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
```

```python
# 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()
```

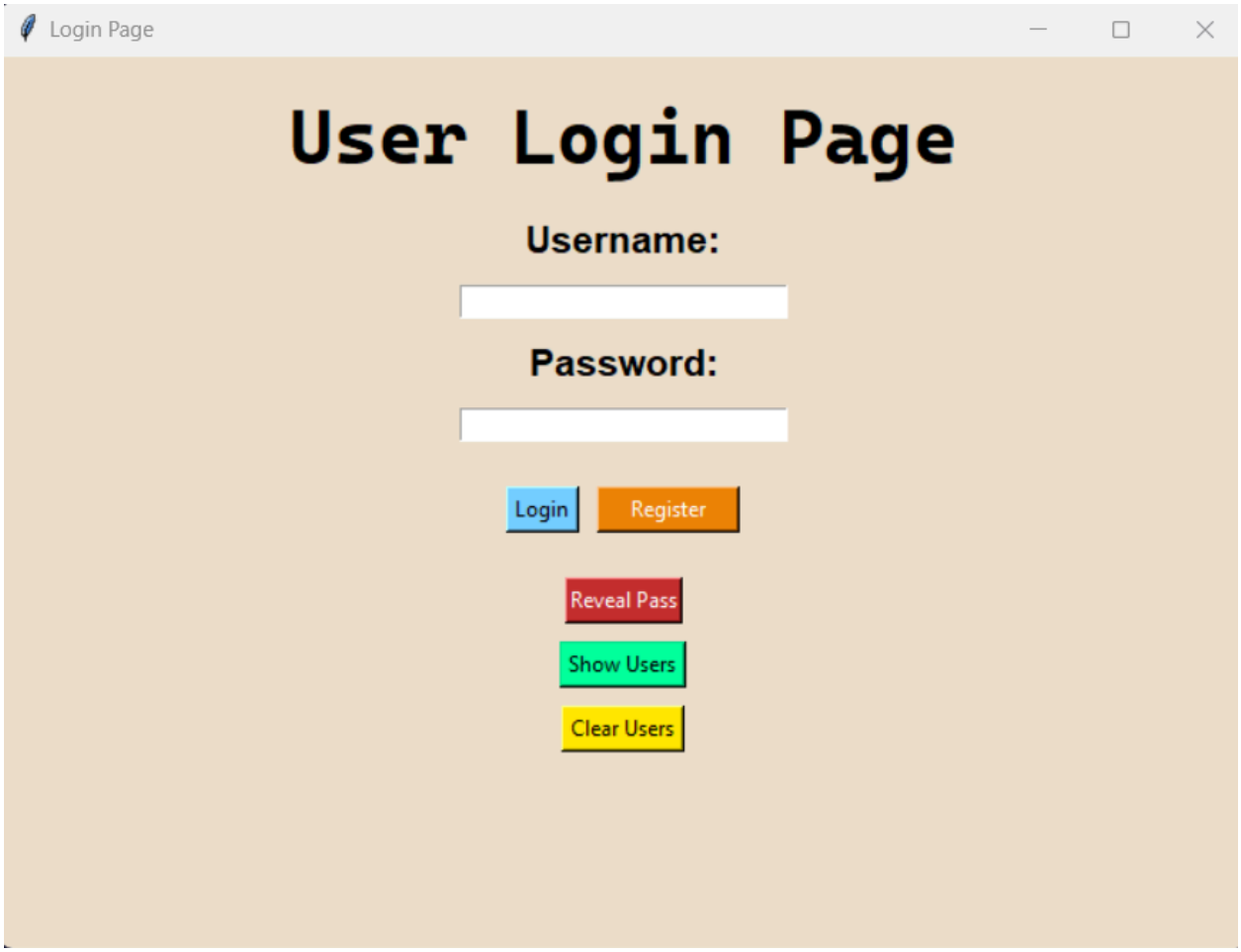
```
        if existing_details:
            # Update the existing user details
            c.execute('UPDATE user_details SET address=?, phone_number=?,
company_name=? WHERE user_id=?',
                      (address, phone_number, company_name, user_id))
        else:
            # Insert new details 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
```

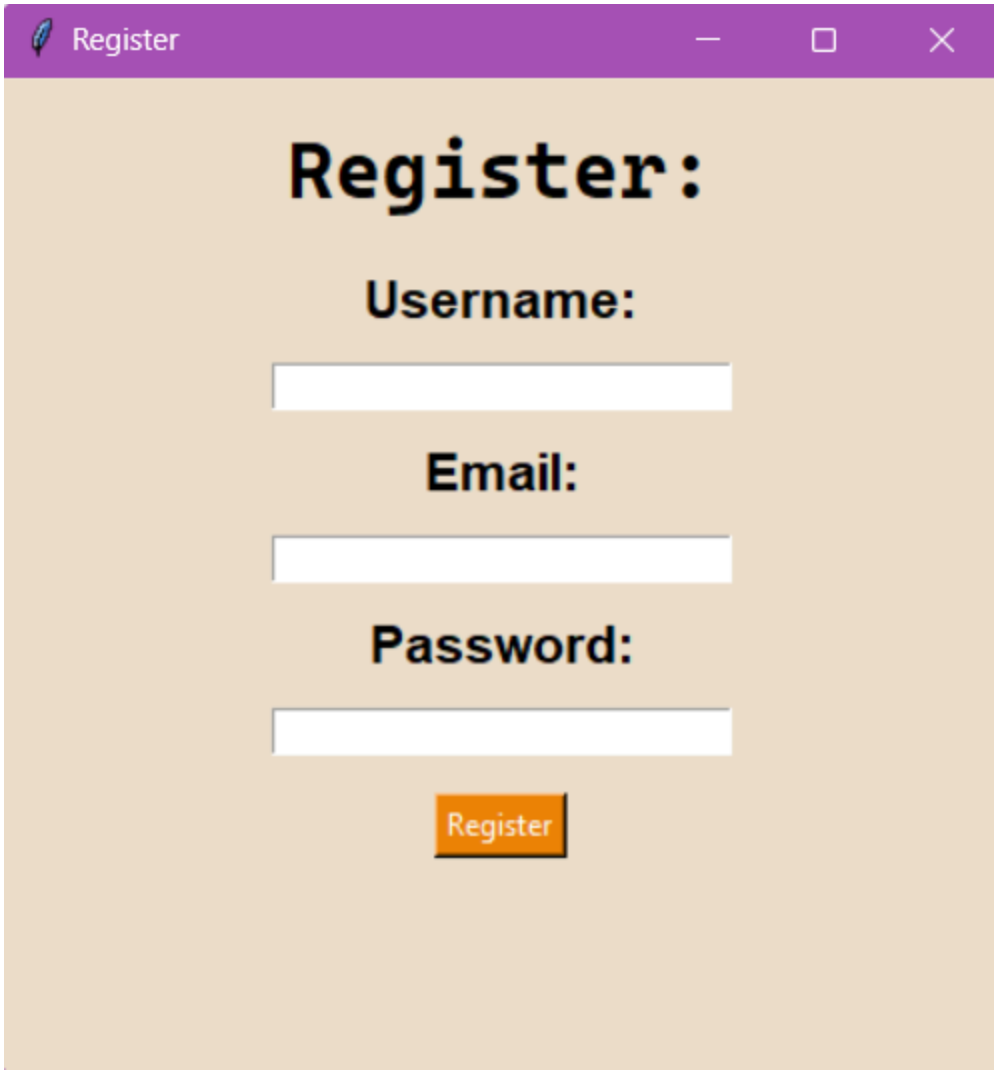
```
label_username = tk.Label(root, text="Username:", font=("", 16, "bold"),
bg="#ECDFCC")
label_username.pack(pady=5)
username_entry = tk.Entry(root, width=30)
username_entry.pack(pady=5)

# 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()
```

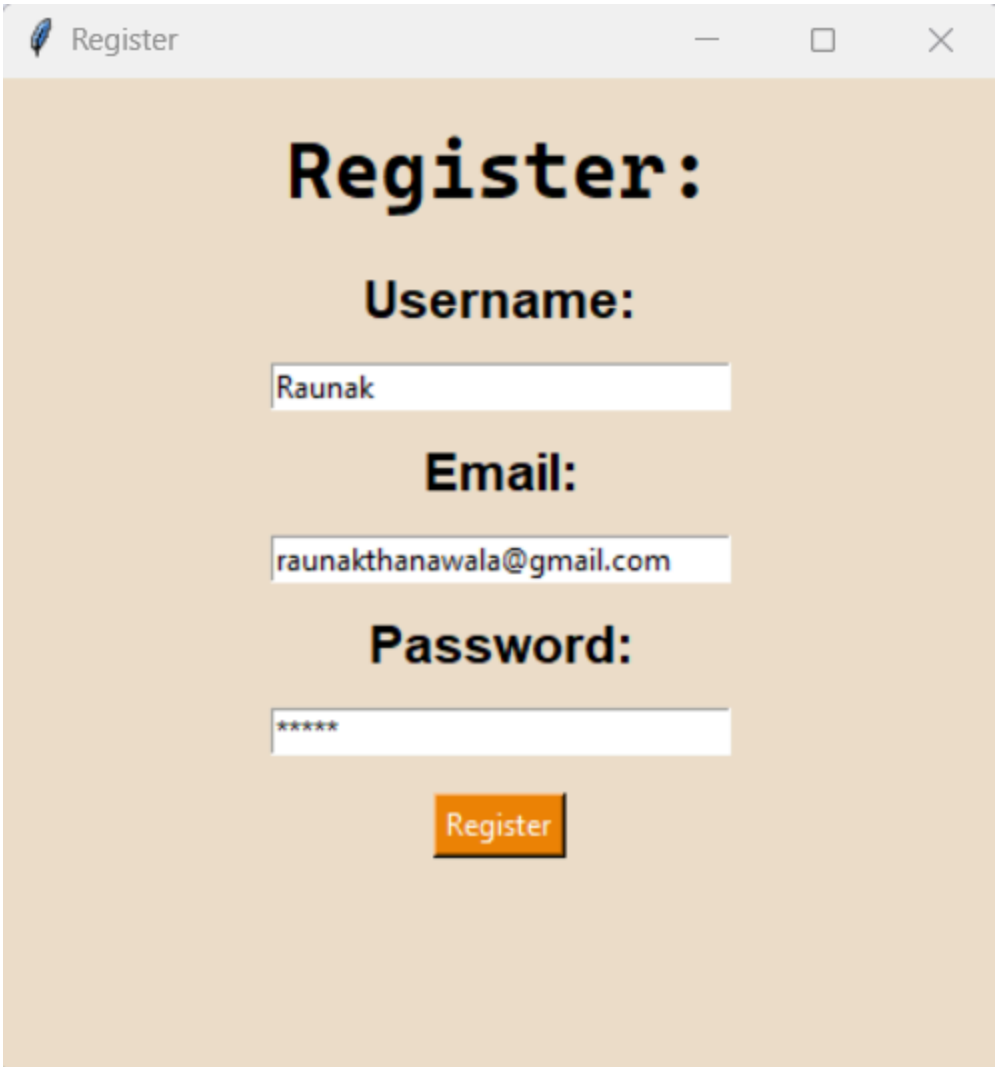
OUTPUT:

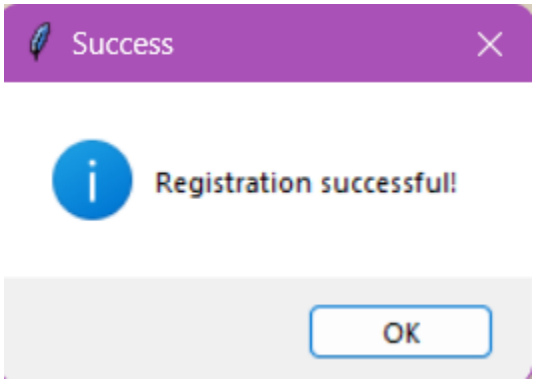


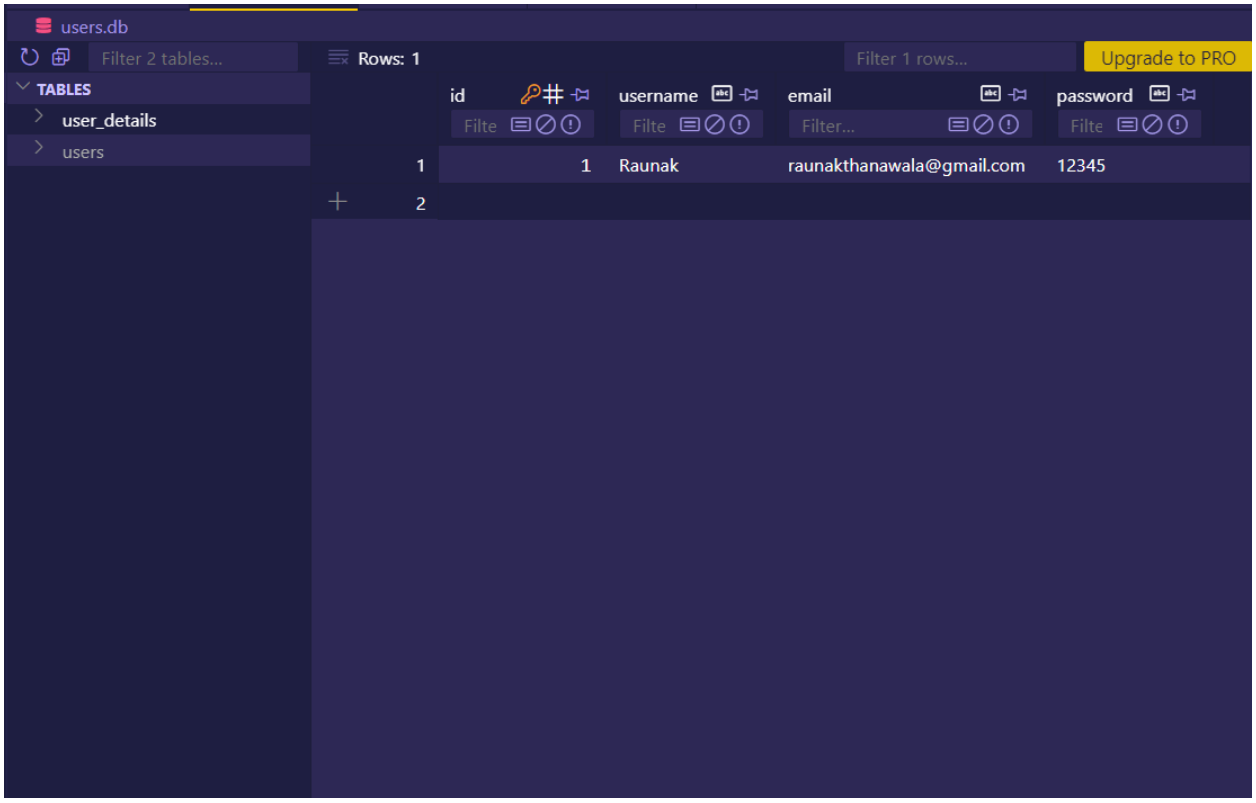
When you run the program this window opens



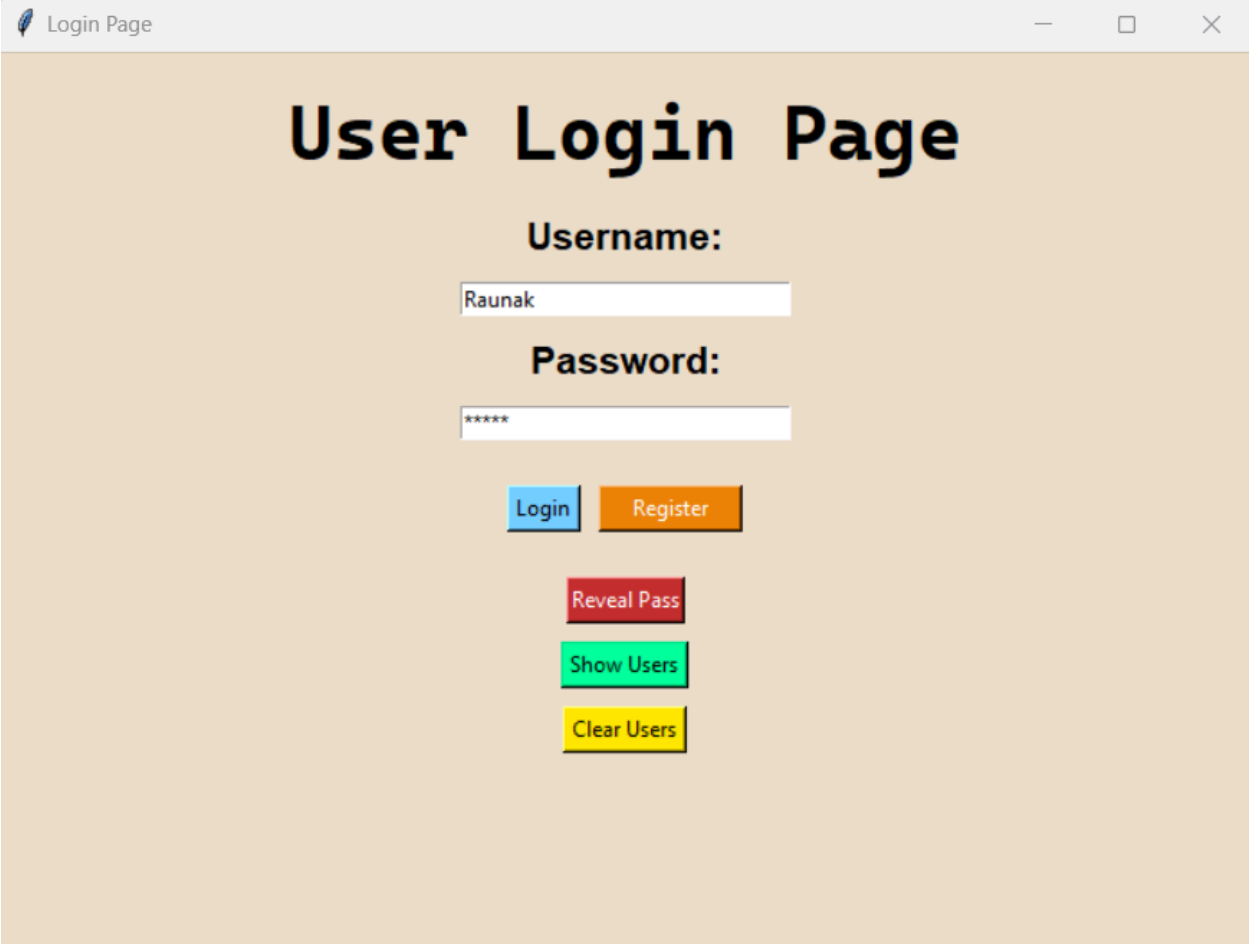
On clicking register this window opens

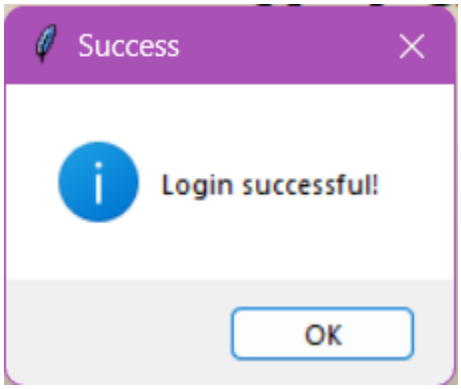

Register
Register:
Username:
Raunak
Email:
raunakthanawala@gmail.com
Password:
*****
Register

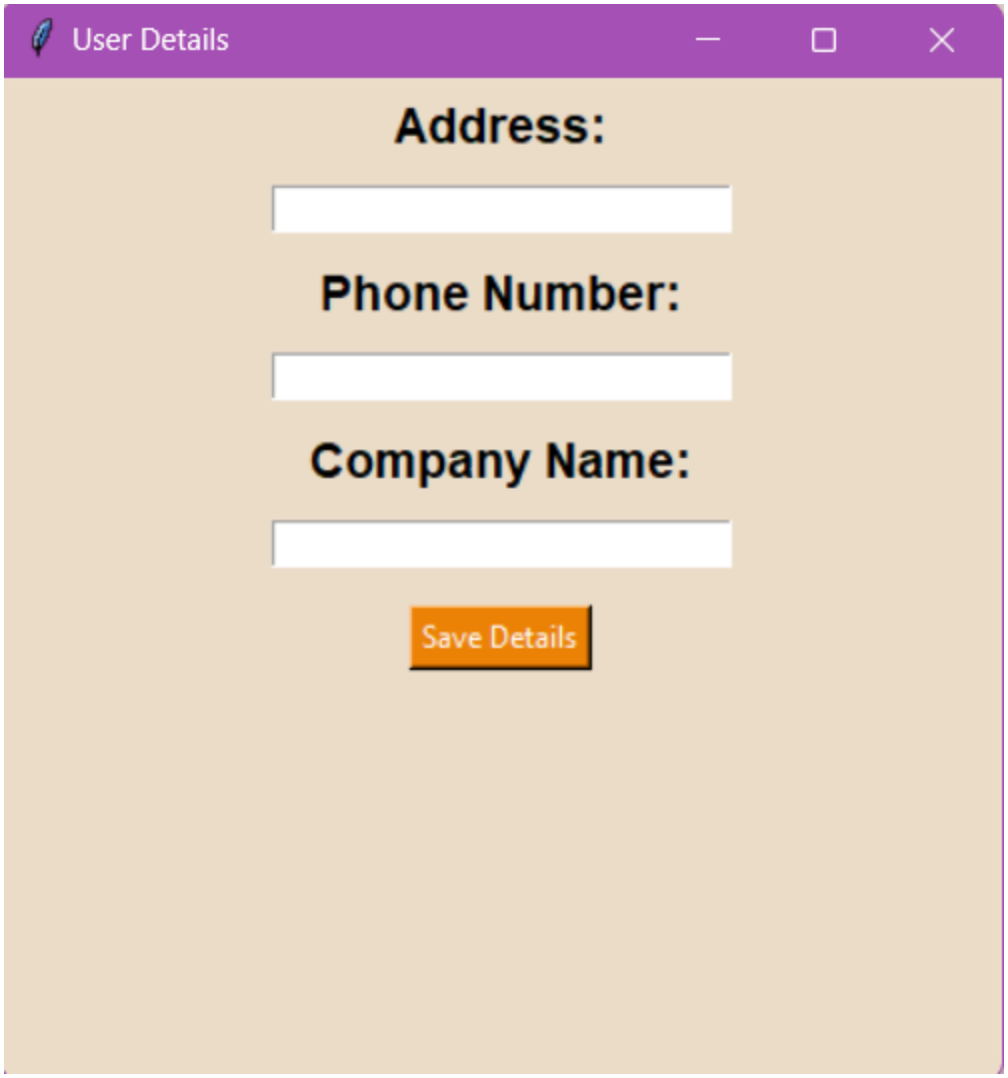

Success
Registration successful!
OK



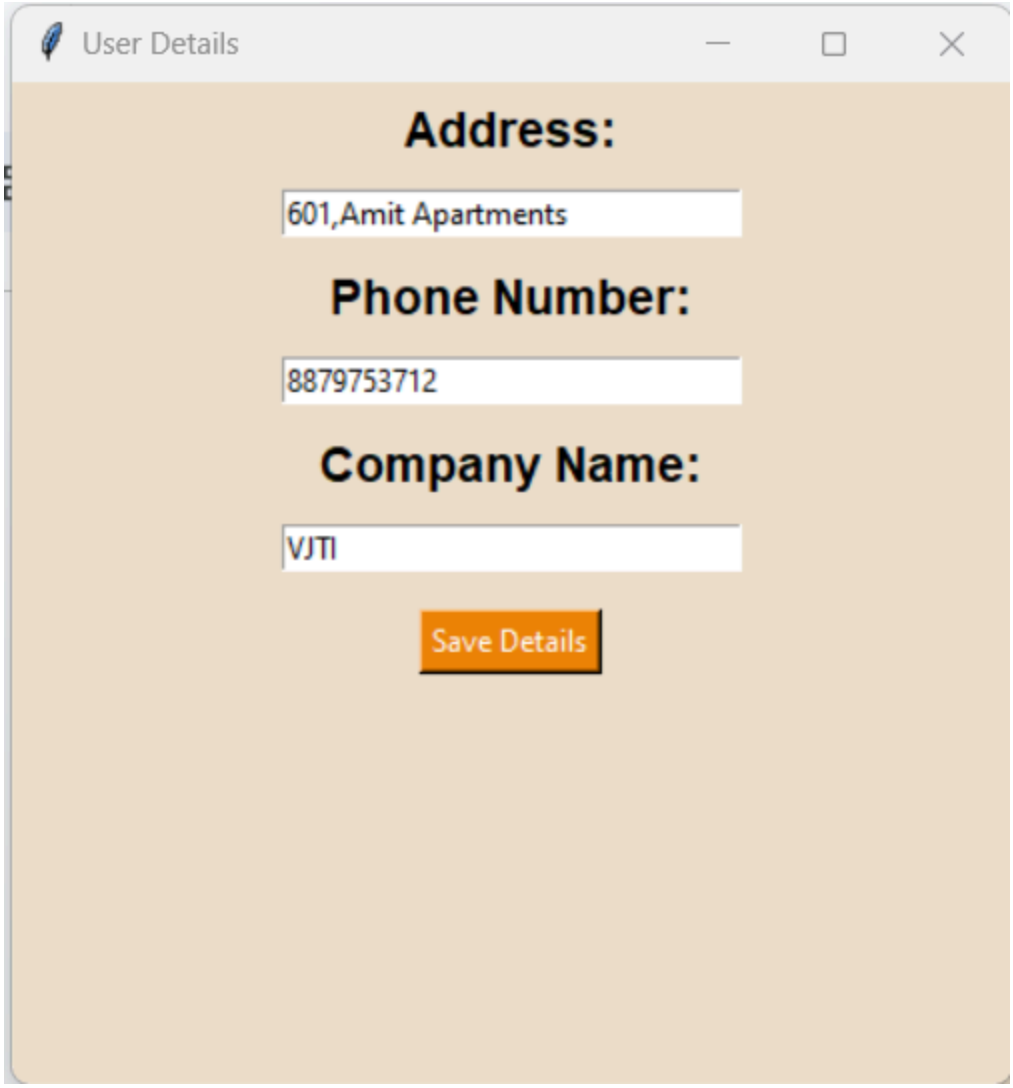
After registering the db file looks like this

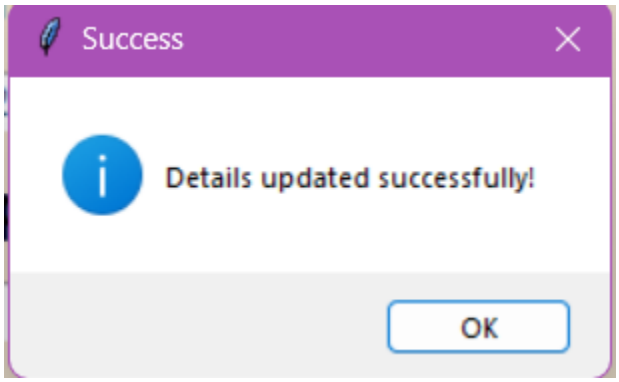

Login Page
User Login Page
Username:
Raunak
Password:
*****
Login
Register
Reveal Pass
Show Users
Clear Users

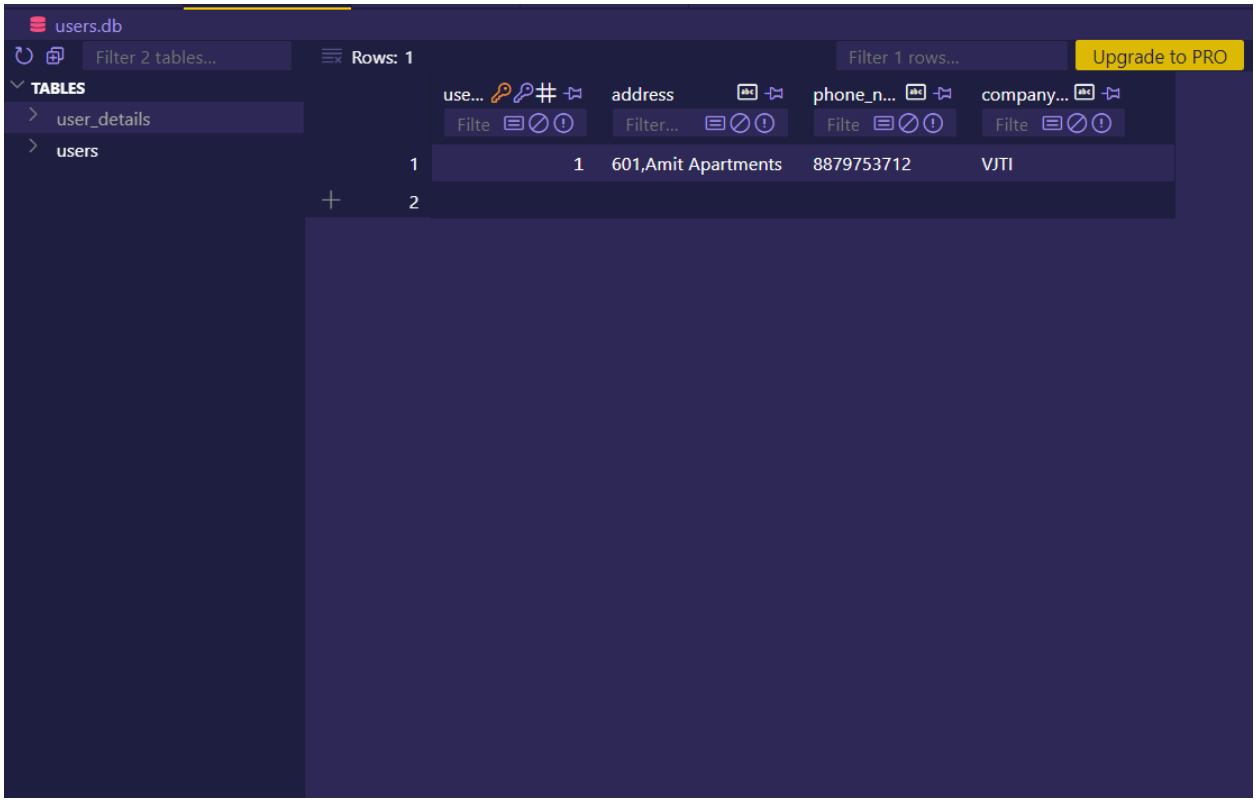

Success
Login successful!
OK



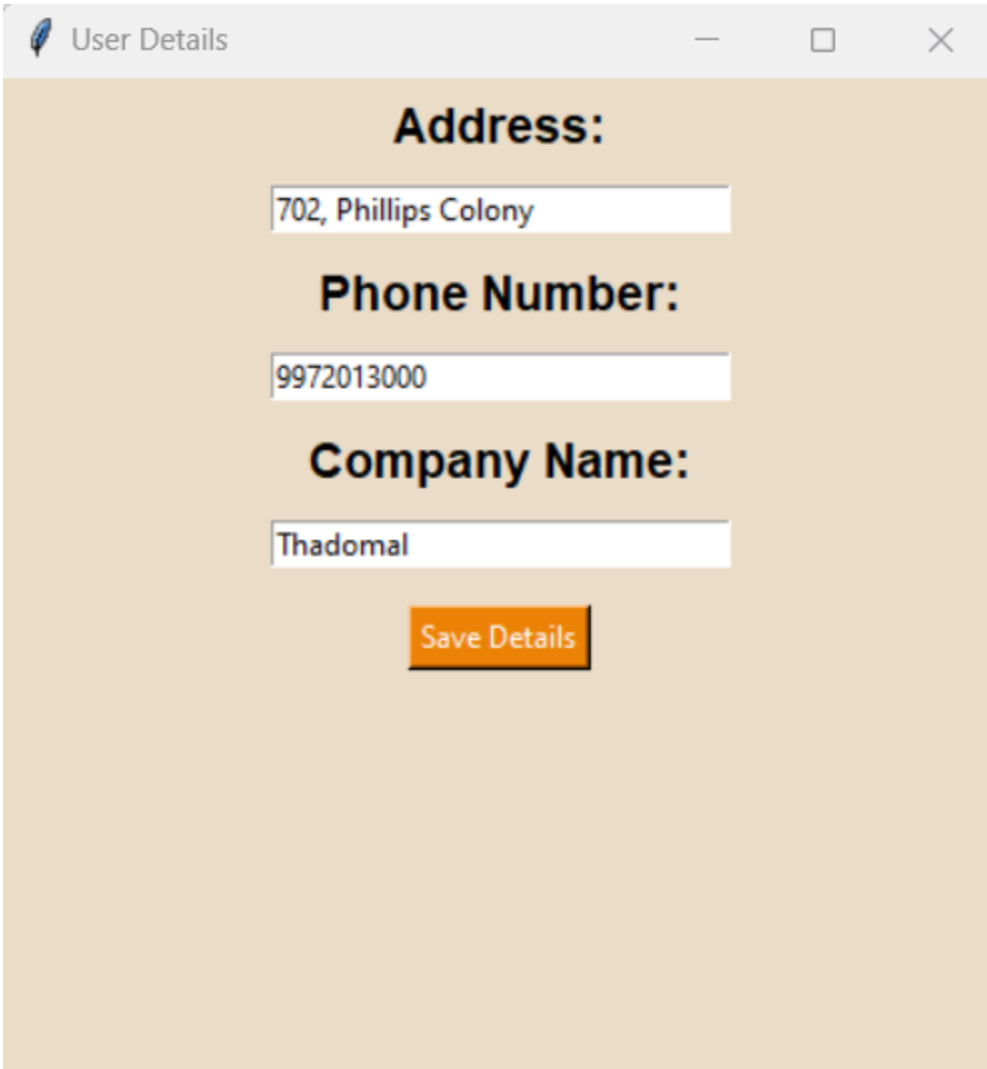
Window opened after logging in with valid username and password from database

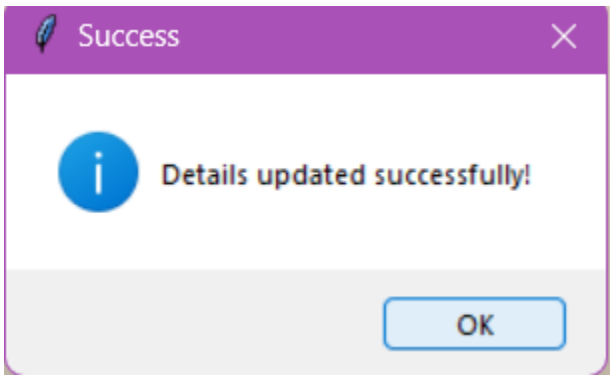

User Details
Address:
601,Amit Apartments
Phone Number:
8879753712
Company Name:
VJTI
Save Details


Success
Details updated successfully!
OK



New table created in database for storing these values


User Details
Address:
702, Phillips Colony
Phone Number:
9972013000
Company Name:
Thadomal
Save Details


Success
Details updated successfully!
OK

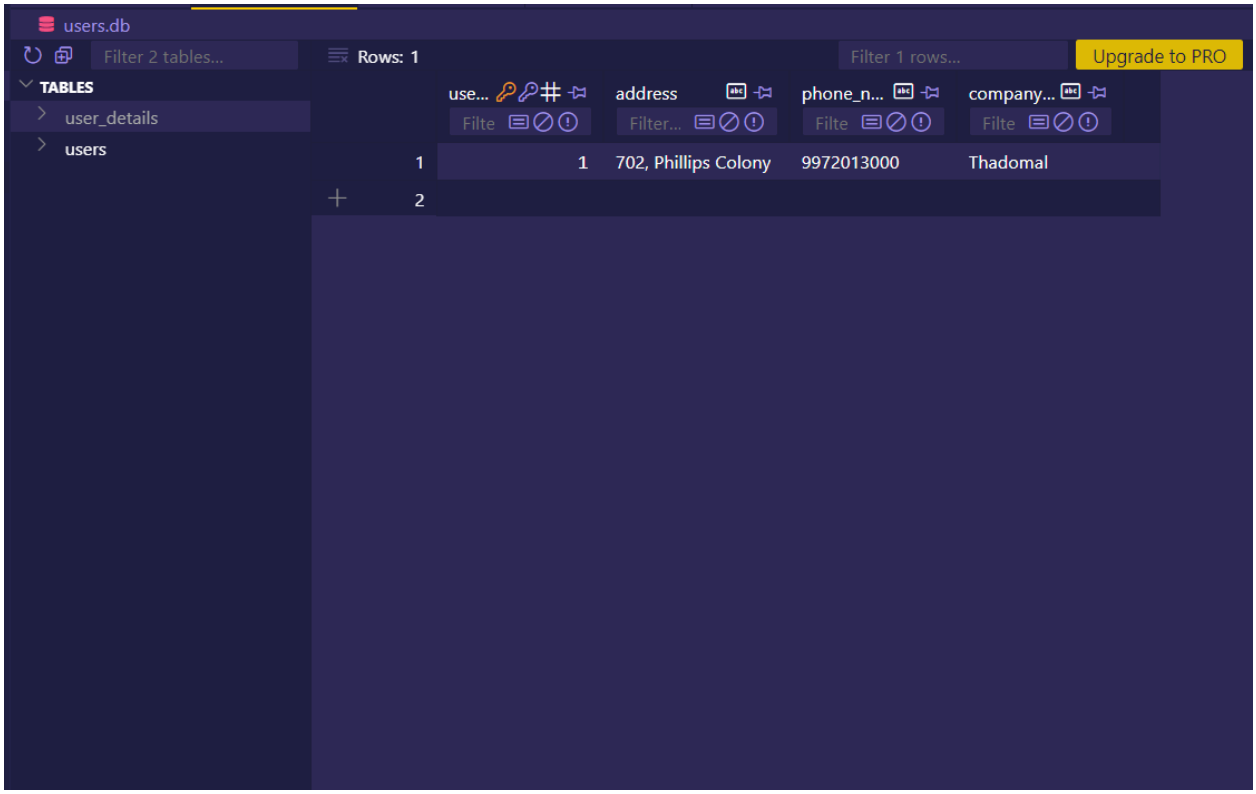
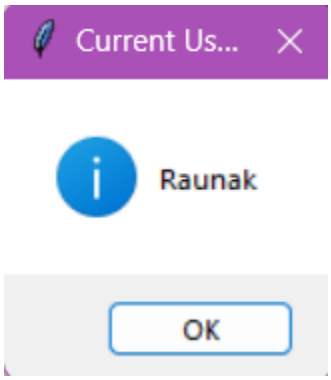
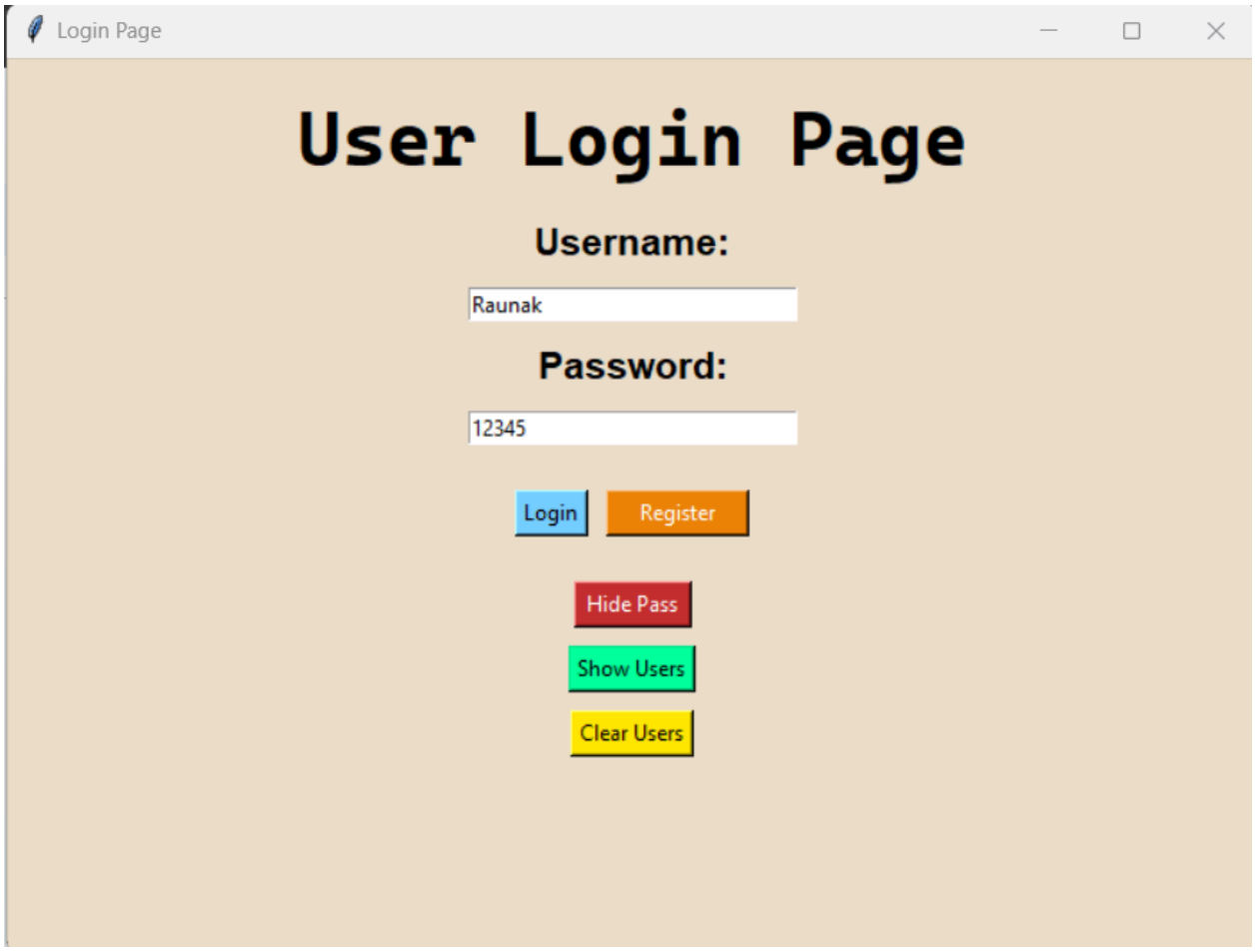


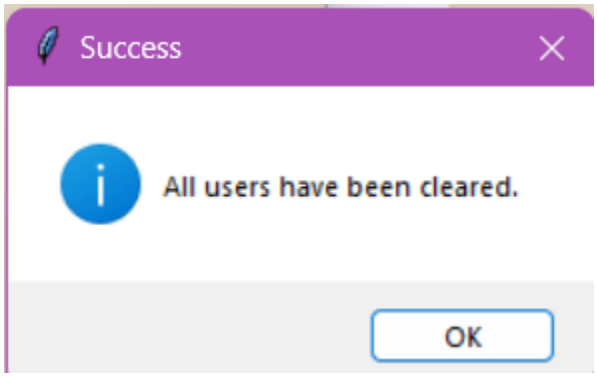
Table in database updated according to the edited values



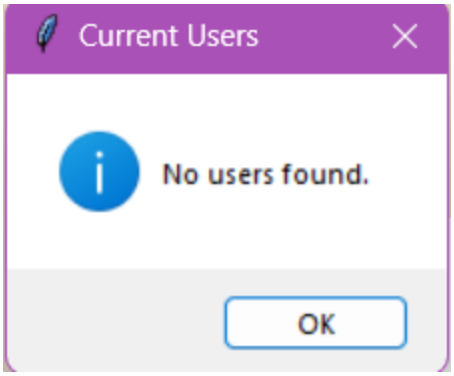
What pressing show all users does



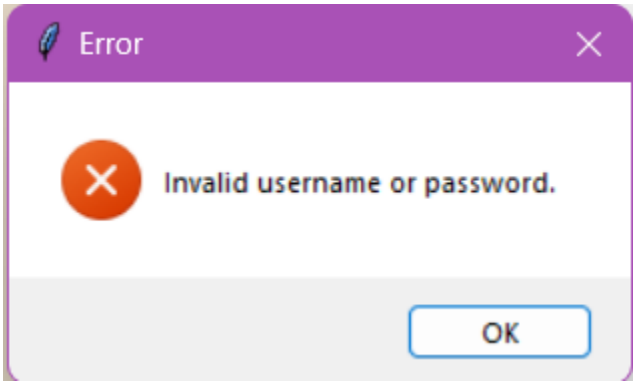
What reveal Password button does



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.