import tkinter as tk

from tkinter import messagebox

import sqlite3

import bcrypt

import secrets

import string

from cryptography.fernet import Fernet

import os

# Generate encryption key if not exists

if not os.path.exists("secret.key"):

with open("secret.key", "wb") as key\_file:

key\_file.write(Fernet.generate\_key())

with open("secret.key", "rb") as key\_file:

key = key\_file.read()

fernet = Fernet(key)

# Database setup

conn = sqlite3.connect("database.db")

cursor = conn.cursor()

cursor.execute("""

CREATE TABLE IF NOT EXISTS users(

id INTEGER PRIMARY KEY AUTOINCREMENT,

username TEXT UNIQUE,

password\_hash TEXT

)

""")

cursor.execute("""

CREATE TABLE IF NOT EXISTS passwords(

id INTEGER PRIMARY KEY AUTOINCREMENT,

user\_id INTEGER,

account TEXT,

username TEXT,

password TEXT,

FOREIGN KEY(user\_id) REFERENCES users(id)

)

""")

conn.commit()

# Application class

class PasswordManager:

def \_\_init\_\_(self, master):

self.master = master

self.master.title("Password Manager")

self.master.geometry("400x300")

self.user\_id = None

self.login\_screen()

def clear\_screen(self):

for widget in self.master.winfo\_children():

widget.destroy()

def login\_screen(self):

self.clear\_screen()

tk.Label(self.master, text="Login", font=("Arial", 16)).pack(pady=10)

tk.Label(self.master, text="Username").pack()

username\_entry = tk.Entry(self.master)

username\_entry.pack()

tk.Label(self.master, text="Password").pack()

password\_entry = tk.Entry(self.master, show="\*")

password\_entry.pack()

def login():

username = username\_entry.get()

password = password\_entry.get()

cursor.execute("SELECT id, password\_hash FROM users WHERE username=?", (username,))

result = cursor.fetchone()

if result and bcrypt.checkpw(password.encode(), result[1]):

self.user\_id = result[0]

self.dashboard()

else:

messagebox.showerror("Error", "Invalid credentials")

def register():

self.register\_screen()

tk.Button(self.master, text="Login", command=login).pack(pady=5)

tk.Button(self.master, text="Register", command=register).pack()

def register\_screen(self):

self.clear\_screen()

tk.Label(self.master, text="Register", font=("Arial", 16)).pack(pady=10)

tk.Label(self.master, text="Username").pack()

username\_entry = tk.Entry(self.master)

username\_entry.pack()

tk.Label(self.master, text="Password").pack()

password\_entry = tk.Entry(self.master, show="\*")

password\_entry.pack()

def register\_user():

username = username\_entry.get()

password = password\_entry.get()

hashed\_pw = bcrypt.hashpw(password.encode(), bcrypt.gensalt())

try:

cursor.execute("INSERT INTO users(username, password\_hash) VALUES (?,?)", (username, hashed\_pw))

conn.commit()

messagebox.showinfo("Success", "User registered successfully")

self.login\_screen()

except sqlite3.IntegrityError:

messagebox.showerror("Error", "Username already exists")

tk.Button(self.master, text="Register", command=register\_user).pack(pady=5)

tk.Button(self.master, text="Back to Login", command=self.login\_screen).pack()

def dashboard(self):

self.clear\_screen()

tk.Label(self.master, text="Password Manager Dashboard", font=("Arial", 14)).pack(pady=10)

tk.Button(self.master, text="Add Password", command=self.add\_password\_screen).pack(pady=5)

tk.Button(self.master, text="View Passwords", command=self.view\_passwords).pack(pady=5)

tk.Button(self.master, text="Logout", command=self.login\_screen).pack(pady=5)

def add\_password\_screen(self):

self.clear\_screen()

tk.Label(self.master, text="Add Password", font=("Arial", 16)).pack(pady=10)

tk.Label(self.master, text="Account").pack()

account\_entry = tk.Entry(self.master)

account\_entry.pack()

tk.Label(self.master, text="Username").pack()

username\_entry = tk.Entry(self.master)

username\_entry.pack()

tk.Label(self.master, text="Password").pack()

password\_entry = tk.Entry(self.master)

password\_entry.pack()

def generate\_password():

chars = string.ascii\_letters + string.digits + string.punctuation

password = ''.join(secrets.choice(chars) for \_ in range(12))

password\_entry.delete(0, tk.END)

password\_entry.insert(0, password)

def save\_password():

account = account\_entry.get()

uname = username\_entry.get()

pwd = fernet.encrypt(password\_entry.get().encode())

cursor.execute("INSERT INTO passwords(user\_id, account, username, password) VALUES (?,?,?,?)",

(self.user\_id, account, uname, pwd))

conn.commit()

messagebox.showinfo("Success", "Password saved successfully")

self.dashboard()

tk.Button(self.master, text="Generate Password", command=generate\_password).pack(pady=5)

tk.Button(self.master, text="Save", command=save\_password).pack(pady=5)

tk.Button(self.master, text="Back", command=self.dashboard).pack()

def view\_passwords(self):

self.clear\_screen()

tk.Label(self.master, text="Stored Passwords", font=("Arial", 16)).pack(pady=10)

cursor.execute("SELECT account, username, password FROM passwords WHERE user\_id=?", (self.user\_id,))

rows = cursor.fetchall()

for account, uname, pwd in rows:

decrypted\_pwd = fernet.decrypt(pwd).decode()

tk.Label(self.master, text=f"{account} | {uname} | {decrypted\_pwd}").pack()

tk.Button(self.master, text="Back", command=self.dashboard).pack(pady=5)

root = tk.Tk()

app = PasswordManager(root)

root.mainloop()