PD LAB

ASSIGNMENT - 8

Name: Raunak Thanawala

Registration Number: 231070051

Branch: Computer Engineering

Batch: 3

**Aim:-**

Write an application to connect to database in python

**Theory:-**

* A database is an organized collection of data stored electronically, which allows for easy access, management, and updating.
* Databases store data in a structured format, making it easier to retrieve, manipulate, and manage large volumes of information efficiently.
* Types of Databases:
  + Relational Databases (RDBMS):
    - Data is organized in tables (rows and columns).
    - Tables can be linked using relationships.
    - Popular RDBMS: MySQL, PostgreSQL, Oracle, SQLite.
  + Non-Relational Databases (NoSQL):
    - Data is stored in formats other than tables (e.g., JSON, key-value pairs).
    - Suited for large-scale distributed data.
    - Examples: MongoDB, Redis, Cassandra.
* SQL is the standard programming language used to manage and manipulate relational databases.
* It allows you to interact with the database by performing various operations such as creating tables, inserting data, querying, updating, and deleting records.
* Advantages of SQL and Databases:
  + Efficient Data Management:
    - Databases allow for storing vast amounts of data in an organized and easily accessible way.
  + Data Integrity:
    - Constraints like primary keys and foreign keys ensure data consistency and prevent data duplication.
  + Scalability:
    - Databases, especially relational ones, can scale well as data grows, providing robust performance.
  + Security:
    - SQL databases offer various mechanisms like user roles, permissions, and encryption to secure sensitive data.
  + Data Manipulation:
    - SQL allows for powerful data operations such as filtering, sorting, aggregating, and grouping data.

**Code and Output:**

CODE:

import tkinter as tk

from tkinter import messagebox

import sqlite3

*# Create or connect to a database*

conn = sqlite3.connect('users.db')

c = conn.cursor()

*# Create a table for users if it doesn't exist (now includes email)*

c.execute('''

CREATE TABLE IF NOT EXISTS users (

id INTEGER PRIMARY KEY,

username TEXT UNIQUE,

email TEXT UNIQUE,

password TEXT

)

''')

conn.commit()

*# Create a table for user details if it doesn't exist*

c.execute('''

CREATE TABLE IF NOT EXISTS user\_details (

user\_id INTEGER PRIMARY KEY,

address TEXT,

phone\_number TEXT,

company\_name TEXT,

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

)

''')

conn.commit()

*# Function to register a user*

def register\_user(username, email, password):

if username and email and password:

try:

c.execute('INSERT INTO users (username, email, password) VALUES (?, ?, ?)',

(username, email, password))

conn.commit()

messagebox.showinfo("Success", "Registration successful!")

register\_window.destroy() *# Close the registration window*

except sqlite3.IntegrityError:

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

else:

messagebox.showwarning("Warning", "Please fill out all fields.")

*# Function to open registration window*

def open\_registration():

*global* register\_window

register\_window = tk.Toplevel(root)

register\_window.title("Register")

register\_window.geometry("400x400")

register\_window.configure(bg="#ECDFCC")

tk.Label(register\_window, text="Register:", font=("Cascadia Code", 24, "bold"), bg="#ECDFCC").pack(pady=10)

tk.Label(register\_window, text="Username:", bg="#ECDFCC", font=("", 16, "bold")).pack(pady=5)

username\_entry = tk.Entry(register\_window, width=30)

username\_entry.pack(pady=5)

tk.Label(register\_window, text="Email:", bg="#ECDFCC", font=("", 16, "bold")).pack(pady=5)

email\_entry = tk.Entry(register\_window, width=30)

email\_entry.pack(pady=5)

tk.Label(register\_window, text="Password:", bg="#ECDFCC", font=("", 16, "bold")).pack(pady=5)

password\_entry = tk.Entry(register\_window, width=30, show='\*')

password\_entry.pack(pady=5)

tk.Button(register\_window, text="Register", bg="#EC8305", fg="white",

command=lambda: register\_user(username\_entry.get(), email\_entry.get(), password\_entry.get())).pack(pady=10)

*# Function to open the user details page*

def open\_user\_details(user\_id):

details\_window = tk.Toplevel(root)

details\_window.title("User Details")

details\_window.geometry("400x400")

details\_window.configure(bg="#ECDFCC")

*# Fetch existing details if any*

c.execute('SELECT \* FROM user\_details WHERE user\_id=?', (user\_id,))

details = c.fetchone()

*# Create form labels and fields for address, phone number, and company name*

tk.Label(details\_window, text="Address:", bg="#ECDFCC", font=("", 14, "bold")).pack(pady=5)

address\_entry = tk.Entry(details\_window, width=30)

address\_entry.pack(pady=5)

tk.Label(details\_window, text="Phone Number:", bg="#ECDFCC", font=("", 14, "bold")).pack(pady=5)

phone\_entry = tk.Entry(details\_window, width=30)

phone\_entry.pack(pady=5)

tk.Label(details\_window, text="Company Name:", bg="#ECDFCC", font=("", 14, "bold")).pack(pady=5)

company\_entry = tk.Entry(details\_window, width=30)

company\_entry.pack(pady=5)

*# Pre-fill the fields if details exist*

if details:

address\_entry.insert(0, details[1])

phone\_entry.insert(0, details[2])

company\_entry.insert(0, details[3])

*# Function to save or update details*

def save\_details():

address = address\_entry.get()

phone\_number = phone\_entry.get()

company\_name = company\_entry.get()

c.execute('SELECT \* FROM user\_details WHERE user\_id=?', (user\_id,))

existing\_details = c.fetchone()

if existing\_details:

*# Update the existing user details*

c.execute('UPDATE user\_details SET address=?, phone\_number=?, company\_name=? WHERE user\_id=?',

(address, phone\_number, company\_name, user\_id))

else:

*# Insert new details if not already existing*

c.execute('INSERT INTO user\_details (user\_id, address, phone\_number, company\_name) VALUES (?, ?, ?, ?)',

(user\_id, address, phone\_number, company\_name))

conn.commit()

messagebox.showinfo("Success", "Details updated successfully!")

*# Save button*

tk.Button(details\_window, text="Save Details", command=save\_details, bg="#EC8305", fg="white").pack(pady=10)

*# Function to login a user*

def login():

username = username\_entry.get()

password = password\_entry.get()

c.execute('SELECT id FROM users WHERE username=? AND password=?', (username, password))

result = c.fetchone()

if result:

user\_id = result[0]

messagebox.showinfo("Success", "Login successful!")

open\_user\_details(user\_id) *# Open user details page after login*

else:

messagebox.showerror("Error", "Invalid username or password.")

*# Function to toggle password visibility*

def toggle\_password():

if password\_entry.cget('show') == '\*':

password\_entry.config(show='')

reveal\_button.config(text='Hide Pass')

else:

password\_entry.config(show='\*')

reveal\_button.config(text='Reveal Pass')

*# Function to display all users*

def show\_users():

c.execute('SELECT username FROM users')

users = c.fetchall()

if users:

user\_list = "\n".join([user[0] for user in users])

messagebox.showinfo("Current Users", user\_list)

else:

messagebox.showinfo("Current Users", "No users found.")

*# Function to clear all users*

def clear\_users():

confirm = messagebox.askyesno("Confirm", "Are you sure you want to delete all users?")

if confirm:

c.execute('DELETE FROM users')

conn.commit()

messagebox.showinfo("Success", "All users have been cleared.")

*# Create the main window*

root = tk.Tk()

root.title("Login Page")

root.geometry("700x500")

root.configure(bg="#ECDFCC")

title\_label = tk.Label(root, text="User Login Page", font=("Cascadia Code", 32, "bold"), bg="#ECDFCC")

title\_label.pack(pady=10)

*# Create and place the username label and entry*

label\_username = tk.Label(root, text="Username:", font=("", 16, "bold"), bg="#ECDFCC")

label\_username.pack(pady=5)

username\_entry = tk.Entry(root, width=30)

username\_entry.pack(pady=5)

*# Create and place the password label and entry*

label\_password = tk.Label(root, text="Password:", font=("", 16, "bold"), bg="#ECDFCC")

label\_password.pack(pady=5)

password\_entry = tk.Entry(root, width=30, show='\*')

password\_entry.pack(pady=5)

*# Create and place the buttons*

frame = tk.Frame(root)

frame.pack(pady=20)

frame.configure(bg="#ECDFCC")

register\_button = tk.Button(frame, text="Register", command=open\_registration, width=10, bg="#EC8305", fg="white")

register\_button.grid(row=0, column=1, padx=5)

login\_button = tk.Button(frame, text="Login", command=login, bg="#77CDFF")

login\_button.grid(row=0, column=0, padx=5)

*# Create and place the reveal password button*

reveal\_button = tk.Button(root, text="Reveal Pass", command=toggle\_password, width=8, bg="#C62E2E", fg="white")

reveal\_button.pack(pady=5)

*# Create and place the show users and clear users buttons*

show\_users\_button = tk.Button(root, text="Show Users", command=show\_users, bg="#00FF9C")

show\_users\_button.pack(pady=5)

clear\_users\_button = tk.Button(root, text="Clear Users", command=clear\_users, bg="#FFE700")

clear\_users\_button.pack(pady=5)

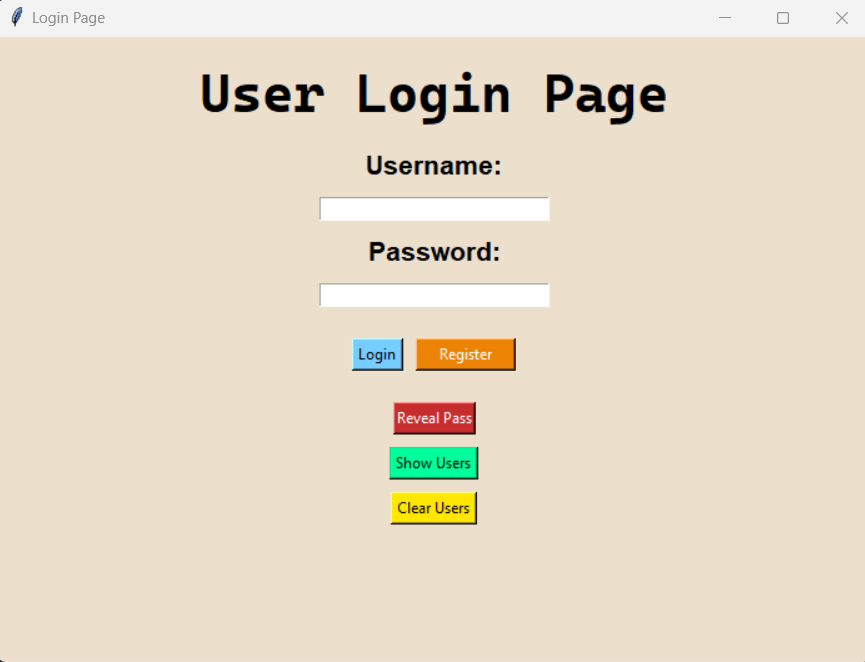
*# Start the Tkinter main loop*

root.mainloop()

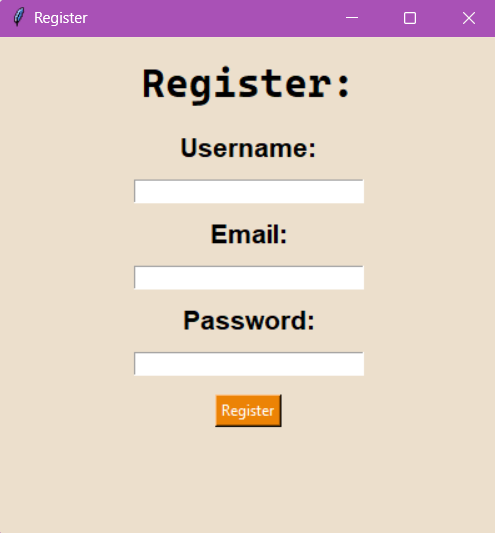
*# Close the database connection when the app is closed*

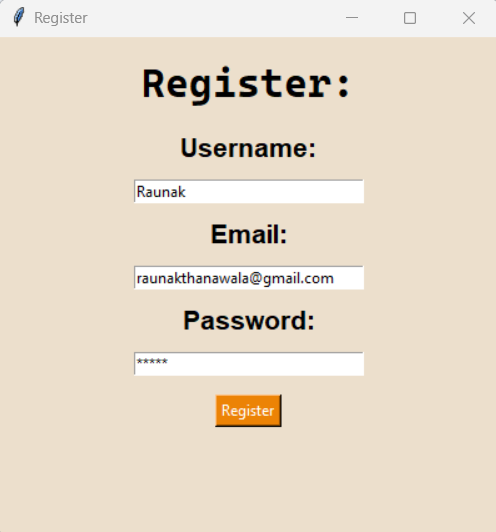
conn.close()

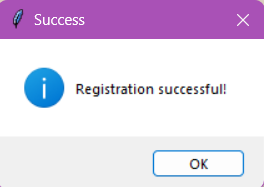
OUTPUT:

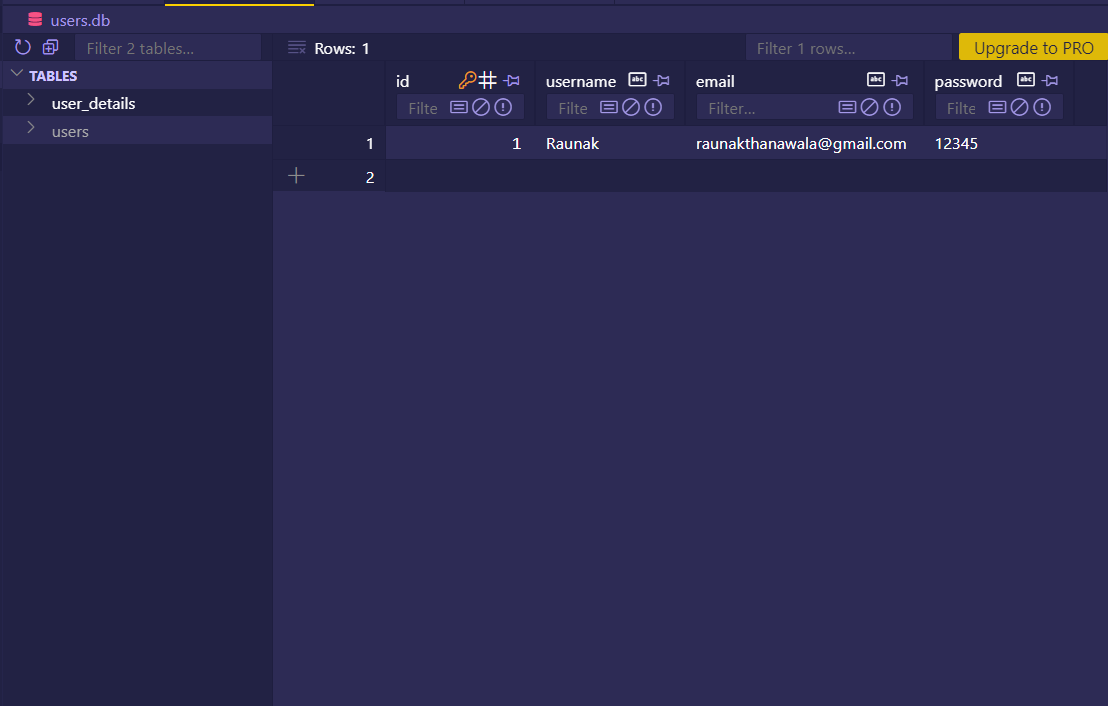


When you run the program this window opens

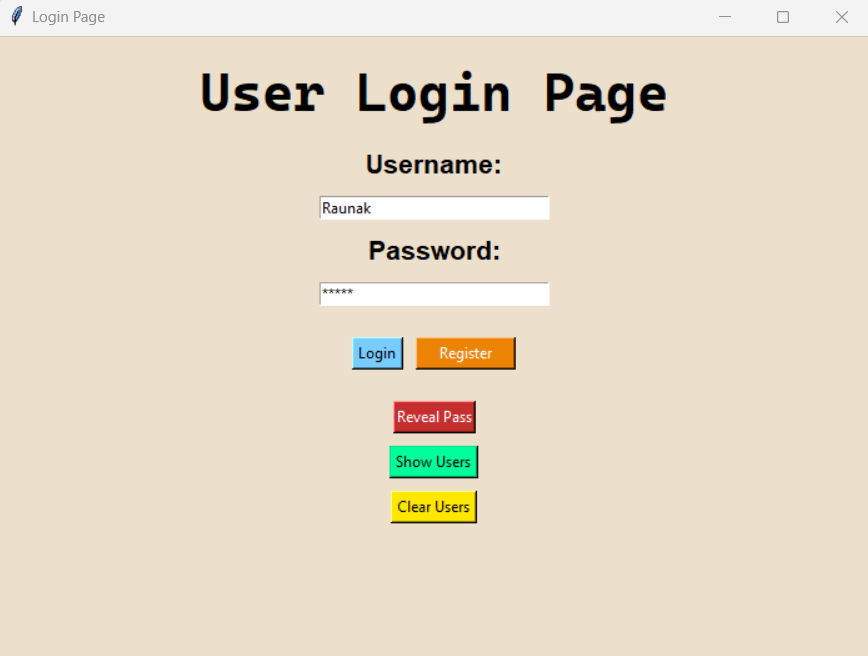
  
On clicking register this window opens

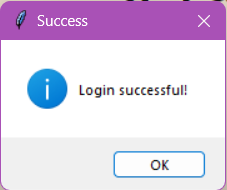


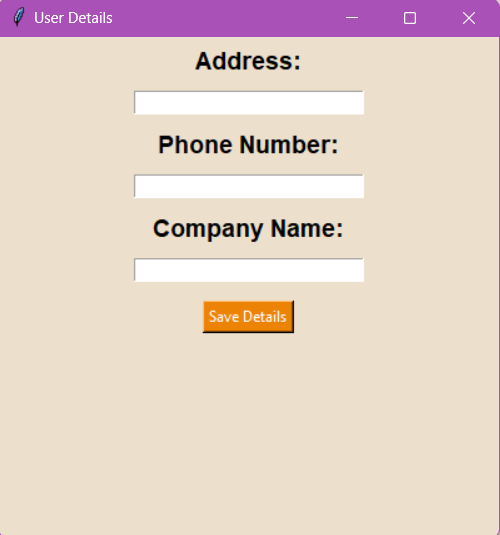


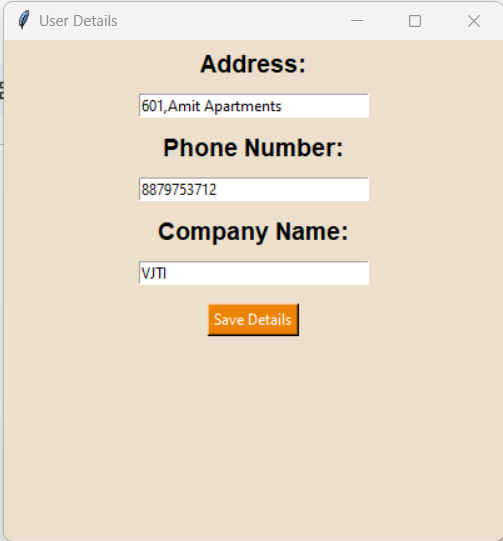


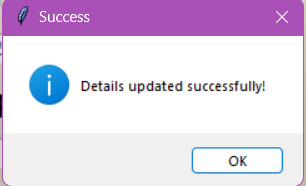
After registering the db file looks like this

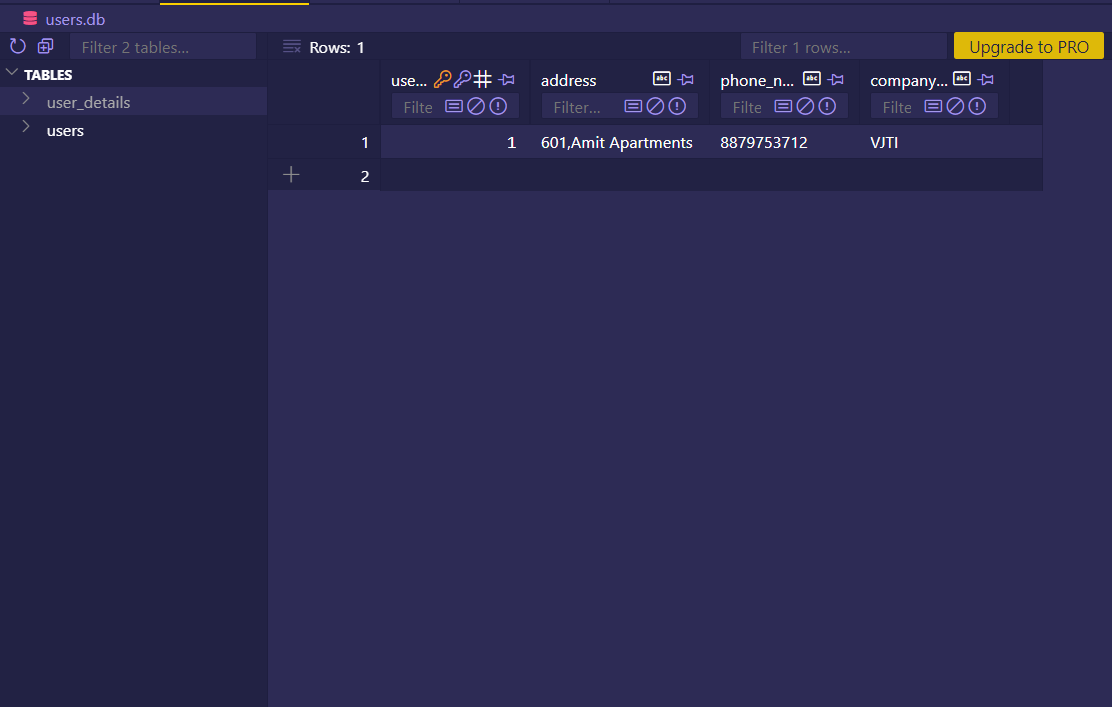




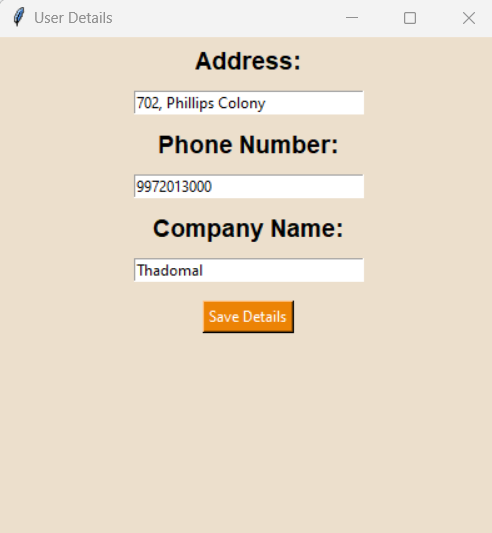
  
Window opened after logging in with valid username and password from database

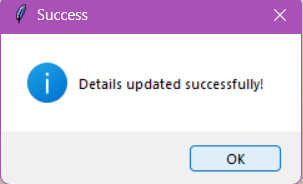


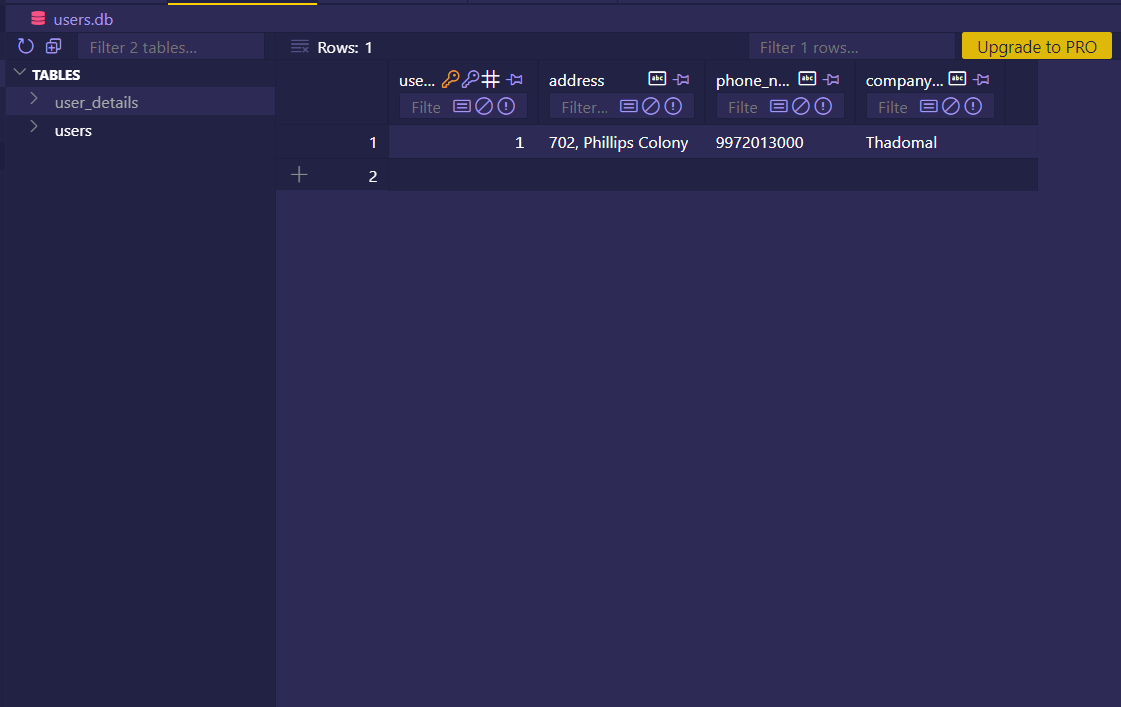


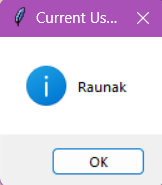


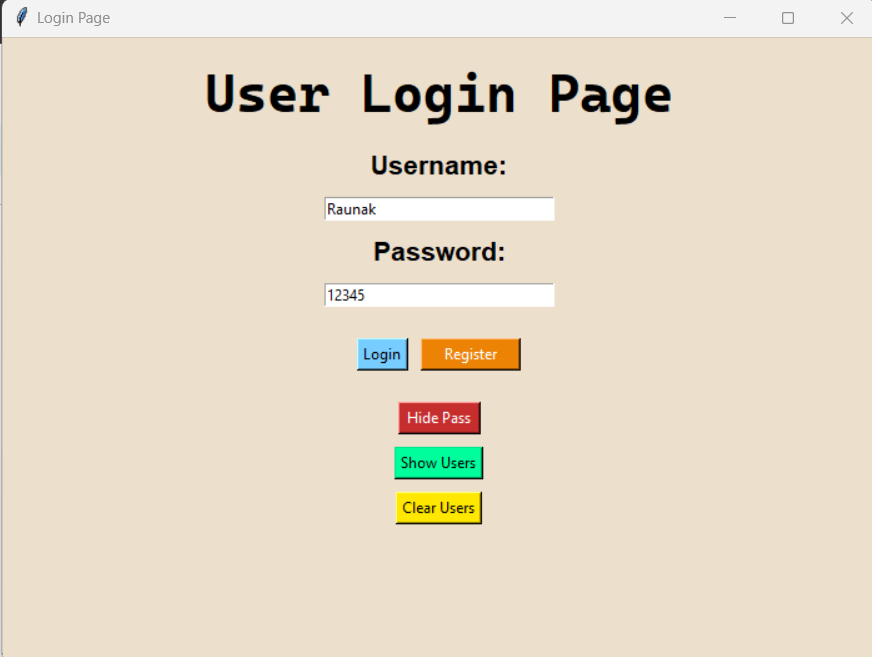
New table created in database for storing these values



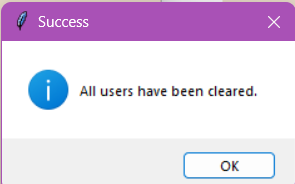


  
Table in database updated according to the edited values

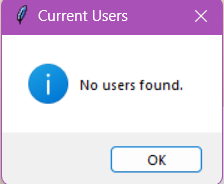
  
What pressing show all users does

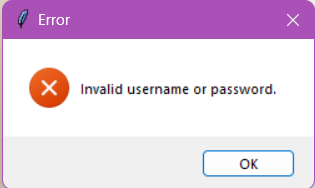


What reveal Password button does



What clear users does

  
Show users after clearing users



Logging in with username/ password not from database

**Conclusion:**

Thus we have written a program to make a basic sign up and login page using tkinter and SQL where when we sign up/register we save our details in a table of a database and then we login by checking if the details entered are in the database.

After logging in we go to the user page where the user can store their address, company name and phone number in a different table of the database.

If we want to edit these values we can just re - enter them and click on Save Details button which changes the values of that entry in the table.