Project Title: GOPI'S PASSWORD MANAGER

Date: 09/05/2022 **Prepared By: K. Gopi Kalyan**

Project Abstract: The main objective of the Password manager is protect user's passwords by using a master password or a security token. The security of them using the master password is weakened if users use weak master passwords. Password managers reduce the difficulties in creating and remembering strong passwords. Password is the most popular and simplest way in which users can authenticate themselves before accessing computer systems or websites. Password management is organization and encryption of many personal passwords using a single login. Some common risk involved in losing password is over the shoulder attack, Brute-force attack, Sniffing attack, Login spoofing attack. Using Password manager adds another layer of security to our password and data protection.

Product Requirements & tools: Python, Visual Studio code

Product Future Scope: Passwords are easy to implement using software. They are also very cheap to run. Unlike two-factor authentication, biometrics, or any of the other advanced systems discussed below, the only real costs are the implementation. There is no fancy hardware or software to maintain over time. In addition, users can delete or modify passwords easily, which is not always possible with other solutions. (Random shower thought: If your company uses facial recognition, would they pay for Botox if you called it "updating your login credentials"?)

DECLARATION

I …**K. Gopi Kalyan Prasad**……, hereby declare that everything proposed in this project proposal is based on my own knowledge and research carried out with exception to printed or electronic content.

SIGNATURE … [illegible] … DATE **09/05/2022**

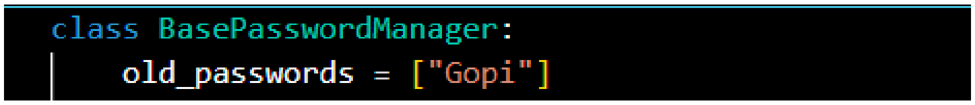
Steps to perform

class **BasePasswordManager**

members

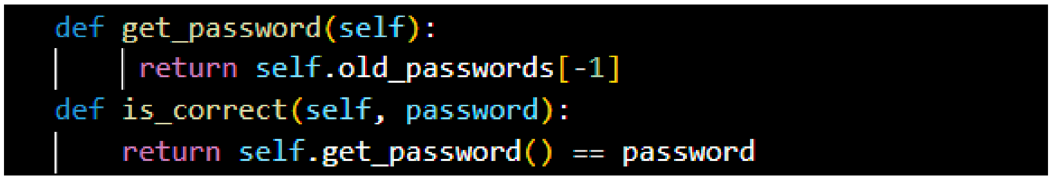
old_passwords: is a list that holds all of the user's past passwords.

The last item of the list is the user's current password. Methods



```
class BasePasswordManager:
    old_passwords = ["Gopi"]
```

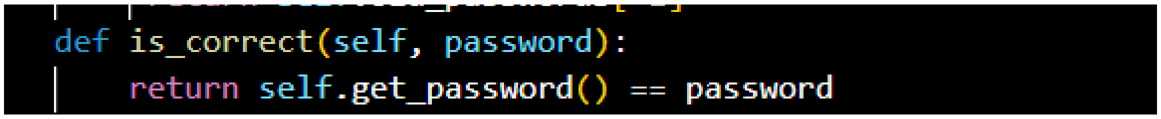
get_password method that returns the current password as a string.



```
def get_password(self):
        return self.old_passwords[-1]
def is_correct(self, password):
    return self.get_password() == password
```

Receives a string and returns a boolean True or False. depending on whether the string is equal to the current password or not.

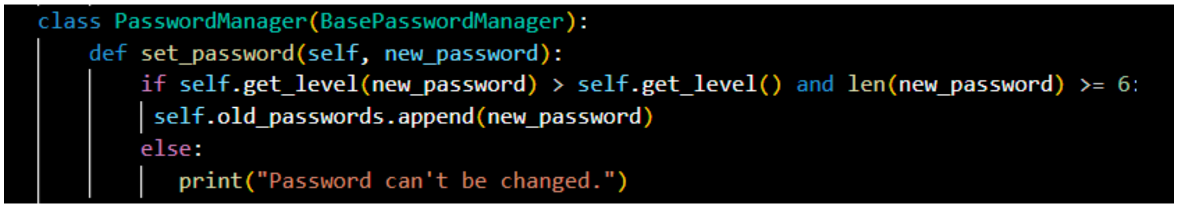
is_correct: Method that receives a string and returns a Boolean. True or False depending on whether the string is equal the current password or not.



```
    def is_correct(self, password):
        return self.get_password() == password
```

class PasswordManager

This class inherits from **BasePasswordManager**



```
class PasswordManager(BasePasswordManager):
    def set_password(self, new_password):
        if self.get_level(new_password) > self.get_level() and len(new_password) >= 6:
            self.old_passwords.append(new_password)
        else:
            print("Password can't be changed.")
```

set_password: Method that sets the user's password.

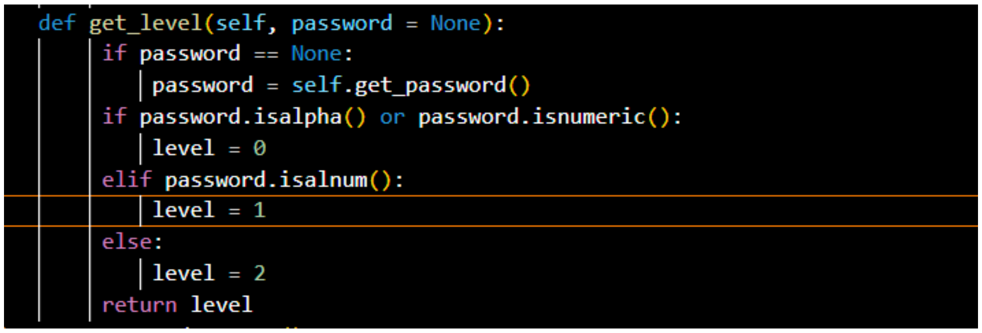
Password change is successful only if:

- Security level of the new password is greater.

- Length of new password is minimum 6

However, if the old password already has the highest security level. New password must be of the highest security level for a successful password change.

get_level: Method that returns the security level of the current password. It can also check and return the security level of a new password passed as a string.



```
def get_level(self, password = None):
    if password == None:
        password = self.get_password()
    if password.isalpha() or password.isnumeric():
        level = 0
    elif password.isalnum():
        level = 1
    else:
        level = 2
    return level
```

***Security levels*:**

level 0* - *password consists of alphabets or numbers only.

level 1* - *Alphanumeric passwords.

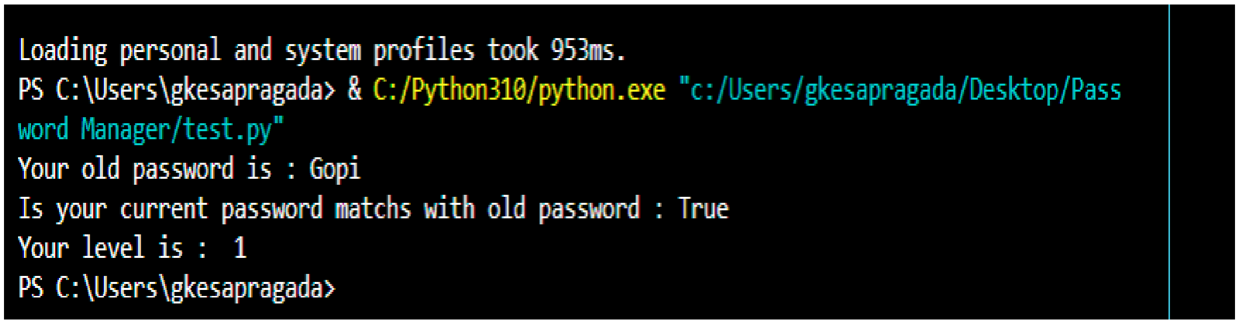
level 2* - *Alphanumeric passwords with special characters.

Source Code:

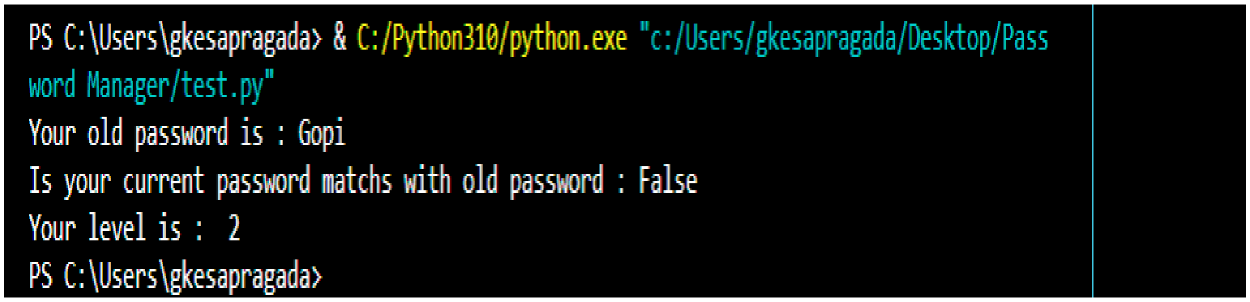
```
class BasePasswordManager:
    old_passwords = ["Gopi"]
    def get_password(self):
         return self.old_passwords[-1]
    def is_correct(self, password):
        return self.get_password() == password
class PasswordManager(BasePasswordManager):
    def set_password(self, new_password):
        if self.get_level(new_password) > self.get_level() and len(new_password) >=
6:
         self.old_passwords.append(new_password)
        else:
           print("Password can't be changed.")
    def get_level(self, password = None):
         if password == None:
             password = self.get_password()
         if password.isalpha() or password.isnumeric():
             level = 0
         elif password.isalnum():
             level = 1
         else:
            level = 2
        return level

sb= BasePasswordManager()
print("Your old password is :", sb.get_password())
print("Is your current password matchs with old password :",sb.is_correct("hopi"))
passwordManager = PasswordManager()
passwordManager.set_password("657657")
print("Your level is : ", passwordManager.get_level())
```

Outputs:



```
Loading personal and system profiles took 953ms.
PS C:\Users\gkesapragada> & C:/Python310/python.exe "c:/Users/gkesapragada/Desktop/Pass
word Manager/test.py"
Your old password is : Gopi
Is your current password matchs with old password : True
Your level is :  1
PS C:\Users\gkesapragada>
```



```
PS C:\Users\gkesapragada> & C:/Python310/python.exe "c:/Users/gkesapragada/Desktop/Pass
word Manager/test.py"
Your old password is : Gopi
Is your current password matchs with old password : False
Your level is :  2
PS C:\Users\gkesapragada>
```

If your password is None it returns the level 0 based on our condition

```
sb= BasePasswordManager()
print("Your old password is :", sb.get_password())
print("Is your current password matchs with old password :",sb.is_correct("hopi"))
passwordManager = PasswordManager()
passwordManager.set_password("gopi@123")
print("Your level is : ", passwordManager.get_level())
```

SOFTWARE REQUIREMENTS SPECIFICATION

1) Hardware Requirements

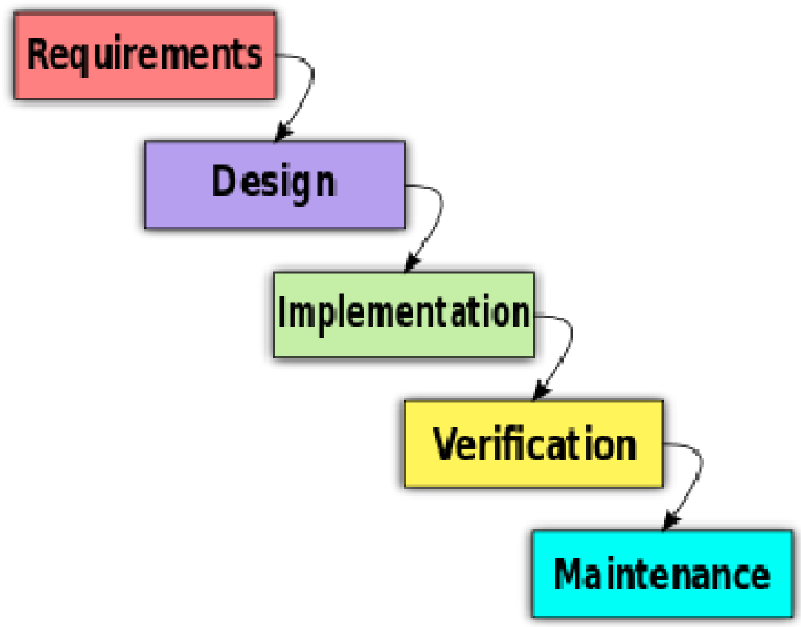
Number	Description
1	PC with 250 GB or more Hard disk.
2	PC with 4GB RAM.
3	PC with Pentium 1 and Above.

2) Software Requirements

Number	Description	Type
1	Operating System	Windows XP / Windows
2	Technology	PYTHON
3	Database	MySQL
4	IDE	Visual Code
5	Browser	Google Chrome

PROCESS MODEL

For my project I plan to use waterfall as a process model. The waterfall model is a sequential design process, often used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Initiation, Analysis, Design, Construction, Testing and Maintenance


Requirements
Design
Implementation
Verification
Maintenance