Nimrawee Nattapat 1TE19248G

Coding Midterm Project Report

**Introduction**:

Forgetting passwords and/or setting the same password for every single account are problems that almost everybody faces. People either to set fairly simple passwords that are easy to hack or set complicated passwords that are hard to memorize and easy to forget. That is why, for my midterm coding project, I created a password managing software that both create complicated and reliable passwords unique to each of your account, and also stores them so remembering such complicated passwords is not required. Those were my two goals when creating this software. This report will be about how I used programming to solve achieve these two goals.

**Goal #1: To Generate Complicated Passwords Unique to each Account.**

 Since the goal of setting a complicated password is to avoid hacking and stealing of personal information, it is very important to consider the amount of time it takes to hack a password. The following table shows how fast passwords can be cracked corresponding to its lengths and compositions. As shown below, most fairly simple passwords that most people use can be hacked within a day, with many even being instantly. Therefore, in order to prevent this, the randomly generated password must be at least 10 characters long, and must include both upper- and lower-case letters, as well as numbers and/or special characters. In order to manifest this idea in Python, we can use libraries **random** and **string**. Setting the length to be random integers between 10 and 20, and randomly selecting through lower-case, upper-case, numbers, and symbols, we were able to achieve such functionality with the following simple, short, yet effective code.

A screenshot of a computer

Description automatically generated with medium confidence

**Goal #2: To Create a User-Based System which Stores and Recalls Accounts and Its Corresponding Randomly Generated Passwords**

Now that we are able to generate passwords, we need to develop a way to store and arrange them to their corresponding username and platform. A user-based system is required to store the accounts of each. The first step is to develop a system to store all the required data of a user. Through trial and error, the following data structure was used:

Diagram

Description automatically generated

As for the user interface, my first instinct was to create a program that simply runs in the terminal. Therefore, I created a quick draft on the and ran the program in the terminal. While it worked out very well, it was not very practical to use and too simple to make. Therefore, I decided to challenge myself to create something I’ve never done before, which was to create a graphic user interface (GUI) using a native Python tool called ***tkinter***. Tkinter allows the creation of frames, buttons, entry boxes, labels, and many other necessary widgets to create a practical user interface. This can be done by simply creating a frame and packing widgets into it. Then, commands can be assigned to each button to execute different functions.

While this sounds very nice and simple, I soon ran into a problem which require quite a lot more work to solve. The problem was that my program requires the use of multiple frames, such as the start page, create user page, log in page, etc. Furthermore, the ability to connect each page and monitor which page shows up while the other pages are hidden is also required. After doing hours of researching and watching tutorial videos, I discovered that the solution was to use classes and objects, which coincidentally, was a new concept that I recently learned in my python class. The following link is the tutorial which my user interface was greatly influenced by: <https://pythonprogramming.net/change-show-new-frame-tkinter/>. This is where I took the structure and the concept of creating interface with multiple frames from and modified it to be my own software. Because each page has to be able to be called up and hidden, a separate class has to be created for each page, with all its widgets packed into the \_\_init\_\_ function, so it is forced to run every time that specific page is called. Each class has to inherit the class “tk.Frame”, which is a Frame widget class from the tkinter library. Then, the frames are created as an instance of their respective class and stored as an object into a dictionary. The following code shows how this was done:

Text

Description automatically generated

This is an example of a class of a page inheriting tk.Frame and having all the widgets under the \_\_init\_\_ function.

Text

Description automatically generated

Next, in order to monitor which page shows up for the user to interact with, a new function was defined to serve such function using the frame.tkraise() method and the parameter as the items in the dictionary.

Text

Description automatically generated

This way, the problem of running multiple frames and connecting them is solved. This workaround method is an attempt of creating a Pseudo-MVC (model, view and controller) architecture without the ‘model’ part.

**Conclusion**

In conclusion, for a beginner to python like me, this project was very challenging but fun at the same time. It was very hard trying to do something you don’t know to do, but achieving it feels very rewarding. Overall, considering my skill level with python at this point, I am quite happy with how my project turned out. While it is absolutely nowhere near being secure enough to be used as a reliable and practical password manager, it was a great practice of manifesting my ideas into code and creating user interface. After all, the internet is a very dangerous place, and whenever I am ready with more skills in python, I really want to develop something that will increase to security and safety on the internet some day.