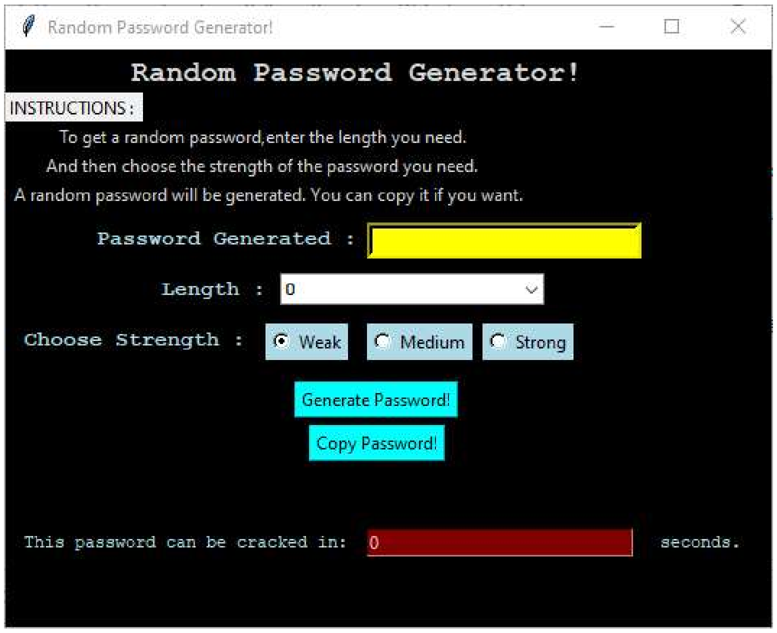
PROJECT 1: RANDOM PASSWORD GENERATOR

I. Introduction to the project

Remember that you store a lot of sensitive information you would not like to get stolen from your account. The first line of defence from unauthorized access to your computer and personal information is provided by passwords. As your password strengthens, your computer is more protected from hackers and malicious software. Paying attention to your password is therefore essential.
The application is a simple, entirely offline **random password generator**, which gives you the required password and also prevents the same from being intercepted. It also tells about **the hack time** of the password to let one know how strong is the password generated. The app can generate random passwords with letters, numeric and special characters. The required **length** and the **strength** of the password can be specified by the user.

II. Technology Used

For the application we used **Tkinter module** present in python which helps developing **Graphic User Interface (GUI)** environment for the application. Python offers a fast and easy way of creating GUI applications when combined with Tkinter. Tkinter provides various **widgets** such as frames, labels, buttons, check buttons, radio buttons, entries and combo boxes.
With the help of Tkinter module and the widget attributes, I created the window as shown below:



To get a random password, enter the length you need.
And then choose the strength of the password you need.
A random password will be generated. You can copy it if you want.

Length : 0

Choose Strength : [✓ Weak] Medium Strong

This password can be cracked in: 0 seconds.

Figure 1: Layout of the Random password Generator

The application asks the user to select the length of the passwords and the strength of the password between low, medium and strong. And generate password creates a random password and copy password copies the password to the clipboard.

III. Flow Chart Related to the Password Generation

The figure below shows the flow chart for the project developed:

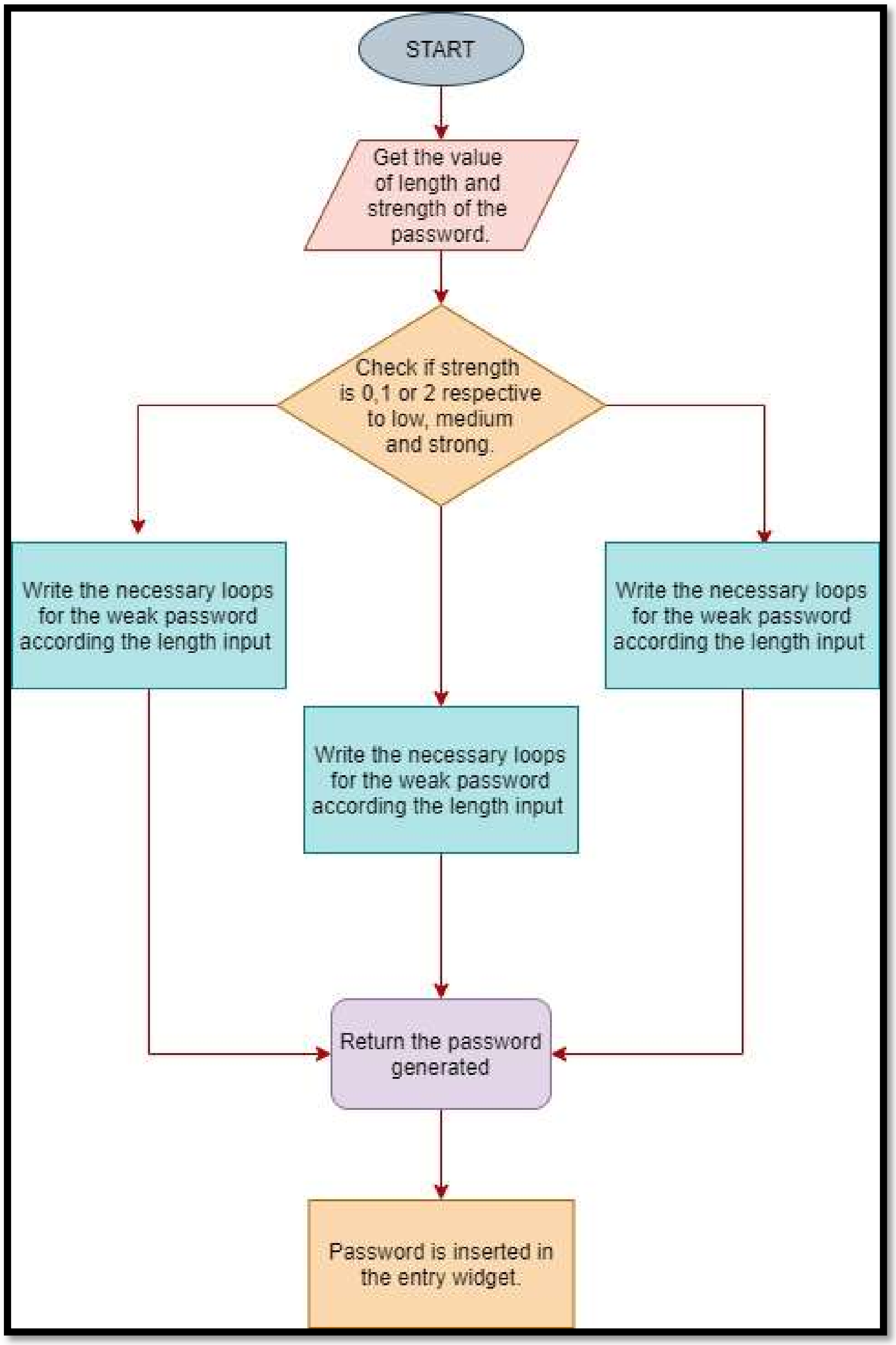

START
Get the value of length and strength of the password.
Check if strength is 0,1 or 2 respective to low, medium and strong.
Write the necessary loops for the weak password according the length input
Write the necessary loops for the weak password according the length input
Write the necessary loops for the weak password according the length input
Return the password generated
Password is inserted in the entry widget.

Figure 2: Flow Chart for the Password Generator

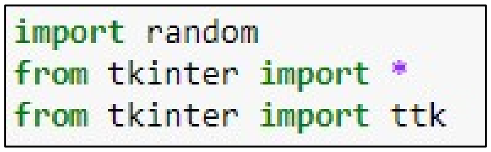
IV. Code for the Project

The GitHub repository link for the project is: https://github.com/Isha1504/Random-Password-Generator

V. Explanation of the code

This section describes the code block by block. And give detailed information for each section.

a. **Importing Modules** – The very first step in order to create any application is to import all the modules required by the application

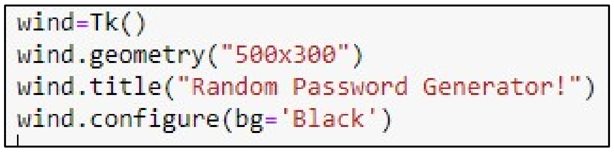


```
import random
from tkinter import *
from tkinter import ttk
```

Figure3: Modules imported

b. **Layout Design –** Now this subsection describes what user sees on the window of the application.

The layout design initiates with basic window design which includes the title, size and other attributes of the window. The code for the same is shown below:



```
wind=Tk()
wind.geometry("500x300")
wind.title("Random Password Generator!")
wind.configure(bg='Black')
```

Figure4: Window creation

The first line **creates a window** object and the next line **geometry** takes up the dimensions of the window. **Title** describes the title of the window and **configure** is used to add extra features to any object. **mainloop** function displays the window. Figure5 shows how the basic window looks.

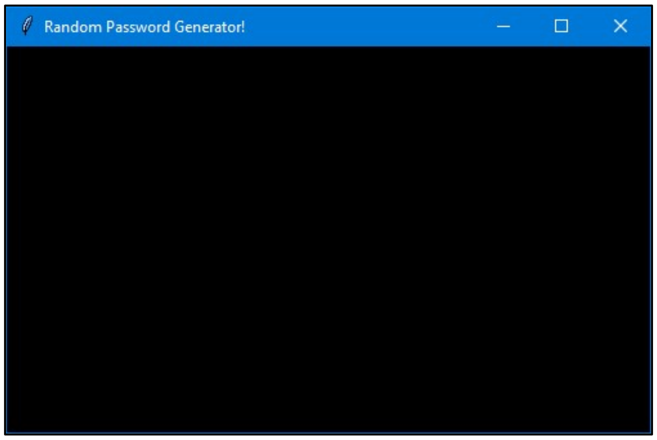
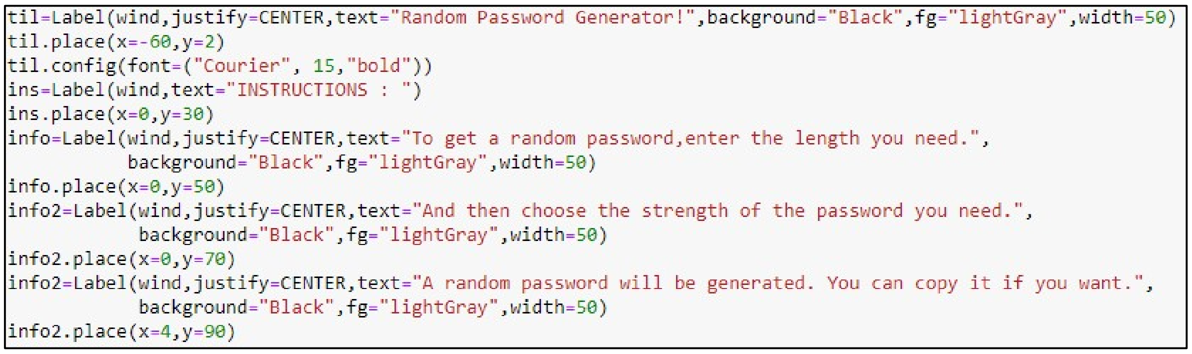


Figure5: Starting Window

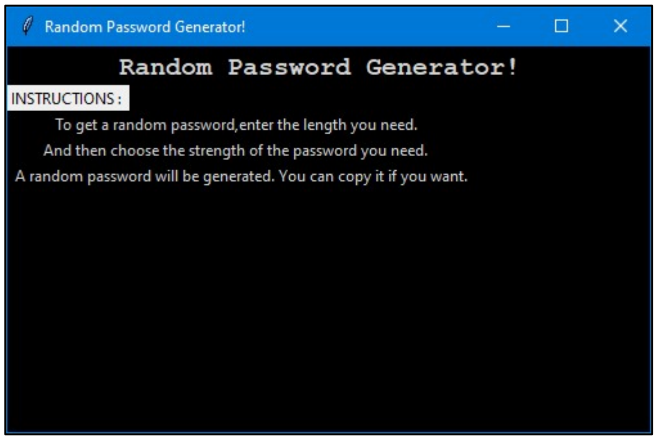
For the designing of the layout I used Tkinter widgets such as Labels, entry boxes, buttons, Radio buttons and many more.

First, I mentioned the instructions on how to get a random password. For the same we use the label widget. Code for the same is shown in Figure6 and Figure7 shows the output of the code of Figure6.



```
til=Label(wind,justify=CENTER,text="Random Password Generator!",background="Black",fg="lightGray",width=50)
til.place(x=-60,y=2)
til.config(font=("Courier", 15,"bold"))
ins=Label(wind,text="INSTRUCTIONS : ")
ins.place(x=0,y=30)
info=Label(wind,justify=CENTER,text="To get a random password,enter the length you need.",
           background="Black",fg="lightGray",width=50)
info.place(x=0,y=50)
info2=Label(wind,justify=CENTER,text="And then choose the strength of the password you need.",
            background="Black",fg="lightGray",width=50)
info2.place(x=0,y=70)
info2=Label(wind,justify=CENTER,text="A random password will be generated. You can copy it if you want.",
            background="Black",fg="lightGray",width=50)
info2.place(x=4,y=90)
```

Figure6: Code for the title and basic info of the application

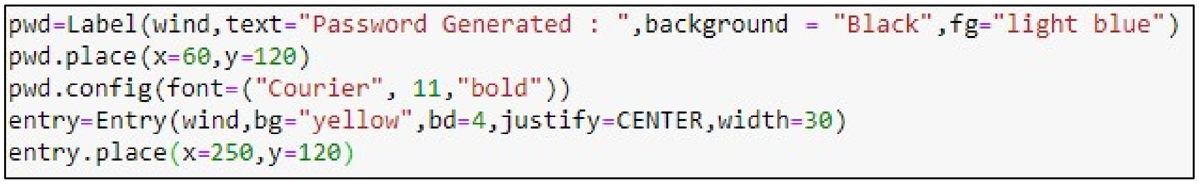


```
Random Password Generator!

Random Password Generator!
INSTRUCTIONS :
To get a random password,enter the length you need.
And then choose the strength of the password you need.
A random password will be generated. You can copy it if you want.
```

Figure7: Output of the code mentioned in Figure6

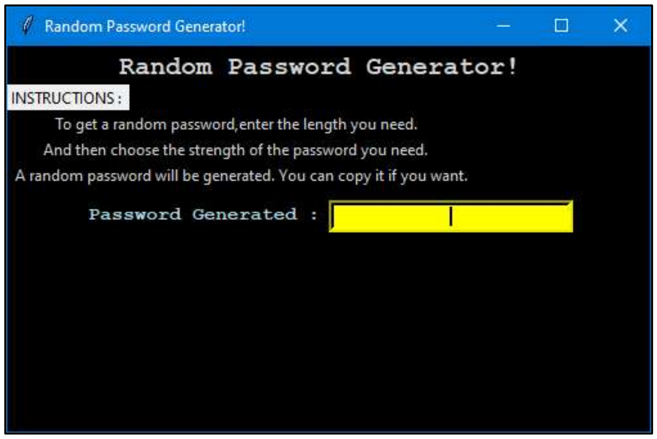
The Label widget takes the window object as the first attribute and other attributes such as **text, background, fore background, width** and other attributes can be added using **config** function. The **place** function takes the exact coordinates of the position of the widget.

The next step is to create the **Label and Entry box** for the password generated in the system. The Entry widget uses a few attributes too. The code and output for the same are shown in Figure8 and Figure9 respectively.



```
pwd=Label(wind,text="Password Generated : ",background = "Black",fg="light blue")
pwd.place(x=60,y=120)
pwd.config(font=("Courier", 11,"bold"))
entry=Entry(wind,bg="yellow",bd=4,justify=CENTER,width=30)
entry.place(x=250,y=120)
```

Figure8: Code for the password box


Random Password Generator!

Random Password Generator!

INSTRUCTIONS :

To get a random password,enter the length you need.

And then choose the strength of the password you need.

A random password will be generated. You can copy it if you want.

Password Generated :

Figure9: Output password entry box

The next step would be to create the dropdown menu to get the length of the password. For the same we used **Combobox** and Label widget. Combo box creates a drop-down menu. The code for the same is shown in Figure10.

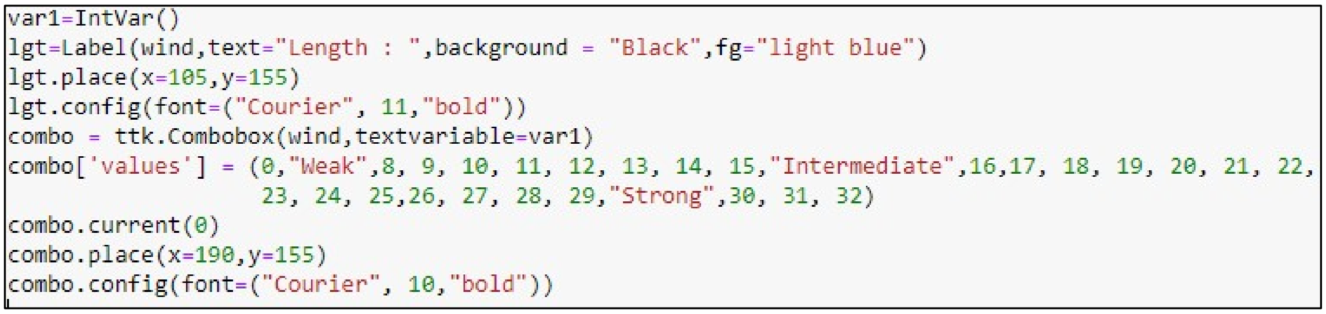
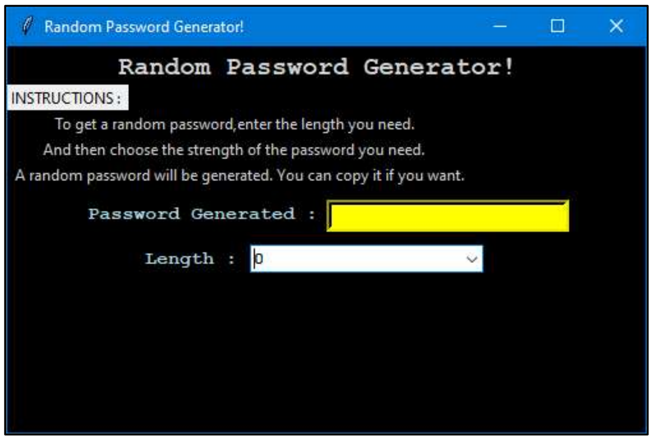

```
var1=IntVar()
lgt=Label(wind,text="Length : ",background = "Black",fg="light blue")
lgt.place(x=105,y=155)
lgt.config(font=("Courier", 11,"bold"))
combo = ttk.Combobox(wind,textvariable=var1)
combo['values'] = (0,"Weak",8, 9, 10, 11, 12, 13, 14, 15,"Intermediate",16,17, 18, 19, 20, 21, 22,
                   23, 24, 25,26, 27, 28, 29,"Strong",30, 31, 32)
combo.current(0)
combo.place(x=190,y=155)
combo.config(font=("Courier", 10,"bold"))
```

Figure10: Code for the length input

The first line gives the int input of an object and it stores the same in var1. The Combobox widget takes values for the drop-down menu. The values are given in form of the list. Current function defines the index whose value is going to be by default on the window. The output for the same is shown in Figure11.


Random Password Generator!

Random Password Generator!

INSTRUCTIONS :

To get a random password,enter the length you need.

And then choose the strength of the password you need.

A random password will be generated. You can copy it if you want.

Password Generated :

Length : 0

Figure11: Output with the length drop-down menu

The next part we have to work on is the buttons to choose the strength of the password. For the same I used Radio buttons which help in getting check buttons. The code for the same is given in Figure12.

```
var=IntVar()
lab=Label(wind,text="Choose Strength : ",bg="black",fg="light blue")
lab.place(x=10,y=190)
lab.config(font=("Courier", 11,"bold"))
low = Radiobutton(wind, text="Weak",variable=var,value=0,activebackground='light green',background = "light blue")
low.place(x=180,y=190)
middle = Radiobutton(wind, text="Medium", variable=var,value=1,activebackground='light green',background = "light blue")
middle.place(x=250,y=190)
strong = Radiobutton(wind, text="Strong",value=2,variable=var,activebackground='light green',background = "light blue")
strong.place(x=330,y=190)
```

Figure12: Code for the strength of the password

First, we have a variable var to store which button is selected. We create a radio button and give them the values to know which is selected by the user. Then we place them in the required position with styling attributes. Figure13 shows the output of the same.

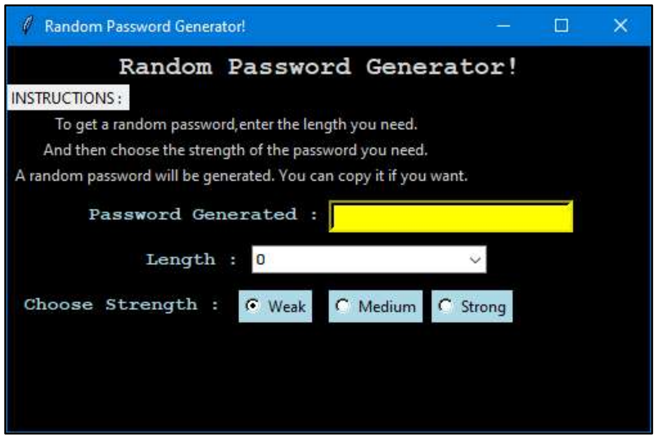

Random Password Generator!
Random Password Generator!
INSTRUCTIONS :
To get a random password,enter the length you need.
And then choose the strength of the password you need.
A random password will be generated. You can copy it if you want.
Password Generated :
Length : 0
Choose Strength :
Weak
Medium
Strong

Figure13: Window having strength of the password

The next and final steps in layout is to create the two buttons one to generate the password and display it and the second one was to copy the password. The code and output for the same is shown in Figure14 and Figure15 respectively.

```
def password_gen():
    pass
def copy_pass():
    pass
gen=Button(wind, text="Generate Password!",activebackground="red",background = "cyan",command=password_gen)
gen.place(x=200,y=230)
cpy=Button(wind,text="Copy Password!",activebackground="red",background = "cyan",command=copy_pass)
cpy.place(x=210,y=260)
```

Figure14: The code for generate and copy button

The **Button** on click obviously need to have some function that needs to be performed when the button is clicked. The same are defined as blank for now. And the functions are called using **command attribute**.

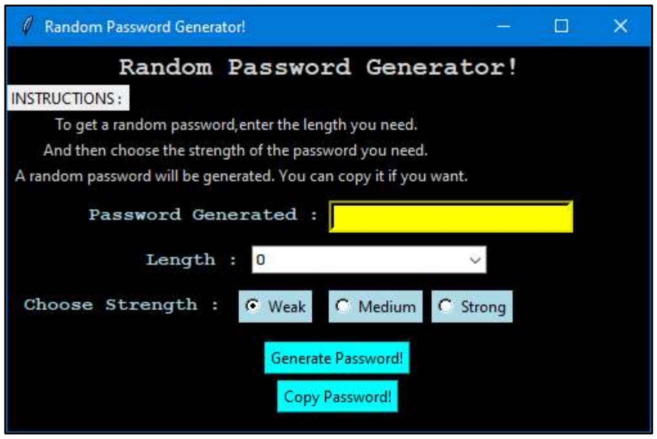
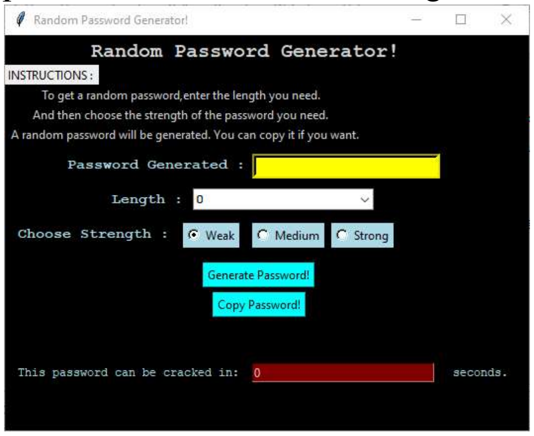

Random Password Generator!
Random Password Generator!
INSTRUCTIONS :
To get a random password,enter the length you need.
And then choose the strength of the password you need.
A random password will be generated. You can copy it if you want.
Password Generated :
Length : 0
Choose Strength :
Weak
Medium
Strong
Generate Password!
Copy Password!

Figure15: Layout with the final buttons

Now an additional feature with the random password generator is it tells the user that how much time will take to crack their password by any hacker. The feature is added using Label and entry layout and the output of the same is shown in Figure16.


Random Password Generator!

Random Password Generator!

INSTRUCTIONS :

To get a random password,enter the length you need.
And then choose the strength of the password you need.
A random password will be generated. You can copy it if you want.

Password Generated :

Length : 0

Choose Strength : [✓ Weak] Medium Strong

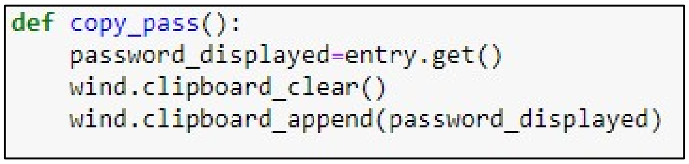
This password can be cracked in: 0 seconds.

Figure16: Final Layout of Application

c. Working behind the window

For the working of the application let's get the code for the copy button which is shown in Figure17. The code will be written inside the empty function created in Figure14 named as **copy_pass.**

The **get function** takes the value which is in the entry box at that time and stores it. The **clipboard_clear** function clears the previously saved things on clipboard and the **clipboard_append** takes the value which needs to be stored on the clipboard further until it is cleared again.

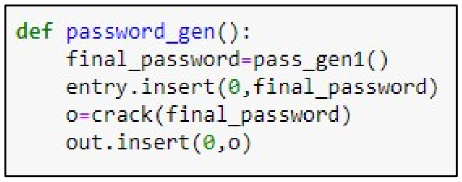


```
def copy_pass():
    password_displayed=entry.get()
    wind.clipboard_clear()
    wind.clipboard_append(password_displayed)
```

Figure17: Code for copying of the password

Next, we move on to the password generator function which needs to be written in the empty function named **password_gen** in Figure14.

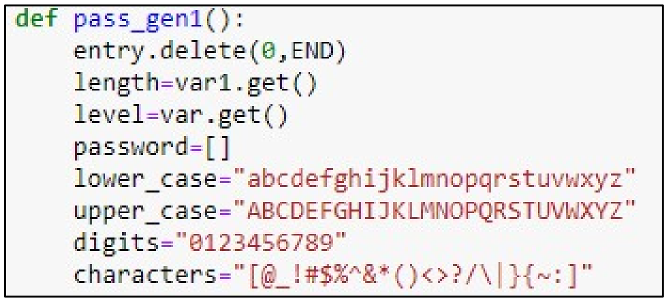
The code for generating the password is explained step by step in the next sections. When generate function is clicked it calls the function **password_gen** and in password_gen function we call the function **pass_gen1** and **crack** function where pass_gen1 function is used to decide the password and crack is used to calculate the hack time of the password generated. Now to insert the values into the respective places we use the **insert function** with the object name which also takes the starting index and the value to be inserted in the respective **entry boxes** on the screen of the application.



```
def password_gen():
    final_password=pass_gen1()
    entry.insert(0,final_password)
    o=crack(final_password)
    out.insert(0,o)
```

Figure18: password_gen function

Now moving ahead, we come to the pass_gen1 function. For the password generation we have **3 strength levels** namely weak, medium and strong. Also, we have **4 acceptable character categories** namely lowercase letters, uppercase letters, digits and special symbols. We create strings for all of them and we also take values from the length and the strength section given by user. Then according to the inputs, we design the password and return the password.



```
def pass_gen1():
    entry.delete(0,END)
    length=var1.get()
    level=var.get()
    password=[]
    lower_case="abcdefghijklmnopqrstuvwxyz"
    upper_case="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    digits="0123456789"
    characters="[@_!#$%^&*()<>?/\|}{~:]"
```

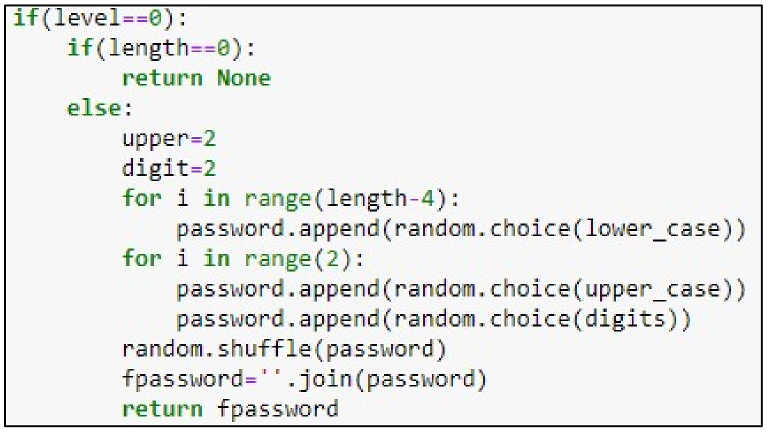
Figure19: Part1 of pass_gen1 function

Now we check the level value and create the password according to our choice the same is shown in Table1.

Table 1: Showing the information about the 3 levels of password

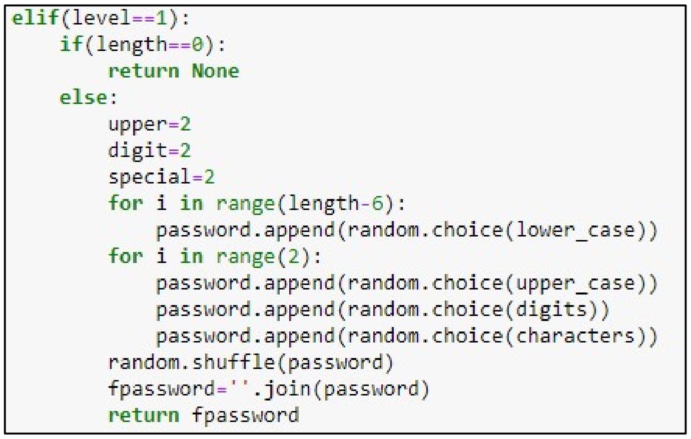
S.no.	Level	Level value	No. of upper case	No. of digits	No. of special characters	No. of lower case
1	Weak	0	2	2	0	Length-4
2	Medium	1	2	2	2	Length-6
3	Strong	2	4	4	2	Length-10

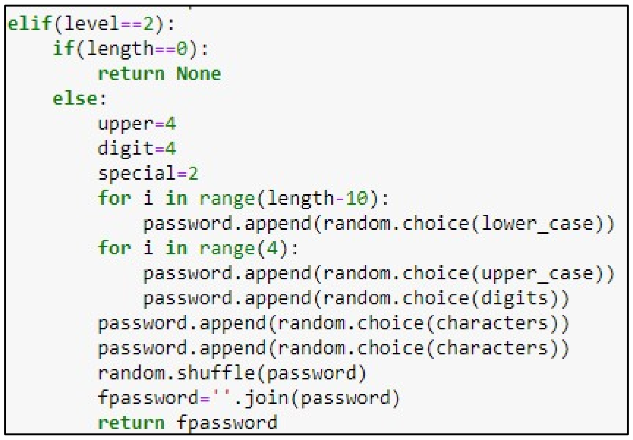
According to Table 1 the passwords are created for weak , medium and strong strength which is respectively shown in Figute20, Figure21 and Figure22.



```
if(level==0):
    if(length==0):
        return None
    else:
        upper=2
        digit=2
        for i in range(length-4):
            password.append(random.choice(lower_case))
        for i in range(2):
            password.append(random.choice(upper_case))
            password.append(random.choice(digits))
        random.shuffle(password)
        fpassword=''.join(password)
        return fpassword
```

Figure20: Password creation code for Weak password


```
elif(level==1):
    if(length==0):
        return None
    else:
        upper=2
        digit=2
        special=2
        for i in range(length-6):
            password.append(random.choice(lower_case))
        for i in range(2):
            password.append(random.choice(upper_case))
            password.append(random.choice(digits))
            password.append(random.choice(characters))
        random.shuffle(password)
        fpassword=''.join(password)
        return fpassword
```
Figure21: Password creation code for Medium password


```
elif(level==2):
    if(length==0):
        return None
    else:
        upper=4
        digit=4
        special=2
        for i in range(length-10):
            password.append(random.choice(lower_case))
        for i in range(4):
            password.append(random.choice(upper_case))
            password.append(random.choice(digits))
        password.append(random.choice(characters))
        password.append(random.choice(characters))
        random.shuffle(password)
        fpassword=''.join(password)
        return fpassword
```
Figure22: Password creation code for Strong password

Now after creating the password it gets returned to password_gen function and then inserted into the respective entry box.

Now the next function is the crack function which is helpful in calculating the hack time of the password. The formula used to calculate the hack time in the application is equation (1).

$$n^m/2000000000 \qquad (1)$$

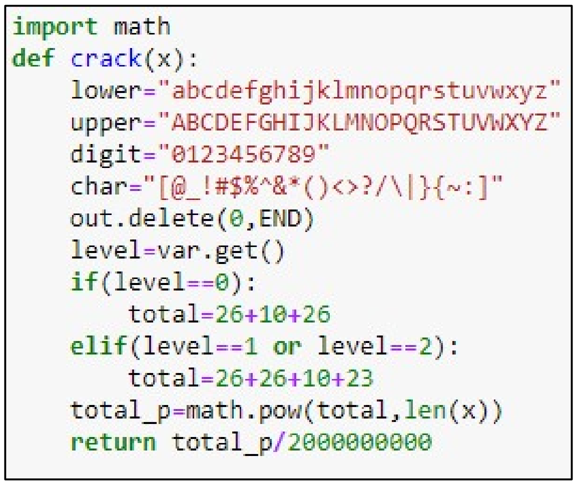
Where the value '2000000000' is total computation time taken by system. And the value n^m describes the total possible combination of my password.

Now the value m is total length of the password and n for each level is given in Table2.

Table2: The table showing value of n for each level

S.no.	Level (level value)	n for the value
1	Weak (0)	26+10+26=62
2	Medium (1)	26+10+26+23=85
3	Strong (2)	26+10+26+23=85

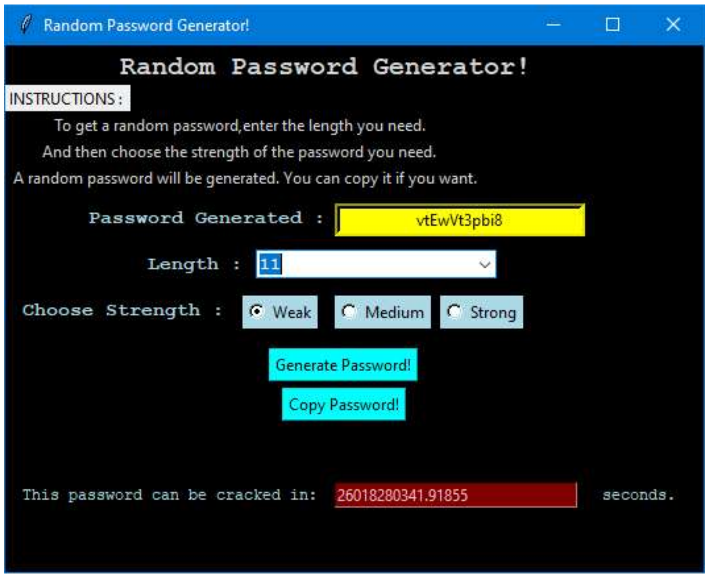
The code for the same calculation is shown in Figure23.



```
import math
def crack(x):
    lower="abcdefghijklmnopqrstuvwxyz"
    upper="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    digit="0123456789"
    char="[@_!#$%^&*()<>?/\|}{~:]"
    out.delete(0,END)
    level=var.get()
    if(level==0):
        total=26+10+26
    elif(level==1 or level==2):
        total=26+26+10+23
    total_p=math.pow(total,len(x))
    return total_p/2000000000
```

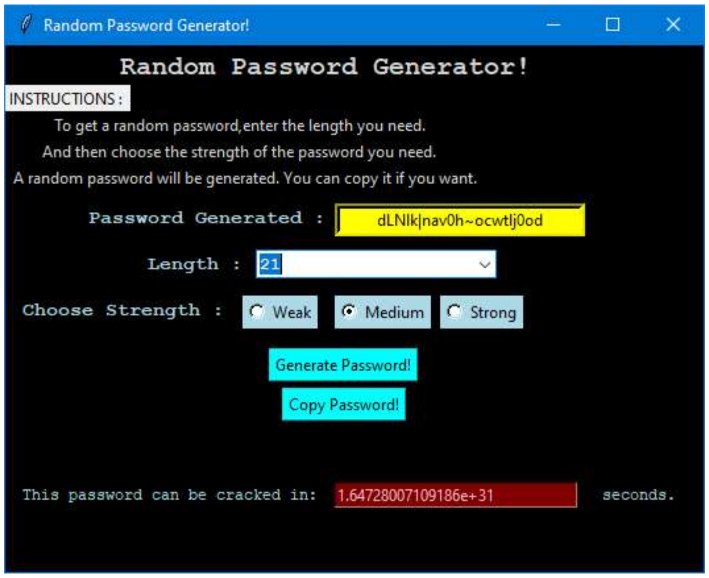
Figure23: Code for calculating the hack time of the password generated

Now the crack/hack time generated gets inserted at the respective position on the screen. The coding part is done. Figures24, Figure25 and Figure26 show the sample results of the application.



```
Random Password Generator!

Random Password Generator!
INSTRUCTIONS :
To get a random password,enter the length you need.
And then choose the strength of the password you need.
A random password will be generated. You can copy it if you want.

Password Generated : vtEwVt3pbi8
Length : 11
Choose Strength : [✓ Weak]  Medium  Strong

Generate Password!
Copy Password!

This password can be cracked in: 26018280341.91855 seconds.
```

Figure24: Sample Output 1



Random Password Generator!

Random Password Generator!

INSTRUCTIONS :

To get a random password,enter the length you need.

And then choose the strength of the password you need.

A random password will be generated. You can copy it if you want.

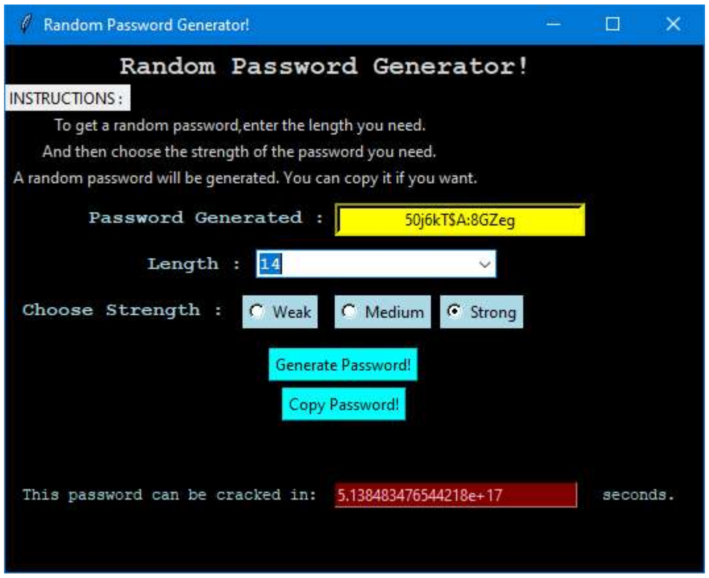
Password Generated : dLNIk|nav0h~ocwtlj0od

Length : 21

Choose Strength : Weak [✓ Medium] Strong

This password can be cracked in: 1.64728007109186e+31 seconds.

Figure25: Sample Output 2



Random Password Generator!

Random Password Generator!

INSTRUCTIONS :

To get a random password,enter the length you need.

And then choose the strength of the password you need.

A random password will be generated. You can copy it if you want.

Password Generated : 50j6kT$A:8GZeg

Length : 14

Choose Strength : Weak Medium [✓ Strong]

This password can be cracked in: 5.138483476544218e+17 seconds.

Figure26: Sample Output 3