**Iteration 3**

import tkinter as tk

from tkinter import messagebox

import sqlite3

import bcrypt

# Database setup

def setup\_database():

    conn = sqlite3.connect("secure\_users.db")

    cursor = conn.cursor()

    cursor.execute("""

        CREATE TABLE IF NOT EXISTS users (

            id INTEGER PRIMARY KEY AUTOINCREMENT,

            username TEXT UNIQUE NOT NULL,

            password\_hash TEXT NOT NULL

        )

    """)

    conn.commit()

    # Create admin account if it doesn't exist

    cursor.execute("SELECT \* FROM users WHERE username = 'admin'")

    if not cursor.fetchone():

        admin\_password = bcrypt.hashpw("admin123".encode('utf-8'), bcrypt.gensalt()).decode('utf-8')

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

        conn.commit()

    conn.close()

# Application logic

def register\_user(username, password):

    if len(password) < 8:

        return False, "Password must be at least 8 characters long."

    conn = sqlite3.connect("secure\_users.db")

    cursor = conn.cursor()

    try:

        password\_hash = bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt()).decode('utf-8')

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

        conn.commit()

        conn.close()

        return True, "Account created successfully!"

    except sqlite3.IntegrityError:

        conn.close()

        return False, "Username already exists."

def login\_user(username, password):

    conn = sqlite3.connect("secure\_users.db")

    cursor = conn.cursor()

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

    result = cursor.fetchone()

    conn.close()

    if result and bcrypt.checkpw(password.encode('utf-8'), result[0].encode('utf-8')):

        return True

    return False

def delete\_account(username):

    conn = sqlite3.connect("secure\_users.db")

    cursor = conn.cursor()

    cursor.execute("DELETE FROM users WHERE username = ?", (username,))

    conn.commit()

    conn.close()

# UI setup

class LoginApp:

    def \_\_init\_\_(self, root):

        self.root = root

        self.root.title("Secure Login Simulation")

        self.logged\_in\_user = None

        self.main\_frame = tk.Frame(self.root)

        self.main\_frame.pack(pady=20)

        self.username\_label = tk.Label(self.main\_frame, text="Username:")

        self.username\_label.grid(row=0, column=0, pady=5)

        self.username\_entry = tk.Entry(self.main\_frame)

        self.username\_entry.grid(row=0, column=1, pady=5)

        self.password\_label = tk.Label(self.main\_frame, text="Password:")

        self.password\_label.grid(row=1, column=0, pady=5)

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

        self.password\_entry.grid(row=1, column=1, pady=5)

        self.login\_button = tk.Button(self.main\_frame, text="Login", command=self.login)

        self.login\_button.grid(row=2, column=0, pady=5)

        self.register\_button = tk.Button(self.main\_frame, text="Register", command=self.register)

        self.register\_button.grid(row=2, column=1, pady=5)

    def login(self):

        username = self.username\_entry.get()

        password = self.password\_entry.get()

        if login\_user(username, password):

            self.logged\_in\_user = username

            messagebox.showinfo("Login Successful", f"Welcome, {username}!")

            self.show\_account\_page()

        else:

            messagebox.showerror("Login Failed", "Invalid username or password.")

    def register(self):

        username = self.username\_entry.get()

        password = self.password\_entry.get()

        success, message = register\_user(username, password)

        if success:

            messagebox.showinfo("Registration Successful", message)

        else:

            messagebox.showerror("Registration Failed", message)

    def show\_account\_page(self):

        for widget in self.main\_frame.winfo\_children():

            widget.destroy()

        tk.Label(self.main\_frame, text=f"Logged in as: {self.logged\_in\_user}").pack(pady=10)

        tk.Button(self.main\_frame, text="Delete Account", command=self.delete\_account).pack(pady=5)

        tk.Button(self.main\_frame, text="Logout", command=self.logout).pack(pady=5)

    def delete\_account(self):

        if messagebox.askyesno("Confirm Deletion", "Are you sure you want to delete your account?"):

            delete\_account(self.logged\_in\_user)

            messagebox.showinfo("Account Deleted", "Your account has been deleted.")

            self.logout()

    def logout(self):

        self.logged\_in\_user = None

        for widget in self.main\_frame.winfo\_children():

            widget.destroy()

        self.\_\_init\_\_(self.root)

# Initialize the application

if \_\_name\_\_ == "\_\_main\_\_":

    setup\_database()

    root = tk.Tk()

    app = LoginApp(root)

    root.mainloop()