Group Project – Password manager

Project name:	Passwordmanager
Group ID:	3358
Project language:	Python
Group members registered:	Samuel9 and Nguyen
Submitted by:	Samuel Hafner and Nguyen Hoang

Table of Content

Overview

The password manager is a Python-based application designed to securely *store, retrieve, and manage* passwords for various accounts. It integrates encryption (via the `cryptography` library) to ensure sensitive data is safeguarded, providing a simple and secure tool for managing credentials. A master password protects the application, ensuring only authorized users can access or modify the stored data.

Use Case

Password security is a critical concern in today's world. Users tend to use simple passwords because they are easier to remember, which can be a major risk in terms of cyber security. This project addresses this issue by providing a centralized, encrypted password storage solution, to generate and store stronger more complex passwords.

Features

1. Master Password Protection
- Ensures only authorized access to the password manager.
- Users set a master password during initialization.
- Includes functionality to change the master password.

2. Password Storage
- Save encrypted passwords for different accounts.
- Data is stored in a file named `passwords.txt`.
- Passwords may be changed or deleted individually by the user.

3. Password Retrieval
- Decrypt and display stored passwords for specified accounts.
- Ensures secure access to saved data.

4. Password Generator
- Create strong, random passwords of user-defined length (min. 8 characters).
- User can choose whether to include symbols.
- Option to save generated passwords directly into the manager.

5. Data Reset
- Deletes all saved passwords, master password, and encryption keys.
- Allows the application to reset for reuse or securely wiped.

6. Encryption Mechanism
- Employs the `Fernet symmetric encryption` from the cryptography library to encrypt and decrypt data.
- Ensures data security during storage and retrieval.

Prerequisites

Users must make sure to install the library `cryptography`. The library is essential for implementing any form of secure data protection, such as storing passwords securely, creating encrypted communications or ensuring data confidentiality through encryption and decryption.

By running the installation command `pip install cryptography` the user makes this functionality available in their Python environment.

Functioning

This chapter gives a quick overview of the functions in the password manager, explaining its purpose and how it works at a surface level. For a more detailed explanation of each function in form of a line-by-line documentation, please find the comments provided directly in the Python code.

1. **Encryption Mechanism**
   The Encryption System provides secure handling of sensitive information within the Password Manager. It generates and stores a unique encryption key, which is critical for protecting stored data and ensuring it remains inaccessible without proper authorization. The system retrieves this key as needed, raising errors if the key is missing to prevent unauthorized access.
   Using the encryption key, plaintext information is securely encrypted into protected data, while encrypted data can be decrypted back into readable text only when authorized. This system ensures that sensitive information is robustly encrypted and protected, forming the backbone of the Password Manager's security.

2. **Master password management**
   The Master Password System serves as the primary security mechanism, ensuring that only authorized users can access or manage the Password Manager. It allows users to set up and securely store a master password during initialization, safeguarding access to the manager. Users are required to authenticate themselves with the stored master password to proceed with any operations, with limited attempts to enhance security.

   Additionally, the system supports the ability to update the master password securely, requiring validation of the current password before making changes. This ensures ongoing protection of sensitive information and provides users with control over their access credentials.

3. **Password Management Functions**
   The Password Management System securely handles the storage, retrieval, and management of encrypted account credentials. Passwords are stored alongside their respective account names in a secure file to protect sensitive data. The system allows users to retrieve, update, or delete passwords for specific accounts, with safeguards to handle missing accounts or decryption errors. The reason behind the password update function is that it's strongly recommended for the users to update their passwords regularly. Additionally, the program ensures that account names are unique by checking for duplicates in the saved passwords file to prevent complications in the password retrieval process in advance. If a name already exists, the user is prompted to choose a different one.

4. **Password Generator**
   This function generates a secure random password with a default length of 12 characters. The password includes a mix of uppercase and lowercase letters, digits, and special characters (symbols) by default. An optional parameter allows restricting the password to only letters and digits, excluding all special characters. This option is provided to ensure compatibility with systems or platforms that may not accept special characters as valid password input while maintaining strong randomness and security.

5. **Reset Manager**
   This function resets the password manager by securely clearing all stored account credentials and related data. It deletes all generated files, such as the password storage file and encryption keys, ensuring no sensitive information remains. The system restores itself to its initial state by

removing all saved passwords and associated encryption data. This provides users the option to start fresh.

Demonstration and user guide

To initiate the code, the user is required to run the code within the respective program (e.g. Visual Studio Code, Replit or any interface alike). In the following demonstration, we simulate the user chronologically generating, saving and retrieving a password in the password manager. (Please note that everything below in bold is the user's input. Please also note that this demonstration does not cover all the functions but rather sticks to the core functions.)

Setting a master password

```
Encryption key not found. Generating a new one...
No master password found. Please set a new one.
Master password: TestTest
Confirm master password: TestTest
Master password successfully set.
Enter the master password: TestTest
Access granted.
```

Generating and saving a password

```
Password Manager
1. Save a password
2. Retrieve a password
3. Generate a password
4. Change master password
5. Reset manager
6. Delete a password
7. Change a password
8. Exit
Choose an option: 3
Password length (minimum 8 characters): 23
Exclude symbols? (yes/no): no
Generated password: 5_f=$mFCn7C.h)kBiP1638D
Save this password? (yes/no): yes
For which account should it be saved? Acc. 1
Password saved!
```

Retrieving a password

```
Password Manager
1. Save a password
2. Retrieve a password
3. Generate a password
4. Change master password
5. Reset manager
6. Delete a password
7. Change a password
8. Exit
Choose an option: 2
Account name: Acc. 1
Password for Acc. 1: 5_f=$mFCn7C.h)kBiP1638D
```

Deleting a password

```
Password Manager
1. Save a password
2. Retrieve a password
3. Generate a password
```

```
4. Change master password
5. Reset manager
6. Delete a password
7. Change a password
8. Exit
Choose an option: 6
Enter the account name to delete: Acc. 1
Password for account 'Acc. 1' has been deleted.
```

Exiting the program

```
Password Manager
1. Save a password
2. Retrieve a password
3. Generate a password
4. Change master password
5. Reset manager
6. Delete a password
7. Change a password
8. Exit
Choose an option: 8
Exiting Password Manager.


...Program finished with exit code 0
Press ENTER to exit console.
```

Summary

The Password Manager is a Python-based application designed to securely store and manage user passwords using encryption from the *cryptography* library. It relies on a master password for access control, ensuring only authorized users can use its features.

The application allows users to save encrypted passwords, retrieve them securely, generate strong random passwords, and delete outdated credentials. It also offers an option to reset all stored data and encryption keys if necessary.

The password manager ensures account names are unique to prevent accidental overwrites. When executed, the user sets a master password, and after authentication, can access functionalities like generating, saving, and retrieving passwords through a simple menu-driven interface. This tool combines ease of use with strong security features to help users manage sensitive credentials efficiently.

Acknowledgment of the use of ChatGPT

While developing this entire project, ChatGPT was utilized as a supplementary tool to assist with specific aspects of the development. Chat GPT was especially used in the coding process itself as well as bug fixing and some parts of the documentation. The collaboration with ChatGPT allowed for a more efficient process while ensuring a clear understanding of all components of the application.