Q. Make a simple login form asking for username and password. Make sure that the password stored in the database is "hashed", either by MD5 or SHA256. Make verification by supplying the password.

A cryptographic hash function is an algorithm that takes an arbitrary amount of data input—a credential—and produces a fixed-size output of enciphered text called a hash value, or just "hash." That enciphered text can then be stored instead of the password itself, and later used to verify the user. Certain properties of cryptographic hash functions impact the security of password storage.

Cryptographic hashes take cleartext passwords and turn them into enciphered text for storage. Attackers who access your database are forced to decipher those hash values if they want to exploit them. In other words, hashes slow down attackers.

Some popular hash functions are:

Message Digest (MD5)

MD5 (Message Digest Algorithm 5) is a cryptographic hash algorithm that can be used to generate a 128-bit string value from a string of any length. Despite the fact that MD5 has security flaws, it is still widely used. The most common method for verifying the integrity of files is MD5. Other security protocols and applications, such as SSH, SSL, and IPSec, use it as well. Some applications improve the MD5 algorithm by adding a salt value to the plaintext or repeating the hash function.

SHA 256

In the field of cryptography and crypt analytics, the SHA-256 algorithm is a crypt-formatted hash function that generates an almost-unique 256-bit (32-byte) signature for a text. SHA-256 is one of the successor hash functions to SHA-1 (collectively referred to as SHA-2), and is one of the strongest hash functions available. SHA-256 is not much more complex to code than SHA-1, and has not yet been compromised in any way.

Description

This program has 2 major functions which are:

i. To create a user id and store the information provided.

ii. To display the message when the user id and password is provided.

In the first function, the program asks for the username and password to create a new user. When we provide that we can now enter the text we want. The password is hashed using the SSH256 algorithm. The username, password and the message is then appended to the csv file i.e. 'db.csv'. In the next function, the user is asked to enter the username and password. If the username and password is matched then the content saved by the user is displayed. If the information provided by the user is not matched then an error message is shown and the process is ended. We can also see the content of the csv file. We will see that the password field contains alphanumeric hashed content of the password.

Code

```
from hashlib import sha256
import pandas as pd
import os
import time


def hash_password(password):
    hashed_password = sha256(password.encode('utf-8')).hexdigest()
    return hashed_password


def check_password(entered_password, true_hashed):
    entered_hashed = sha256(entered_password.encode('utf-8')).hexdigest()
    return entered_hashed == true_hashed
```

```python
def option1(df):

    locker_id = input("Enter the username : ")
    pwd_entered = input("Enter the password : ")

    key = "0"
    the_locker = df.loc[df["id"] == locker_id]
    if len(the_locker) == 0:
        print("Username or Password is invalid")
        print("Back to main menu")
        time.sleep(2)
        key = input("Press anykey to go to menu : ")
    else:
        if check_password(pwd_entered, the_locker.password.values[0]):
            print("Connected to the system")
            print(f"your msg : {the_locker.msg.values[0]}")
            key = input("\n Press anykey to go to menu : ")
        else:
            print("Username or Password is invalid")
            print("Back to main menu")
            key = input("\n Press anykey to go to menu : ")
            time.sleep(2)
    return key
```

```python
def option2(df):

    locker_id = input("Enter the new username : ")
    pwd_entered = input("Enter the new password : ")
    data = {}
    locker_id_added = False
    while not locker_id_added:
        the_locker = df.loc[df["id"] == locker_id]
        if len(the_locker) != 0:
            print("Locker ID is unavailable")
            locker_id = input("Enter the id for your new locker : ")
        else:
            data["id"] = locker_id
            locker_id_added = True

    entered_hash = hash_password(pwd_entered)
    data["password"] = entered_hash

    content = input("Content you want to store in the locker : ")
    data["msg"] = content

    df = df.append(data, ignore_index=True)

    return df
```

```python
if __name__ == "__main__":

    database_name = "db.csv"

    if not os.path.isfile(database_name):
        df = pd.DataFrame(columns=["id", "password", "msg"])
        print("Creating the database")
    else:
        df = pd.read_csv(database_name)
        print("Setting up the database")

    time.sleep(1)
    while True:

        print("1. Show my Content")
        print("2. Create new id")
        print("Press 'q' to quit")

        key = input("Select a option : ")

        if key == "q":
            break

        if key == "1":
            os.system('clear')
            key = option1(df)
```

```python
        elif key == "2":
            os.system('clear')
            df = option2(df)
            df.to_csv(database_name, index=False)
        else:
            print("Invalid option")

        os.system('clear')

    df.to_csv(database_name, index=False)
```

Output

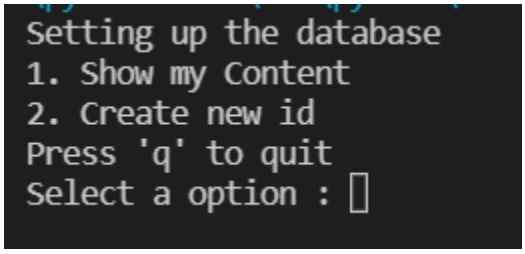

fig: setting up database

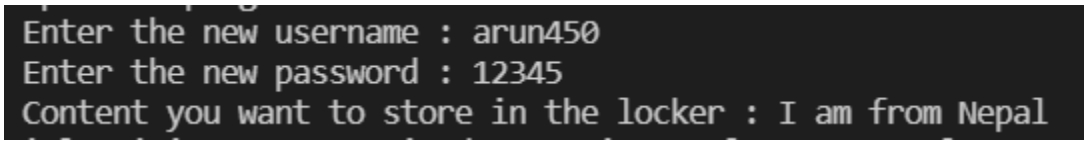

Fig:setting up new user

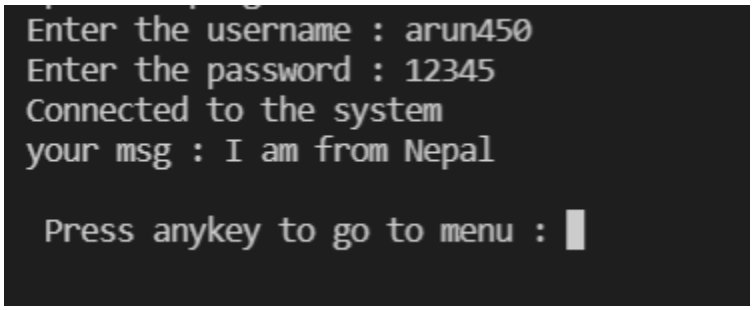

Fig: output when stored data is entered

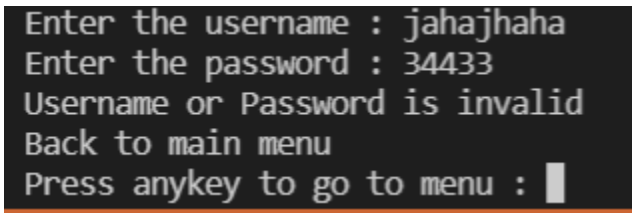

Fig: output when wrong data is entered

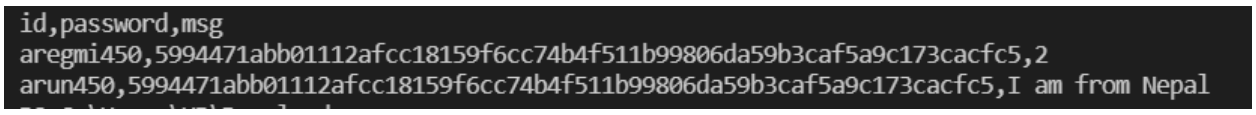

Fig: hashed password shown in the database