# **PROJECT REPORT**

## **TOPIC:**

## PASSWORD GENERATOR WITH A PASSWORD MANAGER

## INDEX:

**Index**

| **Sr.No.** | **Topic** | **Page No.** |
| --- | --- | --- |
| **1** | **INTRODUCTION** | **3** |
| **2** | **NECESSITY** | **4** |
| **3** | **OBJECTIVE AND SCOPE**  **WORKING** | **5** |
| **4** | **SYSTEM DEVELOPMENT** | **7** |
| **5** | **PERFORMANCE ANALYSIS** | **9** |
| **6** | **TEST CASES** | **11** |
| **7** | **CONCLUSION AND FUTURE DEVELOPMENTS** | **14** |

## INTRODUCTION:

We have created a Secure and unique password Generator with a Password Manager to generate a unique and random password and also to store them safely so that we can use those passwords whenever requires. These passwords which are generated from the password generator are highly secure and uncrackable.

We have created our project using Python which randomly generates a string with the specified numbers of characters (alphabets, digits and symbols). These strings can be used as a password which also satisfies the password policies for almost every Web-applications or Software-applications.

This password generator can be used to generate a very safe password for your login form so that any malicious hacker can’t perform any sort of brute-forcing or dictionary attacks to get access of a user account.

## NECESSITY:

As the technologies and IT industries are growing, everything gets hosted on Internet which is more convenient for every individual who wants to use the technology there. But this also comes with a great risk for data confidentiality. Malicious hackers or attackers can perform a data breach for any organization whose software or applications or not secure. One of the main reasons for these attacks are weak passwords being used. An attacker can perform a dictionary-based attack or brute force the login field with every possible password he/she can think of based on their research. If the password gets matched, the hacker can take-over the account and can do any malicious stuff we want.

Thus, it becomes necessary for an organization or every individual who want to be safe from such attacks to have a secure and strong password. Here we came with the same idea to protect everyone from these types of attack and every one with a unique password which is uncrackable.

## OBJECTIVES AND SCOPE:

This tool can be useful to anyone who wants to have a secure and unique password and want to prevent his/her date to get breached in any sort of cyber-attacks.

The password manager can even handle multiple entries and also allows you to update the entries later is any modification is required.

This application/tool also has a feature to recover the master password which is used to login into the vault.

## WORKING:

This tool is programmed using Python and we used ‘tkinter’ library to design a basic UI for the tool. This program simple accepts the number of characters we would like in the password and uses the random module of python to generate a random password string. The entered characters are alphabets, numeric digits and symbols. The range for entering these inputs lies between 3 to 11 for each characters. Every time the generate button is pressed, it generates a different string of password and we are also allowed to copy that password string and use for any login-form we want to register for.

Next, we have our password manager/vault to save the passwords for later use. This vault would be helpful as the generated password is a gibberish string and is very difficult to remember. Thus, at times it becomes necessary to save this password for later use.

Accessing the password vault is also easy. We just need to click on the Open Vault button and provide the password which we are required to set during our very first use.

While we register ourself into the vault for the first time, we are also provided with a reference key, which is basically a hashed string of data. This reference key can be used to re-enter the password for the vault in case the user forgets it’s password.

Now, when we get logged in into our vault after providing correct password, we can see an add button to add up the entries we want. Here, we can save the account name for which we used the password for, a username section and a password section. We can also see an update button incase we want to modify any entry.

## SYSTEM DEVELOPMENT:

In this project we have used Python language as our programming language and several modules of python to make it functioning. We have used Cryptography library of python and also fernet class from that library for encryption and decryption purpose.

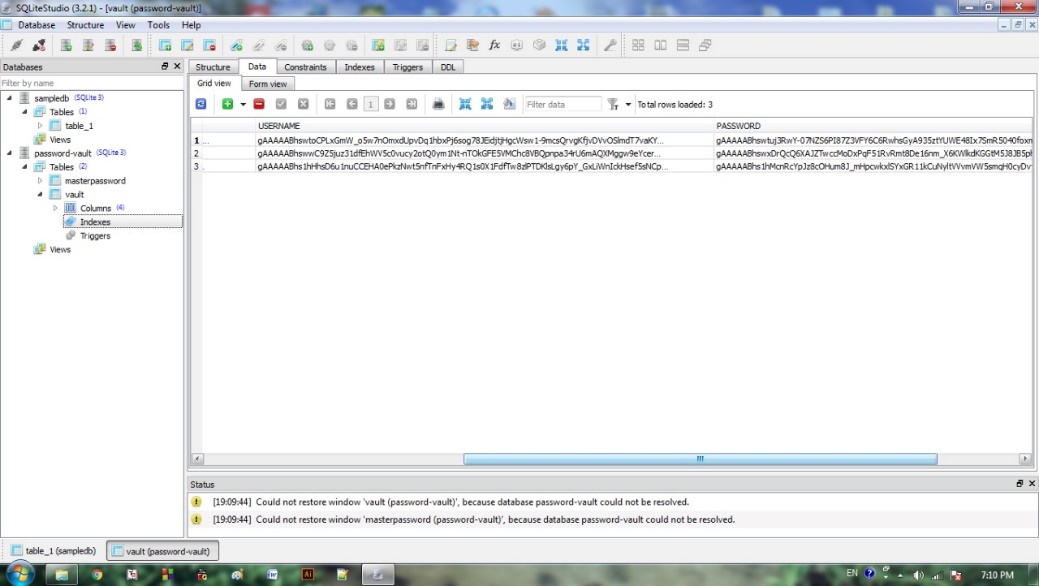
For creating encryption key we pass a encoded utf-8 unique string to kdf.derive(). The main purpose of this function is to take a human readable text password and turn it into the bits and bytes to be used as the key to a cryptographic algorithmand this bytes format of converted string is the passed to urlsafe\_b64encode(s), with the help of base64.urlsafe\_b64encode(s) method, we can encode the string using url and filesystem safe alphabets into the binary form. Then lastly we store this into a variable'encryption\_key'which we will use for our encryption and decryption purpose.

For creating a recovery key we have taken a variable 'key' and we have used the UUID class which provides immutable object,the uuid4() function from uuid class create a unique ID and returns it which we store in our key variable, and then by using ‘.hex()’ we convert it into a 32 character hexadecimal string. Then we encode it into utf-8 version of string and then we again pass this key through sha256 which will be its hash and this value will be referred as our recovery key.

For encryption and decryption of our data (username,password),we have used fernet class. So, after creating an encryption key and passing it and an encoded message to encrypt function, it encrypts data passed as a parameter to the method. The outcome of this encryption is known as a “Fernet token” which is basically the ciphertext. The encrypted token also contains the current timestamp when it was generated in plaintext. The encrypt method throws an exception if the data is not in bytes. This ciphertext is then stored in our database which cannot be changed or altered without our key

For decryption we call the decrypt method of fernet class and pass the key and the ciphertext from the databasewhich was created by encrypt function,it converts that ciphertext to plaintext. The encryption key which we had generated is URL-safe base64 encoded 32-byte key. Without that both keys, our encrypt and decrypt functions are useless.

Below is a snap shot which depicts how the username and passwords are stored in the database.



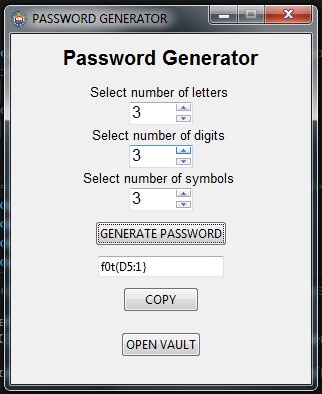
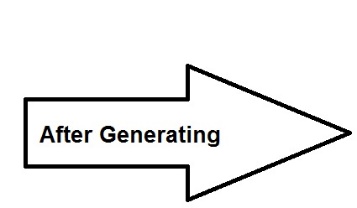
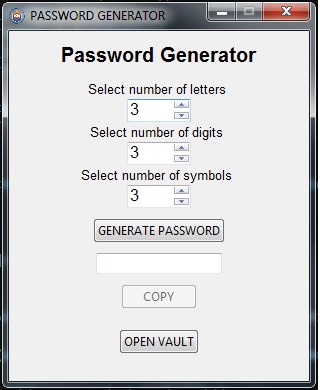
## PERFORMANCE ANALYSIS:

The proposed model works exactly the way we wanted it work. Once the tool is installed and the process of setup is complete, the user needs to enter/generate a Master Password to log in into the vault. This is the first step to do after the setup is complete.

After these steps, the user can use to generate any kind of password he/she wants to have.

Following are some sample outputs, which shows the performance and the functionalities of the tool properly.

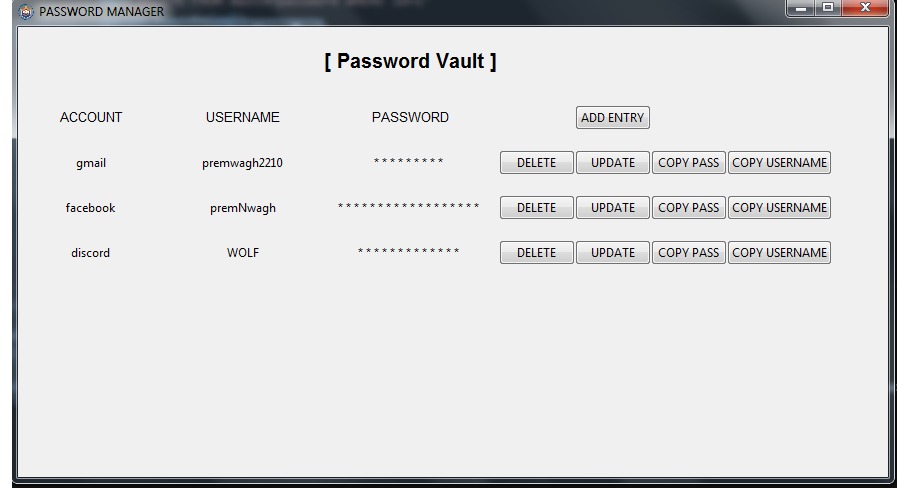
1. Password generation.



1. Login panel for Password Manager/vault.



1. Saved entries in the vault.



## TEST CASES:

The following test cases are made using Unit based testing of the software application for our project. These test cases shows that our project work fine for every functionality.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **Input** | **Expected output** | **Actual output** | **Status** |
| **1** | **Check whether it generates the password with specified number of letters, symbols and digits.** | **Provide the number of letters, symbols and digits in their respective SpinBox widgets.** | **The generated password should contain the specified number of characters.** | **Yes, it generates the password with the specified number of characters.** | **Passed** |
| **2** | **Check whether the password gets randomly generated on every click.** | **Click on the Generate Password button.** | **It should generate a random password string every time of the specified length.** | **It always generates a random password of the specified length.** | **Passed** |
| **3** | **Check whether the generated password gets copied on the clipboard.** | **Click the copy button after the password is generated.** | **The generated password should get copied on the clipboard.** | **Yes, the generated gets copied on the clipboard.** | **Passed** |
| **4** | **Check whether we successfully get logged in to the vault with correct credentials.** | **Provide correct user credentials in the vault login field.** | **We should get logged in to the vault.** | **Yes, we get logged in into the vault.** | **Passed** |
| **5** | **Check whether we get access to the vault on providing wrong password.** | **Enter wrong password.** | **Should throw an error for wrong password.** | **Throws an error.** | **Passed** |
| **6** | **Check whether a unique recovery key is generated after creation of a fresh master password.** | **Use the forget password functionality and provide the recovery key obtained from the very first login.** | **It should generate a unique recovery key every time.** | **Yes, its generates a unique recovery key.** | **Passed** |
| **7** | **Check whether the forget password functionality works only with the correct recovery key.** | **Enter a wrong recovery key.** | **It should not work with wrong recovery key.** | **Yes, it does not accept incorrect key.** | **Passed** |
| **8** | **Check whether we are able to save usernames and passwords.** | **Save a Username and Password.** | **It should save the username and password for later use.** | **Yes, the username and password is saved for later use.** | **Passed** |
| **9** | **Check whether the vault can manage multiple entries.** | **Enter multiple entries in the vault.** | **It should properly manage all the entries.** | **Yes, the vault properly manages all the entries** | **Passed** |
| **10** | **Check whether the entry gets updated while we update it.** | **Update the password by click on update button.** | **The entry should get updated.** | **Yes, the entries get updated.** | **Passed** |
| **11** | **Check whether the passwords from the saved entries can be copied to the clipboard.** | **Click on the copy button at the side of the entry you want to copy the password for.** | **The password from the entry should get copied to the clipboard and is ready to use further.** | **The password gets copied to the clipboard.** | **Passed** |

## CONCLUSION AND FUTUREDEVELOPMENT PLANS:

Thus, we have created a Password Generator along with a Password Manager to create and store a random and unique set a password to use for any sort of login form or any authentication purpose. In this project we have also learnt about some python libraries and their implementations.

In future, several modifications can be done in this project like:

* Enhancing the user interface to make it more interactive to users.
* A more enhanced login panel with more security measure to deal with cyber-attacks properly.
* We will be hosting this tool on a website and enhancing our database to manage a greater number of user entries.