BINUS UNIVERSITY
BINUS INTERNATIONAL

Assignment Cover Letter

(Individual Work)

Student Information:		**Surname**	**Given Names**	**Student ID Number**
	1.	**Rachmadi**	**Cen**	**2301891752**

Course Code	**: COMP6056**	**Course Name**	**: Program Design Methods**
Class	**: LIAC**	**Name of Lecturer(s)**	**: Ida Bagus Kerthyayana**
Major	**: CS**		
Title of Assignment (if any)	: Github Web Scraping		
Type of Assignment	**: Final Project**		
Submission Pattern			
Due Date	**: 14-01-20**	**Submission Date**	**: 14-01-20**

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

"Github Web Scraping"

Name : Radisa Hussein Rachmadi

ID : 2301891752

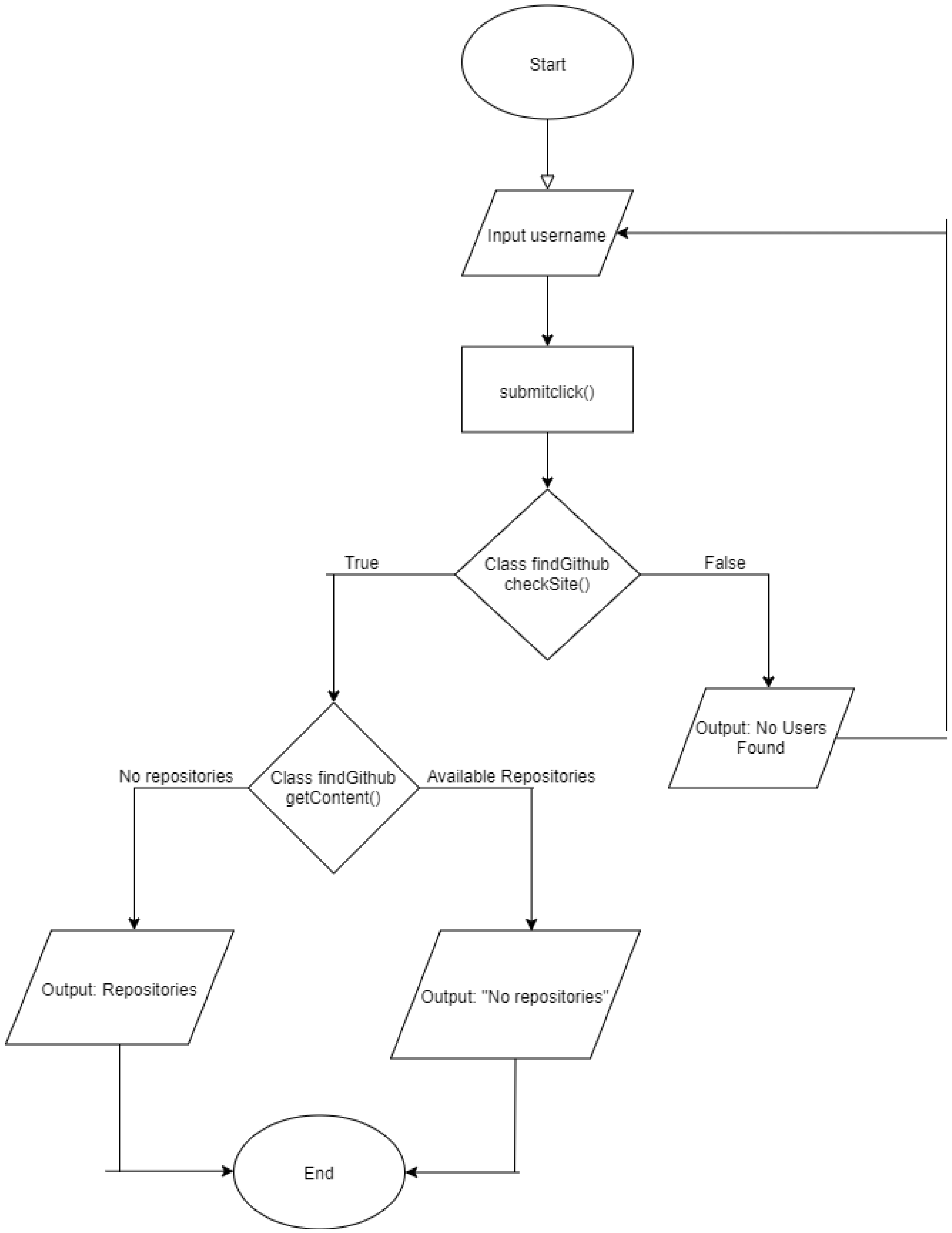
I. Description

The function of this program:

The purpose of this program is to return links of repositories belonging to the GitHub username that is inputted. It uses the requests module for creating an HTTP request to open up the Github page. Then it uses the beautifulsoup4 module to scrape the data off of the user's Github page and return it into links. Then using the tkinter module, to make a GUI, where the user can input a username and it outputs a list of up to 15 repositories.

II.a. Design/Plan

Project's Hierarchy Chart

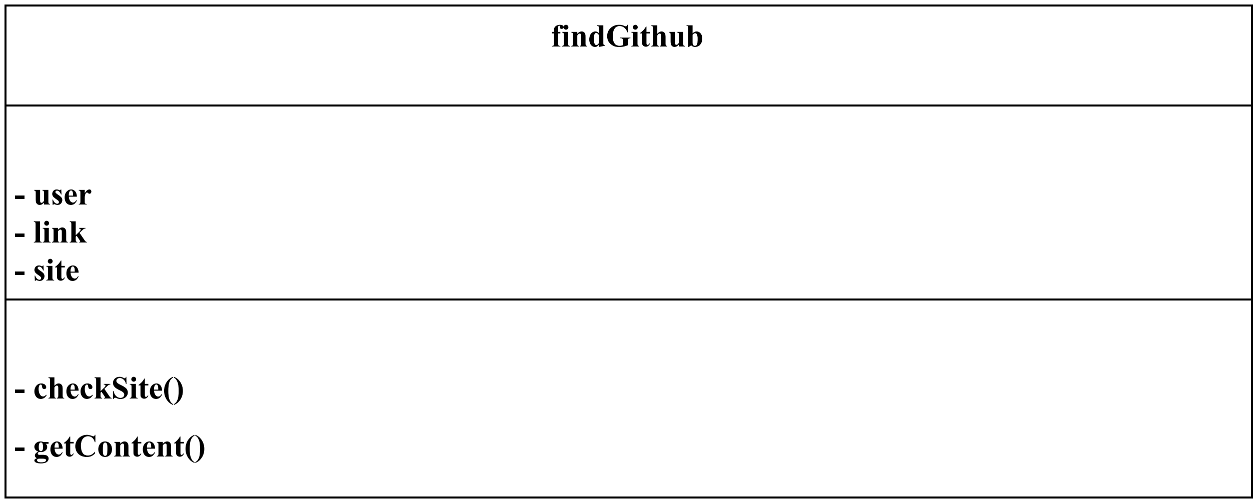


II.b. Explanation of Each Function Inside the findGithub Class

FinalCode.py

- **checkSite (self) :**
    - Opens the Github repository link of the inputted user with the requests module
    - If status code returns 200, site is accessible and no issues persist, returns True.
    - If returns a code other than 200, it could mean no available profile, or poor internet connection, returns False
- **getContent (self) :**
    - Initialize variable “source” to store the contents of the accessed website from requests
    - Using the BeautifulSoup module to process the contents in “source”
    - With a for loop using beautifulsoup4, look for all “h3” tags with class ‘wb-break-all’ to find repositories
    - From those h3 tags, look for the “a” tags supplied with a href attribute
    - Appends all “a” tags with a href attribute to a list called urls
    - Each url is then re-appended to a list called newurls, with “https://www.github.com” + the url from the href attribute, to create a valid link
    - If user has no repositories, will return an error label with “This user has no repositories”
    - Finally using a for loop, printing out the labels, from the newurls list

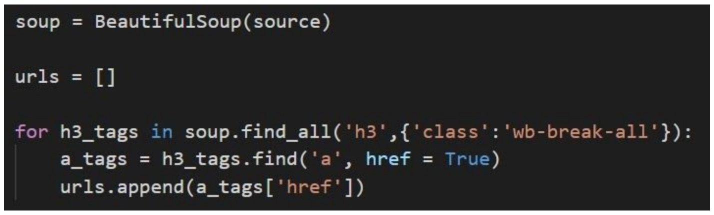
Class Diagram



II.c. Explanation of Each Function outside the findGithub class

- **submitclick() :**
    - This function is called whenever the search button is clicked
    - if checkSite() returns true, it will continue to run getContent()
    - if checkSite() returns false, it will remove all widgets in showing frame and result frame, and then print out error labels.

III.a. Lessons that Have Been Learned



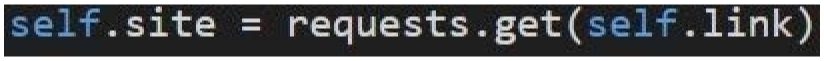
```
soup = BeautifulSoup(source)

urls = []

for h3_tags in soup.find_all('h3',{'class':'wb-break-all'}):
    a_tags = h3_tags.find('a', href = True)
    urls.append(a_tags['href'])
```

1. Web Scraping using Beautiful Soup 4

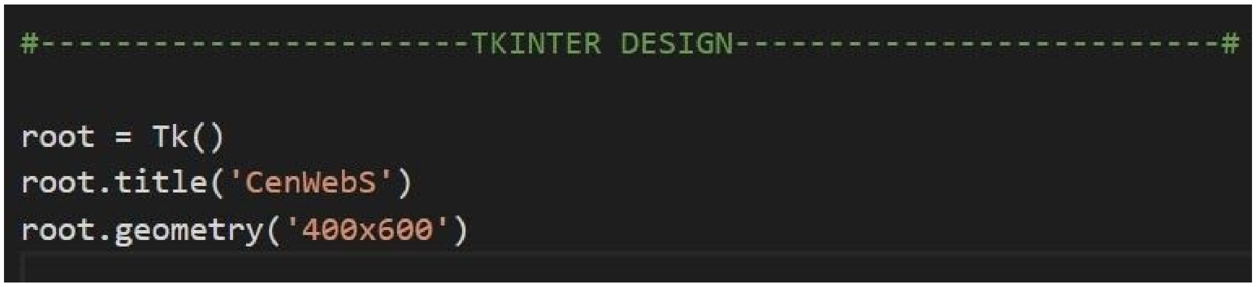
I came across web scraping during my brainstorming phase, and I found it to be very interesting how with just a few lines of codes, one can store the contents of an entire page, and analyze it. I initially wanted to use selenium, but after doing more research I found out that beautiful soup 4 works better for small projects that are simple.



```
self.site = requests.get(self.link)
```

2. Requests Module

After reading documentations about beautifulsoup4, I learned that this module is not able to work entirely on their own, it needs external libraries to help access html pages, this is where the requests module comes in. It is a standard library for making HTML requests, which is just what I needed for web scraping Github.

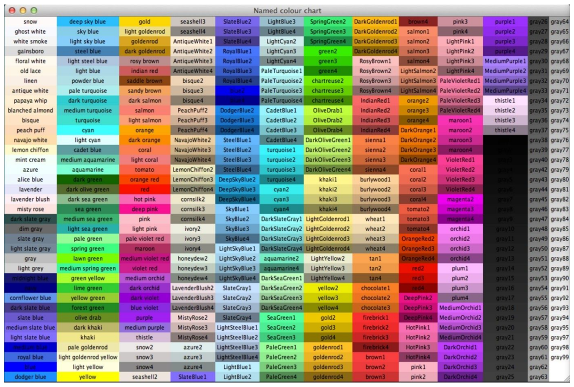


```
#---------------------TKINTER DESIGN-----------------------#


root = Tk()
root.title('CenWebS')
root.geometry('400x600')
```

3. Tkinter Module

I initially started testing out beautifulsoup4, or bs4 for short, and the requests module in a command line program. When I got the hang of it, I thought that this program looked too simple so I looked to find other means to put my program into, that’s where I came across Tkinter. Python’s standard GUI package. It was pretty difficult to get around using Tkinter, but with practice and more researching, I was finally able to get the hang of it. With Tkinter, it is pretty easy as it has a wide option of widgets, and I learned



Named colour chart

snow	deep sky blue	gold	seashell3	SlateBlue2	LightBlue3	SpringGreen2	DarkGoldenrod1	brown4	pink3	purple1	gray26	gray64
ghost white	sky blue	light goldenrod	seashell4	SlateBlue3	LightBlue4	SpringGreen3	DarkGoldenrod2	salmon1	pink4	purple2	gray27	gray65
white smoke	light sky blue	goldenrod	AntiqueWhite1	SlateBlue4	LightCyan2	SpringGreen4	DarkGoldenrod3	salmon2	LightPink1	purple3	gray28	gray66
gainsboro	steel blue	dark goldenrod	AntiqueWhite2	RoyalBlue1	LightCyan3	green2	DarkGoldenrod4	salmon3	LightPink2	purple4	gray29	gray67
floral white	light steel blue	rosy brown	AntiqueWhite3	RoyalBlue2	LightCyan4	green3	RosyBrown1	salmon4	LightPink3	MediumPurple1	gray30	gray68
old lace	light blue	indian red	AntiqueWhite4	RoyalBlue3	PaleTurquoise1	green4	RosyBrown2	LightSalmon2	LightPink4	MediumPurple2	gray31	gray69
linen	powder blue	saddle brown	bisque2	RoyalBlue4	PaleTurquoise2	chartreuse2	RosyBrown3	LightSalmon3	PaleVioletRed1	MediumPurple3	gray32	gray70
antique white	pale turquoise	sandy brown	bisque3	blue2	PaleTurquoise3	chartreuse3	RosyBrown4	LightSalmon4	PaleVioletRed2	MediumPurple4	gray33	gray71
papaya whip	dark turquoise	dark salmon	bisque4	blue4	PaleTurquoise4	chartreuse4	IndianRed1	orange2	PaleVioletRed3	thistle1	gray34	gray72
blanched almond	medium turquoise	salmon	PeachPuff2	DodgerBlue2	CadetBlue1	OliveDrab1	IndianRed2	orange3	PaleVioletRed4	thistle2	gray35	gray73
bisque	turquoise	light salmon	PeachPuff3	DodgerBlue3	CadetBlue2	OliveDrab2	IndianRed3	orange4	maroon1	thistle3	gray36	gray74
peach puff	cyan	orange	PeachPuff4	DodgerBlue4	CadetBlue3	OliveDrab4	IndianRed4	DarkOrange1	maroon2	thistle4	gray37	gray75
navajo white	light cyan	dark orange	NavajoWhite2	SteelBlue1	CadetBlue4	DarkOliveGreen1	sienna1	DarkOrange2	maroon3	[illegible]	gray38	gray76
lemon chiffon	cadet blue	coral	NavajoWhite3	SteelBlue2	turquoise1	DarkOliveGreen2	sienna2	DarkOrange3	maroon4	gray2	gray39	gray77
mint cream	medium aquamarine	light coral	NavajoWhite4	SteelBlue3	turquoise2	DarkOliveGreen3	sienna3	DarkOrange4	VioletRed1	gray3	gray40	gray78
azure	aquamarine	tomato	LemonChiffon2	SteelBlue4	turquoise3	DarkOliveGreen4	sienna4	coral1	VioletRed2	gray4	gray42	gray79
alice blue	dark green	orange red	LemonChiffon3	DeepSkyBlue2	turquoise4	khaki1	burlywood1	coral2	VioletRed3	gray5	gray43	gray80
lavender	dark olive green	red	LemonChiffon4	DeepSkyBlue3	cyan2	khaki2	burlywood2	coral3	VioletRed4	gray6	gray44	gray81
lavender blush	dark sea green	hot pink	cornsilk2	DeepSkyBlue4	cyan3	khaki3	burlywood3	coral4	magenta2	gray7	gray45	gray82
misty rose	sea green	deep pink	cornsilk3	SkyBlue1	cyan4	khaki4	burlywood4	tomato2	magenta3	gray8	gray46	gray83
dark slate gray	medium sea green	pink	cornsilk4	SkyBlue2	DarkSlateGray1	LightGoldenrod1	wheat1	tomato3	magenta4	gray9	gray47	gray84
dim gray	light sea green	light pink	ivory2	SkyBlue3	DarkSlateGray2	LightGoldenrod2	wheat2	tomato4	orchid1	gray10	gray48	gray85
slate gray	pale green	pale violet red	ivory3	SkyBlue4	DarkSlateGray3	LightGoldenrod3	wheat3	OrangeRed2	orchid2	gray11	gray49	gray86
light slate gray	spring green	maroon	ivory4	LightSkyBlue1	DarkSlateGray4	LightGoldenrod4	wheat4	OrangeRed3	orchid3	gray12	gray50	gray87
gray	lawn green	medium violet red	honeydew2	LightSkyBlue2	aquamarine2	LightYellow2	tan1	OrangeRed4	orchid4	gray13	gray51	gray88
light grey	medium spring green	violet red	honeydew3	LightSkyBlue3	aquamarine4	LightYellow3	tan2	red2	plum1	gray14	gray52	gray89
midnight blue	green yellow	medium orchid	honeydew4	LightSkyBlue4	DarkSeaGreen1	LightYellow4	tan4	red3	plum2	gray15	gray53	gray90
navy	lime green	dark orchid	LavenderBlush2	SlateGray1	DarkSeaGreen2	yellow2	chocolate1	red4	plum3	gray16	gray54	gray91
cornflower blue	yellow green	dark violet	LavenderBlush3	SlateGray2	DarkSeaGreen3	yellow3	chocolate2	DeepPink2	plum4	gray17	gray55	gray92
dark slate blue	forest green	blue violet	LavenderBlush4	SlateGray3	DarkSeaGreen4	yellow4	chocolate3	DeepPink3	MediumOrchid1	gray18	gray56	gray93
slate blue	olive drab	purple	MistyRose2	SlateGray4	SeaGreen1	gold2	firebrick1	DeepPink4	MediumOrchid2	gray19	gray57	gray94
medium slate blue	dark khaki	medium purple	MistyRose3	LightSteelBlue1	SeaGreen2	gold3	firebrick2	HotPink1	MediumOrchid3	gray20	gray58	gray95
light slate blue	khaki	thistle	MistyRose4	LightSteelBlue2	SeaGreen3	gold4	firebrick3	HotPink2	MediumOrchid4	gray21	gray59	gray97
medium blue	pale goldenrod	snow2	azure2	LightSteelBlue3	PaleGreen1	goldenrod1	firebrick4	HotPink3	DarkOrchid1	gray22	gray60	gray98
royal blue	light goldenrod yellow	snow3	azure3	LightSteelBlue4	PaleGreen2	goldenrod2	brown1	HotPink4	DarkOrchid2	gray23	gray61	gray99
blue	light yellow	snow4	azure4	LightBlue1	PaleGreen3	goldenrod3	brown2	pink1	DarkOrchid3	gray24	gray62	
dodger blue	yellow	seashell2	SlateBlue1	LightBlue2	PaleGreen4	goldenrod4	brown3	pink2	DarkOrchid4	gray25	gray63	

4. Tkinter Color Charts

Another factor that made it much easier to use Tkinter was it's color coding. Which is just to write the name of the color. The full color chart can be seen in the documentation, but this feature makes it so much easier to implement colors without having to look up hex codes or RGB codes.

III.b. Problem that Have Been Overcome

The hardest problem was choosing a project and sticking to it. I originally started with a random command line program, but I found it to be really simple and boring. I then got an idea to create a website using Django, which is a python web-framework. After about 80% of the progress I got stuck at trying to create animations and drag & drop features through Django, and after researching more about it, turns out that it requires significant knowledge of Javascript, CSS, and HTML. So then after brainstorming for more ideas, I stumbled across Web Scraping using Beautiful Soup 4.

The idea and concept of a Github web scraper was pretty easy, but implementing it into Tkinter was pretty challenging, not as easy as I thought it would be. I found some codes in the internet of a pretty similar web scraping program, but it scrapes your profile stats instead of your repositories. I was able to get some ideas on how to build my program. I then had minor troubles trying to design the layout for Tkinter using pack, which I then solved by using the grid command which allows to put many widgets on a grid like layout.

Resources :

- https://stackoverflow.com (website I used when I was trying to fix the errors)
- http://www.science.smith.edu/dftwiki/images/3/3d/TkInterColorCharts.png (Color chart for Tkinter)
- https://docs.python.org/3/library/tk.html (documentation for Tkinter)
- https://github.com/jamesgeorge007/GitHub-Mate-Desktop (reference for a similar web scraping project)

V. Source Code

```
import requests
from bs4 import BeautifulSoup
from tkinter import *


#--------------------------PROGRAM---------------------------#

class findGithub():                              #class for finding github
repositories

    def __init__(self,user):
        self.user = str(user)
                  self.link = 'https://www.github.com/' + self.user +
'?tab=repositories' #link for repositories page of a given user
        self.site = requests.get(self.link)

    def checkSite(self):
        #function to check if site is accessible
        #if not, could be due to no connection, wrong username, or private
page
        if self.site.status_code == 200:
            return True
        else:
            return False
    def getContent(self):
        #function to get the content from the link

                                        source     =     self.site.content
#variable to store page content of accessed website from requests
                                      soup     =     BeautifulSoup(source)
#beautiful soup module to process the source

        urls = []

             for  h3_tags  in  soup.find_all('h3',{'class':'wb-break-all'}):
#for loop finding all h3 tags with the specified class
                            a_tags   =   h3_tags.find('a',   href   =   True)
#from the h3 tags, finding a tags with a HREF attribute
            urls.append(a_tags['href'])

        finduser = findGithub(self.user)
        global newurls
        newurls = []
        for url in urls:
                        newurls.append('https://github.com'   +   str(url))
#Adding 'https://github.com to the href tags making a valid link


         for widget in resultframe.winfo_children():              #Destroy
Widgets inside Resultframes
            widget.destroy()

         for widget in showingframe.winfo_children():             #Destroy
Widgets inside Showingframes
            widget.destroy()

        labels = []
         if len(newurls) > 15:                                   #Limit on
number of repositories, more than 15, will end up going out of screen
            del newurls[15:]
            newurls.append('and more')
        elif len(newurls) == 0:                                    #If no repositories found, will print out this label
            norepo = 'This user has no repositories'
            norepolbl = Label(showingframe,text=norepo,font=('Arial',13),fg='red' )
            norepolbl.grid(row=2,column=1)

        # "Showing users profile" label

        showingprofile = 'Showing ' + inputuser.get() + "'s Profile"
        showinglabel = Label(showingframe,text=showingprofile,font=('Arial',13),bg='gray26',fg='white')
        showinglabel.grid(row=0,column=1)

        #for loop printing of repositories in the form of labels
        for i in range(len(newurls)):
            labels.append(Label(resultframe,text=newurls[i]))
            labels[i].grid(row=i,column=1)




def submitclick():                                                 #Function called when submit button is clicked
    finduser = findGithub(inputuser.get())                         #Initializing variable as class

    if finduser.checkSite() is True:
        finduser.getContent()

    else:
        for widget in showingframe.winfo_children():               #making sure the showing frame and result frame is clear
            widget.destroy()
        for widgets in resultframe.winfo_children():
            widgets.destroy()
        # Labels for when checksite returns false

        showingprofile = "User not found!"
        internet = 'Make sure you have a good internet connection!'
                                                       showinglabel       =
Label(showingframe,text=showingprofile,font=('Arial',13),fg='firebrick1')
                                                      internetlabel       =
Label(showingframe,text=internet,font=('Arial',13),fg='light slate gray')
        showinglabel.grid(row=0,column=1)
        internetlabel.grid(row=2,column=1)


#-----------------------TKINTER DESIGN--------------------------#

root = Tk()
root.title('CenWebS')
root.geometry('400x600')

#Set                                                                 window
--------------------------------------------------------------------------
--------

cenWebsLabel                                                              =
Label(root,text='CenWebS',font=('Helvetica',40),relief='groove',bd=4,padx=
77,bg='gray26',fg='white smoke')
cenWebsLabel.grid(column=0,row=0)

#github                          label                                frame
------------------------------------------------------------

githubFrame = Frame(root)
githubFrame.grid(row=1,column=0)

githublabel         =         Label(githubFrame,text='Github          Web
Scraping',font=('Arial',20),relief=RIDGE,bd=4)
githublabel.grid(row=0,column=0)
```

```
#input                  frame                  and                  widgets
-------------------------------------------------------

inputframe = Frame(root,bd=4,relief='ridge',padx=10,pady=10)
inputframe.grid(row=2,column=0)

inputuser = Entry(inputframe,font=('Arial',10))
inputuser.grid(row=2,column=2)

labelinput = Label(inputframe,text='Username: ', font=('Arial',15))
labelinput.grid(row=2,column=1)

submitbutton                                                              =
Button(inputframe,text='Search',font=('Arial',13),command=submitclick)
submitbutton.grid(row=3,column=1,columnspan=2)

#Showing           Profile           frame           and           widget
---------------------------------------------------------

showingframe = Frame(root,padx=10,pady=10,)
showingframe.grid(row=3,column=0)

#Result frames and widgets ----------------------------------------------

resultframe = Frame(root,bd=4,relief='flat')
resultframe.grid(row=4,column=0,rowspan=10)



#Main loop ------------------------------------

root.mainloop()
```