Dokumentation Password-Manager Gruppenarbeit Python

Das Programm Password Manager dient zur sicheren Verwaltung und Speicherung von Passwörtern für verschiedene Websites. Es enthält Funktionen zum Hinzufügen, Anzeigen, Bearbeiten und Löschen von Passworteinträgen.

Utility-Functions in password_manager.py:

```python
def generate_key() -> bytes:
    return Fernet.generate_key()
```

- Generiert bzw. gibt einen neuen Verschlüsselung-Schlüssel zurück

```python
def load_key() -> bytes:
    try:
        return open("key.key", "rb").read()
    except FileNotFoundError:
        raise Exception("Encryption key not found. Please initialize the password manager.")
```

- Gibt den Schlüssel aus der „key.key“ Datei zurück und löst andernfalls eine Fehlermeldung aus

```python
def save_key(key: bytes) -> None:
    with open("key.key", "wb") as key_file:
        key_file.write(key)
```

- Speichert den angegebenen Schlüssel in der Datei „key.key“

```python
def encrypt_data(data: str, key: bytes) -> bytes:
    f = Fernet(key)
    encrypted_data = f.encrypt(data.encode())
    return encrypted_data
```

- Verschlüsselt die angegeben Daten mit dem bereitgestellten Schlüssel und liefert diese als Bytes zurück

```python
def decrypt_data(encrypted_data: bytes, key: bytes) -> str:
    f = Fernet(key)
    decrypted_data = f.decrypt(encrypted_data).decode()
    return decrypted_data
```

- Entschlüsselt die angegeben Daten mit dem bereitgestellten Schlüssel und liefert diese als String zurück

```python
def hash_password(password: str) -> str:
    return hashlib.sha256(password.encode()).hexdigest()
```

- Hashing des angegebenen Passwortes und Rückgabe als hexadezimale Zeichenkette

```python
def generate_password(length: int = 12, use_uppercase: bool = True, use_numbers: bool = True, use_special_chars: bool = True) -> str:
    characters = string.ascii_lowercase
    if use_uppercase:
        characters += string.ascii_uppercase
    if use_numbers:
        characters += string.digits
    if use_special_chars:
        characters += string.punctuation

    password = ''.join(random.choice(characters) for _ in range(length))
    return password
```

- Generiert ein zufälliges Passwort mit 12 Zeichen, Großbuchstaben, Nummern und Sonderzeichen

```python
# PasswordManager class
class PasswordManager:
    def __init__(self, master_password: str):
        self.master_password = master_password
        self.hashed_master_password = hash_password(master_password)

        if not os.path.exists("key.key"):
            self.key = generate_key()
            save_key(self.key)
        else:
            self.key = load_key()

        if not os.path.exists("passwords.json"):
            with open("passwords.json", "w") as file:
                json.dump({}, file)
```

- Initialisiert den Passwort Manager bzw. nutzt die oben beschriebenen Funktionen, um ein Master-Passwort zu „Hashen“, einen Schlüssel ggf. zu generieren und stellt sicher ob eine passwords.json Datei existiert.

```python
def save_password(self, username: str, password: str, url: str, notes: str, categories: str) -> None:
    encrypted_password = encrypt_data(password, self.key)
    encrypted_notes = encrypt_data(notes, self.key)
    encrypted_categories = encrypt_data(categories, self.key)
    creation_date = datetime.now().isoformat()
    encrypted_creation_date = encrypt_data(creation_date, self.key)

    password_entry = {
        'username': username,
        'password': encrypted_password.decode(),
        'url': url,
        'notes': encrypted_notes.decode(),
        'categories': encrypted_categories.decode(),
        'creation_date': encrypted_creation_date.decode(),
        'password_history': []
    }

    try:
        with open("passwords.json", "r") as file:
            passwords = json.load(file)
    except json.JSONDecodeError:
        passwords = {}

    if self.hashed_master_password not in passwords:
        passwords[self.hashed_master_password] = []

    passwords[self.hashed_master_password].append(password_entry)

    with open("passwords.json", "w") as file:
        json.dump(passwords, file, indent=4)
```

- Verschlüsselt und speichert einen Passwort-Eintrag in „passwords.json“

```python
def view_passwords(self) -> None:
    try:
        with open("passwords.json", "r") as file:
            passwords = json.load(file)
    except json.JSONDecodeError:
        passwords = {}

    if self.hashed_master_password in passwords:
        for entry in passwords[self.hashed_master_password]:
            decrypted_password = decrypt_data(entry['password'].encode(), self.key)
            decrypted_notes = decrypt_data(entry['notes'].encode(), self.key)
            decrypted_categories = decrypt_data(entry['categories'].encode(), self.key)
            decrypted_creation_date = decrypt_data(entry['creation_date'].encode(), self.key)

            print(f"Username: {entry['username']}")
            print(f"Password: {decrypted_password}")
            print(f"URL: {entry['url']}")
            print(f"Notes: {decrypted_notes}")
            print(f"Categories: {decrypted_categories}")
            print(f"Creation Date: {decrypted_creation_date}")
            print("-----")
    else:
        print("Invalid master password!")
```

- Entschlüsselt und zeigt alle Passwort-Einträge an

```python
def generate_password(self, length: int = 12, use_uppercase: bool = True, use_numbers: bool = True, use_special_chars: bool = True) -> str:
    return generate_password(length, use_uppercase, use_numbers, use_special_chars)
```

- Generiert per der bereits oben beschriebenen Funktion ein zufälliges Passwort

```python
def retrieve_password(self, search_term: str) -> None:
    try:
        with open("passwords.json", "r") as file:
            passwords = json.load(file)
    except json.JSONDecodeError:
        passwords = {}

    if self.hashed_master_password in passwords:
        for entry in passwords[self.hashed_master_password]:
            decrypted_url = entry['url']
            decrypted_username = entry['username']
            decrypted_password = decrypt_data(entry['password'].encode(), self.key)
            decrypted_notes = decrypt_data(entry['notes'].encode(), self.key)
            decrypted_categories = decrypt_data(entry['categories'].encode(), self.key)
            decrypted_creation_date = decrypt_data(entry['creation_date'].encode(), self.key)

            if search_term in decrypted_url or search_term in decrypted_username:
                print(f"URL: {decrypted_url}")
                print(f"Username: {decrypted_username}")
                print(f"Password: {decrypted_password}")
                print(f"Notes: {decrypted_notes}")
                print(f"Categories: {decrypted_categories}")
                print(f"Creation Date: {decrypted_creation_date}")
                print("-----")
    else:
        print("Invalid master password!")
```

- Sucht und zeigt einen geeigneten Passwort-Eintrag an, der den Suchkriterien entspricht

```
def edit_password(self, search_term: str) -> None:
    try:
        with open("passwords.json", "r") as file:
            passwords = json.load(file)
    except json.JSONDecodeError:
        passwords = {}

    if self.hashed_master_password in passwords:
        for entry in passwords[self.hashed_master_password]:
            decrypted_url = entry['url']
            if search_term in decrypted_url or search_term in entry['username']:
                print("Editing entry:")
                print(f"URL: {decrypted_url}")
                print(f"Username: {entry['username']}")
                new_username = input("Enter new username (leave blank to keep current): ")
                new_password = getpass("Enter new password (leave blank to keep current): ")
                new_notes = input("Enter new notes (leave blank to keep current): ")
                new_categories = input("Enter new categories (leave blank to keep current): ")

                if new_username:
                    entry['username'] = new_username
                if new_password:
                    entry['password_history'].append(entry['password'])
                    entry['password'] = encrypt_data(new_password, self.key).decode()
                if new_notes:
                    entry['notes'] = encrypt_data(new_notes, self.key).decode()
                if new_categories:
                    entry['categories'] = encrypt_data(new_categories, self.key).decode()

                break
        else:
            print("No matching entry found.")

        with open("passwords.json", "w") as file:
            json.dump(passwords, file, indent=4)
    else:
        print("Invalid master password!")
```

- Entschlüsselt und bearbeitet ggf. einen Passwort-Eintrag

```
def delete_password(self, search_term: str) -> None:
    try:
        with open("passwords.json", "r") as file:
            passwords = json.load(file)
    except json.JSONDecodeError:
        passwords = {}

    if self.hashed_master_password in passwords:
        for i, entry in enumerate(passwords[self.hashed_master_password]):
            decrypted_url = entry['url']
            if search_term in decrypted_url or search_term in entry['username']:
                print(f"Deleting entry for {decrypted_url} - {entry['username']}")
                del passwords[self.hashed_master_password][i]
                break
        else:
            print("No matching entry found.")

        with open("passwords.json", "w") as file:
            json.dump(passwords, file, indent=4)
    else:
        print("Invalid master password!")
```

- Löscht je nach Suchkriterium einen Passwort-Eintrag

Main.py

Hier muss noch def run erklärt werden. Der User kann bis jetzt nur Einträge erstellen und abrufen. Generation, Änderung, Löschen sind bis jetzt noch nicht in der Benutzeroberfläche implementiert.

Dokumentation ist on Hold bis das Programm fertig ist

Tests müssen auch noch dokumentiert werden, sowie die Abhänigkeiten der Funtktionen

Wiedemann:
Es gibt hier keine strikte Vorgabe. Versuchen Sie einfach grob zu beschreiben, was Sie sich bei der Entwicklung gemacht haben. Wie und wieso haben Sie die Module erstellt, was macht welches Modul, welche Abhängigkeiten haben diese, ...

Machen Sie ein paar Screenshots des Benutzerinterfaces und erklären Sie, wie man das Programm bedient. Versuchen Sie auch kurz zu erklären, was das Programm im Hintergrund macht. Also z.B. erst die Datei einlesen, dann das Interface laden usw

Verwendete Biblotheken:

pylint=3.2.3

coverage=7.5.3

mypy=1.10.0

cryptography=42.0.0