentOverlay(face\_cigar\_roi\_color, cigar, (int(w/2), int(sh\_cigar/2)))  
 # transparentOverlay(face\_mus\_roi\_color, mustache)  
  
 cv2.imshow('Thug Life', img)  
 key = cv2.waitKey(1) & 0xFF  
 if key == ord("q"):  
 break  
  
 k = cv2.waitKey(30) & 0xff  
 if k == 27:  
 cv2.imwrite('img.jpg', img)  
 break  
  
 cap.release()  
 cv2.destroyAllWindows()  
  
 # Account Number : 10 ------------ Password : trial  
  
 def bank():  
 # Account Number : 10 ------------ Password : trial  
  
 ARIAL = ("arial", 10, "bold")  
  
 class Bank:  
 def \_\_init\_\_(self, root):  
 self.conn = sqlite3.connect("atm\_databse.db", timeout=100)  
 self.login = False  
 self.root = root  
 self.header = Label(self.root, text="B~K BANK", bg="dark blue", fg="white", font=("arial", 20, "bold"))  
 self.header.pack(fill=X)  
 self.frame = Frame(self.root, bg="white", width=600, height=400)  
 # Login Page Form Components  
  
 self.userlabel = Label(self.frame, text="Account Number", bg="pink", font=ARIAL)  
 self.uentry = Entry(self.frame, width=30, bg="light green")  
 self.plabel = Label(self.frame, text="Password", bg="pink", font=ARIAL)  
 self.pentry = Entry(self.frame, show="\*", bg="light green")  
 self.button = Button(self.frame, text="LOGIN", bg="yellow", font=ARIAL, command=self.verify)  
 self.q = Button(self.frame, text="Quit", bg="pink", font=ARIAL, command=self.root.destroy)  
 self.frame.config(bg="cyan")  
 self.userlabel.place(x=145, y=100, width=120, height=20)  
 self.uentry.place(x=153, y=130, width=200, height=20)  
 self.plabel.place(x=145, y=160, width=120, height=20)  
 self.pentry.place(x=153, y=190, width=200, height=20)  
 self.button.place(x=155, y=230, width=120, height=20)  
 self.q.place(x=480, y=360, width=120, height=20)  
 self.frame.pack()  
  
 def database\_fetch(self):  
 # Fetching Account data from database  
 self.acc\_list = []  
 self.temp = self.conn.execute("select name,pass,acc\_no,acc\_type,bal from atm where acc\_no = ? ",  
 (self.ac,))  
 for i in self.temp:  
 self.acc\_list.append("Name = {}".format(i[0]))  
 self.acc\_list.append("Account no = {}".format(i[2]))  
 self.acc\_list.append("Account type = {}".format(i[3]))  
 self.ac = i[2]  
 self.acc\_list.append("Balance = {}".format(i[4]))  
  
 def verify(self):  
 # verifying of authorised user  
 ac = False  
 self.temp = self.conn.execute("select name,pass,acc\_no,acc\_type,bal from atm where acc\_no = ? ",  
 (int(self.uentry.get()),))  
 for i in self.temp:  
 self.ac = i[2]  
 if i[2] == self.uentry.get():  
 ac = True  
 elif i[1] == self.pentry.get():  
 ac = True  
 m = "{} Login SucessFull".format(i[0])  
 self.database\_fetch()  
 messagebox.\_show("Login Info", m)  
 self.frame.destroy()  
 self.MainMenu()  
 else:  
 ac = True  
 m = " Login UnSucessFull ! Wrong Password"  
 messagebox.\_show("Login Info!", m)  
 if not ac:  
 m = " Wrong Acoount Number !"  
 messagebox.\_show("Login Info!", m)  
  
 def MainMenu(self):  
 # Main App Appears after logined !  
 self.frame = Frame(self.root, bg="white", width=800, height=400)  
 root.geometry("800x400")  
 self.frame.config(bg="cyan")  
 self.detail = Button(self.frame, text="Account Details", bg="blue", font=ARIAL,  
 command=self.account\_detail)  
 self.enquiry = Button(self.frame, text="Balance Enquiry", bg="pink", font=ARIAL, command=self.Balance)  
 self.deposit = Button(self.frame, text="Deposit Money", bg="pink", font=ARIAL,  
 command=self.deposit\_money)  
 self.withdrawl = Button(self.frame, text="Withdrawl Money", bg="pink", font=ARIAL,  
 command=self.withdrawl\_money)  
 self.q = Button(self.frame, text="Quit", bg="yellow", font=ARIAL, command=self.root.destroy)  
 self.detail.place(x=50, y=50, width=200, height=50)  
 self.enquiry.place(x=50, y=200, width=200, height=50)  
 self.deposit.place(x=500, y=50, width=200, height=50)  
 self.withdrawl.place(x=500, y=200, width=200, height=50)  
 self.q.place(x=340, y=340, width=120, height=20)  
 self.frame.pack()  
  
 def account\_detail(self):  
 self.database\_fetch()  
 text = self.acc\_list[0] + "\n" + self.acc\_list[1] + "\n" + self.acc\_list[2]  
 self.label = Label(self.frame, text=text, font=ARIAL)  
 self.label.place(x=200, y=100, width=300, height=100)  
  
 def Balance(self):  
 self.database\_fetch()  
 self.label = Label(self.frame, text=self.acc\_list[3], font=ARIAL)  
 self.label.place(x=200, y=100, width=300, height=100)  
  
 def deposit\_money(self):  
 self.money\_box = Entry(self.frame, width=20)  
 self.submitButton = Button(self.frame, text="Submit", bg="lightyellow", font=ARIAL)  
 self.money\_box.place(x=200, y=100, width=200, height=20)  
 self.submitButton.place(x=445, y=100, width=55, height=20)  
 self.submitButton.bind("<Button-1>", self.deposit\_trans)  
  
 def deposit\_trans(self, flag):  
 self.label = Label(self.frame, text="Transaction Completed !", font=ARIAL)  
 self.label.place(x=200, y=100, width=300, height=100)  
 self.conn.execute("update atm set bal = bal + ? where acc\_no = ?", (self.money\_box.get(), self.ac))  
 self.conn.commit()  
  
 def withdrawl\_money(self):  
 self.money\_box = Entry(self.frame, width=20)  
 self.submitButton = Button(self.frame, text="Submit", bg="lightyellow", font=ARIAL)  
 self.money\_box.place(x=200, y=100, width=200, height=20)  
 self.submitButton.place(x=445, y=100, width=55, height=20)  
 self.submitButton.bind("<Button-1>", self.withdrawl\_trans)  
  
 def withdrawl\_trans(self, flag):  
 self.label = Label(self.frame, text="Money Withdrawl !", font=ARIAL)  
 self.label.place(x=200, y=100, width=300, height=100)  
 self.conn.execute("update atm set bal = bal - ? where acc\_no = ?", (self.money\_box.get(), self.ac))  
 self.conn.commit()  
  
 root = Toplevel()  
 root.title("Sign In")  
 root.geometry("600x420")  
 root.config(bg="cyan")  
 obj = Bank(root)  
  
 outputtext = Text(top, bd=0, bg="white", height="8", width="50", font=("Arial", 12))  
 outputtext.configure(highlightbackground='lightgrey', highlightthickness=1)  
 scrollbar = Scrollbar(top, command=outputtext.yview)  
 outputtext['yscrollcommand'] = scrollbar.set  
 scrollbar.place(x=376, y=40, height=352)  
 outputtext.place(x=6, y=40, height=352, width=370)  
 inputentry.place(x=6, y=401, height=90, width=265)  
 panel.place(x=6, y=1, height=39, width=39)  
 SendButton.place(x=285, y=418, height=60, width=62)  
 up.place(x=360, y=3, height=32, width=32)  
 up1.place(x=320, y=3, height=32, width=32)  
 ggt.place(x=280, y=3, height=32, width=32)  
 hst.place(x=240, y=3, height=32, width=32)  
 user\_lb.place(x=45, y=4, height=18)  
 ip\_lb.place(x=50, y=26, height=10)  
  
 while True:  
 t2 = Thread(target=recv())  
 t2.setDaemon(True)  
 t2.start()  
  
  
def onclosing(arg, s):  
 # print("trying to close the window")  
 if messagebox.askokcancel("Quit", "Do you want to close H-PINGER?"):  
 s.close()  
 root.destroy()  
  
  
def accept\_connection():  
 s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
 s.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)  
 ip = '0.0.0.0'  
 port = 4141  
 s.bind((ip, port))  
 s.listen(5)  
  
 time.sleep(1)  
 act\_lb1.destroy()  
 sp.destroy()  
 panelu.destroy()  
 panelunew.place(x=6, y=6, height=35)  
 label1.place(x=20, y=70, height=39, width=180)  
 panel.place(x=230, y=67, height=39, width=39)  
 # panel.load("checked2.gif")  
 labelw.place(x=20, y=120, height=39, width=180)  
 panelw.place(x=230, y=120, height=39, width=65)  
 panelw.load('124.gif')  
  
 connected\_clients = []  
  
 while True:  
 root.protocol("WM\_DELETE\_WINDOW", lambda arg=(root): onclosing(arg, s))  
 c, addr = s.accept()  
  
 connected\_clients.append(addr[0])  
 print(connected\_clients)  
 user\_name = getpass.getuser()  
 user\_name = user\_name.encode("ascii")  
 c.send(user\_name)  
  
 t = Thread(target=handle\_client, args=(c, addr))  
 t.setDaemon(True)  
 t.start()  
  
  
def con():  
 t = Thread(target=accept\_connection)  
 t.setDaemon(True)  
 t.start()  
  
  
sp = Button(text='Activate', anchor='center', font=30, width=18, height=3, bd=0, command=con)  
act\_img = PhotoImage(file='e1.png')  
sp.config(image=act\_img)  
sp.place(x=130, y=100, height=68, width=68)  
act\_lb1 = Label(text='Press the button to activate')  
act\_lb1.place(x=100, y=40, height=55)  
imgu = ImageTk.PhotoImage(Image.open("user1.png"))  
panelu = Label(root, image=imgu)  
panelu.place(x=6, y=6, height=35)  
slabel1 = Label(text=getpass.getuser())  
slabel1.place(x=48, y=8, height=14)  
slabel2 = Label(text=socket.gethostbyname(socket.gethostname()))  
slabel2.place(x=48, y=30, height=10)  
imgunew = ImageTk.PhotoImage(Image.open("user4.png"))  
panelunew = Label(root, image=imgunew)  
img = ImageTk.PhotoImage(Image.open("checked1.png"))  
panel = Label(root, image=img)  
prompt1 = "Server Activated"  
label1 = Label(root, text=prompt1, width=len(prompt1), font=("Arial", 12))  
prompt2 = "Waiting new connections"  
labelw = Label(root, text=prompt2, width=len(prompt2), font=("Arial", 12))  
panelw = ImageLabel(root)  
  
root.lift()  
root.attributes('-topmost', True)  
root.after\_idle(root.attributes, '-topmost', False)  
root.mainloop()

**CLIENT-SIDE**

import getpass  
from tkinter import \*  
import tkinter as tk  
from PIL import ImageTk, Image  
from tkinter.scrolledtext import ScrolledText  
from threading import Thread  
from tkinter import messagebox  
import os.path  
import socket  
from tkinter import filedialog  
import os  
from plyer import notification  
import winsound  
import ctypes  
import os  
import sqlite3  
import cv2  
import wikipedia  
  
kernel32 = ctypes.WinDLL('kernel32')  
user32 = ctypes.WinDLL('user32')  
SW\_HIDE = 0  
hWnd = kernel32.GetConsoleWindow()  
user32.ShowWindow(hWnd, SW\_HIDE)  
master = Tk()  
master.title("H-PING")  
master.geometry('350x200')  
  
  
def center(toplevel):  
 toplevel.update\_idletasks()  
 screen\_width = toplevel.winfo\_screenwidth()  
 screen\_height = toplevel.winfo\_screenheight()  
 size = tuple(int(\_) for \_ in toplevel.geometry().split('+')[0].split('x'))  
 x = screen\_width / 2.5 - size[0] / 3  
 y = screen\_height / 2.5 - size[1] / 3  
 toplevel.geometry("+%d+%d" % (x, y))  
  
  
def handle\_server(ip, s):  
 master.withdraw()  
 server\_username = s.recv(4141)  
 server\_username = server\_username.decode('ascii')  
  
 top = Toplevel()  
 top.title('H-PING')  
 top.geometry("400x500")  
 top.attributes('-topmost', 1)  
 top.attributes('-topmost', 0)  
 center(top)  
 top.resizable(width=FALSE, height=FALSE)  
 inputentry = Text(top, bd=0, bg="white", width="29", height="5", font=("Arial", 12))  
 inputentry.configure(highlightbackground='lightgrey', highlightthickness=1)  
 inputentry.bind('<Return>', (lambda event: send()))  
 img = ImageTk.PhotoImage(Image.open("user4.png"))  
 panel = Label(top, image=img)  
 prompt = server\_username  
 user\_lb = Label(top, text=prompt, width=len(prompt), font=(12,))  
 prompt = ip  
 ip\_lb = Label(top, text=prompt, width=len(prompt), font=("Arial", 8))  
 SendButton = Button(top, font=30, text="Send", width="12", height=5, bd=0, command=(lambda: send()))  
 send\_img = PhotoImage(file="e2.png") # make sure to add "/" not "\"  
 SendButton.config(image=send\_img)  
 up = Button(top, text="Attach", font=30, width="18", height=3, bd=0, command=(lambda: FileTransfer()))  
 upbt\_img = PhotoImage(file="at3.png") # make sure to add "/" not "\"  
 up.config(image=upbt\_img)  
 up1 = Button(top, text="money transfer ", font=30, width="18", height=3, bd=0, command=(lambda: stk()))  
 abt\_img = PhotoImage(file="money transfer.png")  
 up1.config(image=abt\_img)  
 ggt = Button(top, text="google ", font=30, width="18", height=3, bd=0, command=(lambda: google()))  
 sept\_img = PhotoImage(file="google.png")  
 ggt.config(image=sept\_img)  
 hst = Button(top, text="filter ", font=30, width="18", height=3, bd=0, command=(lambda: filters()))  
 tit\_img = PhotoImage(file="glass.png")  
 hst.config(image=tit\_img)  
  
 outputtext = Text(top, bd=0, bg="white", height="8", width="50", font=("Arial", 12))  
 outputtext.configure(highlightbackground='lightgrey', highlightthickness=1)  
 scrollbar = Scrollbar(top, command=outputtext.yview)  
 outputtext['yscrollcommand'] = scrollbar.set  
  
 def send():  
 varContent = inputentry.get("1.0", END)  
 varContent = varContent.strip()  
 if varContent and (not varContent.isspace()):  
 message = varContent.encode("ascii")  
 s.send(message)  
 message = "\n" + varContent + "\n\n"  
 outputtext.tag\_config('user\_message', justify='right', wrap='word')  
 outputtext.insert(tk.END, message, 'user\_message')  
 outputtext.see(tk.END)  
 inputentry.delete('1.0', END)  
  
 def recv():  
 while True:  
 reply = s.recv(4141)  
 reply = reply.decode('ascii')  
  
 chek = 'START\_TRANSFER\_FILE\_NAME#3@41$\*='  
 if chek in reply:  
 # print(reply)  
 file\_name = reply.split("=", 1)[1]  
 scc = socket.socket()  
 port = 6767  
 scc.connect((ip, port))  
 received\_path = os.path.expanduser('~\\Downloads\\')  
 with open(received\_path + file\_name, 'wb') as f:  
 while True:  
  
 data = scc.recv(1024)  
  
 f.write(data)  
  
 if not data:  
 break  
  
 fmessage = "\n" + file\_name + "\n\n"  
 outputtext.tag\_config('r', background="lightsteelblue", foreground="royalblue")  
 outputtext.insert(tk.END, fmessage, 'r')  
 f.close()  
 scc.close()  
 popup2 = Toplevel()  
 popup2.title('File Received')  
 popup2.geometry('250x100')  
 popup2.attributes('-topmost', 1)  
 popup2.attributes('-topmost', 0)  
 prompt = file\_nam  
 e + " received"  
 label1 = Label(popup2, text=prompt, width=len(prompt), font=("Arial", 10))  
 label1.place(x=45, y=32, height=39, width=200)  
 imgn = ImageTk.PhotoImage(Image.open("tick.png"))  
 paneln = Label(popup2, image=imgn)  
 paneln.place(x=14, y=30, height=39, width=30)  
  
 def close\_after\_2s():  
 popup2.destroy()  
  
 popup2.after(4000, close\_after\_2s)  
  
 else:  
 reply = "\n" + reply + "\n\n"  
 outputtext.tag\_config('reply', background="lightsteelblue", foreground="black", wrap='word')  
 outputtext.insert(tk.END, reply, 'reply')  
 outputtext.see(tk.END)  
  
 winsound.PlaySound("SystemExit", winsound.SND\_ALIAS)  
 winsound.PlaySound("notif.wav", winsound.SND\_ALIAS)  
  
 if 'normal' != top.state():  
 popup3 = Toplevel()  
 popup3.title('Notification')  
 popup3.geometry('250x100')  
 popup3.attributes('-topmost', 1)  
 popup3.attributes('-topmost', 0)  
 prompt = "Message received from\n " + server\_username + ""  
 label3 = Label(popup3, text=prompt, width=len(prompt), font=("Arial", 10))  
 label3.place(x=45, y=32, height=39, width=200)  
 imgn = ImageTk.PhotoImage(Image.open("tick.png"))  
 paneln3 = Label(popup3, image=imgn)  
 paneln3.place(x=14, y=30, height=39, width=30)  
  
 def close\_after\_2s():  
 popup3.destroy()  
  
 popup3.after(4000, close\_after\_2s)  
  
 notification.notify(  
 title='New message received',  
 message="Message received from \n" + server\_username + "",  
 app\_name='H-PING',  
 timeout=20,  
 app\_icon='3.ico')  
  
 def FileTransfer():  
 File\_path = filedialog.askopenfilename(title='Choose file to send')  
  
 if File\_path:  
 File\_name = os.path.basename(File\_path)  
 trnsfr\_st = 'START\_TRANSFER\_FILE\_NAME#3@41$\*=' + File\_name  
 message = trnsfr\_st.encode("ascii")  
 s.send(message)  
 port = 7676  
 ss = socket.socket()  
 host = "0.0.0.0"  
 ss.bind((host, port))  
 ss.listen(5)  
 conns, addr = ss.accept()  
 while True:  
 b = os.path.getsize(File\_path)  
 f = open(File\_path, 'rb')  
 l = f.read(b)  
  
 while (l):  
 conns.send(l)  
  
 l = f.read(b)  
 f.close()  
 break  
 conns.close()  
 ss.close()  
 # messagebox.showinfo("Success", File\_name + " Sent")  
 popup4 = Toplevel()  
 popup4.title('Success')  
 popup4.geometry('250x100')  
 popup4.attributes('-topmost', 1)  
 popup4.attributes('-topmost', 0)  
 prompt = File\_name + " Sent"  
 label4 = Label(popup4, text=prompt, width=len(prompt), font=("Arial", 10))  
 label4.place(x=45, y=32, height=39, width=200)  
 img4 = ImageTk.PhotoImage(Image.open("ms1.png"))  
 panel4 = Label(popup4, image=img4)  
 panel4.place(x=14, y=30, height=39, width=30)  
  
 def close\_after\_2s():  
 popup4.destroy()  
  
 popup4.after(4000, close\_after\_2s)  
 ftmessage = "\n" + File\_name + "\n\n"  
 outputtext.tag\_config('u', justify='right', foreground="royalblue")  
 outputtext.insert(tk.END, ftmessage, 'u')  
  
 def google():  
 def get\_me():  
 entry\_value = entry.get()  
 answer.delete(1.0, END)  
 try:  
 answer\_value = wikipedia.summary(entry\_value)  
 answer.insert(INSERT, answer\_value)  
 except:  
 answer.insert(INSERT, "please check you input or internet connection")  
  
 root = Tk()  
  
 topframe = Frame(root)  
 entry = Entry(topframe)  
 entry.pack()  
 button = Button(topframe, text="search", command=get\_me)  
 button.pack()  
 topframe.pack(side=TOP)  
  
 bottomframe = Frame(root)  
 scroll = Scrollbar(bottomframe)  
 scroll.pack(side=RIGHT, fill=Y)  
 answer = Text(bottomframe, width=70, height=20, yscrollcommand=scroll.set, wrap=WORD, bg="aqua")  
 scroll.config(command=answer.yview)  
 answer.pack()  
 bottomframe.pack()  
  
 root.mainloop()  
  
 def filters():  
 face\_cascade = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')  
  
 specs\_ori = cv2.imread('glass.png', -1)  
 cigar\_ori = cv2.imread('cigar.png', -1)  
 mus\_ori = cv2.imread('mustache.png', -1)  
  
 # Camera Init  
 cap = cv2.VideoCapture(0)  
 cap.set(cv2.CAP\_PROP\_FPS, 30)  
  
 def transparentOverlay(src, overlay, pos=(0, 0), scale=1):  
 overlay = cv2.resize(overlay, (0, 0), fx=scale, fy=scale)  
 h, w, \_ = overlay.shape # Size of foreground  
 rows, cols, \_ = src.shape # Size of background Image  
 y, x = pos[0], pos[1] # Position of foreground/overlay image  
  
 for i in range(h):  
 for j in range(w):  
 if x + i >= rows or y + j >= cols:  
 continue  
 alpha = float(overlay[i][j][3] / 255.0) # read the alpha channel  
 src[x + i][y + j] = alpha \* overlay[i][j][:3] + (1 - alpha) \* src[x + i][y + j]  
 return src  
  
 while 1:  
 ret, img = cap.read()  
 gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
 faces = face\_cascade.detectMultiScale(img, 1.2, 5, 0, (120, 120), (350, 350))  
  
 for (x, y, w, h) in faces:  
 if h > 0 and w > 0:  
 glass\_symin = int(y + 1.5 \* h / 5)  
 glass\_symax = int(y + 2.5 \* h / 5)  
 sh\_glass = glass\_symax - glass\_symin  
  
 cigar\_symin = int(y + 4 \* h / 6)  
 cigar\_symax = int(y + 5.5 \* h / 6)  
 sh\_cigar = cigar\_symax - cigar\_symin  
  
 mus\_symin = int(y + 3.5 \* h / 6)  
 mus\_symax = int(y + 5 \* h / 6)  
 sh\_mus = mus\_symax - mus\_symin  
  
 face\_glass\_roi\_color = img[glass\_symin:glass\_symax, x:x + w]  
 face\_cigar\_roi\_color = img[cigar\_symin:cigar\_symax, x:x + w]  
 face\_mus\_roi\_color = img[mus\_symin:mus\_symax, x:x + w]  
  
 specs = cv2.resize(specs\_ori, (w, sh\_glass), interpolation=cv2.INTER\_CUBIC)  
 cigar = cv2.resize(cigar\_ori, (w, sh\_cigar), interpolation=cv2.INTER\_CUBIC)  
 mustache = cv2.resize(mus\_ori, (w, sh\_mus), interpolation=cv2.INTER\_CUBIC)  
  
 transparentOverlay(face\_glass\_roi\_color, specs)  
 # transparentOverlay(face\_cigar\_roi\_color, cigar, (int(w/2), int(sh\_cigar/2)))  
 # transparentOverlay(face\_mus\_roi\_color, mustache)  
  
 cv2.imshow('Thug Life', img)  
 key = cv2.waitKey(1) & 0xFF  
 if key == ord("q"):  
 break  
  
 k = cv2.waitKey(30) & 0xff  
 if k == 27:  
 cv2.imwrite('img.jpg', img)  
 break  
  
 cap.release()  
 cv2.destroyAllWindows()  
  
 def stk():  
 # Account Number : 10 ------------ Password : trial  
  
 ARIAL = ("arial", 10, "bold")  
  
 class Bank:  
 def \_\_init\_\_(self, root):  
 self.conn = sqlite3.connect("atm\_databse.db", timeout=100)  
 self.login = False  
 self.root = root  
 self.header = Label(self.root, text="B~K BANK", bg="dark blue", fg="white", font=("arial", 20, "bold"))  
 self.header.pack(fill=X)  
 self.frame = Frame(self.root, bg="white", width=600, height=400)  
 # Login Page Form Components  
  
 self.userlabel = Label(self.frame, text="Account Number", bg="pink", font=ARIAL)  
 self.uentry = Entry(self.frame, width=30, bg="light green")  
 self.plabel = Label(self.frame, text="Password", bg="pink", font=ARIAL)  
 self.pentry = Entry(self.frame, show="\*", bg="light green")  
 self.button = Button(self.frame, text="LOGIN", bg="yellow", font=ARIAL, command=self.verify)  
 self.q = Button(self.frame, text="Quit", bg="pink", font=ARIAL, command=self.root.destroy)  
 self.frame.config(bg="cyan")  
 self.userlabel.place(x=145, y=100, width=120, height=20)  
 self.uentry.place(x=153, y=130, width=200, height=20)  
 self.plabel.place(x=145, y=160, width=120, height=20)  
 self.pentry.place(x=153, y=190, width=200, height=20)  
 self.button.place(x=155, y=230, width=120, height=20)  
 self.q.place(x=480, y=360, width=120, height=20)  
 self.frame.pack()  
  
 def database\_fetch(self):  
 # Fetching Account data from database  
 self.acc\_list = []  
 self.temp = self.conn.execute("select name,pass,acc\_no,acc\_type,bal from atm where acc\_no = ? ",  
 (self.ac,))  
 for i in self.temp:  
 self.acc\_list.append("Name = {}".format(i[0]))  
 self.acc\_list.append("Account no = {}".format(i[2]))  
 self.acc\_list.append("Account type = {}".format(i[3]))  
 self.ac = i[2]  
 self.acc\_list.append("Balance = {}".format(i[4]))  
  
 def verify(self):  
 # verifying of authorised user  
 ac = False  
 self.temp = self.conn.execute("select name,pass,acc\_no,acc\_type,bal from atm where acc\_no = ? ",  
 (int(self.uentry.get()),))  
 for i in self.temp:  
 self.ac = i[2]  
 if i[2] == self.uentry.get():  
 ac = True  
 elif i[1] == self.pentry.get():  
 ac = True  
 m = "{} Login SucessFull".format(i[0])  
 self.database\_fetch()  
 messagebox.\_show("Login Info", m)  
 self.frame.destroy()  
 self.MainMenu()  
 else:  
 ac = True  
 m = " Login UnSucessFull ! Wrong Password"  
 messagebox.\_show("Login Info!", m)  
 if not ac:  
 m = " Wrong Acoount Number !"  
 messagebox.\_show("Login Info!", m)  
  
 def MainMenu(self):  
 # Main App Appears after logined !  
 self.frame = Frame(self.root, bg="white", width=800, height=400)  
 root.geometry("800x400")  
 self.frame.config(bg="cyan")  
 self.detail = Button(self.frame, text="Account Details", bg="blue", font=ARIAL,  
 command=self.account\_detail)  
 self.enquiry = Button(self.frame, text="Balance Enquiry", bg="pink", font=ARIAL, command=self.Balance)  
 self.deposit = Button(self.frame, text="Deposit Money", bg="pink", font=ARIAL,  
 command=self.deposit\_money)  
 self.withdrawl = Button(self.frame, text="Withdrawl Money", bg="pink", font=ARIAL,  
 command=self.withdrawl\_money)  
 self.q = Button(self.frame, text="Quit", bg="yellow", font=ARIAL, command=self.root.destroy)  
 self.detail.place(x=50, y=50, width=200, height=50)  
 self.enquiry.place(x=50, y=200, width=200, height=50)  
 self.deposit.place(x=500, y=50, width=200, height=50)  
 self.withdrawl.place(x=500, y=200, width=200, height=50)  
 self.q.place(x=340, y=340, width=120, height=20)  
 self.frame.pack()  
  
 def account\_detail(self):  
 self.database\_fetch()  
 text = self.acc\_list[0] + "\n" + self.acc\_list[1] + "\n" + self.acc\_list[2]  
 self.label = Label(self.frame, text=text, font=ARIAL)  
 self.label.place(x=200, y=100, width=300, height=100)  
  
 def Balance(self):  
 self.database\_fetch()  
 self.label = Label(self.frame, text=self.acc\_list[3], font=ARIAL)  
 self.label.place(x=200, y=100, width=300, height=100)  
  
 def deposit\_money(self):  
 self.money\_box = Entry(self.frame, width=20)  
 self.submitButton = Button(self.frame, text="Submit", bg="lightyellow", font=ARIAL)  
 self.money\_box.place(x=200, y=100, width=200, height=20)  
 self.submitButton.place(x=445, y=100, width=55, height=20)  
 self.submitButton.bind("<Button-1>", self.deposit\_trans)  
  
 def deposit\_trans(self, flag):  
 self.label = Label(self.frame, text="Transaction Completed !", font=ARIAL)  
 self.label.place(x=200, y=100, width=300, height=100)  
 self.conn.execute("update atm set bal = bal + ? where acc\_no = ?", (self.money\_box.get(), self.ac))  
 self.conn.commit()  
  
 def withdrawl\_money(self):  
 self.money\_box = Entry(self.frame, width=20)  
 self.submitButton = Button(self.frame, text="Submit", bg="lightyellow", font=ARIAL)  
 self.money\_box.place(x=200, y=100, width=200, height=20)  
 self.submitButton.place(x=445, y=100, width=55, height=20)  
 self.submitButton.bind("<Button-1>", self.withdrawl\_trans)  
  
 def withdrawl\_trans(self, flag):  
 self.label = Label(self.frame, text="Money Withdrawl !", font=ARIAL)  
 self.label.place(x=200, y=100, width=300, height=100)  
 self.conn.execute("update atm set bal = bal - ? where acc\_no = ?", (self.money\_box.get(), self.ac))  
 self.conn.commit()  
  
 root = Toplevel()  
 root.title("Sign In")  
 root.geometry("600x420")  
 root.config(bg="cyan")  
 obj = Bank(root)  
  
 scrollbar.place(x=376, y=40, height=352)  
 outputtext.place(x=6, y=40, height=352, width=370)  
 inputentry.place(x=6, y=401, height=90, width=265)  
 panel.place(x=6, y=1, height=39, width=39)  
 SendButton.place(x=285, y=418, height=60, width=62)  
 up.place(x=360, y=3, height=32, width=32)  
 up1.place(x=320, y=3, height=32, width=32)  
 ggt.place(x=280, y=3, height=32, width=32)  
 hst.place(x=240, y=3, height=32, width=32)  
 user\_lb.place(x=45, y=4, height=18)  
 ip\_lb.place(x=50, y=26, height=10)  
  
 def onclosing(arg, s):  
  
 s.close()  
 top.destroy()  
 sys.exit()  
  
 top.protocol("WM\_DELETE\_WINDOW", lambda arg=(top): onclosing(arg, s))  
 while True:  
 t2 = Thread(target=recv())  
 t2.start()  
  
  
def chat\_window(userInput):  
 try:  
  
 e1.configure(text=userInput.get())  
 ip = userInput.get()  
 s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
 s.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)  
 port = 4141  
 s.connect((ip, port))  
 user\_name = getpass.getuser()  
 user\_name = user\_name.encode("ascii")  
 s.send(user\_name)  
 t = Thread(target=handle\_server, args=(ip, s))  
 t.start()  
  
 except Exception as e:  
 messagebox.showerror("Error", "Please activate H PINGER in target system first or check IP")  
 master.destroy()  
  
  
def connect(userInput):  
 t1 = Thread(target=chat\_window(userInput))  
 t1.start()  
  
  
userInput = StringVar()  
ll = Label(master, text="IP Address:", font=("Arial", 10))  
ll.config(height=2, width=15)  
ll.place(x=15, y=65, height=18)  
e1 = Entry(master, textvariable=userInput, width=20, font=("Arial", 12))  
e1.bind('<Return>', (lambda event: connect(userInput)))  
e1.place(x=120, y=60, height=25)  
imgu = ImageTk.PhotoImage(Image.open("user3.png"))  
panelu = Label(master, image=imgu)  
panelu.place(x=6, y=6, height=35)  
slabel1 = Label(text=getpass.getuser(), font=("Arial", 10))  
slabel1.place(x=48, y=18, height=14)  
ss = Button(text='connect', anchor='center', font=30, width=18, height=3, bd=0, command=(lambda: connect(userInput)))  
act\_img = PhotoImage(file='e1.png')  
ss.config(image=act\_img)  
ss.place(x=200, y=110, height=68, width=68)  
  
master.mainloop()

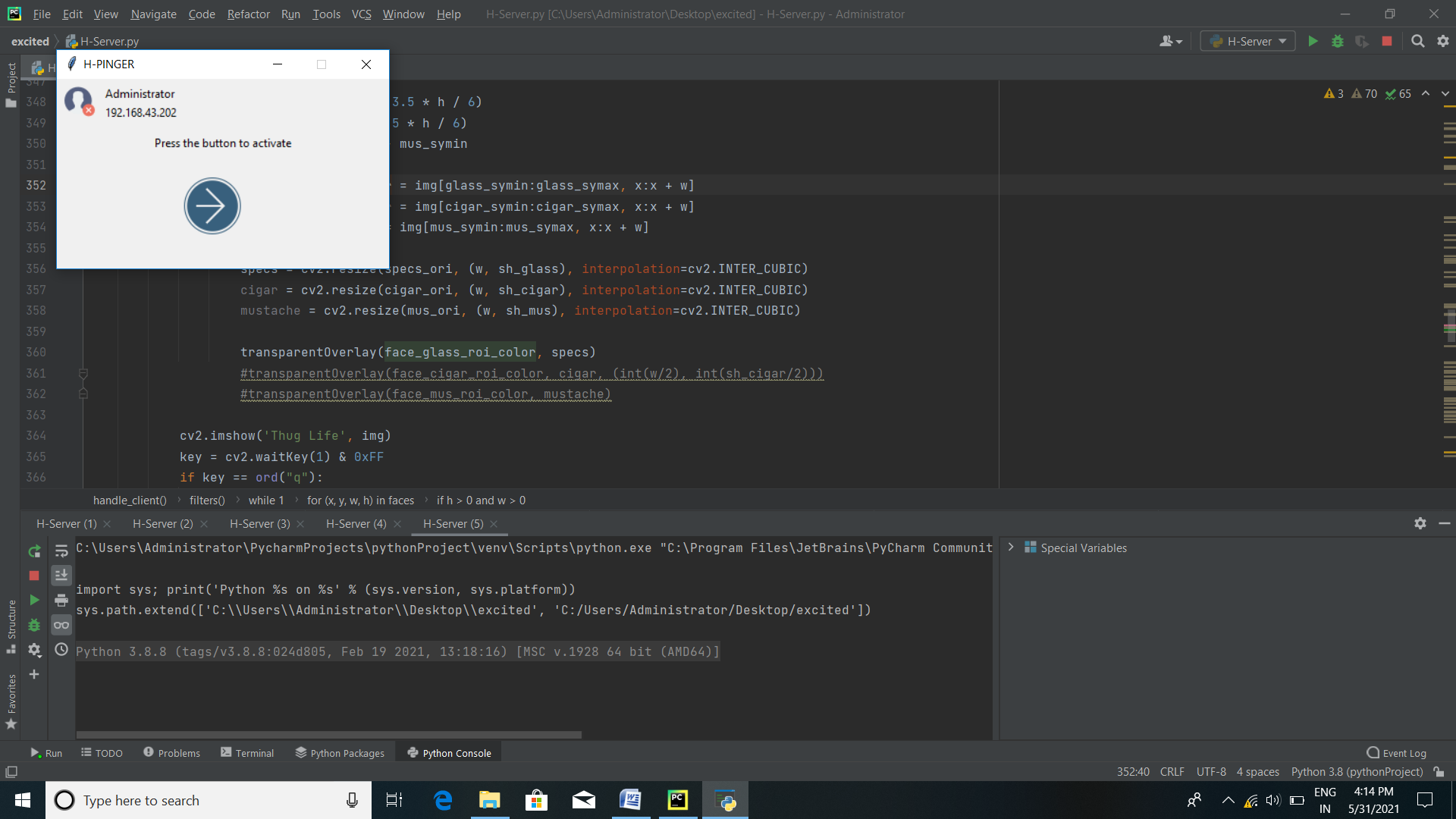
**CHAPTER-4**

**SCREENSHOTS OF OUTPUTS**:

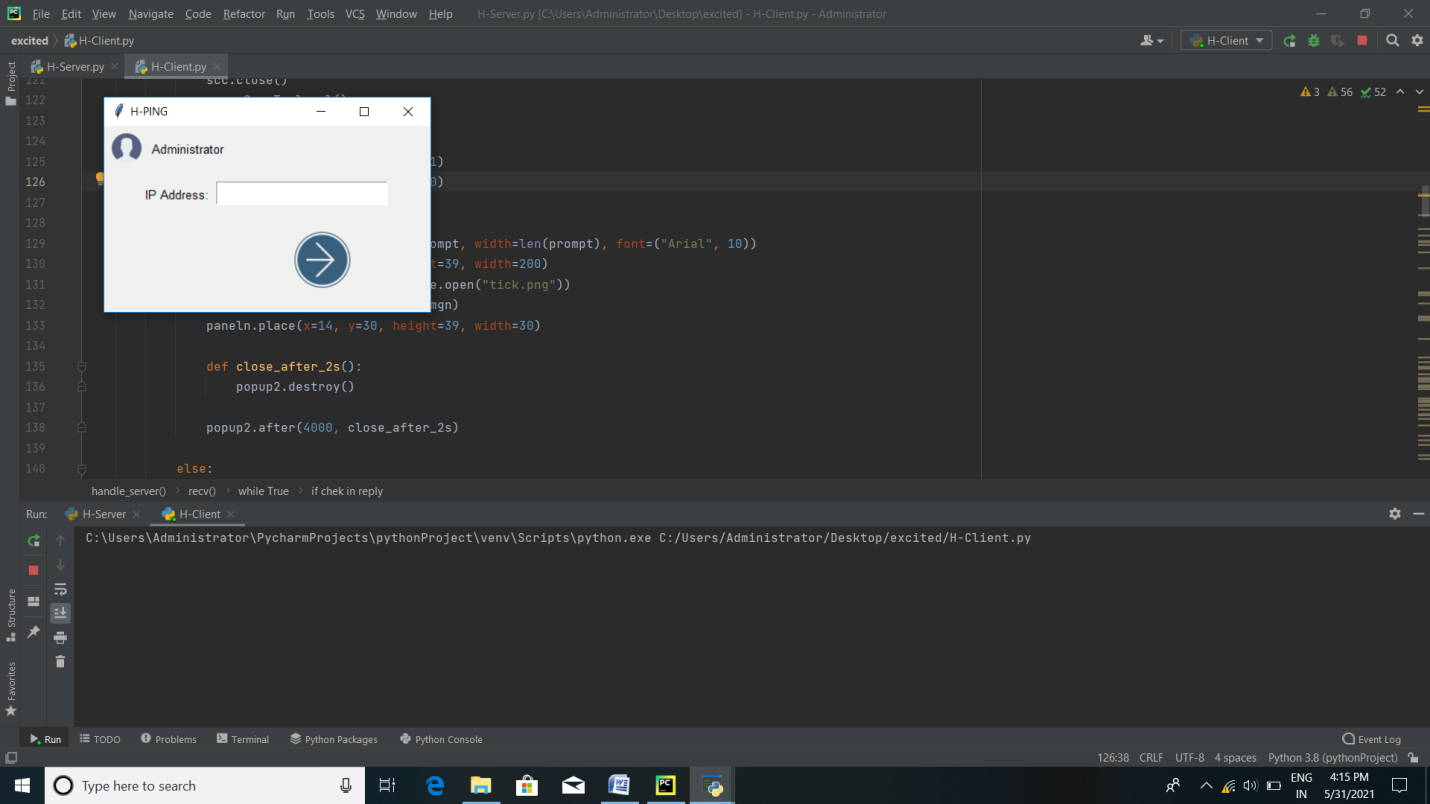
1.I place my project folder on desktop.



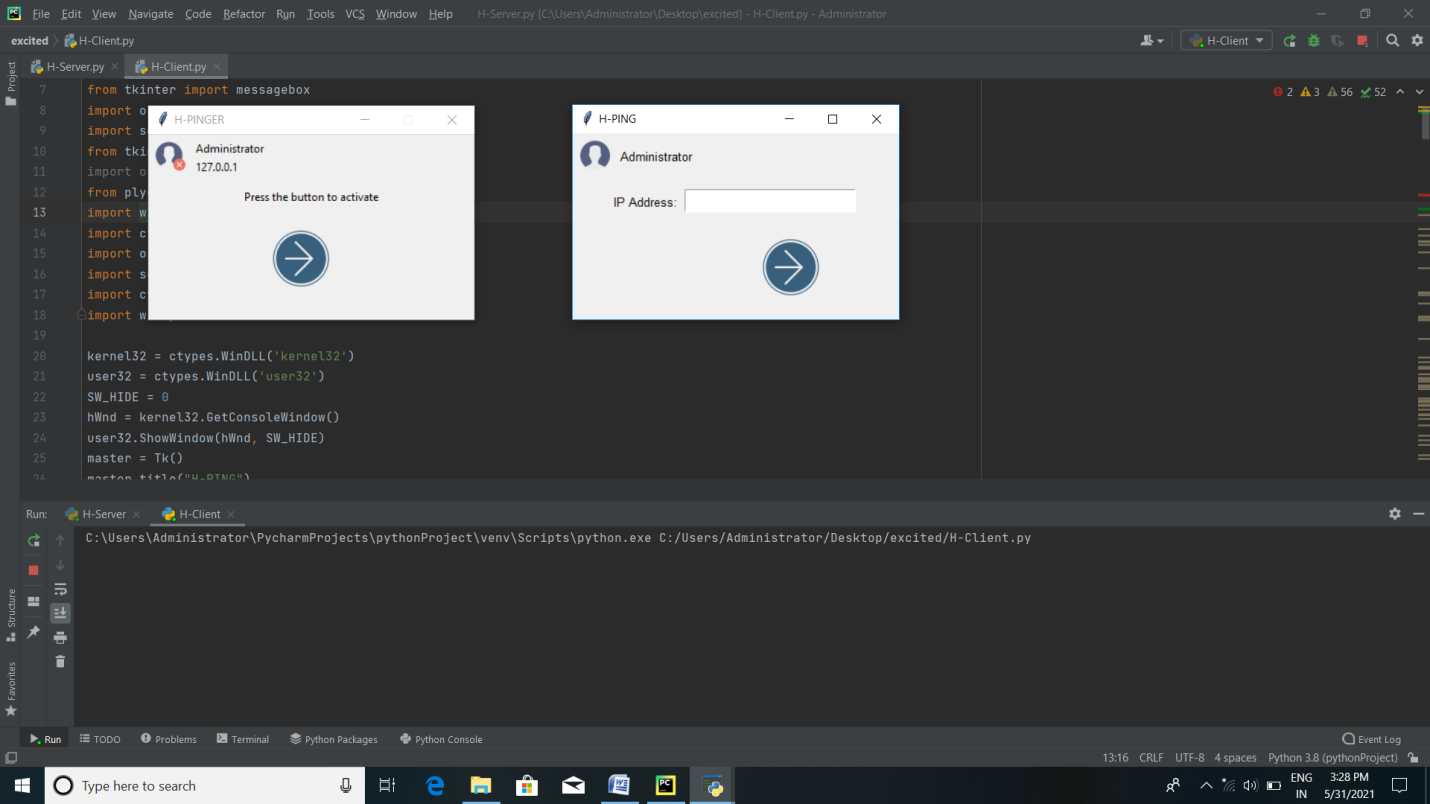
2.open the project place in folder using Pycharm and run server side program.



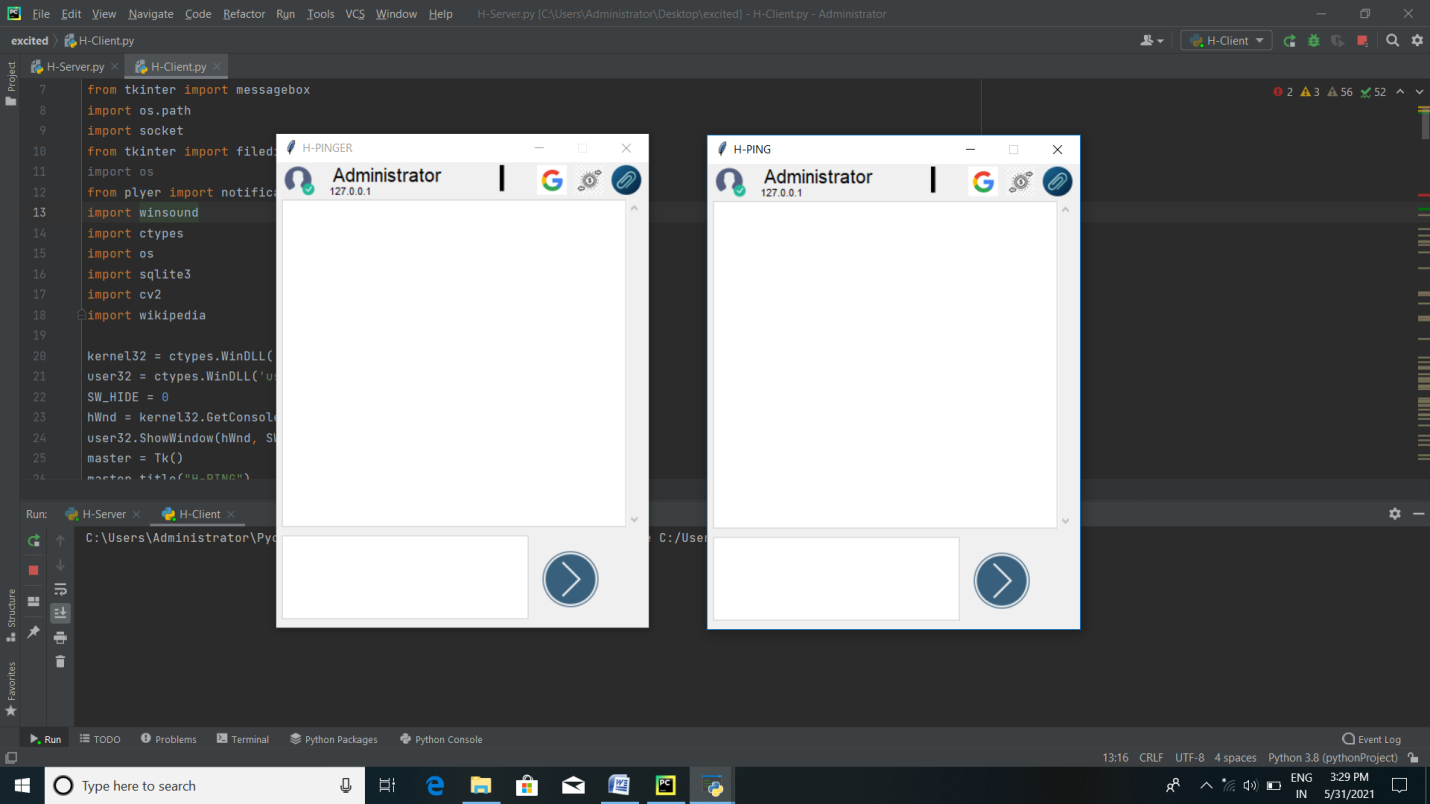
3 run clientt side project



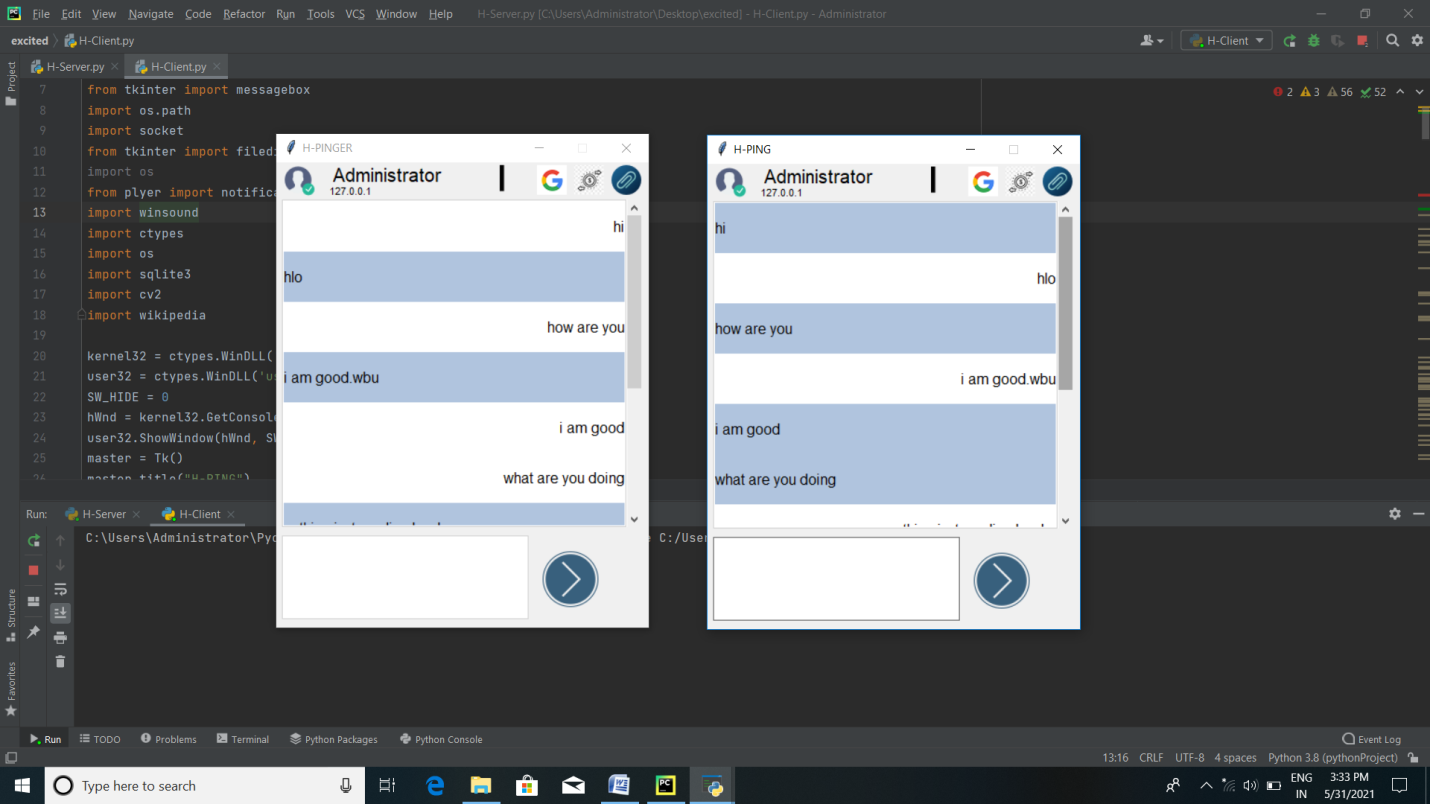
4. open server.



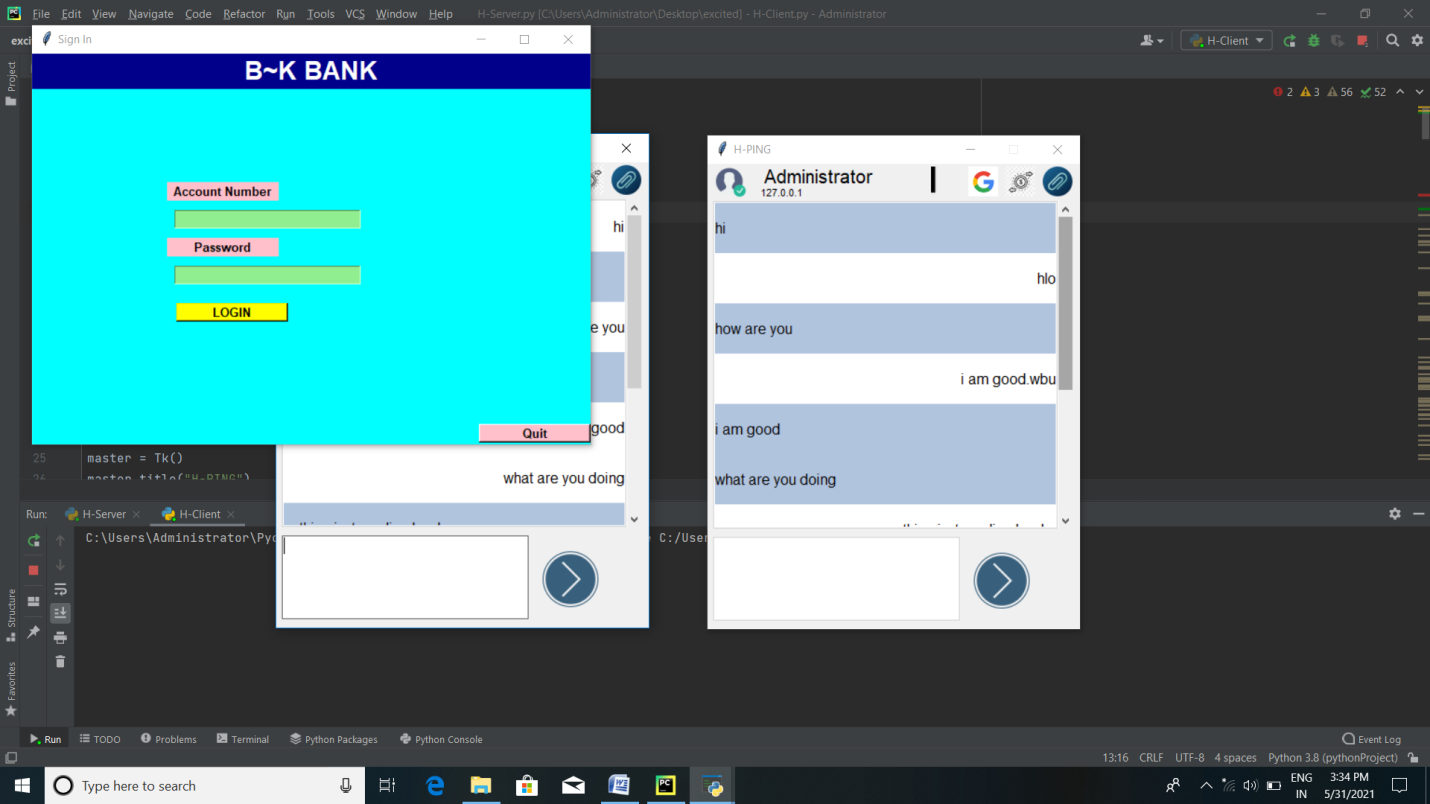
4.fill the ip address and connect.on connection I got.

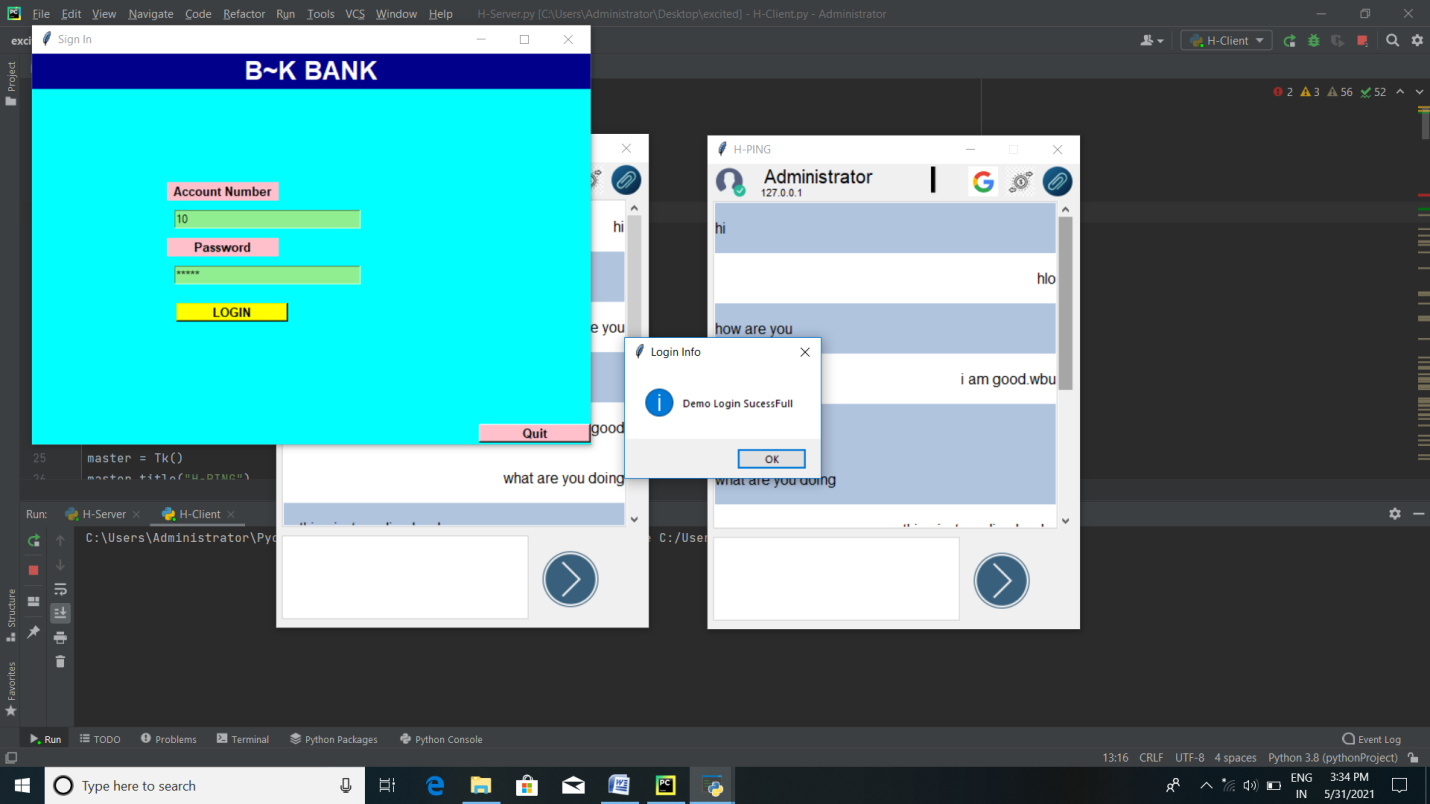


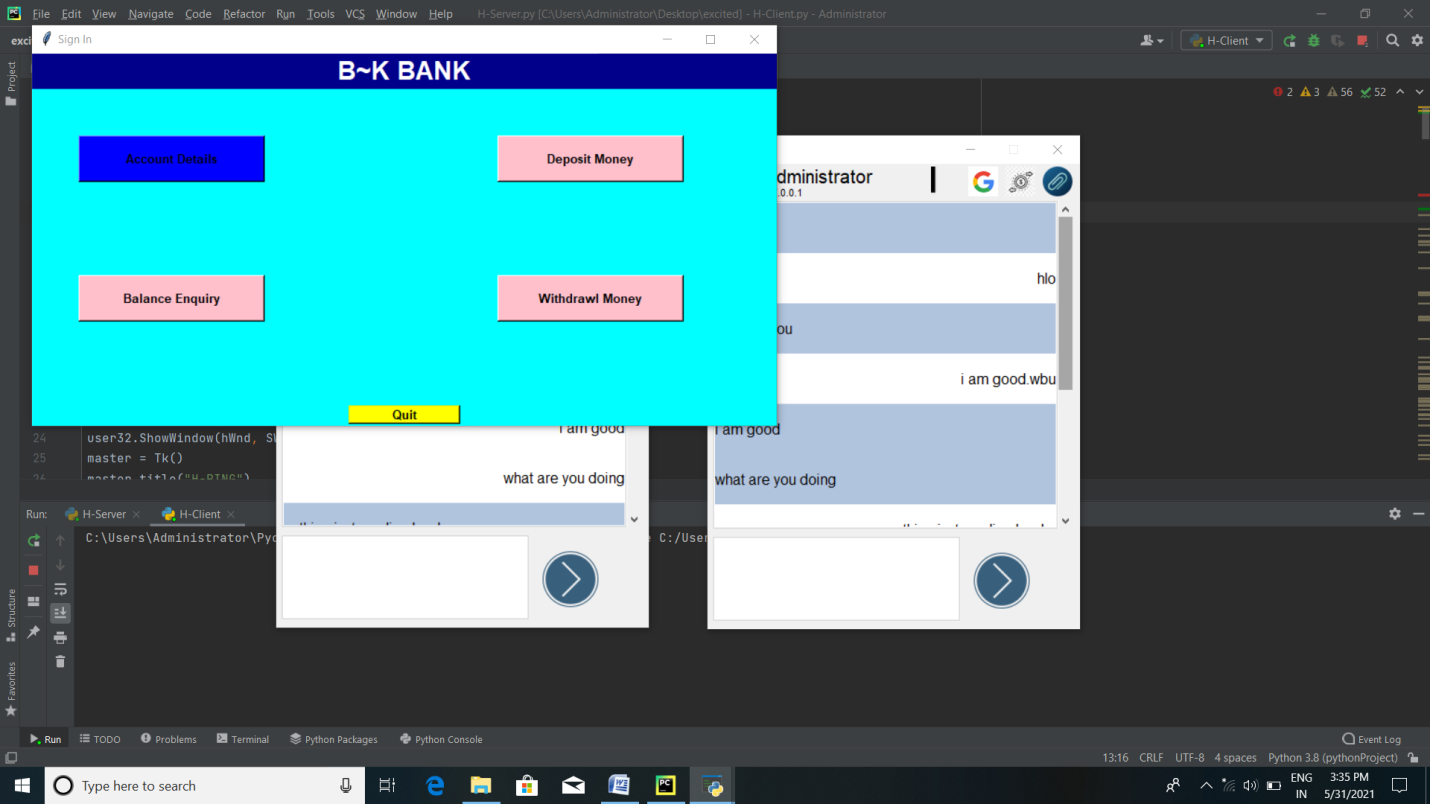
5. Start chatting.

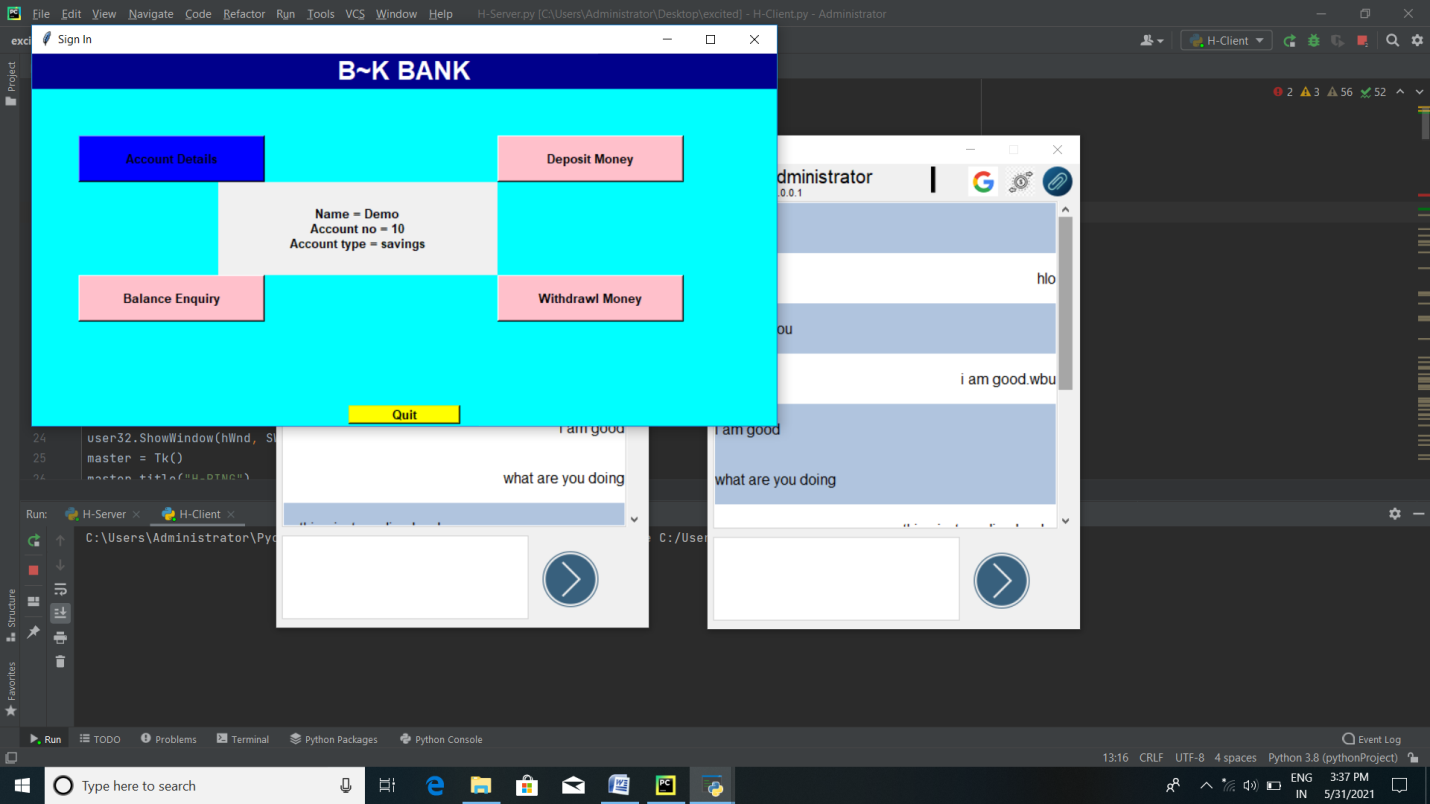


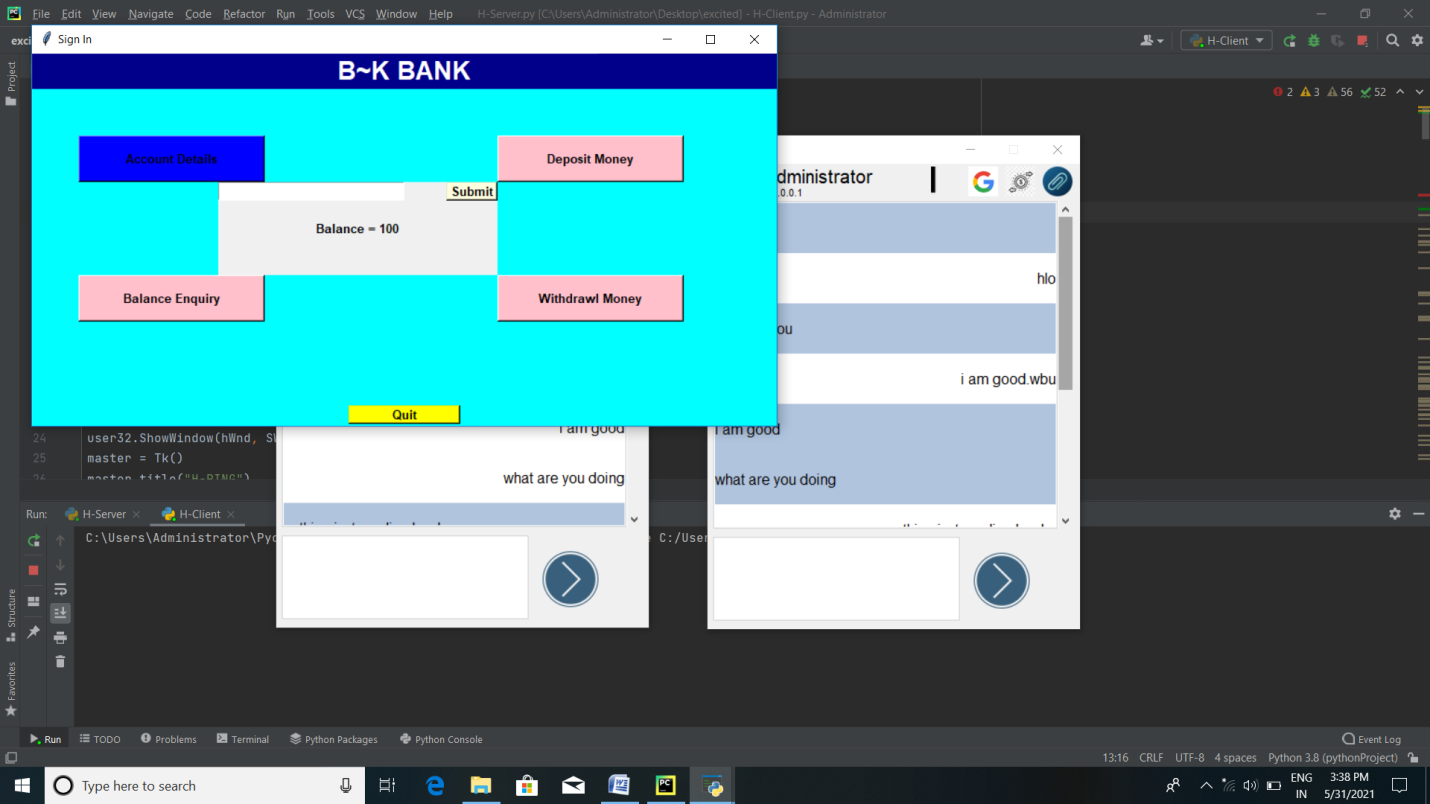
6. checking bank balance.

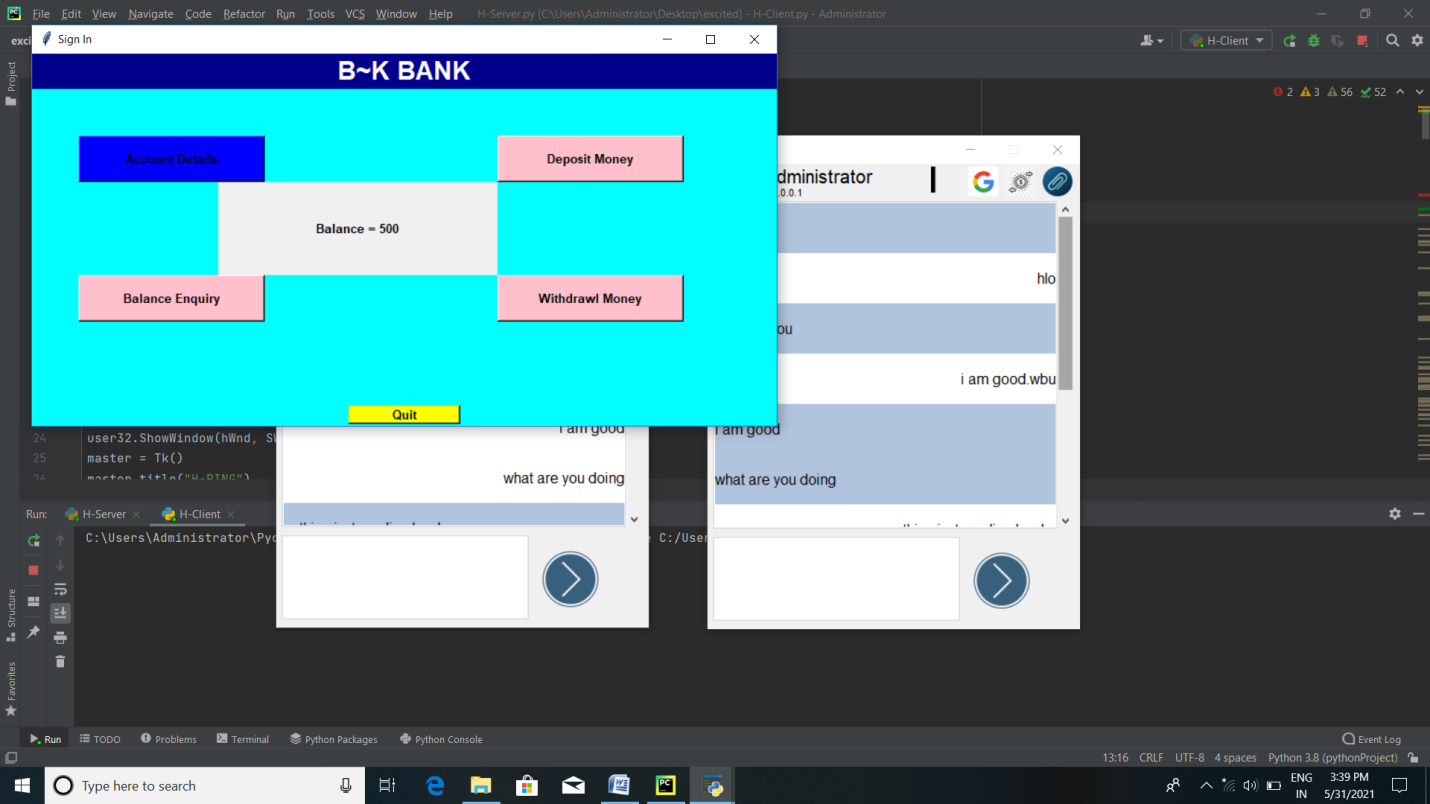


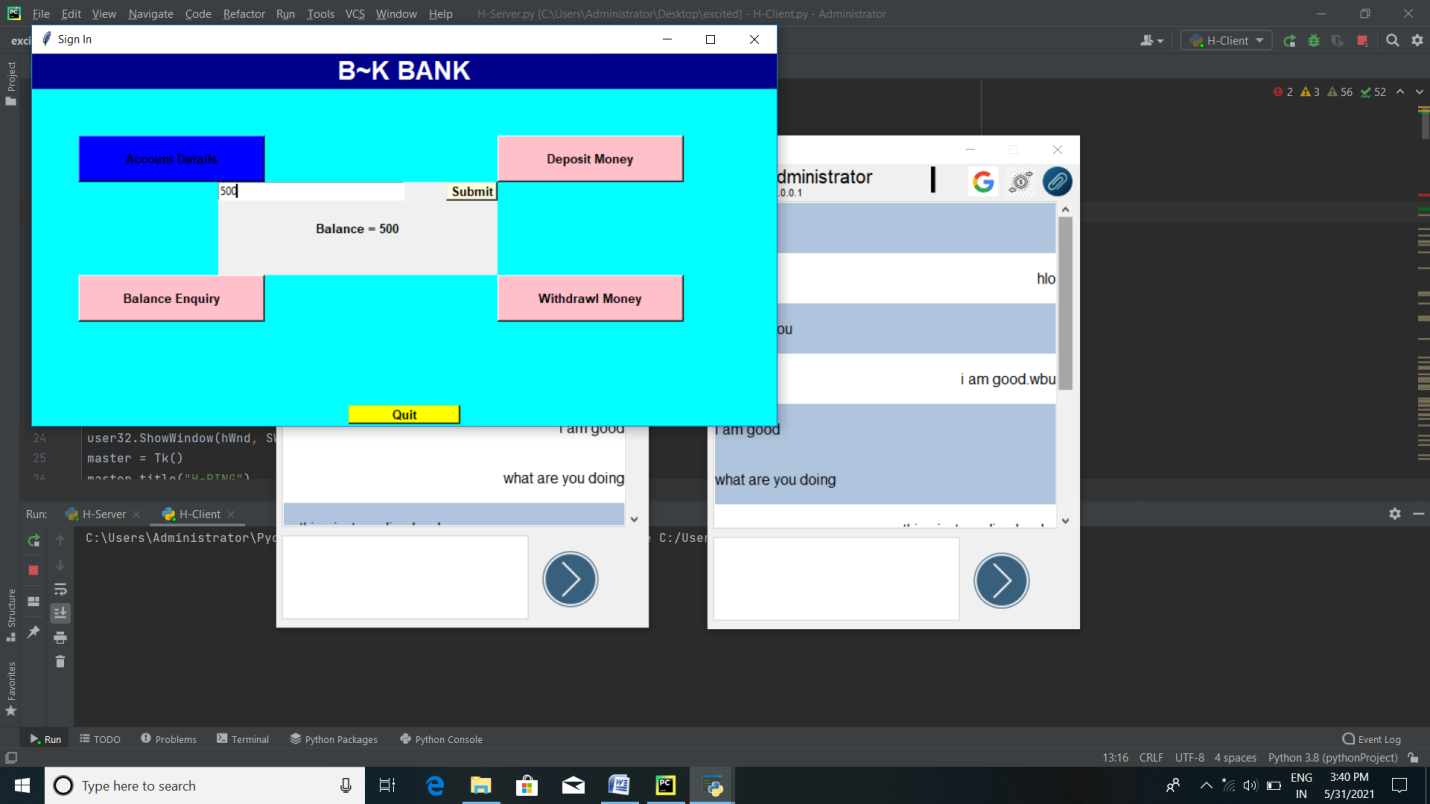


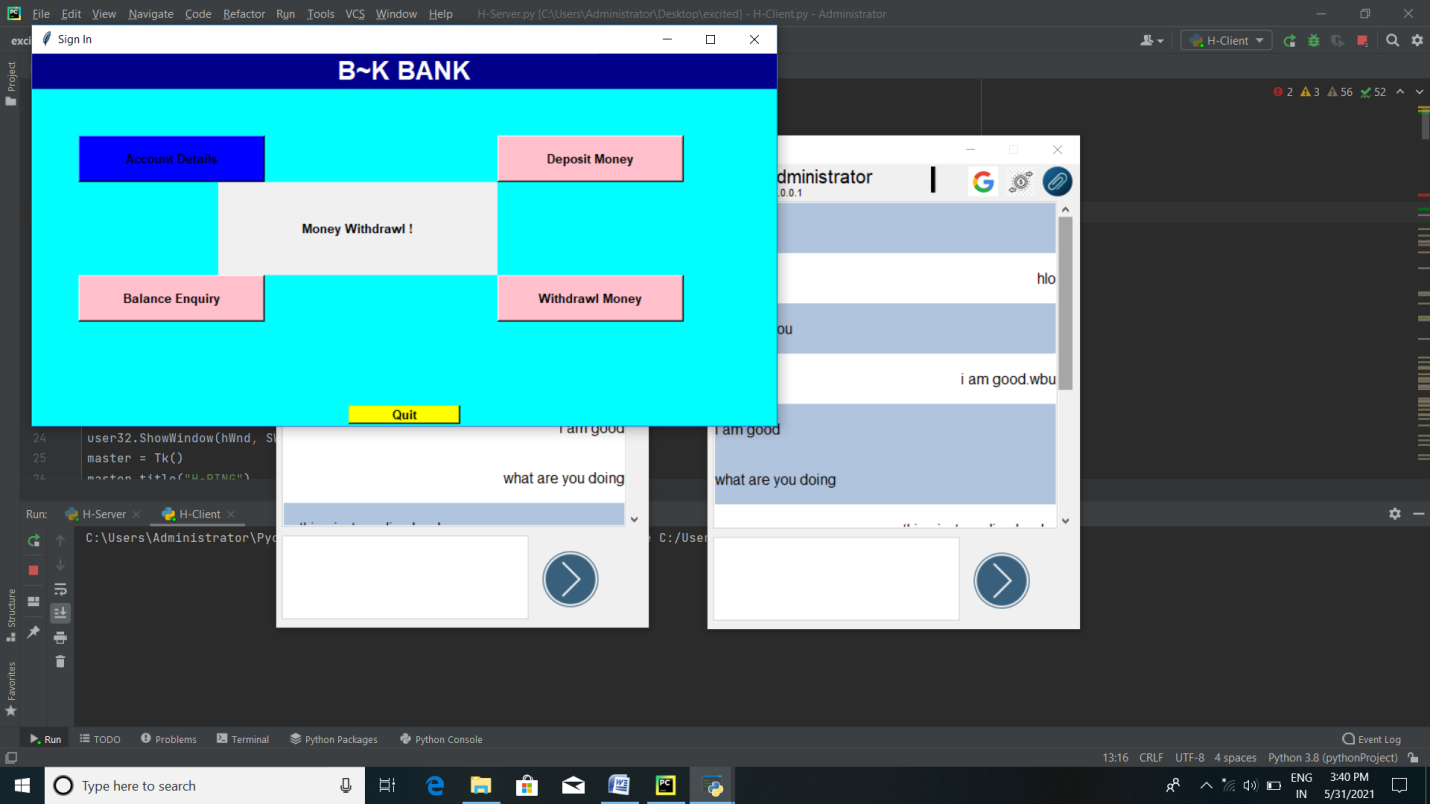




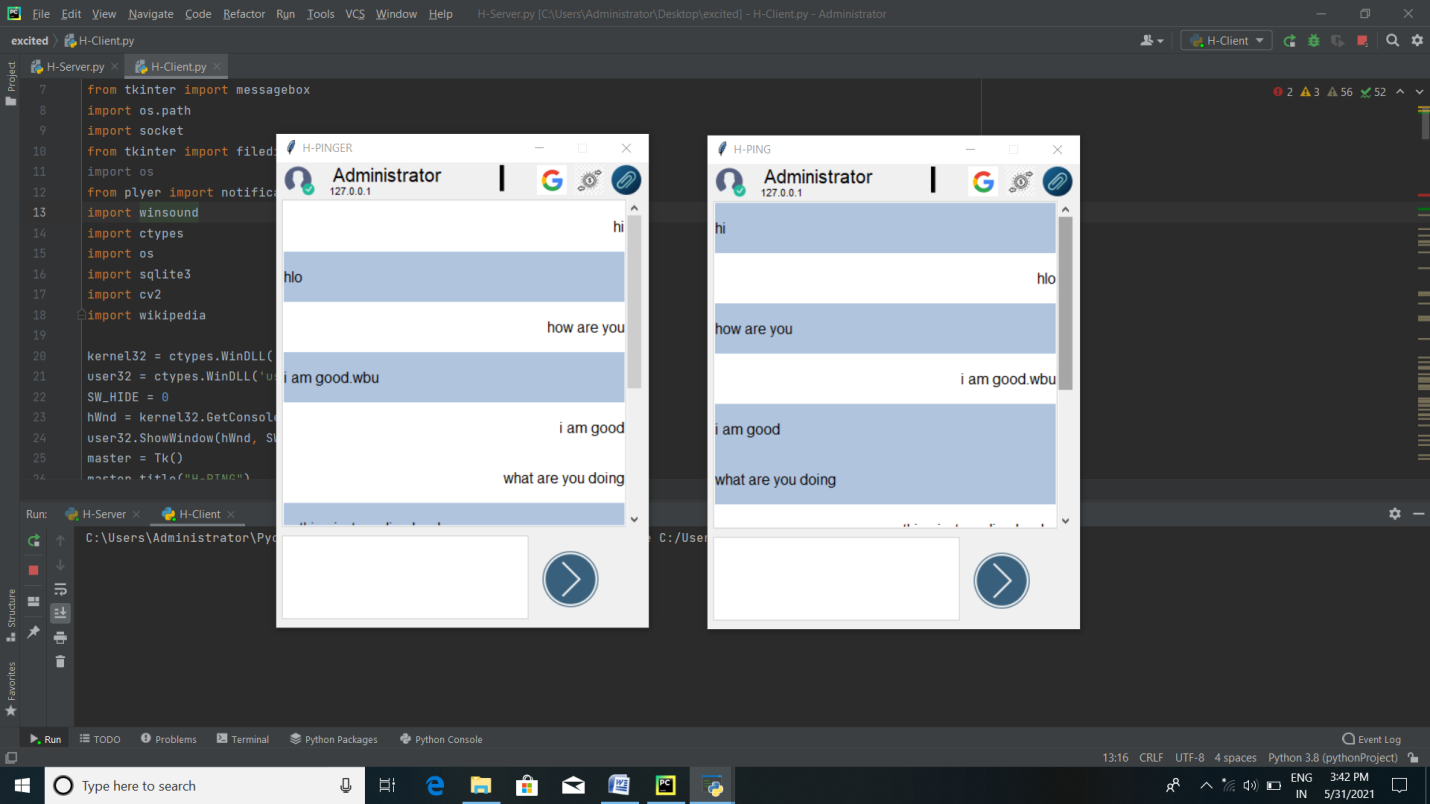


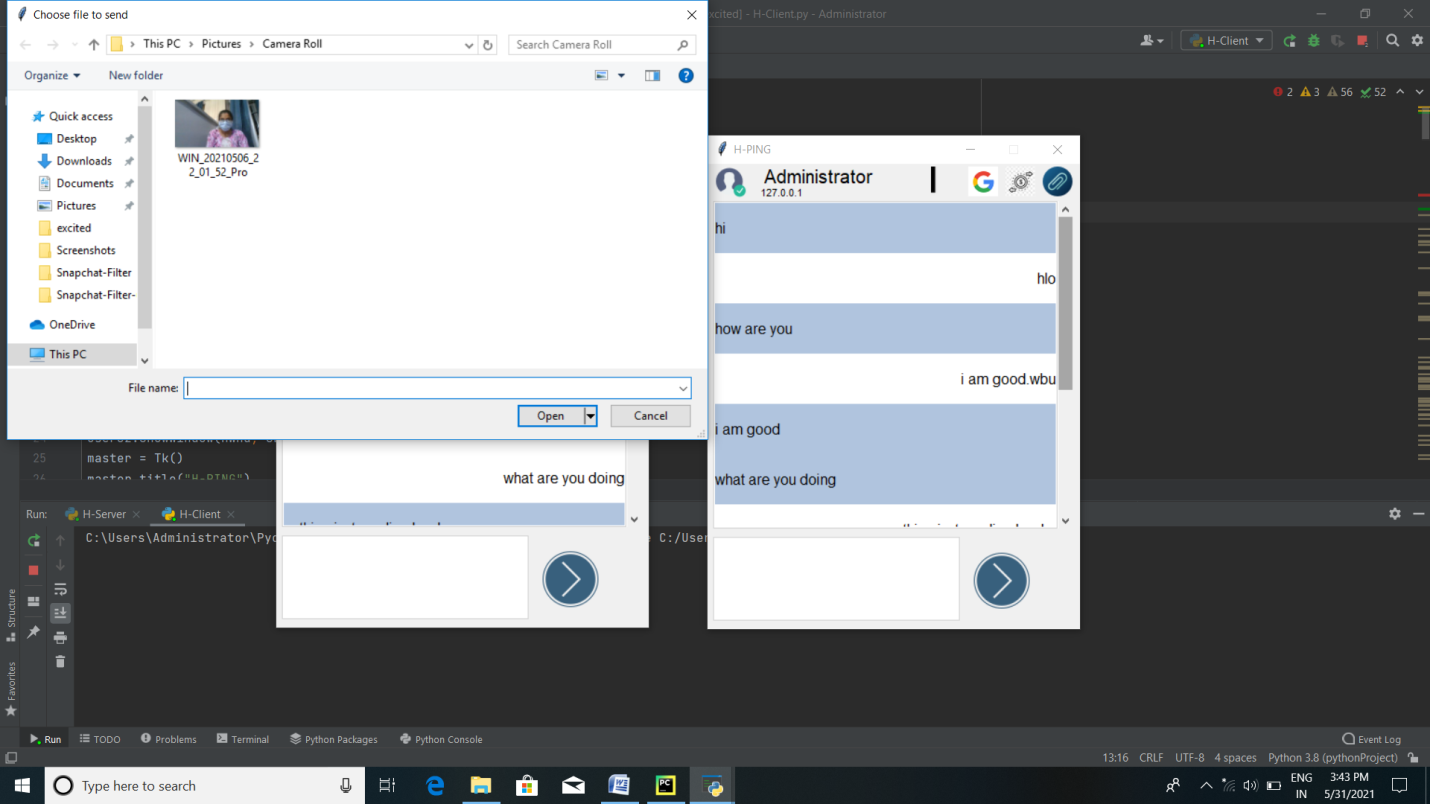


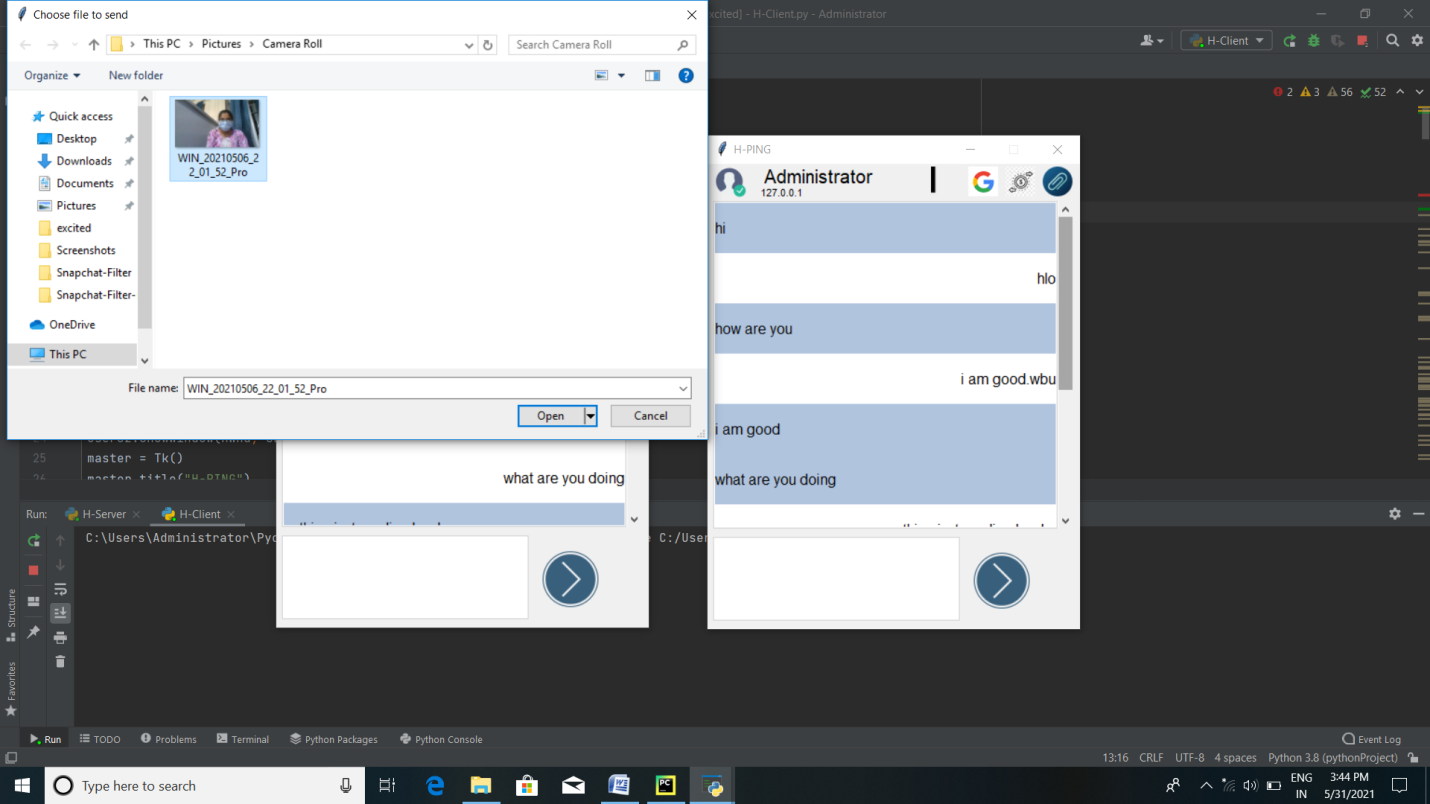


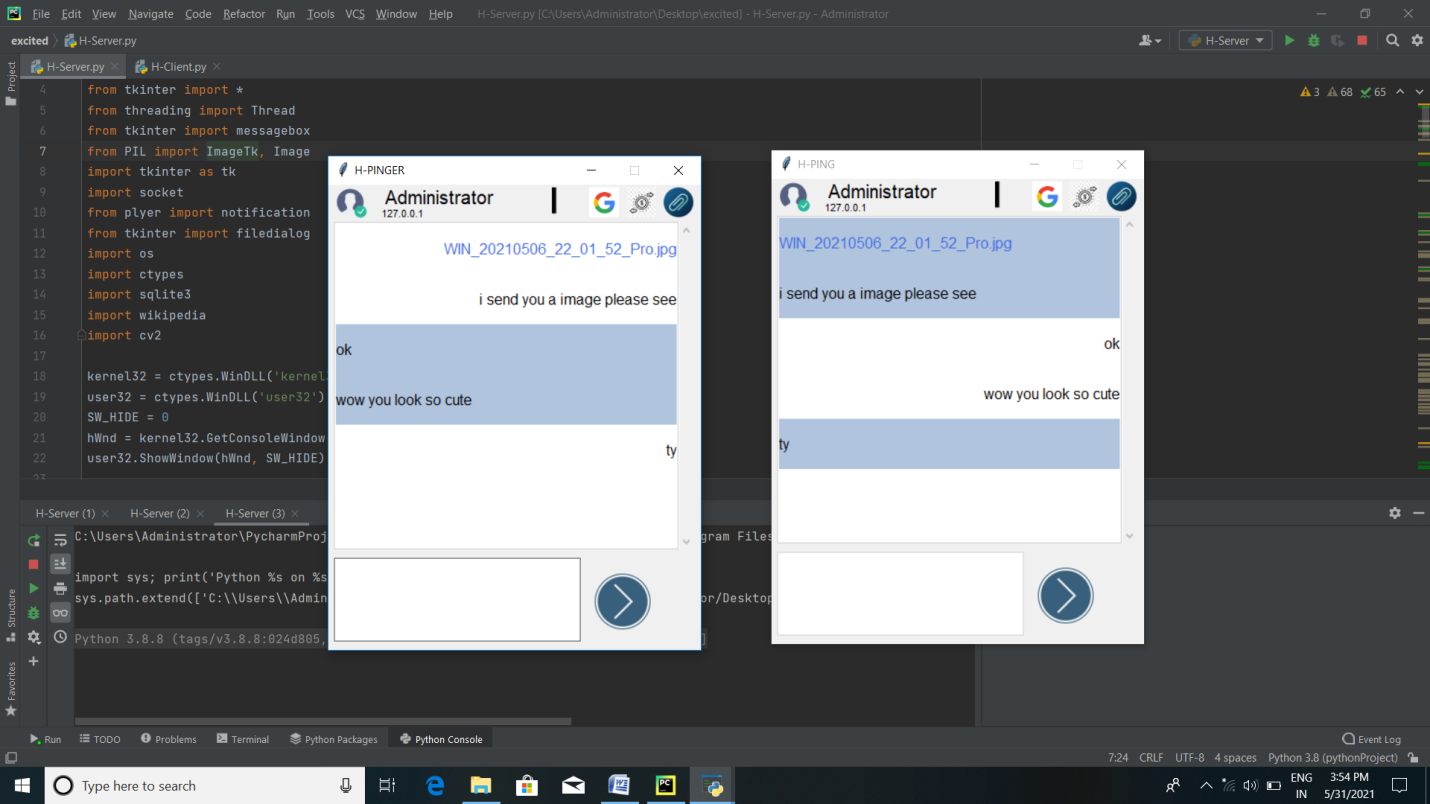


7. Transfering files.

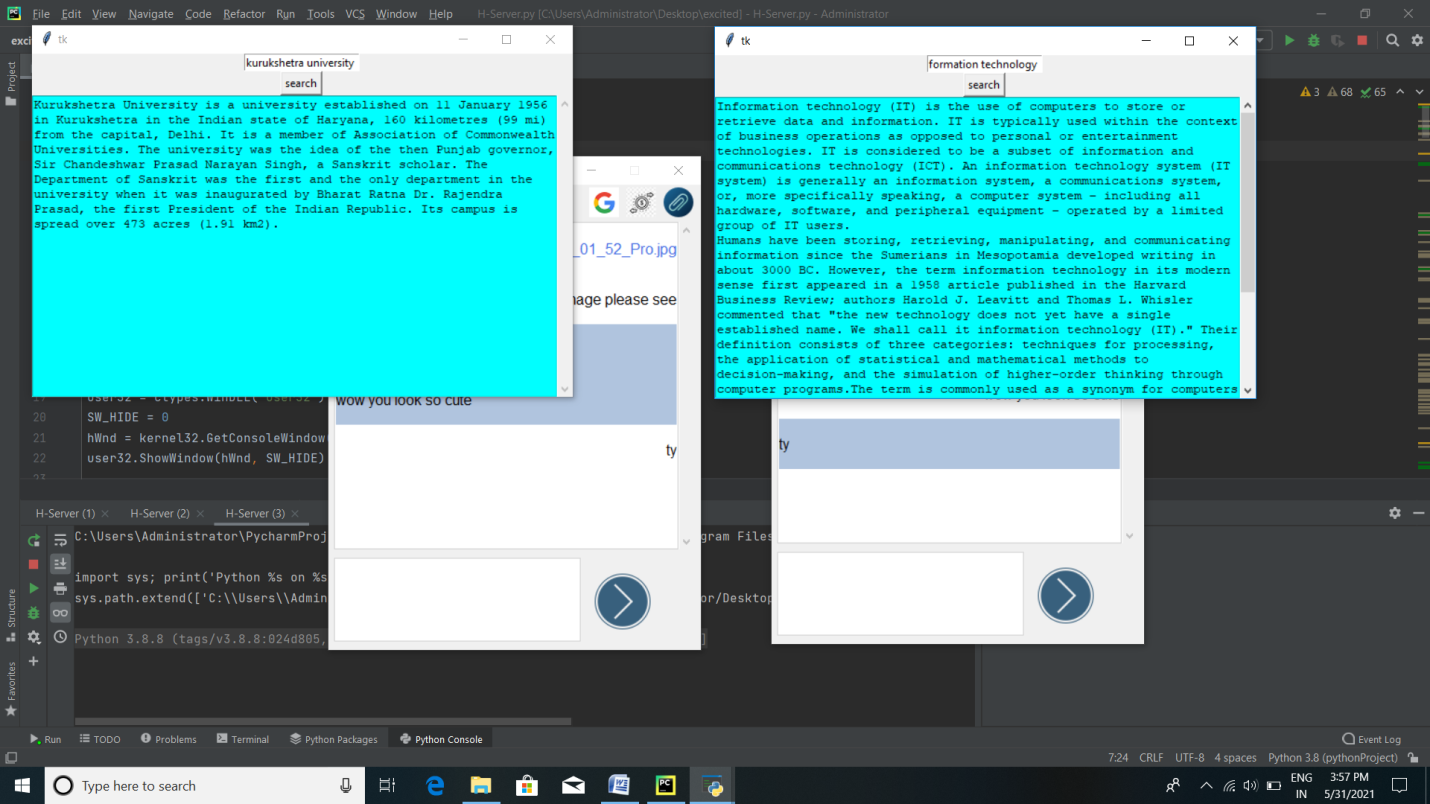




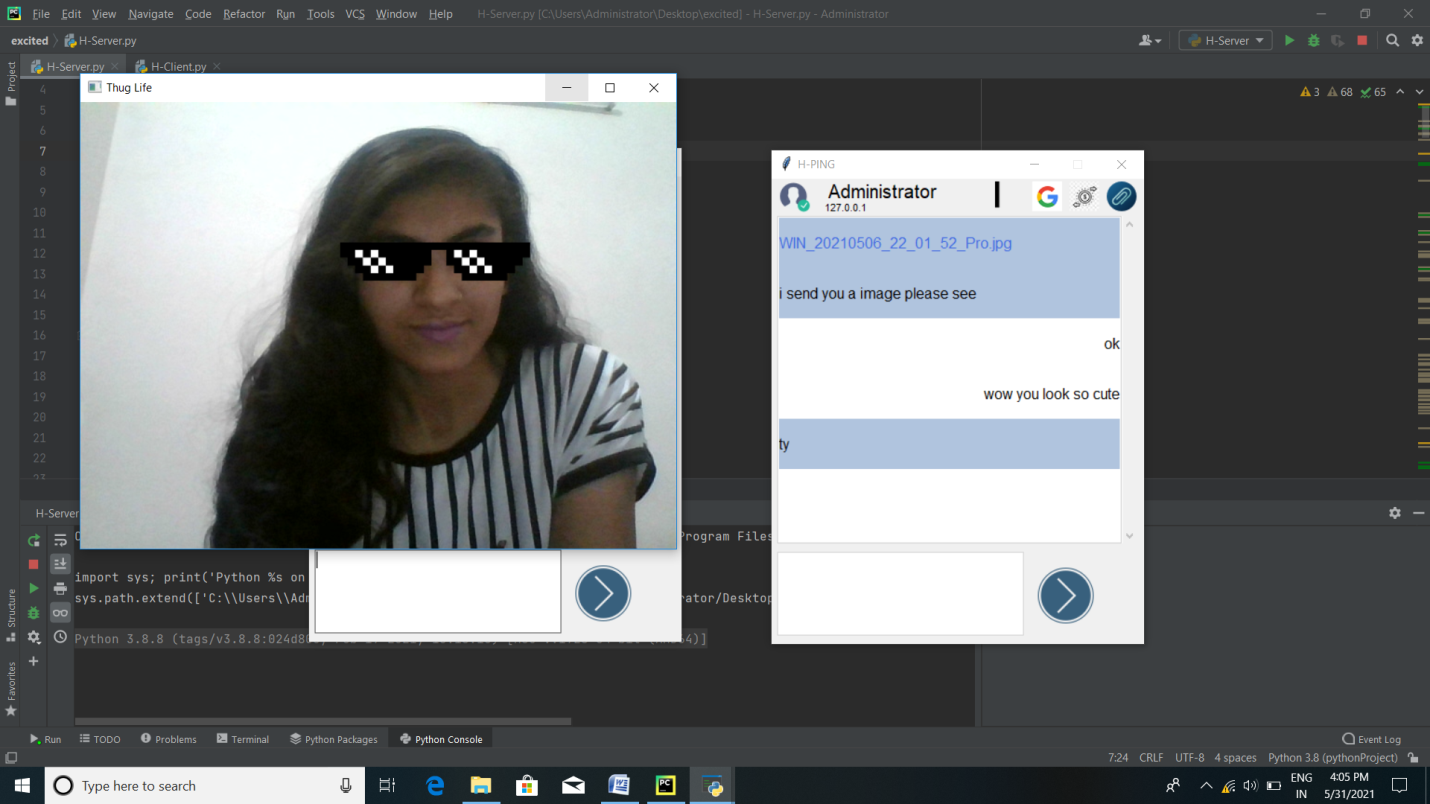




8. Open Google :server side Google is on right side and client side Google is on left side



9. Open filters



**CHAPTER -5**

**CONCLUSION**

While developing the system a conscious effort has been made to create and develop a software package, making use of available tools, techniques and resources – that would generate a proper system. While making the system, an eye has been kept on making it as user-friendly. As such one may hope that the system will be acceptable to any user and will adequately meet his/her needs. As in case of any system development process where there are a number of short comings, there has been some shortcomings in the development of this system also. There are some of the areas of improvement which couldn’t be implemented due to time constraints.