import sqlite3

import re

import bcrypt

import os

from time import time

# Connect to SQLite Database

database = sqlite3.connect("log\_and\_pass.db")

cursor = database.cursor()

# Create the users table if it doesn't exist

cursor.execute("""CREATE TABLE IF NOT EXISTS users (

login TEXT UNIQUE,

password TEXT,

login\_attempts INTEGER DEFAULT 0,

blocked\_until INTEGER DEFAULT 0)""")

database.commit()

# Constants

MAX\_LOGIN\_ATTEMPTS = 5

LOCKOUT\_DURATION = 15 # in minutes

# Utility Functions

def get\_choice():

"""Function to ask the user for their choice."""

return input("Вход/Регистрация/Конец? 1/2/3: ")

def is\_valid\_password(password):

"""Function to validate the strength of the password."""

if len(password) < 8:

return False

if not re.search("[A-Z]", password): # Check for at least one uppercase letter

return False

if not re.search("[0-9]", password): # Check for at least one digit

return False

if not re.search("[^a-zA-Z0-9]", password): # Check for at least one special character

return False

return True

def hash\_password(password):

"""Function to hash the password using bcrypt."""

return bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt())

def verify\_password(stored\_hash, entered\_password):

"""Function to verify entered password with stored hash."""

return bcrypt.checkpw(entered\_password.encode('utf-8'), stored\_hash.encode('utf-8'))

def register\_user():

"""Function for registering a new user."""

login = input("Введите логин: ")

password = input("Введите пароль: ")

# Validate password

if not is\_valid\_password(password):

print("Пароль не соответствует требованиям. Пароль должен содержать хотя бы 8 символов, одну заглавную букву, цифру и специальный символ.")

return

# Hash the password before saving

hashed\_password = hash\_password(password)

try:

# Insert the new user into the database

cursor.execute("INSERT INTO users (login, password) VALUES (?, ?)", (login, hashed\_password))

database.commit()

print(f"Пользователь {login} успешно зарегистрирован!")

except sqlite3.IntegrityError:

print(f"Ошибка: пользователь с логином {login} уже существует.")

except Exception as e:

print(f"Произошла ошибка при регистрации: {e}")

def login\_user():

"""Function to log in an existing user."""

login = input("Введите логин: ")

password = input("Введите пароль: ")

# Fetch the user's stored data

cursor.execute("SELECT password, login\_attempts, blocked\_until FROM users WHERE login = ?", (login,))

result = cursor.fetchone()

if result:

stored\_hashed\_password, login\_attempts, blocked\_until = result

# Check if the account is blocked

current\_time = int(time())

if current\_time < blocked\_until:

blocked\_time = time\_to\_str(blocked\_until)

print(f"Аккаунт заблокирован. Попробуйте снова после {blocked\_time}.")

return

# Verify the password

if verify\_password(stored\_hashed\_password, password):

print(f"Добро пожаловать, {login}!")

reset\_login\_attempts(login) # Reset the failed login attempts on success

else:

print("Неверный пароль.")

increment\_login\_attempts(login, login\_attempts)

else:

print("Пользователь с таким логином не найден.")

def increment\_login\_attempts(login, login\_attempts):

"""Increment the failed login attempts and possibly lock the account."""

login\_attempts += 1

if login\_attempts >= MAX\_LOGIN\_ATTEMPTS:

block\_account(login)

else:

cursor.execute("UPDATE users SET login\_attempts = ? WHERE login = ?", (login\_attempts, login))

database.commit()

def reset\_login\_attempts(login):

"""Reset the login attempts for the user after successful login."""

cursor.execute("UPDATE users SET login\_attempts = 0 WHERE login = ?", (login,))

database.commit()

def block\_account(login):

"""Block the account for a certain duration after multiple failed attempts."""

blocked\_until = int(time()) + (LOCKOUT\_DURATION \* 60) # Lock for LOCKOUT\_DURATION minutes

cursor.execute("UPDATE users SET blocked\_until = ?, login\_attempts = 0 WHERE login = ?", (blocked\_until, login))

database.commit()

print(f"Слишком много неверных попыток. Аккаунт заблокирован на {LOCKOUT\_DURATION} минут.")

def time\_to\_str(time\_int):

"""Convert a Unix timestamp to a human-readable time."""

from datetime import datetime

return datetime.utcfromtimestamp(time\_int).strftime('%Y-%m-%d %H:%M:%S')

def main():

"""Main function to control the flow of the application."""

while True:

choice = get\_choice()

if choice == '1': # Login

login\_user()

elif choice == '2': # Register

register\_user()

elif choice == '3': # Exit

print("Завершение работы.")

break

else:

print("Неверный выбор. Пожалуйста, выберите 1, 2 или 3.")

# Run the application

main()