(Password Generator and Saver)

Bachelors of Engineering

by

(Shaheema Shaikh)

(Roll No.31)

Guide:

Prof. Shiburaj Pappu

Electronics & Telecommunication Engineering Department

Rizvi College of Engineering

University of Mumbai

2020-2021

CERTIFICATE

This is to certify that the project entitled **“Password Generator and Saver”** is a bonafide work of “**Shaheema Shaikh” (Roll No.:31)** submitted in partial fulfillment of the requirement for the Mini-Project in **“Skill Based Lab : Python Programming”**.

Prof Shiburaj Pappu
(Name and sign)
Guide

ABSTRACT

The users of computer technology and internet are increasing day by day. Every person using different online services is concerned with the security and privacy for protecting individual information from the intruders. Many authentication systems are available for the protection of individuals' data, and the password authentication system is one of them. Due to the increment of information sharing, internet popularization, electronic commerce transactions, and data transferring, both password security and authenticity have become an essential and necessary subject. But it is also mandatory to ensure the strength of the password. For that reason, all cyber experts recommend intricate password patterns. But most of the time, the users forget their passwords because of those complicated patterns.In this project, we discuss a method of generating random passwords, which are strong enough to combat the attacks. This password will he based on the information, i.e. (some alphabets,special character and numbers) provided by the users. Also, the design of this system is pretty simple so that the user won’t get any difficulties while working on it. The password generator project will be build using python modules like Tkinter, random, string.

Keywords : Random password, Security, information,Strength,Strong.

Index

Sr. No	Title	Page No
1.	Introduction	1
2.	Review and Literature	2
	2.1. Paper 1	3
	2.2. Paper 2	4
3.	Theory, Methodology and Algorithm	5
	3.1 Section	6
	3.1.1. Subsection	7
4.	Results and Discussions	8
5.	Conclusion	9
6.	References	10
	Appendix	11
	Acknowledgement	12

Chapter 1
Introduction

Password is indispensable and inevitable one in today‟s communication process and provides security to user‟s data. Password is a sequence of character string used to authenticate personal identity of user and to provide or refuse the access to system resources. The password is not only denying any access to the system from unauthorized person, but also prevent users who are previously logged in from doing unauthorized process in system. Security risk from unauthorized entry involves more than the risk to a single user via their system account.User ID and password combination is the one of the simplest forms of user authentication.

Password is a secret word, which is used to authorize the user to particular system or particular application. The identity of the user is tested using the password. If the passwords are not strong or easily guessable, the intruders can attack the system and attack the data assets.Guessing attacks have had major business implications, such as a 2009 incident in which a vandal guessed a Twitter executive's password and was able to leak all of the company's internal documents .

Because of the scenario of widespread re-use of passwords across sites , an emerging attack model is to compromise accounts by a guessing attack against a low-security website and attempt to re-use the credentials at critical websites . Most organizations specify a password rules that form the requirements for the composition and usage of passwords.They are minimum length, required categories such as upper and lower case,numbers, and special characters, prohibited elements such as own name, D.O.B., address, telephone number. Some governments have national authentication framework that define requirements for user authentication to government services, including requirements for passwords.

In this project, we discuss a method of generating random passwords, which are strong enough to combat the attacks. This password will be based on the information, i.e. (some alphabets,special character and numbers) provided by the users. Also, the design of this system is pretty simple so that the user won't get any difficulties while working on it. The password generator project will be build using python modules like Tkinter, random, string.

Chapter 2
Review of Literature

2.1 paper1:

Entropy in software engineering is the haphazardness created by an application, with the end goal of cryptography. In software engineering, we consider entropy because of the accompanying reasons. Entropy is basic for PCs to recreate different physical, compound and organic marvel that are regularly irregular. Entropy is important for cryptography, and a few plan methods, e.g., inspecting and property testing. Despite the fact that proficient deterministic calculations are obscure for some issues, straightforward, exquisite and quick randomized calculations are known [3]. We need to know whether these randomized calculations can be DE randomized without losing proficiency. Irregularity is an effective device for scrambling delicate information. For example, the event that utilizes an irregular secret key generator 5, 10, 100 or 1000 times with similar parameters, there's a little shot that the generator will make a similar outcome twice since it should be arbitrary, which means the outcomes will be capricious, won't take after a set-design, and a past outcome will have no impact on any of the accompanying results. The outcome saw in such a situation are pseudo irregular, i.e., they are not generally arbitrary yet they seem, by all accounts, to be irregular. Conventional PC frameworks are not awesome at creating irregular outcomes. They are deterministic, which implies that on the off chance that we ask a similar inquiry we will find a similar solution without fail. Indeed, such machines are particularly and deliberately modified to dispense with irregularity in comes about. They do this by following standards and depending on calculations when they process. A very deterministic machine cannot create genuinely irregular number arrangements since it takes after a similar calculation to deliver its outcomes. Ordinarily, that implies it begins with a typical 'seed' number and after that takes after an example. Therefore the randomized passwords are not genuinely irregular since the same algorithm [4] made them.

2.2 Paper2:

Priti Jadhao and Lalit Dole [5] present a survey on authentication password techniques. They discussed about many techniques but do not ensure the security of a user, since there is a very high chance of a user, coming across a previously used password, thus making the password generators unreliable. Zhouetal. [6] discussed about one-time password generating method. N. Halleretal. [7] discussed about A One-Time Password System. In spite of the fact that being very arbitrary, pseudo-irregular secret key generators are not 100% dependable. They normally do not; however, can furnish us with a formerly utilized secret key, accordingly making the whole thought of an irregular watchword generator outdated. In this manner, the ideal approach to enhance the execution and security of pseudo-random generators is high entropy, i.e., eccentrics of assembled information utilized for cryptographic and security capacities. This is typically accomplished by utilizing cryptographically secure pseudo-arbitrary number generator. The entropy in a cryptographically Secure Pseudo-Random Number Generator is higher than a customary Pseudo-Random Number Generator, on the grounds that it is generally created by an eccentric wonder or a physical action, for example, organize action, hard drive action, console strokes, mouse developments, no. of dynamic procedures, time stamp, and so forth., i.e., everything that is continuously. This information is then used to make the 'seed' number, which makes our new irregular secret key. In entirety, the higher the entropy, the harder an arbitrary number or secret word ought to be to foresee. A. Nitinetal. [8] discussed a new technique for generating a complex password based on real components. They proposed a new technique based on entropy to generate the password. An entropy based password generator, will use components that change in real time, like time stamp, to give us a new password, in every single hit of a button.

Chapter 3

Report on the Present Investigation

3.1 Theory:-

So before starting the actual coding let’s first understand some theory. In this password generator GUI application, we are going to use python modules. Firstly, the Tkinter module for creating an application window. Secondly, messagebox module for dispalying the message. Now, remember these two modules are not inbuilt modules. So you have to install them using the pip install command. The third and final module we are going to use is a random module to generate the random password finally. The quick logic for this GUI application is going to be like this: Firstly, We will create import the necessary modules. Then we will create the application window. After that, we will generate a random password and Save it. Afterward, In the end, we will create some buttons to make our password generator application more interactive for the user.

3.1.1 Hardware Requirements:-

Operating system: windows 10 ,processor: intel core .

3.1.2 Software Requirements:-

Python IDE ,version above 3,Python libraries – Tkinter, messagebox, random, string.

and Code Editor.

3.2 Methodology:-

Random password generator is to produce random password with high security. Generally, random passwords have various benefits over user-chosen password where it enhances security and confidentiality. The new methodology has been created to generate random password which consists of both upper & lower case letter and digits from 0 to 9 and special characters. The is a simple algorithm that generates random password with predetermined length. The password generator algorithm selects a random character form random character list and forms the password, which is combination of numbers, lower & upper-case letters and special characters .

The random string method can also be called keyboard mashing. You can probably guess what the process entails.Just mash your fingers across the keyboard.

Here's what I get: hi"ePb&f9rg9(*gK.L.Now, that password is quite strong. It includes a random string, multiple letter cases, characters, and numbers, plus its 18 characters.

To build this project we will use the basic concept of python and libraries – Tkinter, messagebox, random, string.

3.2.1 Proposed technique consists of following phases:

1)User is asked to enter the URL for which user wants to generate the password.The second input from user is Username/Email.

2)Then User is asked to select the length of the password.Then user has to select the strength of the password whether the strength they wants in their password contain high level of security,medium level of security and low level of security.

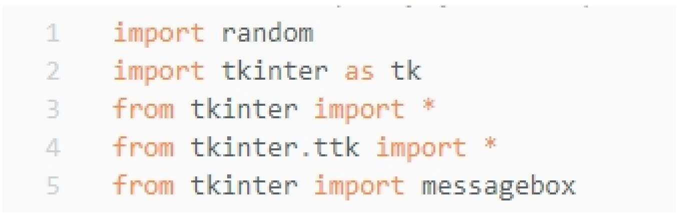
3)High level of secure password contains the combination of Upper-case letters,lower-case letters,digits and special characters.whereas the medium level of secure password contains the combination of upper-case, lower-case letters and digits and low level of security contains the lower-case letters and digits.

4)Then user has to click the generate button then password is successfully generated as per user requirements.then user can save their details in the text file on user folder by save button.

3.3 Algorithm:-

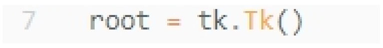
3.3.1 Steps To develop Random password generator and saver project.

Step1: Firstly, Import necessary modules (Tkinter, messagebox and random).The tkinter.ttk module provides access to the Tk themed widget.



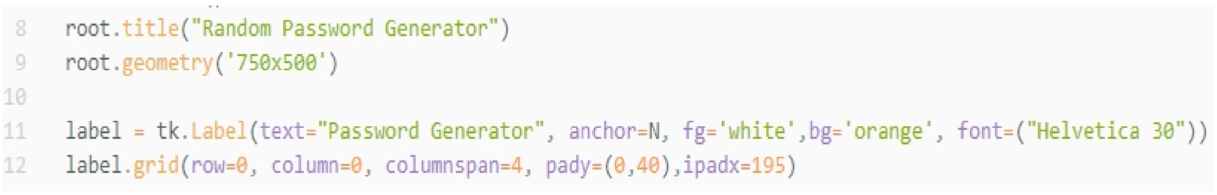
```python
import random
import tkinter as tk
from tkinter import *
from tkinter.ttk import *
from tkinter import messagebox
```

Step2: Then, Initialize Tkinter using Tk() method



```python
root = tk.Tk()
```

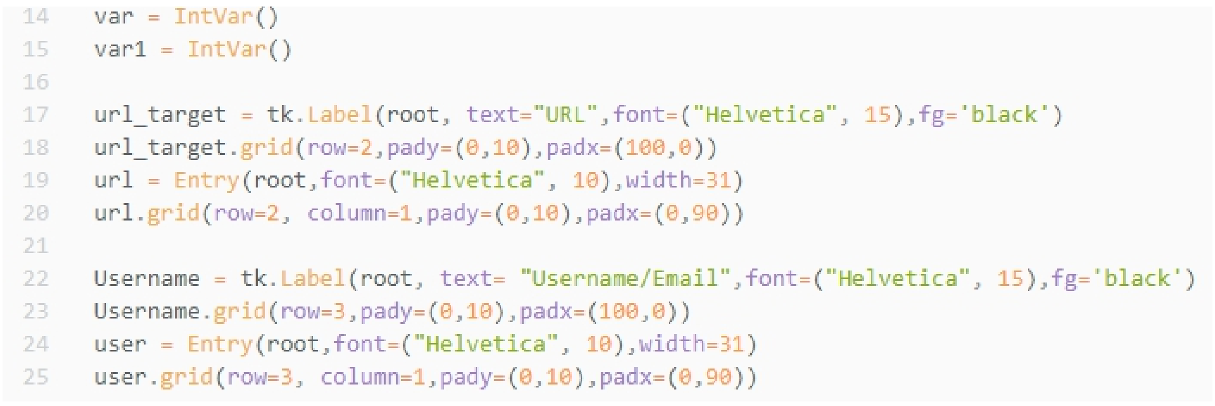
Step 3: Create a root window. Give the root window a title(using title()) and dimension(using geometry()). All other widgets will be inside the root window.



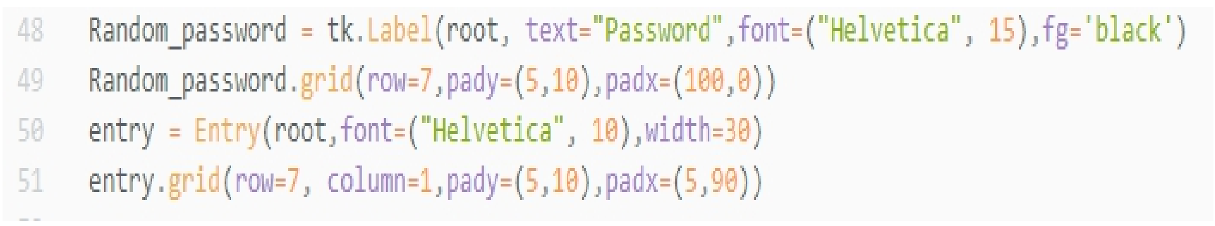
```python
root.title("Random Password Generator")
root.geometry('750x500')

label = tk.Label(text="Password Generator", anchor=N, fg='white',bg='orange', font=("Helvetica 30"))
label.grid(row=0, column=0, columnspan=4, pady=(0,40),ipadx=195)
```

Label() widget use to display one or more than one line of text that users can’t able to modify.
root is the name which we refer to our window
text which we display on the label
font in which the text is written

Step 4: Declare two variables for taking password length and strength and take inputs from user.Take Inputs Such as URL, Username/Email, Length and Strength of the password from user.

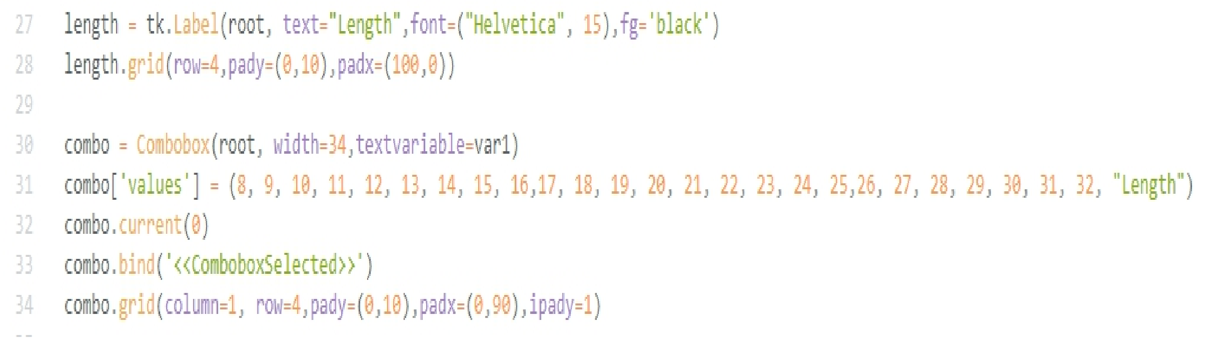


```
var = IntVar()
var1 = IntVar()

url_target = tk.Label(root, text="URL",font=("Helvetica", 15),fg='black')
url_target.grid(row=2,pady=(0,10),padx=(100,0))
url = Entry(root,font=("Helvetica", 10),width=31)
url.grid(row=2, column=1,pady=(0,10),padx=(0,90))

Username = tk.Label(root, text= "Username/Email",font=("Helvetica", 15),fg='black')
Username.grid(row=3,pady=(0,10),padx=(100,0))
user = Entry(root,font=("Helvetica", 10),width=31)
user.grid(row=3, column=1,pady=(0,10),padx=(0,90))
```



```
Random_password = tk.Label(root, text="Password",font=("Helvetica", 15),fg='black')
Random_password.grid(row=7,pady=(5,10),padx=(100,0))
entry = Entry(root,font=("Helvetica", 10),width=30)
entry.grid(row=7, column=1,pady=(5,10),padx=(5,90))
```

Entry() widget used to create an input text field.
textvariable used to retrieve the current text to the entry widget.



```
length = tk.Label(root, text="Length",font=("Helvetica", 15),fg='black')
length.grid(row=4,pady=(0,10),padx=(100,0))

combo = Combobox(root, width=34,textvariable=var1)
combo['values'] = (8, 9, 10, 11, 12, 13, 14, 15, 16,17, 18, 19, 20, 21, 22, 23, 24, 25,26, 27, 28, 29, 30, 31, 32, "Length")
combo.current(0)
combo.bind('<<ComboboxSelected>>')
combo.grid(column=1, row=4,pady=(0,10),padx=(0,90),ipady=1)
```

To select the password length we use Combobox() widget.
Combobox() widget is used to select from a fixed number of values. Here the value from 8 to 32.

```python
Strength = tk.Label(root, text="Strength",font=("Helvetica", 15),fg='black')
Strength.grid(row=5,column=0,padx=(90,55),sticky='e')

style = Style(root)
style.configure("TRadiobutton",foreground = "black", font = ("Helvetica", 15))
radio_low = Radiobutton(root, text="Low", variable=var, value=1)
radio_low.grid(row=5,column=1,columnspan=3,padx=(0,360))
radio_middle = Radiobutton(root, text="Medium", variable=var, value=3)
radio_middle.grid(row=5, column=1,padx=(0,110))
radio_strong = Radiobutton(root, text="Strong", variable=var, value=0)
radio_strong.grid(row=5, column=1,padx=(90,25))
```

Use Radio Buttons for deciding the strength of password.Default strength is Strong. Radiobuttons can contain text and you can associate a Python function or method with each button. When the button is pressed, Tkinter automatically calls that function or method.

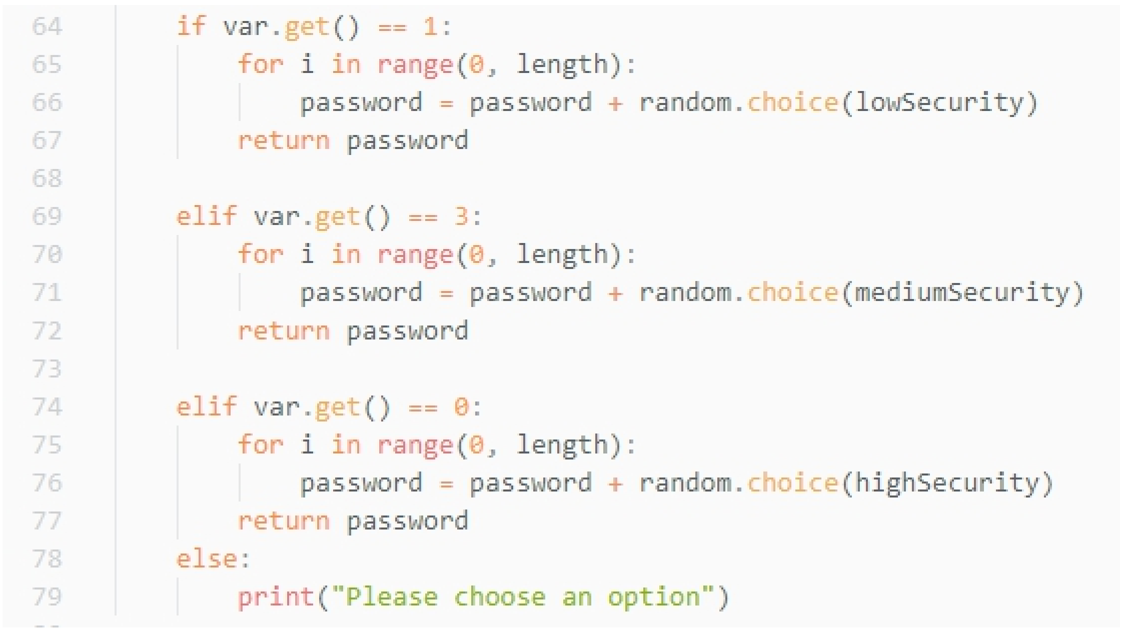
Step5: Define the low function for calculation of password then add .delete to remove the last entry.Then create a string of upper-case letters , lower-case letters,special characters and number for high security strength similarly string of upper-case letters , lower-case letters and number for medium security strength and string of lower-case letters and number for low security strength .

```python
def low():
    entry.delete(0, END)

    length = var1.get()

    lowSecurity = "abcdefghijklmnopqrstuvwxyz0123456789"
    mediumSecurity = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789"
    highSecurity = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$%^&*()"
```

Step 6: Create an empty list and assign it to the password variable.

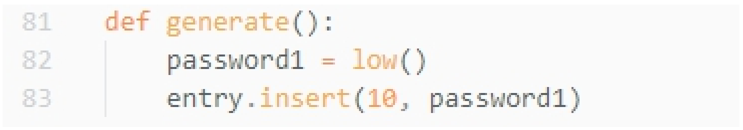
```python
    password = ""
```

Step 7: Use if elif loop for selected strength of the password .Suppose the password length selected by user is 8 to make sure that it runs enough times to generate 8 characters, a for loop is used inside if loop to tell it that it needs to run 8 times.The i represents the number of times that it needs to run based on the information from the variable length. Use the random.choice() method to returns any random predefined character from the lowSecurity or highSecurity or mediumSecurity variable.



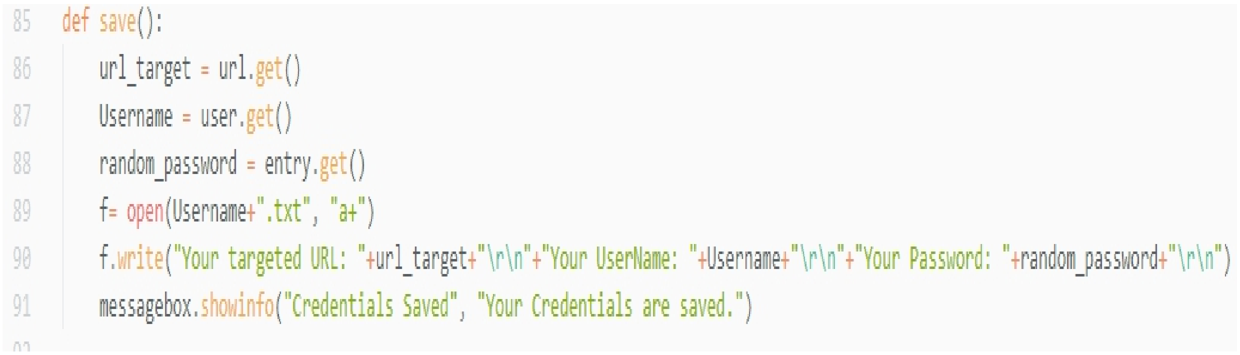
```python
    if var.get() == 1:
        for i in range(0, length):
            password = password + random.choice(lowSecurity)
        return password

    elif var.get() == 3:
        for i in range(0, length):
            password = password + random.choice(mediumSecurity)
        return password

    elif var.get() == 0:
        for i in range(0, length):
            password = password + random.choice(highSecurity)
        return password
    else:
        print("Please choose an option")
```

Step 8: Define generate Function for generation of password add .insert to show the generated password.



```python
def generate():
    password1 = low()
    entry.insert(10, password1)
```

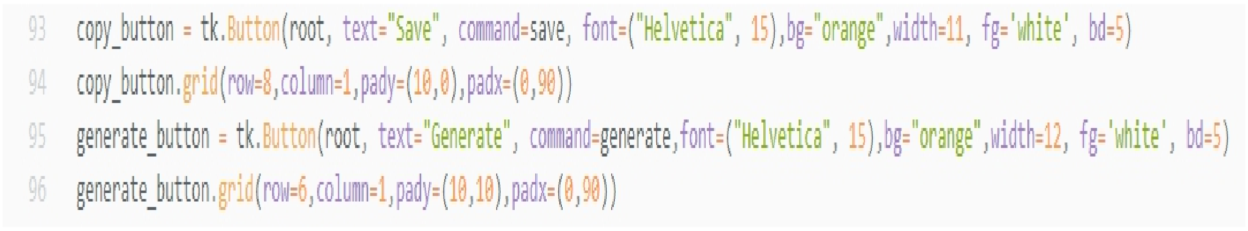
Step 9: Define Save method to save the generated password and other information entered by the user in your computer as a text file.First,we use the .get() method to get the information then we open the file using the open() method.a+ Opens a file for both appending and reading.then we write() the file and append information in it. \r\n is used to create a new line.Use messagebox to show the message that your information has been saved successfully.



```
def save():
    url_target = url.get()
    Username = user.get()
    random_password = entry.get()
    f= open(Username+".txt", "a+")
    f.write("Your targeted URL: "+url_target+"\r\n"+"Your UserName: "+Username+"\r\n"+"Your Password: "+random_password+"\r\n")
    messagebox.showinfo("Credentials Saved", "Your Credentials are saved.")
```

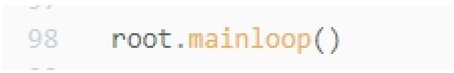
.

Step10: Then, Create Button widgets and set commands as per the above defined methods.



```
copy_button = tk.Button(root, text="Save", command=save, font=("Helvetica", 15),bg="orange",width=11, fg='white', bd=5)
copy_button.grid(row=8,column=1,pady=(10,0),padx=(0,90))
generate_button = tk.Button(root, text="Generate", command=generate,font=("Helvetica", 15),bg="orange",width=12, fg='white', bd=5)
generate_button.grid(row=6,column=1,pady=(10,10),padx=(0,90))
```

Step11: Finally, Run infinite mainloop to run the application.



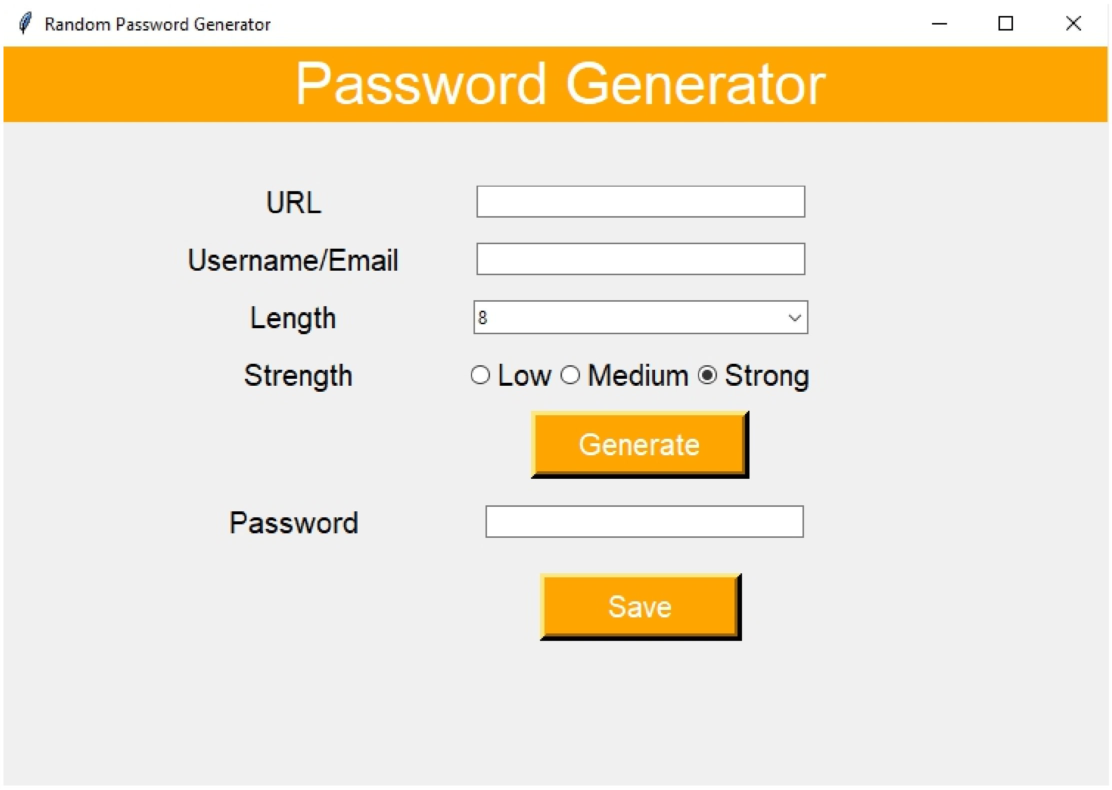
```
root.mainloop()
```

Chapter 4

Results and Discussions

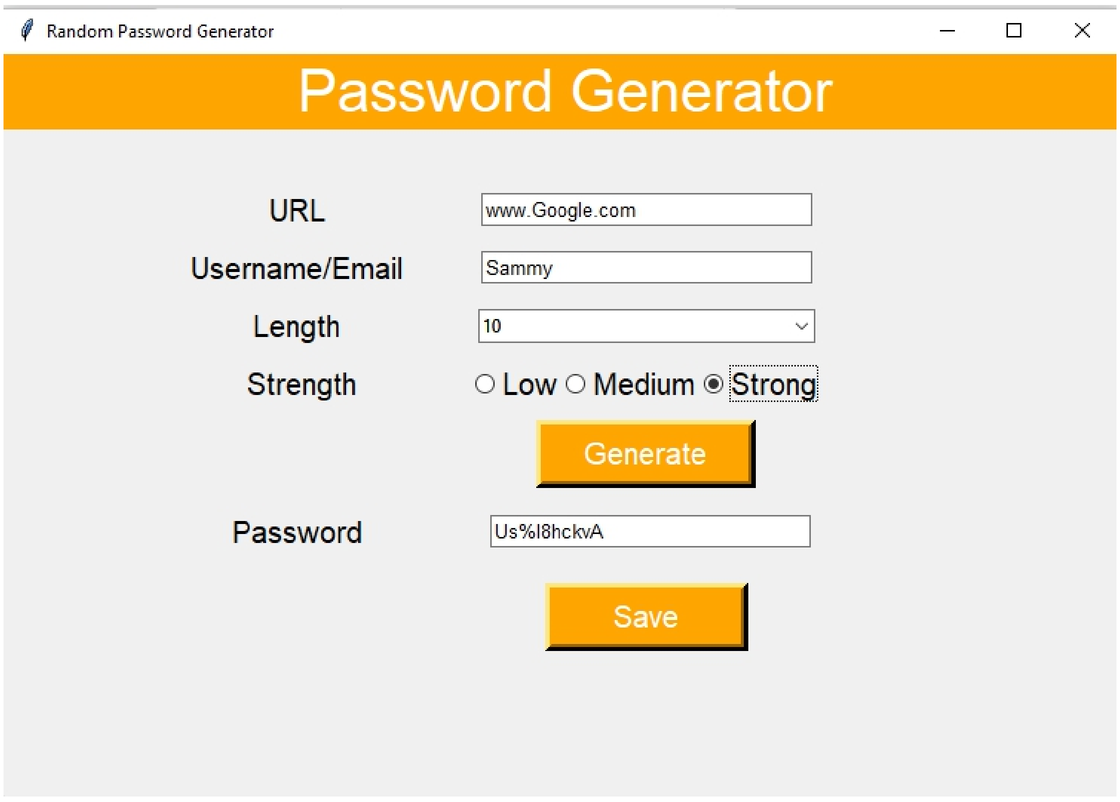
4.1Results:-

Step1:Run the code ,following window will appear in which user has to enter the URL,Username/Email and select the length of the password you want then select the strength of your password.



URL	
Username/Email	
Length	8
Strength	Low Medium [✓ Strong]
Password	

Step2:After Entering all the details click on the generate button your password is now generated Successfully.


Random Password Generator

Password Generator

URL: www.Google.com

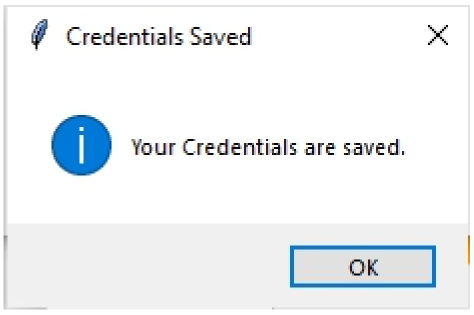
Username/Email: Sammy

Length: 10

Strength: Low ( ) Medium ( ) [✓ Strong]

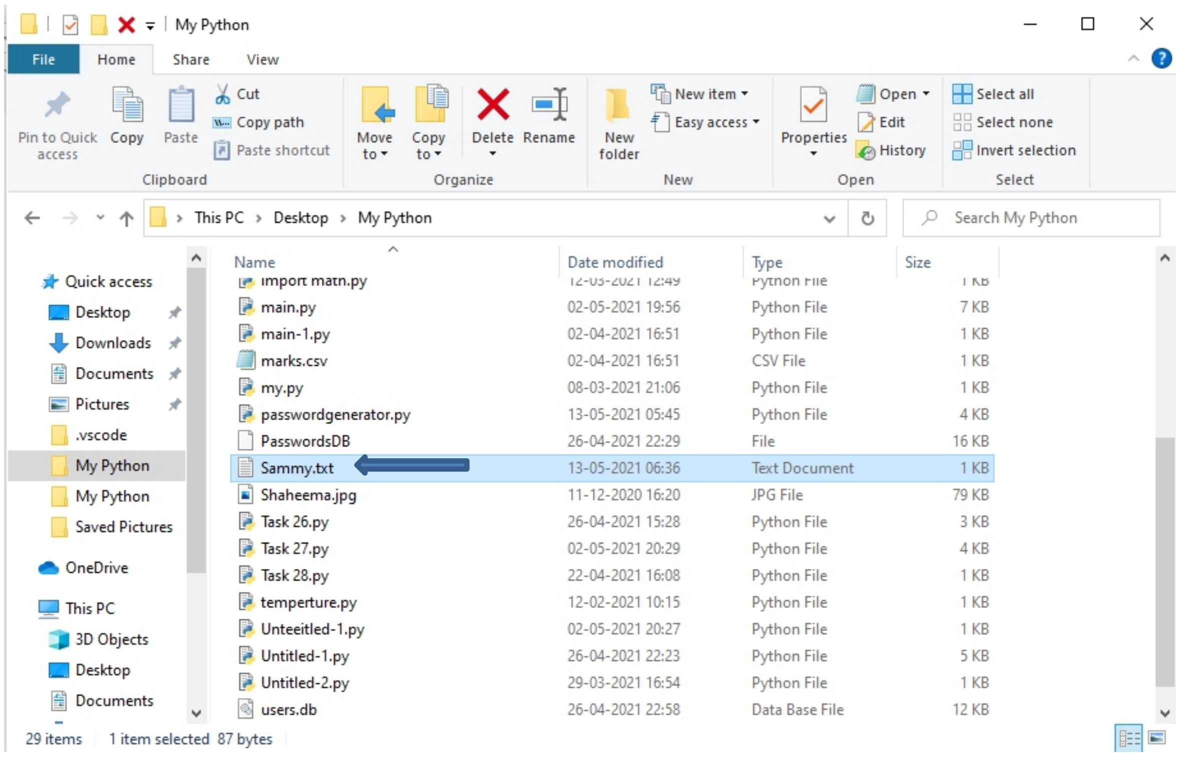
Password: Us%l8hckvA

Step3:Click on the Save button to save your details and then messagebox is generated to show you that your details is saved Successfully then press OK.


Credentials Saved

Your Credentials are saved.

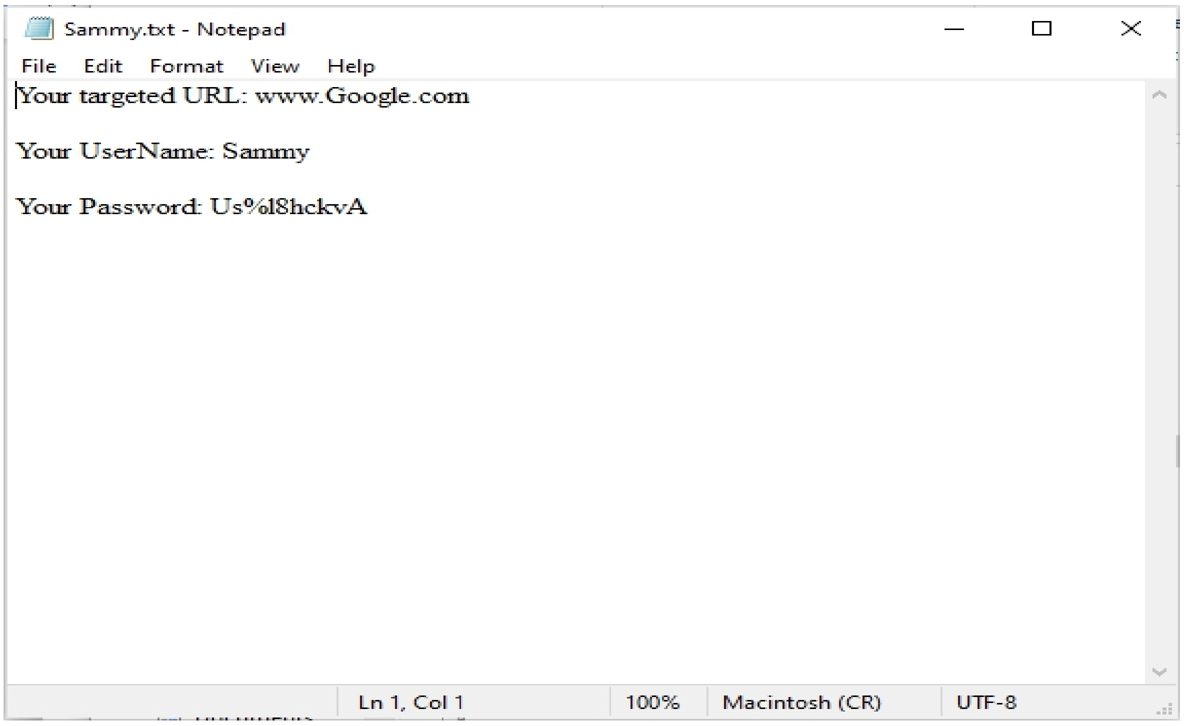
Step4:Go to your File Explorer and open your python folder now search for the text file by your Username/Email.



Name	Date modified	Type	Size
import math.py	12-05-2021 12:49	Python File	1 KB
main.py	02-05-2021 19:56	Python File	7 KB
main-1.py	02-04-2021 16:51	Python File	1 KB
marks.csv	02-04-2021 16:51	CSV File	1 KB
my.py	08-03-2021 21:06	Python File	1 KB
passwordgenerator.py	13-05-2021 05:45	Python File	4 KB
PasswordsDB	26-04-2021 22:29	File	16 KB
Sammy.txt	13-05-2021 06:36	Text Document	1 KB
Shaheema.jpg	11-12-2020 16:20	JPG File	79 KB
Task 26.py	26-04-2021 15:28	Python File	3 KB
Task 27.py	02-05-2021 20:29	Python File	4 KB
Task 28.py	22-04-2021 16:08	Python File	1 KB
temperture.py	12-02-2021 10:15	Python File	1 KB
Unteeitled-1.py	02-05-2021 20:27	Python File	1 KB
Untitled-1.py	26-04-2021 22:23	Python File	5 KB
Untitled-2.py	29-03-2021 16:54	Python File	1 KB
users.db	26-04-2021 22:58	Data Base File	12 KB

29 items 1 item selected 87 bytes

Step5:Open your file whenever you need to check your password.



```
Your targeted URL: www.Google.com

Your UserName: Sammy

Your Password: Us%l8hckvA
```

4.2Discussion:-

With these steps, we have successfully created a random password generator and saver project using python. We used popular tkinter library to rendering graphics in our display window and we also learned about random library.We learned how to create buttons, input textfield, labels, and messagebox. In this way, we successfully created our password generator and saver python project.Since all the applications are protected with passwords, more research can be accomplished for secured automatic password generations. The proposed method uses only the alphabets and numerical values and special symbols for random character string.The password length also can be extended to make the password strong. New encryption and decryption standard could be implemented with the randomly selected passwords.Cryptography technique can also be implemented on future work.For future work, machine learning algorithms can be added to the technique that would create sentence out of the passwords that is generated. It is always easier to remember passwords that resemble a sentence. Say, for example, password APR4$ can be associated with the sentence All People Run For (4) Money ($).The experimental study can be done with large number of samples in future.

Chapter 5

Conclusions

The password generated using random password mechanism that was illustrated above is practical and can be used with great results. When the password is selected manually, most of the time, the users select the password that are related to himself or herself and related to any of the event. This gives the space for the intruders to deploy various attacks in breaking the passwords. The random generated passwords avoid this particular situation. One of the drawbacks could be the difficulty in memorizing the randomly generated password.To Overcome this drawbacks we introduce the method of Saving the password. When comparing the security achieved through the randomly generated password, it is much preferable than the manually chosen password.The above done work also creates awareness and interest to start exploring this field more.

Appendix

Appendix I:

Common guidelines for creating a good password

Guidelines for choosing good passwords are typically designed to make passwords harder to discover by intelligent guessing. Common guidelines advocated by proponents of software system security include:

- Use a minimum password length of 20 or more characters if permitted.
- Include lowercase and uppercase alphabetic characters, numbers and symbols if Permitted.
- Generate passwords randomly where feasible.
- Avoid using the same password twice (e.g., across multiple user accounts and/or software systems).
- Avoid character repetition, keyboard patterns, dictionary words, letter or number sequences.
- Avoid using information that is or might become publicly associated with the user or the account, such as username, ancestors' names or dates.
- Avoid using information that the user's colleagues and/or acquaintances might know to be associated with the user, such as relative or pet names, romantic links (current or past) and biographical information (e.g., ID numbers, ancestors' names or dates)..
- Do not use passwords which consist wholly of any simple combination of the aforementioned weak components.

Some guidelines advise against writing passwords down, while others, noting the large numbers of password protected systems users must access, encourage writing down passwords as long as the written password lists are kept in a safe place, not attached to a monitor or in an unlocked desk drawer.Use of a password manager is recommended by the NCSC.

The possible character set for a password can be constrained by different web sites or by the range of keyboards on which the password must be entered.

Appendix II:

Type and strength of password generated

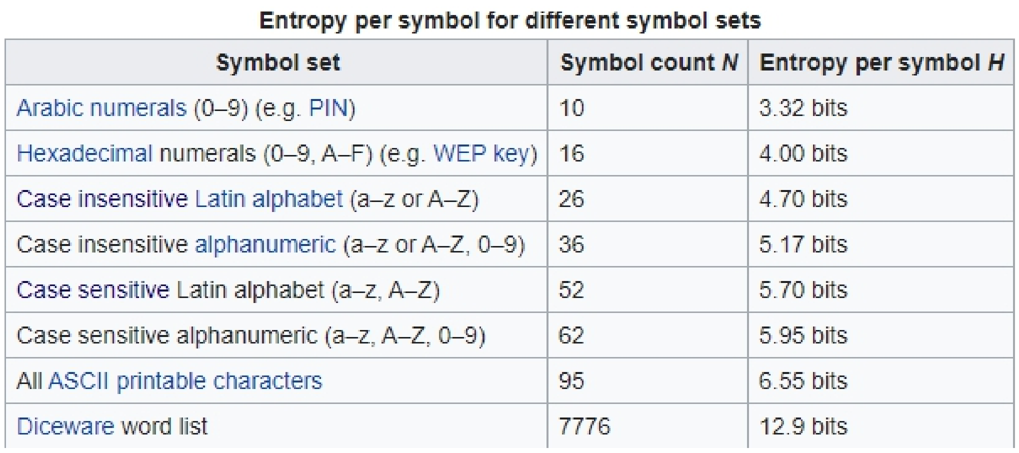
Password strength is a measure of the effectiveness of a password against guessing or brute-force attacks. In its usual form, it estimates how many trials an attacker who does not have direct access to the password would need, on average, to guess it correctly. The strength of a password is a function of length, complexity, and unpredictability.

Random password generators normally output a string of symbols of specified length. These can be individual characters from some character set, syllables designed to form pronounceable passwords, or words from some word list to form a passphrase. The program can be customized to ensure the resulting password complies with the local password policy, say by always producing a mix of letters, numbers and special characters. Such policies typically reduce strength slightly below the formula that follows, because symbols are no longer independently produced.

The Password strength of a random password against a particular attack (brute-force search), can be calculated by computing the information entropy of the random process that produced it. If each symbol in the password is produced independently and with uniform probability, the entropy in bits is given by the formula

$$H = \log_2 N^L = L \log_2 N = L \frac{\log N}{\log 2}$$

where N is the number of possible symbols and L is the number of symbols in the password. The function $\log_2$ is the base-2 logarithm. H is typically measured in bits.



Entropy per symbol for different symbol sets

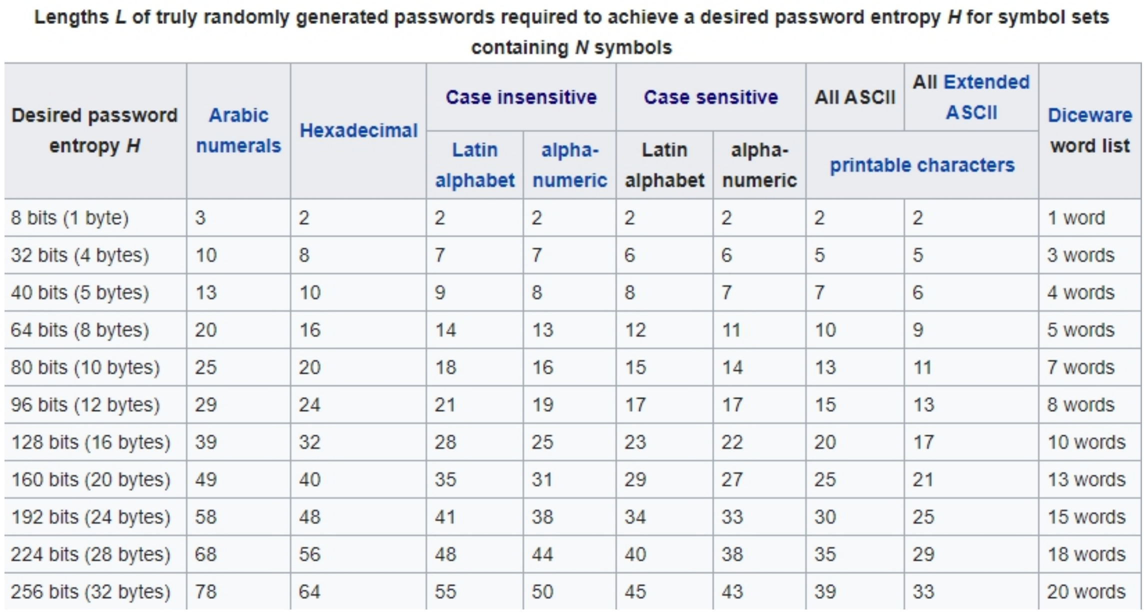
Symbol set	Symbol count N	Entropy per symbol H
Arabic numerals (0–9) (e.g. PIN)	10	3.32 bits
Hexadecimal numerals (0–9, A–F) (e.g. WEP key)	16	4.00 bits
Case insensitive Latin alphabet (a–z or A–Z)	26	4.70 bits
Case insensitive alphanumeric (a–z or A–Z, 0–9)	36	5.17 bits
Case sensitive Latin alphabet (a–z, A–Z)	52	5.70 bits
Case sensitive alphanumeric (a–z, A–Z, 0–9)	62	5.95 bits
All ASCII printable characters	95	6.55 bits
Diceware word list	7776	12.9 bits

A binary byte is usually expressed using two hexadecimal characters.To find the length, *L,* needed to achieve a desired strength *H,* with a password drawn randomly from a set of *N* symbols, one computes:

$$L = \left\lceil \frac{H}{\log_2 N} \right\rceil$$

Where[ ] denotes rounding up to the next largest whole number.

The following table uses this formula to show the required lengths of truly randomly generated passwords to achieve desired password entropies for common symbol sets:


Lengths *L* of truly randomly generated passwords required to achieve a desired password entropy *H* for symbol sets containing *N* symbols

Desired password entropy *H*	Arabic numerals	Hexadecimal	Case insensitive		Case sensitive		All ASCII	All Extended ASCII	Diceware word list
			Latin alphabet	alpha-numeric	Latin alphabet	alpha-numeric	printable characters		
8 bits (1 byte)	3	2	2	2	2	2	2	2	1 word
32 bits (4 bytes)	10	8	7	7	6	6	5	5	3 words
40 bits (5 bytes)	13	10	9	8	8	7	7	6	4 words
64 bits (8 bytes)	20	16	14	13	12	11	10	9	5 words
80 bits (10 bytes)	25	20	18	16	15	14	13	11	7 words
96 bits (12 bytes)	29	24	21	19	17	17	15	13	8 words
128 bits (16 bytes)	39	32	28	25	23	22	20	17	10 words
160 bits (20 bytes)	49	40	35	31	29	27	25	21	13 words
192 bits (24 bytes)	58	48	41	38	34	33	30	25	15 words
224 bits (28 bytes)	68	56	48	44	40	38	35	29	18 words
256 bits (32 bytes)	78	64	55	50	45	43	39	33	20 words

Chapter 6
References

[1]Art Conklin, Glenn Dietrich, Diane Walz, “Password-Based Authentication: A System Perspective”, Proceedings of the 37th Hawaii International Conference on System Sciences – 2004.

[2]NikCubrilovic, The Anatomy of the Twitter Attack, TechCrunch, July 2009.

[3] Thorsten Brantz and Alex Franz,The Google Web 1T 5-gram corpus, Technical Report LDC2006T13, Linguistic Data Consortium, 2006.

[4] Mike Bond, Comments on authentication, www.cl.cam.ac.uk/~mkb23/ research/GridsureComments.pdf, 2008.

[5] Manoj Kumar Singh,” Password Based a Generalize Robust Security System Design using Neural Network”, IJCSI-International Journal of Computer Science Issues, Vol. 4, No. 2, 2009.

[6] Michael D. Leonhard, V. N. Venkatakrishnan, “A Comparative Study of Three Random Password Generators”, IEEE EIT 2007 Proceedings.

[7] Ayushi, “A Symmetric Key Cryptographic Algorithm”, International Journal of Computer Applications, Volume 1 – No. 15, 2010.

[8] Kamini H. Solanki, Chandni R. Patel, “New Symmetric Key Cryptographic algorithm for Enhancing Security of Data”, International Journal of Research in Computer Engineering and Electronics, volume 1, issue 3,Dec 2012.

[9]Google, YouTube.

Acknowledgment

I have great pleasure in presenting the report on **"Password Generator and Saver"**. I take this opportunity to express my sincere thanks to all those who have taken keen interest in directing my efforts towards a successful completion of the report work. At the outset, I wish to register my deepest sense of gratitude and respect to my guide **Prof. Shiburaj Pappu** for their constant support and guidance throughout the duration of the project. The critical analysis and timely suggestions have helped me in coming out with this fruitful result. I also wish to register my feelings towards my family and friends who have been so encouraging and understanding. Lastly, I would like to thank Mumbai University and Rizvi College Of Engineering for providing us with the opportunity of using our knowledge and skills to help in the development of technology and to help us increase our experience.

Date: 10 / 05 / 2021 Shaheema Shaikh