**DE LA SALLE UNIVERSITY - MANILA**

**PCGM: Password Checker, Generator, and Manager**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

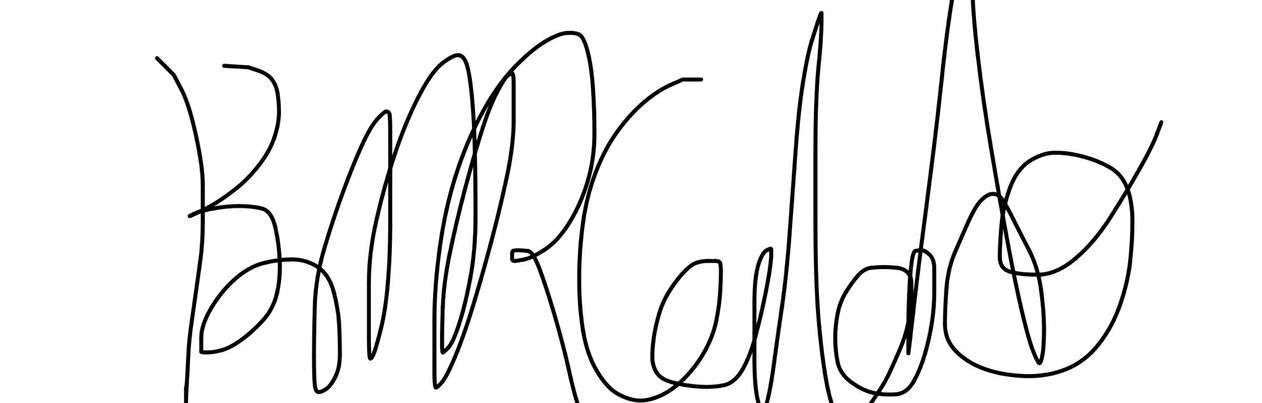
A Term Project

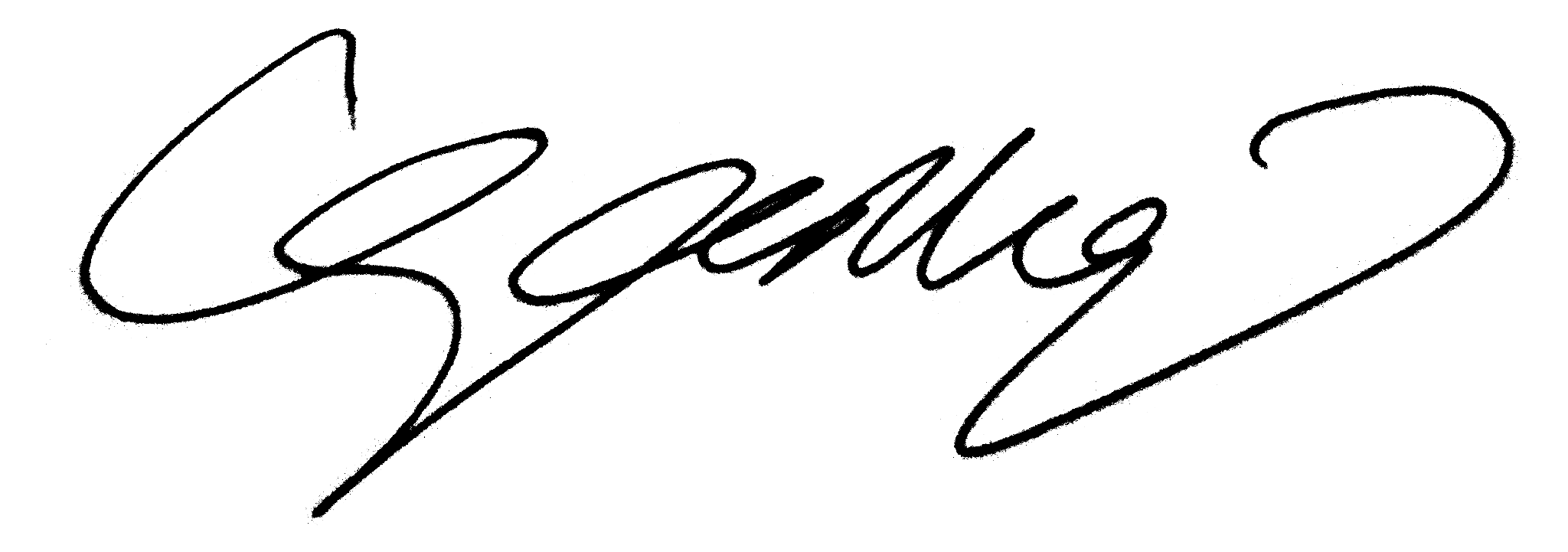
Presented to Mr. Ramon Stephen L. Ruiz

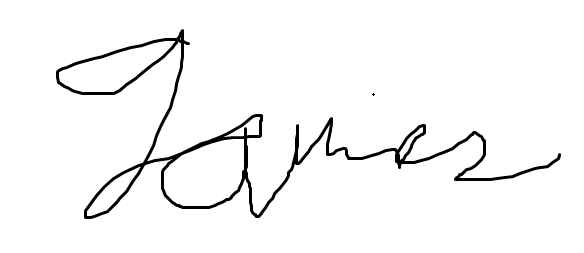
In Partial Fulfillment of the

Requirements for the Course Programming Logic and Design (PROLOGI)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

by

CALADO, Brent Mitchel R. -

SANTIAGO, Ezekiel Inigo P. - 

TEPACE, James Louis C. -

EQ3

Monday (F2F) 12:45PM - 1:45PM, Thursday (Online) 12:45PM - 1:45PM

April, 2023

# Table of Contents

**Introduction** 2

Background of the Study 3

Problem Statement 3

Objectives 5

General Objectives 5

Specific Objectives 6

Significance of the Project 6

**Review of Related Literature** 7

**Methodology** 8

Conceptual Framework 8

Hierarchy Chart 10

Flowchart 12

Pseudocode 13

**Results** 19

**Discussion of the Results** 20

**Analysis, Conclusion, and Future Directives** 20

**References** 21

**Appendices** 23

Appendix A - User’s Manual 23

Appendix B - Source Code 24

Appendix C - Work Breakdown 29

Appendix D - Personal Data Sheet 30

**List of Tables & Figures**

Table 1.1 - IPO Table 8

Figure 1.1 - Hierarchy Chart 10

Figure 2.1 - Flowchart 12

Figure 3.1 - Results 19

# Introduction

Passwords are one of the cornerstones and most crucial elements of network security and cybersecurity. Ever since the establishment of the internet, passwords have been present in almost every online site. Although, along with the existence of passwords also comes the threat of breaking and cracking them. Hence, throughout the years, experts, scientists, programmers, and even companies have attempted to take safety measures to improve password security. Some have even successfully abandoned password mechanisms and moved on to biometrics, which uses new technologies for identification such as touch ID, voice recognition, and facial recognition. Though this is the case, many are still dependent on the utilization of passwords since they are easy and efficient enough to use.

To address the problems that password security has, password-checking systems and applications were created and invented by numerous experts and programmers. These applications use various criteria such as use of uppercase and lowercase letters, numbers, the presence of symbols, and password length. The applications then base the strength of passwords through these criteria and categorizes them as either week, moderate, or strong.

Other than password-checking, the field of network security has also made strides in research and development into password-generating applications. These applications are congruent with password-checking systems. Basically, password-generators apply the use of computers or machines to create new and strong passwords. In other words, these systems output computer-generated defenses. Tech companies such as Apple implement such kind of technology into their own products, like the iPhone and the iPad.

Password-management technologies have also emerged as a source of solutions for breaches in cybersecurity. These systems also cover the risky possibility of human error, as people might be able to forget their passwords after some time or if there are too many passwords to memorize. What these applications do is store the credentials or files in a system or device storage for later use.

Essentially, with the above statements in mind, we aim to create a tool or application that unifies and implements all these technologies – password checking, password generating, and password managing. Through this research, as well as the creation of the tool, we hope to contribute to ensuring that network security measures are kept at the utmost level.

## Background of the Study

A particularly secure password is important as it acts as your sensitive information’s main line of defense against unauthorized access by cybercriminals. In 2019, 80% of all data breaches were caused by compromised passwords alone, resulting in financial losses for both businesses and consumers themselves (Alkhatib, 2022). According to a 1000-sample survey by Lord (2020), around 56% of all respondents reported using complex passwords that use a mixture of properly cased letters, numbers, and special characters (e.g. “iLOVEf00tball!”). However, there are still around 38% that use simple passwords (e.g. “Football1”) and the remaining 7% that use no security at all (e.g. “football”). Using passwords with sufficient complexity exponentially increases the time it takes for them to be cracked. A 20-character password of such would take a computer around 3 sextillion years to correctly guess it. That value is equivalent to a 3 followed by 21 zeros, which is a very long time. Nevertheless, a 20-character password will also be a challenge for the user themselves. It is simply impossible for someone to remember each of the different 20-character passwords they use for every site they visit. Despite being only used by 28% of the respondents, password managers remain to be the best help to overcome this issue. These are applications that generate strong unique passwords for each of your credentials, while some even have the capacity to store these passwords securely within themselves for ease of access (Key, 2021).

## Problem Statement

Network Security is one of the most important yet fragile aspects of modern society and technology. The ideal for this field is that the conversation, protection, and management of information are kept at a high level. Although, despite the field evolving over the years because of the continued advancement of technology, the reality is that there are still some problems that need to be resolved when it comes to protecting data and information on the web. Many fields have resorted to big data processing technology in practical applications, making enterprise decisions more reasonable and scientific, and playing the role of data information, but at the same time also exposes computer networks to many security threats (Luo, 2020). One of these problems encompasses the most fundamental part of network security – password protection.

A password is a secret word or phrase that must be used to gain access to websites. Generally, a password should consist of something that is hard to guess, so that it will remain a secret (Kurunthachalam & Karthiga, 2013). It is without a doubt that passwords, despite the presence of new technology biometrics such as touch ID and facial recognition, are still the most commonly used identity authentication measure. However, with the advent of the information age, the number of websites has been increasing, and the databases of these websites have been exposed to the Internet. Driven by huge benefits, passwords have become the target of hackers, and password-cracking incidents have become increasingly frequent (He et al, 2020). Moreover, studies have shown that the percentage in recent years, specifically in 2020, of password-cracking is 68.63%, and the average guess time was 51.99 seconds (He et al, 2020). Because of this, despite numerous studies and attempts to safeguard online information, there still exists a contemporary and demanding problem with regard to password security.

Moreover, other than the threat of password intrusion, there exists yet another problem with password security – password management. In online identity authentication, a small amount of human error is also involved in the efficacy and quality of password management. In current society, it has become a noteworthy common trend among online users to forget or misplace their passwords to websites and applications. Research has found that, in 2023 alone, 73% of people reset at least one password every 90 days because they forgot what the password was, and 3 out of every 4 people find managing their passwords annoying and frustrating (Todorov, 2023).

Almost one thing is certain, the ideal for network security and password protection and management is still not yet met. The reality is that there are still some gaps in knowledge as well as in application when it comes to password security. As such, network intrusion and inefficient password management have become detrimental consequences. These call for a need of a recheck or restrengthening of password checking and security at the most fundamental level. Through this study, the researchers hope to contribute to that notion by implementing a program, created using the programming language Python, that checks the strength of a password in order to avoid outside access and intrusion, create a computer-generated password to replace weak identification credentials, and store the inputted credentials and/or generated passwords into a text file in the user’s device storage.

## Objectives

## C.1 General Objective

People often decide on passwords that have some resemblance or relation to their lives. So counterintuitively, information such as addresses, birthdates, names, or common words are more frequently found in passwords. This is a reason why computer-generated passwords are much more secure than human-generated passwords. Computers randomly choose between pre-defined characters so there is truly no way for a person to guess one without the use of a computer running algorithms specifically for the purpose of cracking passwords.

The general objective of this password manager is to help provide users with a convenient way for them to manage their online accounts by allowing them to generate their passwords instead of creating them themselves, which can be a grave source of security risk.

This project also encourages the use of password requirements. Research by Dinei and Cormac (2007), found that users employ much weaker passwords to websites that enforce lax rules and those that they deem unimportant, than to websites that strictly require a password format and those that they give importance to. According to them, an example of a user-perceived unimportant website is the New York Times, a newspaper company, while an example of an important website is Paypal, an online payment company.

Password managers aim to provide these kinds of users with an easily accessible platform to create strong passwords and store their account information, enhancing the security of their online browsing experience without having to go through the tiring process by themselves.

## C.2 Specific Objectives

Password strength evaluation - The program is able to read password inputs from the user, where it can detect the absence of the usual requirements of a strong password. It is will display these absences in order for the user to edit their password to be acceptable. These requirements include uppercase, lowercase, digits, and symbols.

Password generation - The program is also able to randomly generate a password within the desired length of the user. It uses all requirement aforementioned; lowercase and uppercase letters, digits, and punctuations, among the characters it can use in generation.

Password storage - The program securely stores the generated passwords by hashing using the SHA-256 algorithm. It is a process of transforming the password into a random long string of characters. Over other types of encryption methods, hashes provide security by irreversibly encrypting these passwords. Once a password is hashed, you cannot change it back (Chanda).

## Significance of the Project

The features of this program are beneficial to people seeking more security in their online accounts. They essentially comprise the basic functions and are the main ideas behind the modern commercial password managers that you see today. These password managers revolve around doing these three things: evaluate your current passwords, suggest new ones, and store them securely. It is a continuous arms race in this field as they develop more features and strengthen their security. Some integrate other services such as VPN and cloud storage, while others were designed for business purposes (Bassili & Henges, 2022).

# Review of Related Literature

A study by Chanda (2016) discusses what goes into making a password strong. It compared multiple formats of human-generated passwords such as purely alphabetical, alphanumerical, double-cased passwords, and more, to determine how they compare against each other in how fast they are able to be cracked using brute force attacks. The study also took into account often overlooked parameters such as entropy and how it ties into password strength. Entropy in this context refers to how often characters are repeated in a password, with passwords such as ‘BBBBBBBBBB’ being easily guessed despite having a significant length.

A large-scale study done by Microsoft researchers, Dinei and Cormac (2007) observed the web password habits of over half a million users over the span of three months. It allowed them to derive a better understanding and estimation of how the average person uses passwords. Specific information acquired by the study includes the number of accounts, the number of passwords, how often they are forgotten, and most importantly, how they are used. Detailed evaluation on password strength and password usage gave insight into how little people gave importance to their online security habits. Unfortunately, no similar study of the same scale has been done recently.

A thesis by Mahesh (2018) proved that even security-literate individuals fall victim to not adhering to proper password security. The majority of respondents rated their most secure passwords to be sufficiently secure, meaning that they believe their passwords were strong enough to withstand a brute-force attack and adhere to the recommended conditions of having the right combination of numbers, letters, special characters, and length. However, when it comes to password habits, the majority also reported using just one or two passwords across all accounts, a habit widely considered to be unsafe. The research found that the issue in password insecurity does not lie in creating a secure password, but in remembering them altogether.

Three different papers written by Hu (2018), Dell’Amico, Pietro, & Roudier (2010), and Pereira, Ferreira, & Mendes (2020), provide an overview of the accuracy of password strength checkers. Passwords that satisfy these password strength checkers, also known as strength meters (PSMs), are tested against multiple guessing attacks. The paper by Pereira et. al. also takes into account how users respond to the feedback and suggestions displayed by these strength meters.

# Methodology

The project was planned for 7 days, developed for around 20 days, and documented for 4, overall taking a span of around a month to complete. GitHub and Trello pages were set up during the first week of development, as well as feature brainstorming and assignment. For over two more weeks, coding was done in Python. An initial draft was first developed to get the general idea of the program, with the final product being the same but with GUI improvements. For documentation, both versions were uploaded to Github along with the other documentation requirements of the project. The methodology is further explained in the following:

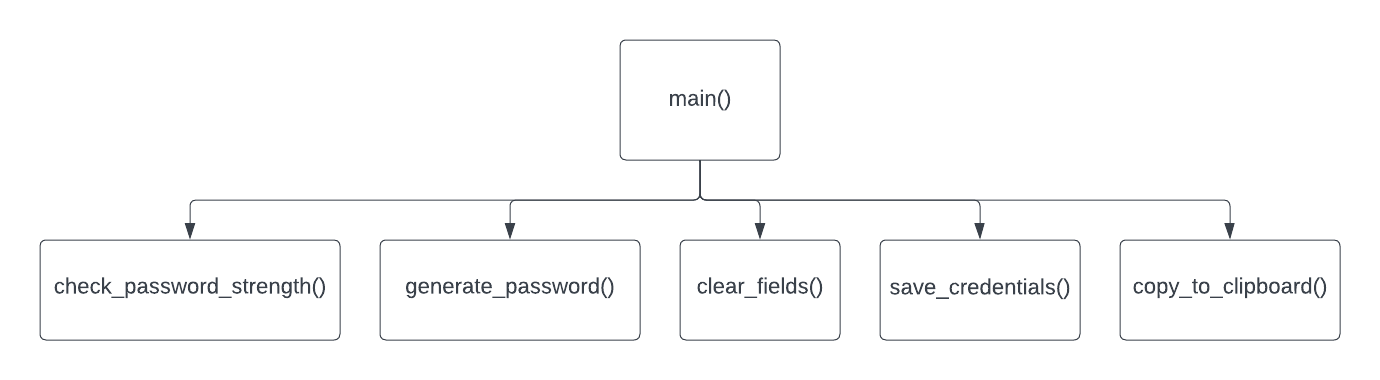
## Conceptual Framework – IPO Chart (Input-Process-Output-Chart)

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| Website/Application  Username/Email Address/Phone Number  Password  Password Length **(Optional)**  File Name **(Optional)** | Checks the strength of the inputted password based on the following criteria:  **If len(password) < 8:**  Display "Weak password: too short"  **Else If len(password) > 32:**  Display "Weak password: too long"  **Else If not any(char.isdigit() for char in password):**  Display "Weak password: no digits”  **Else If not any(char.isupper() for char in password):**  Display "Weak password: no uppercase letters"  **Else If not any(char.islower() for char in password):**  Display "Weak password: no lowercase letters"  **Else If not any(char in string.punctuation for char in password):**  Display "Weak password: no special characters"  **Else**  Display "Strong password!"  Generates a new strong password should the user want to. The generation of the password involves the use of a random yet strong and strategized algorithm (**Optional).**  Saves the credentials (Website/Application, Username/Email Address/Phone Number, Password) to a file that the user named **(Optional).** | Password strength result  Newly generated password **(Optional)**  Saved file **(Optional)** |

**Table 1.1 - IPO Table**

Table 1.1 demonstrates the basic overall process of the project. It can be determined that the user would ask for the following inputs: the Website/Application, the Username/Email Address/Phone Number, and the Password – these three are essential in the application of the project, since the project mainly revolves around the credentials, specifically the password, that the user will input in the program. Once the inputs have been received, the program will then move on to the process. In the process phase, the program will check for the strength of the password that the user inputted into the program. It will then use different criteria to determine if the password is strong or weak. Moreover, there are two optional fields that the user may or may not use in the program – the Password Length and the File Name. The password length is only used when the user wishes to let the program generate the password. Likewise, the file name is also only implemented when the user wants to save the credentials into a file in the user’s respective device. Lastly, with the processes and the inputs in mind, the expected outcomes are the password strength result and the optional newly generated password and saved file.

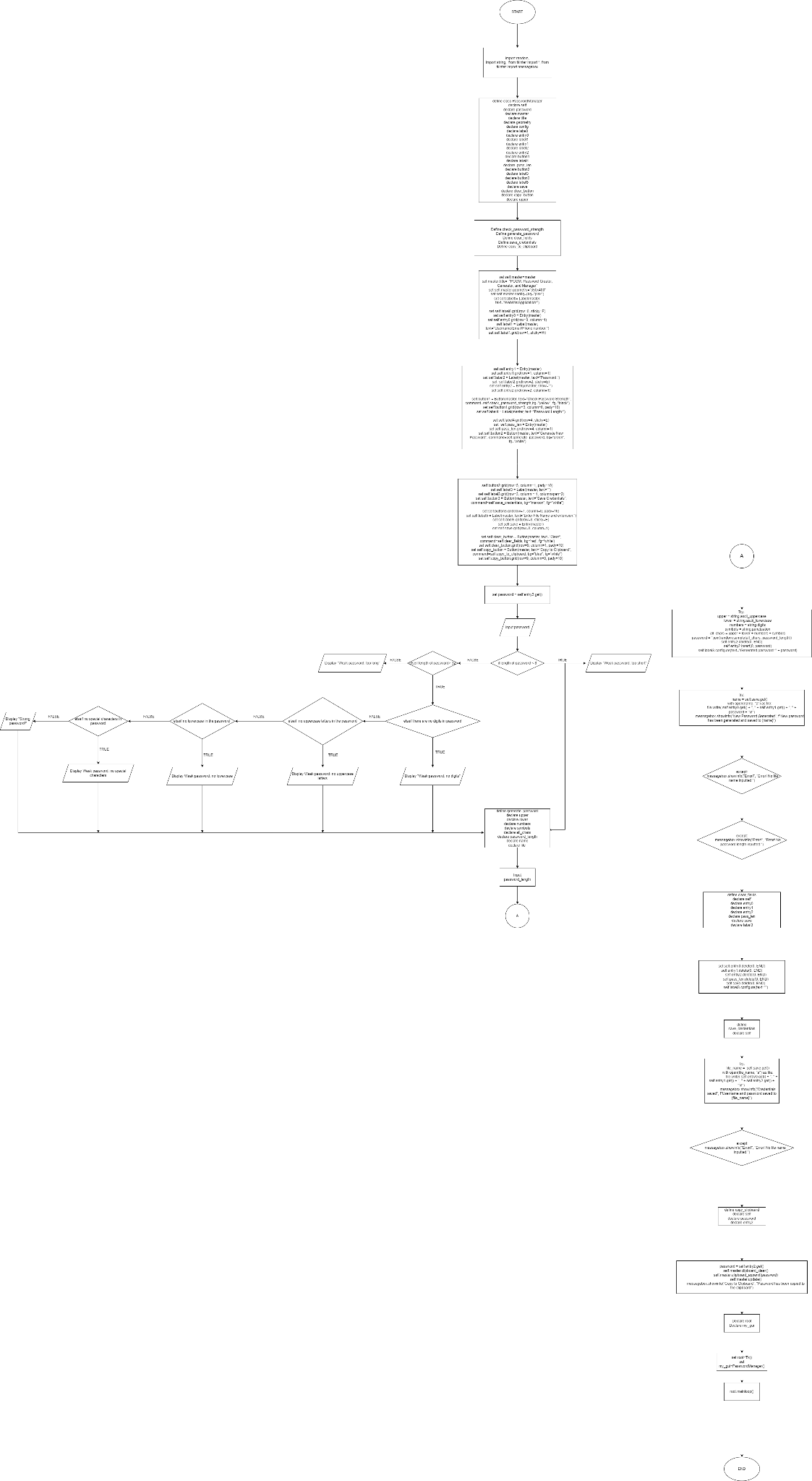
## Hierarchy Chart



**Figure 1.1 - Hierarchy Chart**

The hierarchy chart of the program provides an organized view of the functions within it. This hierarchy in particular makes use of only five different functions; check\_password\_strength(), generate\_password(), clear\_fields(), save\_credentials(), and copy\_to\_clipboard(). The check\_password\_strength() function takes care of the password strength evaluation. It receives the password and returns any missing requirements for a strong password. The generate\_password() function takes care of the random password generation. It receives an integer denoting the length of the password to be generated, and using the import random module returns a generated password of different characters. The clear\_fields() function is simply used to clear all the input fields of the program. The save\_credentials() function receives all the current values entered in the input fields, and exports them to an external file indicated by the user. The copy\_to\_clipboard() function simply uses the Tkinter module to copy the password field to the clipboard. These functions are all found inside one file along with the main module and they essentially help the whole program maintain organization.

## Flowchart

The flowchart for the program/project is too large to be properly seen in this documentation. Hence, a google drive link is provided for easier navigation and readability: <https://drive.google.com/file/d/14bIDxLnG_vdsb9p1b1kYxpQpQnfkQbDO/view?usp=sharing>

**Figure 2.1 - Flowchart**

## Pseudocode

**// Importing of modules**

import random

import string

from tkinter import \*

from tkinter import messagebox

Module main()

Set self.master = master

Set master.title("PCGM: Password Creator, Generator, and Manager")

Set self.master.geometry("350x400")

Set self.master.config(bg="pink")

**// Create labels and entry widgets for website/application, username/email/phone number and password**

Set self.label0 = Label(master, text="Website/Application:")

Display self.label0.grid(row=0, sticky=E)

Set self.entry0 = Entry(master)

Display self.entry0.grid(row=0, column=1)

Set self.label1 = Label(master, text="Username/Email/Phone number:")

Display self.label1.grid(row=1, sticky=W)

Set self.entry1 = Entry(master)

Display self.entry1.grid(row=1, column=1)

Set self.label2 = Label(master, text="Password:")

Display self.label2.grid(row=2, sticky=E)

Set self.entry2 = Entry(master, show="")

Display self.entry2.grid(row=2, column=1)

**// Create button for checking password strength and generating a new password**

Set self.button1 = Button(master, text="Check Password Strength", command=self.check\_password\_strength,bg="yellow", fg="black")

Display self.button1.grid(row=3, column=0, pady=10)

Call check\_password\_strength(self)

Set self.label4 = Label(master, text="Password Length:")

Display self.label4.grid(row=4, sticky=E)

Set self.pass\_len = Entry(master)

Display self.pass\_len.grid(row=4, column=1)

Set self.button2 = Button(master, text="Generate New Password", command=self.generate\_password, bg="green", fg="white")

Display self.button2.grid(row=5, column=1, pady=10)

Call generate\_password(self)

**// Create label for displaying password strength and feedback**

Set self.label3 = Label(master, text="")

Display self.label3.grid(row=3, column = 1, columnspan=2)

**// Button to save login credentials and password to a text file**

Set self.button3 = Button(master, text="Save Credentials", command=self.save\_credentials, bg="maroon", fg="white")

Display self.button3.grid(row=7, column=0, pady=10)

Set self.label5 = Label(master, text="Enter File Name and extension:")

Display self.label5.grid(row=8, sticky=E)

Set self.save = Entry(master)

Display self.save.grid(row=8, column=1)

Call save\_credentials(self)

**// Create button for clearing the entry fields**

Set self.clear\_button = Button(master, text="Clear", command=self.clear\_fields, bg='red', fg='white')

Display self.clear\_button.grid(row=9, column=1, pady=10)

Call clear\_fields(self)

**// Create button for copying the password to the clipboard**

Set self.copy\_button = Button(master, text="Copy to Clipboard", command=self.copy\_to\_clipboard, bg="blue", fg="white")

Display self.copy\_button.grid(row=9, column=0, pady=10)

Call copy\_to\_clipboard(self)

End Module

**// Module that checks the strength of a password**

Module check\_password\_strength(self)

Set password = self.entry2.get()

If len(password) < 8 Then

Display self.label3.configure(text="Weak password: too short")

Else If len(password) > 32 Then

Display self.label3.configure(text="Weak password: too long")

Else If not any(char.isdigit() for char in password) Then

Display self.label3.configure(text="Weak password: no digits")

Else If not any(char.isupper() for char in password) Then

Display self.label3.configure(text="Weak password: no uppercase letters")

Else If not any(char.islower() for char in password) Then

Display self.label3.configure(text="Weak password: no lowercase letters")

Else If not any(char in string.punctuation for char in password) Then

Display self.label3.configure(text="Weak password: no special characters")

Else

Display self.label3.configure(text="Strong password!")

End If

End Module

**// Module that generates a new password**

Module generate\_password(self)

Try

Set upper = string.ascii\_uppercase

Set lower = string.ascii\_lowercase

Set numbers = string.digits

Set symbols = string.punctuation

Set all\_chars = upper + lower + numbers + symbols

Set password\_length = int(self.pass\_len.get())

Set password = ''.join(random.sample(all\_chars, password\_length))

Set self.entry2.delete(0, END)

Set self.entry2.insert(0, password)

Display self.label3.configure(text="Generated password: " + password)

**// Save the login credentials and password to a file**

Try

Set name = self.save.get()

With open(name, "a") as file:

file.write( self.entry0.get() + ", " + self.entry1.get() + ", " + password + "\n")

Display messagebox.showinfo("New Password Generated", f"New password has been generated and saved to {name}")

**// Raise error if no file name is inputted**

Except

Display messagebox.showinfo("Error!", "Error! No file name inputted.")

**// Raise error if no password length is inputted**

Except

Display messagebox.showinfo("Error!", "Error! No password length inputted.")

End Module

**// Module that clears the fields**

Module clear\_fields(self)

Set self.entry0.delete(0, END)

Set self.entry1.delete(0, END)

Set self.entry2.delete(0, END)

Set self.pass\_len.delete(0, END)

Set self.save.delete(0, END)

Set self.label3.configure(text="")

End Module

**// Module that saves the credentials to a new file**

Module save\_credentials(self)

Try

Set file\_name = self.save.get()

With open(file\_name, "a") as file:

file.write( self.entry0.get() + ", " + self.entry1.get() + ", " + self.entry2.get() + "\n")

Display messagebox.showinfo("Credentials saved", f"Username and password saved to {file\_name}")

**// Raise error if no file name is inputted**

Except

Display messagebox.showinfo("Error!", "Error! No file name inputted.")

End Module

**// Module that copies the password to the clipboard of the user**

Module copy\_to\_clipboard(self):

Set password = self.entry2.get()

Set self.master.clipboard\_clear()

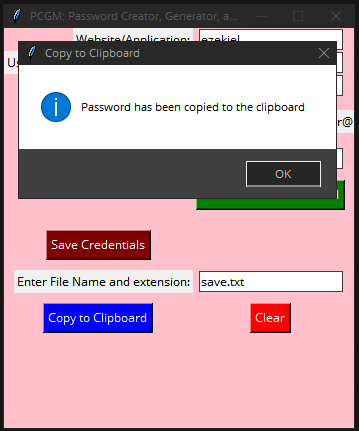
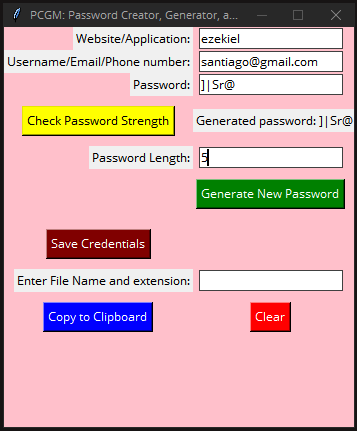
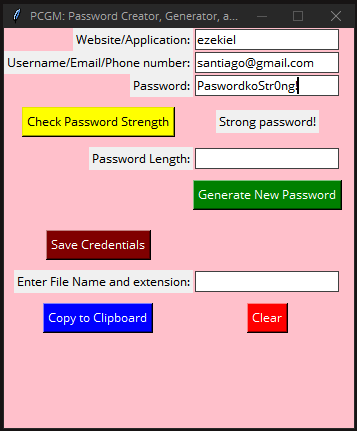
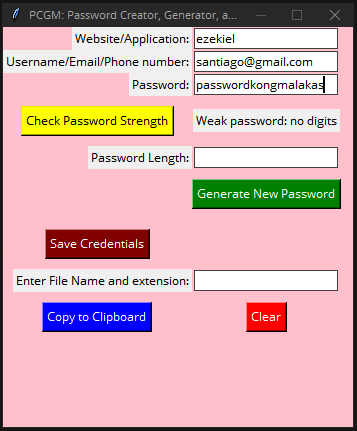
Set self.master.clipboard\_append(password)

Set self.master.update()

Display messagebox.showinfo("Copy to Clipboard", "Password has been copied to the clipboard")

End Module

# Results



**Figure 3.1 - Results**

# Discussion of Results

Simply run the code using your preferred Python compiler or interpreter. The program should be able to receive inputs from the user. These inputs are the website or application the credentials will be used for, the username or email address, and the password. Using the ‘check password strength’ button evaluates the inputted password and determines its strength and recommends changes. The program also allows generating of a random password based on the user’s set length. After all information is entered, the program also allows them to be saved according to a user’s set file name and extension. Finally, for convenience, the program also allows copying generated passwords and clearing all information.

# Analysis, Conclusion and Future Directives

It is undeniable that this project still has its own flaws. Certain limitations proved some implementations to be too challenging to be applied satisfactorily. The initial draft of the project contained most of what was intended but many needed to be removed in the final revision due to constraints.

One of the most important flaws of the project was proper security. Both the draft and final did not offer any sort of security when saving passwords. Passwords are only stored in a text file by the user. However, there are many available ways to secure passwords. Hashing is one of the most commonly used methods for encrypting them. Hashing is the process of converting data into random characters that cannot be converted back to their original string. However, it is already very well known to easily be cracked by generating hashes and finding one that matches. Modern computers of today are now able to use processes such as rainbow tables and brute force, which used to take a very long time, but can now be used efficiently (Chanda, 2016).

Both versions of the project also tried to offer an easily accessible and visually appealing interface, but due to the lack of experience, it may appear to be lacking for some. The draft initially featured a password strength meter to give users a clear indication of the strength of their passwords, but the program, as a whole, was limited to displaying information to the console or terminal. The revision fixed this issue by using a built-in Python module called Tkinter that displays a proper graphical user interface for better ease of access.

The program was also designed with multiple user support in mind. The draft achieved that feature to some extent, but it still lacked the security and privacy aspects between users. Due to the aforementioned lack of encryption, accessing other users’ information was just a matter of finding their saved text file in the file explorer.

There were other program features that were cut from the final product due to constraints, namely; the use of a master key to access all of your credentials and the ability to edit and remove existing passwords. Overall, the project was simply a proof of concept in Python. Improving upon these features and the general user experience will make it a more valuable tool for password management.

# References

Alkhatib, B. (2022, September 14). *Why is Password Security Important?* Keeper Security Blog - Cybersecurity News & Product Updates. <https://www.keepersecurity.com/blog/2022/09/14/why-is-password-security-important/>

Bassili, A., & Henges, E. (2022, March 17). The Best Password Managers of 2023. *How-To Geek.* <https://www.howtogeek.com/780233/best-password-manager/>

Dell’Amico, M., Michiardi, P., &#38; Roudier, Y. (2010, April 19). Password Strength: An Empirical Analysis. *Institute of Electrical and Electronics Engineers.* <https://www.researchgate.net/publication/224136922_Password_Strength_An_Empirical_Analysis>

Dinei, F., & Cormac, H. (2007). A Large-Scale Study of Web Password Habits. *Microsoft*. <https://rist.tech.cornell.edu/6431papers/FlorencioHerley2007.pdf>

Hu, G. (2018, June 1). On Password Strength: A Survey and Analysis. Unknown. <https://www.researchgate.net/publication/318154948_On_Password_Strength_A_Survey_and_Analysis>

He, S., Fu, J., Chen, C., & Guo, Z. (2020). Research on Password Cracking Technology Based on Improved Transformer. *Journal of Physics*. <https://doi.org/10.1088/1742-6596/1631/1/012161>

Hu, G. (2017). On Password Strength: A Survey and Analysis. In Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (pp. 165–186). *Springer International Publishing.* <https://doi.org/10.1007/978-3-319-62048-0_12>

Key, K. (2021, December 13). *Why You Need a Password Manager, and How to Choose the Right One.* PCMag. <https://www.pcmag.com/how-to/why-you-need-a-password-manager-and-how-to-choose-the-right-one>

Khan, J. A. (2022). Why Do People Forget Passwords? How To Keep Track Of Your Password? *PhoneWorld*. <https://www.phoneworld.com.pk/why-do-people-forget-passwords-how-to-keep-track-of-your-password/>

Kurunthachalam, A., & Karthiga, R. R. (2013). One-time Password: A Survey. *ResearchGate*. <https://www.researchgate.net/publication/344518837_One-time_Password_A_Survey>

Lord, N. (2020, September 29). *Uncovering Password Habits: Are Users’ Password Security Habits Improving? (Infographic).* Digital Guardian. <https://www.digitalguardian.com/blog/uncovering-password-habits-are-users-password-security-habits-improving-infographic>

Luo, G. (2020). RETRACTED: Research on Computer Network Security Problems and Protective Measures under the Background of Big Data. *Journal of Physics*, *1607*(1), 012092. <https://doi.org/10.1088/1742-6596/1607/1/012092>

Mahesh, N. (2018). *Password Habits of Security Literate Individuals.* <https://dspace.sunyconnect.suny.edu/bitstream/handle/1951/70380/Namrata%20Mahesh%20Thesis%20Final%20%281%29.pdf?sequence=1&#38;isAllowed=y>

Pereira, D., Ferreira, J., & Mendes, A. (2020). Evaluating the Accuracy of Password Strength Meters using Off-The-Shelf Guessing Attacks. *Institute of Electrical and Electronics Engineers.* <https://ieeexplore.ieee.org/document/9307670/references#references>

Security, R., & Security, R. (2020, October 12). Guide to Password Management in Network Security. *RSI Security*. <https://blog.rsisecurity.com/guide-to-password-management-in-network-security/>

Todorov, G. (2023, April 12). How to promote your blog. *Learn Digital Marketing*. <https://thrivemyway.com/password-statistics/#:~:text=Password%20Hacking%20Statistics%202023%2090%25%20of%20users%20a,been%20hacked%20because%20they%20used%20a%20duplicate%20password>.

# Appendices

## User’s Manual

Simply run the code using your preferred Python compiler or interpreter. The program should be able to receive inputs from the user. These inputs are the website or application the credentials will be used for, the username or email address, and the password. Using the ‘check password strength’ button evaluates the inputted password and determines its strength and recommends changes. The program also allows generating of a random password based on the user’s set length. After all information is entered, the program also allows them to be saved according to a user’s set file name and extension. Finally, for convenience, the program also allows copying generated passwords and clearing all information.

## Source Code

**# Import modules**

import random

import string

from tkinter import \*

from tkinter import messagebox

**# Main program**

class PasswordManager:

def \_\_init\_\_(self, master):

self.master = master

master.title("PCGM: Password Creator, Generator, and Manager")

self.master.geometry("350x400")

self.master.config(bg="pink")

**# Create labels and entry widgets for website/application, username/email/phone number and password**

self.label0 = Label(master, text="Website/Application:")

self.label0.grid(row=0, sticky=E)

self.entry0 = Entry(master)

self.entry0.grid(row=0, column=1)

self.label1 = Label(master, text="Username/Email/Phone number:")

self.label1.grid(row=1, sticky=W)

self.entry1 = Entry(master)

self.entry1.grid(row=1, column=1)

self.label2 = Label(master, text="Password:")

self.label2.grid(row=2, sticky=E)

self.entry2 = Entry(master, show="")

self.entry2.grid(row=2, column=1)

**# Create button for checking password strength and generating a new password**

self.button1 = Button(master, text="Check Password Strength", command=self.check\_password\_strength,bg="yellow", fg="black")

self.button1.grid(row=3, column=0, pady=10)

self.label4 = Label(master, text="Password Length:")

self.label4.grid(row=4, sticky=E)

self.pass\_len = Entry(master)

self.pass\_len.grid(row=4, column=1)

self.button2 = Button(master, text="Generate New Password", command=self.generate\_password, bg="green", fg="white")

self.button2.grid(row=5, column=1, pady=10)

**# Create label for displaying password strength and feedback**

self.label3 = Label(master, text="")

self.label3.grid(row=3, column = 1, columnspan=2)

**# Button to save login credentials and password to a text file**

self.button3 = Button(master, text="Save Credentials", command=self.save\_credentials, bg="maroon", fg="white")

self.button3.grid(row=7, column=0, pady=10)

self.label5 = Label(master, text="Enter File Name and extension:")

self.label5.grid(row=8, sticky=E)

self.save = Entry(master)

self.save.grid(row=8, column=1)

**# Create button for clearing the entry fields**

self.clear\_button = Button(master, text="Clear", command=self.clear\_fields, bg='red', fg='white')

self.clear\_button.grid(row=9, column=1, pady=10)

**# Create button for copying the password to the clipboard**

self.copy\_button = Button(master, text="Copy to Clipboard", command=self.copy\_to\_clipboard, bg="blue", fg="white")

self.copy\_button.grid(row=9, column=0, pady=10)

**# Checks the strength of the inputted password**

def check\_password\_strength(self):

password = self.entry2.get()

if len(password) < 8:

self.label3.configure(text="Weak password: too short")

elif len(password) > 32:

self.label3.configure(text="Weak password: too long")

elif not any(char.isdigit() for char in password):

self.label3.configure(text="Weak password: no digits")

elif not any(char.isupper() for char in password):

self.label3.configure(text="Weak password: no uppercase letters")

elif not any(char.islower() for char in password):

self.label3.configure(text="Weak password: no lowercase letters")

elif not any(char in string.punctuation for char in password):

self.label3.configure(text="Weak password: no special characters")

else:

self.label3.configure(text="Strong password!")

**# User chooses password length**

def generate\_password(self):

try:

upper = string.ascii\_uppercase

lower = string.ascii\_lowercase

numbers = string.digits

symbols = string.punctuation

all\_chars = upper + lower + numbers + symbols

password\_length = int(self.pass\_len.get())

password = ''.join(random.sample(all\_chars, password\_length))

self.entry2.delete(0, END)

self.entry2.insert(0, password)

self.label3.configure(text="Generated password: " + password)

**# Save the login credentials and password to a file**

try:

name = self.save.get()

with open(name, "a") as file:

file.write( self.entry0.get() + ", " + self.entry1.get() + ", " + password + "\n")

messagebox.showinfo("New Password Generated", f"New password has been generated and saved to {name}")

**# Raise error if no file name is inputted**

except:

messagebox.showinfo("Error!", "Error! No file name inputted.")

**# Raise error if no password length is inputted**

except:

messagebox.showinfo("Error!", "Error! No password length inputted.")

**# Clears or erases the fields**

def clear\_fields(self):

self.entry0.delete(0, END)

self.entry1.delete(0, END)

self.entry2.delete(0, END)

self.pass\_len.delete(0, END)

self.save.delete(0, END)

self.label3.configure(text="")

**# Saves the credentials to a new file**

def save\_credentials(self):

try:

file\_name = self.save.get()

with open(file\_name, "a") as file:

file.write( self.entry0.get() + ", " + self.entry1.get() + ", " + self.entry2.get() + "\n")

messagebox.showinfo("Credentials saved", f"Username and password saved to {file\_name}")

**# Raise error if no file name is inputted**

except:

messagebox.showinfo("Error!", "Error! No file name inputted.")

**# Copies the password to the clipboard of the user**

def copy\_to\_clipboard(self):

password = self.entry2.get()

self.master.clipboard\_clear()

self.master.clipboard\_append(password)

self.master.update()

messagebox.showinfo("Copy to Clipboard", "Password has been copied to the clipboard")

root = Tk()

my\_gui = PasswordManager(root)

root.mainloop()

## Work breakdown

|  |  |  |
| --- | --- | --- |
| **Student Name** | **Tasks Assigned** | **Percentage of the Work Contribution** |
| CALADO, Brent Mitchel R. | Introduction  Problem Statement  Conceptual Framework  Hierarchy Chart  Pseudocode  Project Poster  Function/Module save\_credentials - the process of saving the credentials to a text file  Contributed to the initial draft of the program | 40% |
| SANTIAGO, Ezekiel Inigo P. | Background of the Study  Objectives  Review of Related Literature  Methodology Introduction  Results  Analysis, Conclusion, & Future Directives  User’s Manual  Initial draft of the program | 40% |
| TEPACE, James Louis C. | Flowchart  Methodology – Pseudocode  created half of the program with GUI function without the save credential function | 20% |

## Personal Data Sheet

|  |
| --- |
| **Student’s Personal Data Sheet**    Name: Brent Mitchel R. Calado  Degree Program: Bachelor of Science in Computer Engineering  Address: 479 Cuyegkeng St. Pasay City, Metro Manila  Telephone No.: N/A Cell Phone No.: 09175112305  E-Mail Addresses: [brent\_calado@dlsu.edu.ph](mailto:brent_calado@dlsu.edu.ph) (School)  [brentmrc23@gmail.com](mailto:brentmrc23@gmail.com) (Personal)  Birthday: May 23, 2004 Age: 18 years old  Schools:  High School: Malate Catholic School  Elementary: Malate Catholic School |

|  |
| --- |
| **Student’s Personal Data Sheet**    Name: Ezekiel Inigo P. Santaigo  Degree Program: Bachelor of Science in Computer Engineering  Address: 496 V. Baltazar St., Sto. Tomas, Pasig City  Telephone No.: N/A Cell Phone No.: 09670264127  E-Mail Addresses: [ezekiel\_santiago@dlsu.com.ph](mailto:ezekiel_santiago@dlsu.com.ph) (School)  [ezekielsantiago19@gmail.com](mailto:ezekielsantiago19@gmail.com) (Personal)  Birthday: November 19, 2003 Age: 19 years old  Schools:  High School: Pasig Catholic College  Elementary: Pasig Catholic College |

|  |
| --- |
| **Student’s Personal Data Sheet**    Name: James Louis Tepace  Degree Program: Bachelor of Science in Computer Engineering  Address: 23-D Macabagdal St. Caloocan City  Telephone No.: N/A Cell Phone No.: 09270446581  E-Mail Addresses: [james\_tepace@dlsu.com.ph](mailto:james_tepace@dlsu.com.ph) (School)  [jamestepacee@gmail.com](mailto:jamestepacee@gmail.com) (Personal)  Birthday: August 18, 2003 Age: 19 years old  Schools:  High School: University of the East Calooocan  Elementary: St Mary’s Academy of Caloocan City |