**Weather Prediction and Data Management System**

**# Overview**

The Weather Prediction and Data Management System is a simple desktop application built with Python and the CustomTkinter library. It allows users to perform various tasks related to weather data management, user registration, and weather prediction.

**# Features**

**User Authentication:**

- Users can log in with their username and password.

- If they forget their password, they can recover it by providing their mobile number and email.

**User Registration:**

- Users can create new accounts by providing a username, password, mobile number, and email.

- Registration data is stored in an SQLite database.

**Data Management:**

- Users can add, update, and delete temperature data for specific days.

- Data is stored in a database for analysis and prediction.

**Weather Prediction:**

- The system can predict the temperature for the next day based on the available data.

- Predictions are displayed to the user.

**Data Visualization:**

- Users can view temperature data on a line graph using Matplotlib.

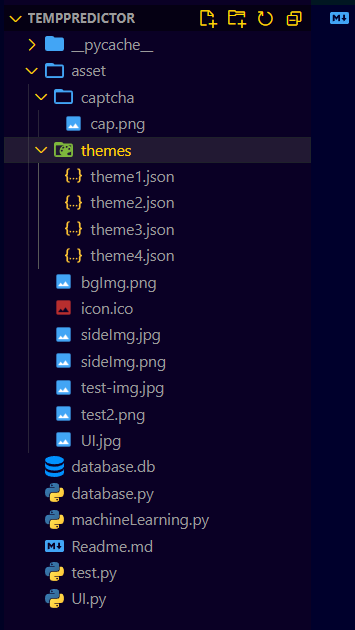
- Graphs help users visualize temperature trends.

**Project Structure**

- **database.py:** Contains functions for database management, including user registration, data addition, and more.

- **machineLearning.py:** Implements machine learning for temperature prediction.

- **UI.py:** The main script for the GUI application.



**Directory Management:**

# **Installation**

- Make sure you have Python 3.x installed on your system.

- Install required dependencies using: pip install matplotlib scikit-learn customtkinter captcha pillow

- If you are using IDLE:

-- Copy the project files into the default python IDLE's location

-- Run UI.py

**Usage**

- Run UI.py to start the application.

- Log in or create a new account.

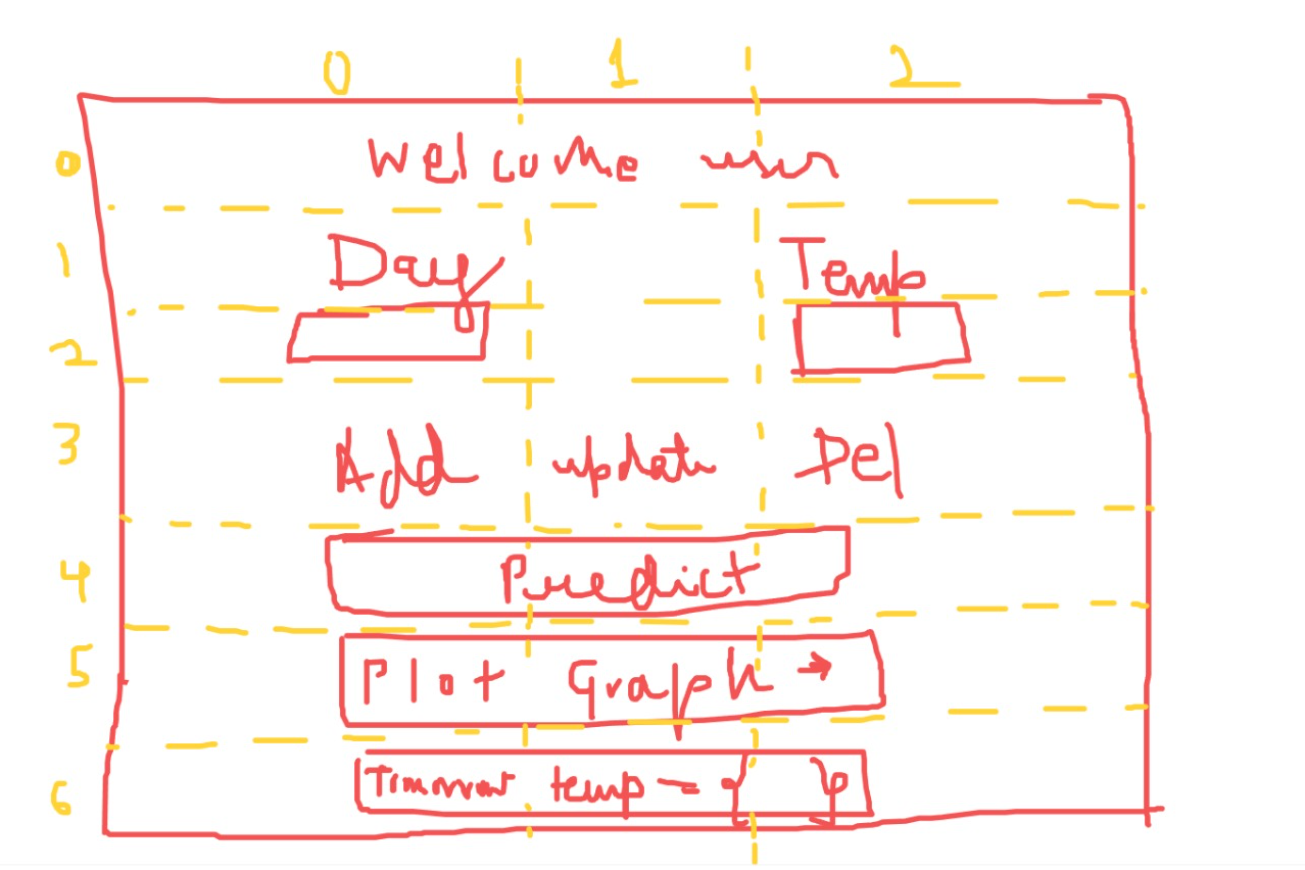
- Manage temperature data, recover a forgotten password, or make predictions.

- Visualize temperature data on a graph.

# **Personal words**

- The \*\*test.py\*\* file was created for testing the functions of different modules.

- I would like to share a rough sketch of how I designed the UI for this app.



**# Contributors**

**Priyanshu Batham**

**GitHub URL**: https://github.com/Priyanshu-Batham/tempPredictor

Code for UI

import tkinter as tk

from captcha.image import ImageCaptcha

import random

from customtkinter import \*

from PIL import Image

from tkinter import messagebox

from database import \*

from machineLearning import \*

#--------------------------UTILITY FUNCTIONS------------------------>>>>>>>>

#LOGIN FUNCTION

def loginBtn():

username = username\_entry.get()

password = password\_entry.get()

if isUserPresent(username, password): #returns true or false

welcomeHeading.configure(text=f"Welcome {username}")

login\_frame.pack\_forget()

home\_frame.pack(expand=True, fill=tk.BOTH)

else:

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

#--------------------------------------------------------------->>>>>>>>>>

#ADMIN FUNCTION

def adminLoginBtn():

username = username\_entry.get()

password = password\_entry.get()

if username == "admin" and password == "123":

login\_frame.pack\_forget()

admin\_frame.pack(expand=True, fill=tk.BOTH)

else:

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

# ------------------------------------------------->>>>>>>>>>>>>>>

def userInfoBtn():

str = " ID NAME PASSWORD PH.NO EMAIL\n\n"

data = showUserData()

for record in data:

str += f"{record}\n"

# outputHeading.configure(text=f"{data}", font=("aerial", 20))

dataWindow = tk.Toplevel(root)

dataLabel=CTkLabel(master=dataWindow, width=1000, height=100, text=str, font=("aerial", 50), text\_color="black")

dataLabel.place(anchor="center", relx=0.5, rely=0.2)

dataLabel.pack()

print(data)

# ------------------------------------------------->>>>>>>>>>>>>>>

def auditTableBtn():

str = " ID TEMPERATURE OPERATION DATE TIME\n\n"

data = getDatasetAudit()

for record in data:

str += f"{record}\n"

# outputHeading.configure(text=f"{data}", font=("aerial", 20))

dataWindow = tk.Toplevel(root)

dataLabel=CTkLabel(master=dataWindow, width=1000, height=100, text=str, font=("aerial", 50), text\_color="black")

dataLabel.place(anchor="center", relx=0.5, rely=0.2)

dataLabel.pack()

print(data)

# ------------------------------------------------------->>>>>>>>>>>>>>

#CAPTCHA GENERATING FUNCTION

def generateCaptcha():

lower = "abcdefghijklmnopqrstuvwxyz"

upper = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

nums = "1234567890"

total = nums+lower+upper

chars = "".join(random.sample(total, 6))

capObj = ImageCaptcha(height=150, width=250)

capObj.write(chars=chars, output="asset/captcha/cap.png")

print(chars)

return chars

#REGISTER BUTTON

def registerBtn():

login\_frame.pack\_forget()

register\_frame.pack(expand=True, fill=tk.BOTH)

def signUpBtn():

username = register\_username\_entry.get()

password = register\_password\_entry.get()

mobile = register\_mobile\_entry.get()

email = register\_email\_entry.get()

captcha = cap\_entry.get()

if(captcha != chars):

print("Incorrect Captcha Letters")

register\_error.configure(text = "Incorrect Captcha Letters!")

return

if(username == "" or password == "" or mobile == "" or email == ""):

print("Fill All the entries")

register\_error.configure(text = "Fill All the entries!")

return

createAccount(username, password, mobile, email)

welcomeHeading.configure(text=f"Welcome {username}")

register\_frame.pack\_forget()

home\_frame.pack(expand=True, fill=tk.BOTH)

#FORGOT PASSWORD FUNCTION

def forgotPasswordBtn():

login\_frame.pack\_forget()

forgot\_frame.pack(expand=True, fill=tk.BOTH)

#VALIDATE CREDENTIALS IF PASSWORD FORGOTTEN

def validateBtn():

username = forgot\_username\_entry.get()

mobile = forgot\_mobile\_entry.get()

email = forgot\_email\_entry.get()

if isCredentialsCorrect(username, mobile, email): #match the credentials

forgot\_frame.pack\_forget()

change\_frame.pack(expand=True, fill=tk.BOTH)

else:

messagebox.showerror("Error", "Incorrect Credentials.")

#CHANGE PASSWORD

def changeBtn():

username = forgot\_username\_entry.get()

newPassword = change\_password\_entry.get()

changePassword(username, newPassword)

welcomeHeading.configure(text=f"Welcome {username}")

change\_frame.pack\_forget()

home\_frame.pack(expand=True, fill=tk.BOTH)

#PREDICT AND UPDATE THE OUTPUT TEXT

def predictBtn():

response = predict()

outputHeading.configure(text=f"Tomorrow's temperature for Day{response[0]} is {response[1]} \u00b0C", font=("aerial", 40))

#DATASET-------------------------------------------------------->>>>>>>>>>

def showData():

str = " Day Temperature\n\n"

data = getData()

for record in data:

str += f"{record[0]} \t {record[1]}\n"

# outputHeading.configure(text=f"{data}", font=("aerial", 20))

dataWindow = tk.Toplevel(root)

dataLabel=CTkLabel(master=dataWindow, width=1000, height=100, text=str, font=("aerial", 50), text\_color="black")

dataLabel.place(anchor="center", relx=0.5, rely=0.2)

dataLabel.pack()

print(data)

def addingData():

res = addData(temp\_entry.get())

outputHeading.configure(text=f"{res}", font=("aerial", 30))

def updatingData():

res = updateData(day\_entry.get(), temp\_entry.get())

outputHeading.configure(text=f"{res}", font=("aerial", 30))

def deletingData():

res = deleteData(day\_entry.get())

outputHeading.configure(text=f"{res}", font=("aerial", 30))

#NAVIGATION------------------------------->>>>>>>>>>>>>

def adminToLogin():

admin\_frame.pack\_forget()

login\_frame.pack(expand=True, fill=tk.BOTH)

def registerToLogin():

register\_frame.pack\_forget()

login\_frame.pack(expand=True, fill=tk.BOTH)

def forgotToLogin():

forgot\_frame.pack\_forget()

login\_frame.pack(expand=True, fill=tk.BOTH)

def changeToLogin():

change\_frame.pack\_forget()

login\_frame.pack(expand=True, fill=tk.BOTH)

def homeToLogin():

home\_frame.pack\_forget()

login\_frame.pack(expand=True, fill=tk.BOTH)

# ---------------------------------FRAMES--------------------------------->>>>>>>>>

# MAIN

root = CTk()

set\_default\_color\_theme("asset/themes/theme4.json")

root.geometry("1920x1080")

root.title("Login Page")

root.iconbitmap("asset/icon.ico")

# LOGIN

login\_frame = CTkFrame(master=root)

heading=CTkLabel(master=login\_frame, width=1000, height=100, text="Temperature Predictor ☀️", font=("aerial", 72), text\_color="white")

heading.place(anchor="center", relx=0.5, rely=0.2)

username\_entry = CTkEntry(master=login\_frame, width=500, height=50, placeholder\_text="Username")

username\_entry.place(rely = 0.35, relx=0.5,anchor="center")

password\_entry = CTkEntry(master=login\_frame, width=500, height=50, show="\*", placeholder\_text="Password")

password\_entry.place(rely = 0.4, relx=0.5, anchor="center")

login\_button = CTkButton(master=login\_frame, text="Login", command=loginBtn)

login\_button.place(rely = 0.47, relx=0.5, anchor="center")

register\_button = CTkButton(master=login\_frame, text="Register", command=registerBtn)

register\_button.place(rely = 0.51, relx=0.5, anchor="center")

forgot\_button = CTkButton(master=login\_frame, text="Forgot Password?", command=forgotPasswordBtn)

forgot\_button.place(rely = 0.55, relx=0.5, anchor="center")

admin\_button = CTkButton(master=login\_frame, text="Admin Login", command=adminLoginBtn)

admin\_button.place(rely = 0.59, relx=0.5, anchor="center")

# ADMIN

admin\_frame = CTkFrame(master=root)

admin\_to\_login\_button = CTkButton(master=admin\_frame, text="\u2190", width=10, height=50, corner\_radius=100, command=adminToLogin)

admin\_to\_login\_button.place(rely = 0.05, relx=0.05, anchor="center")

adminHeading=CTkLabel(master=admin\_frame, width=1000, height=100, text="Admin Panel", font=("aerial", 72), text\_color="white")

adminHeading.place(anchor="center", relx=0.5, rely=0.1)

userInfo\_button = CTkButton(master=admin\_frame, text="User Info", command=userInfoBtn, height= 100)

userInfo\_button.place(rely = 0.5, relx=0.4, anchor="center")

auditTable\_button = CTkButton(master=admin\_frame, text="Audit Table", command=auditTableBtn, height= 100)

auditTable\_button.place(rely = 0.5, relx=0.6, anchor="center")

#REGISTER

register\_frame = CTkFrame(master=root)

back\_to\_login\_button = CTkButton(master=register\_frame, text="\u2190", width=10, height=50, corner\_radius=100, command=registerToLogin)

back\_to\_login\_button.place(rely = 0.05, relx=0.05, anchor="center")

registerHeading=CTkLabel(master=register\_frame, width=1000, height=100, text="Register", font=("aerial", 72), text\_color="white")

registerHeading.place(anchor="center", relx=0.5, rely=0.1)

register\_username\_entry = CTkEntry(master=register\_frame, width=500, height=50, placeholder\_text="Username")

register\_username\_entry.place(rely = 0.25, relx=0.5,anchor="center")

register\_password\_entry = CTkEntry(master=register\_frame, width=500, height=50, placeholder\_text="Password")

register\_password\_entry.place(rely = 0.3, relx=0.5,anchor="center")

register\_mobile\_entry = CTkEntry(master=register\_frame, width=500, height=50, placeholder\_text="Mobile Number")

register\_mobile\_entry.place(rely = 0.35, relx=0.5,anchor="center")

register\_email\_entry = CTkEntry(master=register\_frame, width=500, height=50, placeholder\_text="Email")

register\_email\_entry.place(rely = 0.4, relx=0.5,anchor="center")

# captcha inside register---------------------------->>>

chars = generateCaptcha()

path = "asset/captcha/cap.png"

i = CTkImage(light\_image=Image.open(path), size=(250,100))

cap=CTkLabel(master=register\_frame, width=1000, height=100, image=i, text="")

cap.place(anchor="center", relx=0.5, rely=0.5)

cap\_entry = CTkEntry(master=register\_frame, width=250, height=30, placeholder\_text="Captcha:")

cap\_entry.place(rely = 0.57, relx=0.5,anchor="center")

signUp\_button = CTkButton(master=register\_frame, text="Sign Up", command=signUpBtn, height=50, width=100)

signUp\_button.place(rely = 0.65, relx=0.5, anchor="center")

register\_error=CTkLabel(master=register\_frame, width=1000, height=100, text="", font=("aerial", 50), text\_color="red")

register\_error.place(anchor="center", relx=0.5, rely=0.72)

#FORGOT PASSWORD

forgot\_frame = CTkFrame(master=root)

forgot\_to\_login\_button = CTkButton(master=forgot\_frame, text="\u2190", width=10, height=50, corner\_radius=100, command=forgotToLogin)

forgot\_to\_login\_button.place(rely = 0.05, relx=0.05, anchor="center")

registerHeading=CTkLabel(master=forgot\_frame, width=1000, height=100, text="Enter Credentials", font=("aerial", 72), text\_color="white")

registerHeading.place(anchor="center", relx=0.5, rely=0.2)

forgot\_username\_entry = CTkEntry(master=forgot\_frame, width=500, height=50, placeholder\_text="Username")

forgot\_username\_entry.place(rely = 0.35, relx=0.5,anchor="center")

forgot\_mobile\_entry = CTkEntry(master=forgot\_frame, width=500, height=50, placeholder\_text="Mobile")

forgot\_mobile\_entry.place(rely = 0.4, relx=0.5,anchor="center")

forgot\_email\_entry = CTkEntry(master=forgot\_frame, width=500, height=50, placeholder\_text="Email")

forgot\_email\_entry.place(rely = 0.45, relx=0.5,anchor="center")

validate\_button = CTkButton(master=forgot\_frame, text="Sign Up", command=validateBtn)

validate\_button.place(rely = 0.55, relx=0.5, anchor="center")

#CHANGE PASSWORD

change\_frame = CTkFrame(master=root)

change\_to\_login\_button = CTkButton(master=change\_frame, text="\u2190", width=10, height=50, corner\_radius=100, command=changeToLogin)

change\_to\_login\_button.place(rely = 0.05, relx=0.05, anchor="center")

changeHeading=CTkLabel(master=change\_frame, width=1000, height=100, text="Set New Password", font=("aerial", 72), text\_color="white")

changeHeading.place(anchor="center", relx=0.5, rely=0.2)

change\_password\_entry = CTkEntry(master=change\_frame, width=500, height=50, placeholder\_text="New Password")

change\_password\_entry.place(rely = 0.4, relx=0.5,anchor="center")

change\_button = CTkButton(master=forgot\_frame, text="Change", command=validateBtn)

change\_button.place(rely = 0.55, relx=0.5, anchor="center")

# HOME

home\_frame = CTkFrame(master=root)

home\_to\_login\_button = CTkButton(master=home\_frame, text="\u2190", width=10, height=50, corner\_radius=100, command=homeToLogin)

home\_to\_login\_button.place(rely = 0.05, relx=0.05, anchor="center")

welcomeHeading=CTkLabel(master=home\_frame, width=1000, height=100, text="Welcome", font=("aerial", 72), text\_color="white")

welcomeHeading.place(anchor="center", relx=0.5, rely=0.1)

day\_entry = CTkEntry(master=home\_frame, width=300, height=50, placeholder\_text="Enter the Day Number: ", font=("aerial", 20))

day\_entry.place(rely = 0.2, relx=0.4,anchor="center")

temp\_entry = CTkEntry(master=home\_frame, width=300, height=50, placeholder\_text="Enter the Temperature: ", font=("aerial", 20))

temp\_entry.place(rely = 0.2, relx=0.6,anchor="center")

add\_button = CTkButton(master=home\_frame, text="ADD to dataset", height = 50, command=addingData)

add\_button.place(rely = 0.3, relx=0.4, anchor="center")

update\_button = CTkButton(master=home\_frame, text="UPDATE in dataset", height = 50, command=updatingData)

update\_button.place(rely = 0.3, relx=0.5, anchor="center")

delete\_button = CTkButton(master=home\_frame, text="DELETE from dataset", height = 50, command=deletingData)

delete\_button.place(rely = 0.3, relx=0.6, anchor="center")

predict\_button = CTkButton(master=home\_frame, text="PREDICT", command=predictBtn, width = 400, height=50)

predict\_button.place(rely = 0.42, relx=0.5, anchor="center")

plot\_button = CTkButton(master=home\_frame, text="PLOT", command=plotDataFromDataset, width = 400, height=50)

plot\_button.place(rely = 0.49, relx=0.5, anchor="center")

show\_button = CTkButton(master=home\_frame, text="SHOW DATA", command=showData, width = 400, height=50)

show\_button.place(rely = 0.56, relx=0.5, anchor="center")

outputHeading=CTkLabel(master=home\_frame, width=1000, height=100, text="Tomorrow's Temperature: ", font=("aerial", 50), text\_color="white")

outputHeading.place(anchor="center", relx=0.5, rely=0.7)

#----------------------------INITIALIZING--------------------->>>>>>>>>>>>>

createTable()

createDataset()

createDatasetAudit()

createDatasetTrigger()

login\_frame.pack(expand=True, fill=tk.BOTH)

root.mainloop()

# clearDataset()

Code for Machine Learning

import numpy as np

from sklearn.linear\_model import LinearRegression

from database import getData

import matplotlib.pyplot as plt

# Sample dataset

def predict():

rawData = getData()

nextDay = rawData[-1][0]+1

data = np.array(rawData)

# Split the data into input sequences and target values

X = data[:, 0].reshape(-1,1)

y = data[:, 1]

# Create and train a Linear Regression model

model = LinearRegression()

model.fit(X, y)

# Predict the next number in the sequence

nextDayTemp = model.predict([[nextDay]])[0]

nextDayTemp = round(nextDayTemp, 2)

print(f"The predicted temperature for tomorrow, (day:{nextDay}) is: {nextDayTemp}")

return [nextDay, nextDayTemp]

# -------------------------------------------------------------------------->>>>>>>>>>>>

def plotDataFromDataset():

data = getData()

# Separate data into x and y values

days = [item[0] for item in data]

temps = [item[1] for item in data]

# Create a line plot

plt.figure(figsize=(8, 6))

plt.plot(days, temps, marker='o', linestyle='--', color='b')

# Add labels and a title

plt.xlabel('Day')

plt.ylabel('Temperature')

plt.title('Temperature Data Over Time')

# Show the plot

plt.grid(True)

plt.show()

Code for Database

import sqlite3

import datetime

# --------------------------------------------------------------->>>>>>>>>>>

def createTable():

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

cursor = conn.cursor()

create\_table\_query = '''

CREATE TABLE IF NOT EXISTS userInfo (

id INTEGER PRIMARY KEY,

username TEXT NOT NULL UNIQUE,

password TEXT NOT NULL,

mobileno TEXT,

email TEXT

);

'''

cursor.execute(create\_table\_query)

conn.commit()

conn.close()

# --------------------------------------------------------------->>>>>>>>>>>

def createAccount(username, password, mobile, email):

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

cursor = conn.cursor()

insert\_query = '''

INSERT INTO userInfo (username, password, mobileno, email)

VALUES (?, ?, ?, ?)

'''

try:

cursor.execute(insert\_query, (username, password, mobile, email))

conn.commit()

except sqlite3.Error as e:

print("Error:",e)

conn.close()

# --------------------------------------------------------------->>>>>>>>>>>

def showUserData():

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

cursor = conn.cursor()

fetch\_query = '''

SELECT \* FROM userInfo;

'''

try:

cursor.execute(fetch\_query)

data = cursor.fetchall()

print(data)

except:

print("Couldn't Fetch Results")

conn.commit()

conn.close()

return data

# --------------------------------------------------------------->>>>>>>>>>>

def isUserPresent(username, password):

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

cursor = conn.cursor()

login\_query = '''

SELECT \* FROM userInfo

WHERE username = ? AND password = ?

'''

cursor.execute(login\_query,(username, password))

data = cursor.fetchall()

conn.commit()

conn.close()

return True if data else False

# --------------------------------------------------------------->>>>>>>>>>>

def isCredentialsCorrect(username, mobile, email):

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

cursor = conn.cursor()

credential\_query = '''

SELECT \* FROM userInfo

WHERE username = ? AND mobileno = ? AND email = ?

'''

cursor.execute(credential\_query,(username, mobile, email))

data = cursor.fetchall()

conn.commit()

conn.close()

return True if data else False

# --------------------------------------------------------------->>>>>>>>>>>

def changePassword(username, password):

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

cursor = conn.cursor()

change\_query = '''

UPDATE userInfo

SET password = ?

WHERE username = ?

'''

try:

cursor.execute(change\_query,(password, username))

print(cursor.rowcount)

print("password changed successfully")

except:

print("Couldn't Change Password")

conn.commit()

conn.close()

# --------------------------------------------------------------->>>>>>>>>>>

def clearUserInfo():

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

cursor = conn.cursor()

cursor.execute("DELETE FROM userInfo;")

conn.commit()

conn.close()

# -------------------------------------------------------------------------------------->>>>>>>>

# -----------------------------------DATASET--QUERIES----------------------------------->>>>>>>>

# -------------------------------------------------------------------------------------->>>>>>>>

def createDataset():

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

cursor = conn.cursor()

create\_table\_query = '''

CREATE TABLE IF NOT EXISTS dataset (

id INTEGER PRIMARY KEY,

temp INTEGER NOT NULL

);

'''

cursor.execute(create\_table\_query)

conn.commit()

conn.close()

# --------------------------------------------------------------->>>>>>>>>>>

def getData():

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

cursor = conn.cursor()

query = '''

SELECT \* FROM dataset;

'''

cursor.execute(query)

data = cursor.fetchall()

conn.close()

return data

# --------------------------------------------------------------->>>>>>>>>>>

def addData(temp):

if temp == '':

print("Enter some Data")

return "Enter some Data"

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

cursor = conn.cursor()

query = '''

INSERT INTO dataset (temp)

VALUES (?)

'''

if all(char.isdigit() for char in temp):

try:

cursor.execute(query,(temp,))

conn.commit()

conn.close()

print("Temperature Recorded")

return "Temperature Recorded"

except KeyError as e:

print(e)

else:

conn.commit()

conn.close()

print("Give Numeric Values")

return "Give Numeric Values"

# --------------------------------------------------------------->>>>>>>>>>>

def deleteData(day):

if(day == ""):

return "Enter the Day Number"

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

cursor = conn.cursor()

query = '''

DELETE FROM dataset

WHERE id = (?)

'''

if all(char.isdigit() for char in day):

try:

cursor.execute(query,(day,))

conn.commit()

conn.close()

except:

print("Error occured")

else:

return "Temperature Deleted"

else:

conn.commit()

conn.close()

print("Enter Numeric Values")

return "Enter Numeric Values"

# --------------------------------------------------------------->>>>>>>>>>>

def updateData(day, temp):

if(day == "" or temp == ""):

return "Enter Data in both fields"

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

cursor = conn.cursor()

query = '''

UPDATE dataset

SET temp = (?)

WHERE id = (?)

'''

if all(char.isdigit() for char in day) and all(char.isdigit() for char in temp):

try:

cursor.execute(query,(temp, day))

except:

conn.commit()

conn.close()

print(f"Error occured")

return "Error Occured"

else:

conn.commit()

conn.close()

return "Temperature Updated"

else:

print("Enter Numeric Values")

return "Enter Numeric Values"

# --------------------------------------------------------------->>>>>>>>>>>

def clearDataset():

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

cursor = conn.cursor()

cursor.execute("DELETE FROM dataset;")

print("Done")

conn.commit()

conn.close()

#DATASET TRIGGERS

# --------------------------------------------------------------->>>>>>>>>>>

def createDatasetAudit():

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

cursor = conn.cursor()

create\_table\_query = '''

CREATE TABLE IF NOT EXISTS datasetAudit (

id INTEGER PRIMARY KEY,

temp INTEGER NOT NULL,

opr VARCHAR(10),

time TIME

);

'''

cursor.execute(create\_table\_query)

conn.commit()

conn.close()

# print("done")

# --------------------------------------------------------------->>>>>>>>>>>

def createDatasetTrigger():

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

cursor = conn.cursor()

query1 = '''

CREATE TRIGGER IF NOT EXISTS datasetAddTrigger AFTER INSERT

ON dataset

BEGIN

INSERT INTO datasetAudit(temp, opr, time)

VALUES (NEW.temp, 'INS', datetime('now'));

END;

'''

query2 = '''

CREATE TRIGGER IF NOT EXISTS datasetUpdateTrigger BEFORE UPDATE

ON dataset

BEGIN

INSERT INTO datasetAudit(temp, opr, time)

VALUES (NEW.temp, 'UPDATE', datetime('now'));

END;

'''

query3 = '''

CREATE TRIGGER IF NOT EXISTS datasetDeleteTrigger BEFORE DELETE

ON dataset

BEGIN

INSERT INTO datasetAudit(temp, opr, time)

VALUES (OLD.temp, 'DEL', datetime('now'));

END;

'''

cursor.execute(query1)

cursor.execute(query2)

cursor.execute(query3)

conn.commit()

conn.close()

# print("done maybe")

# --------------------------------------------------------------->>>>>>>>>>>

def dropDatasetDeleteTrigger():

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

cursor = conn.cursor()

cursor.execute('DROP TRIGGER datasetDeleteTrigger;')

conn.commit()

conn.close()

print("done maybe")

# --------------------------------------------------------------->>>>>>>>>>>

def dropDatasetAddTrigger():

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

cursor = conn.cursor()

cursor.execute('DROP TRIGGER datasetAddTrigger;')

conn.commit()

conn.close()

print("done maybe")

# --------------------------------------------------------------->>>>>>>>>>>

def dropDatasetUpdateTrigger():

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

cursor = conn.cursor()

cursor.execute('DROP TRIGGER datasetUpdateTrigger;')

conn.commit()

conn.close()

print("done maybe")

# --------------------------------------------------------------->>>>>>>>>>>

def getDatasetAudit():

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

cursor = conn.cursor()

query = '''

SELECT \* FROM datasetAudit;

'''

cursor.execute(query)

data = cursor.fetchall()

conn.close()

return data

def clearDatasetAudit():

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

cursor = conn.cursor()

cursor.execute("DELETE FROM datasetAudit;")

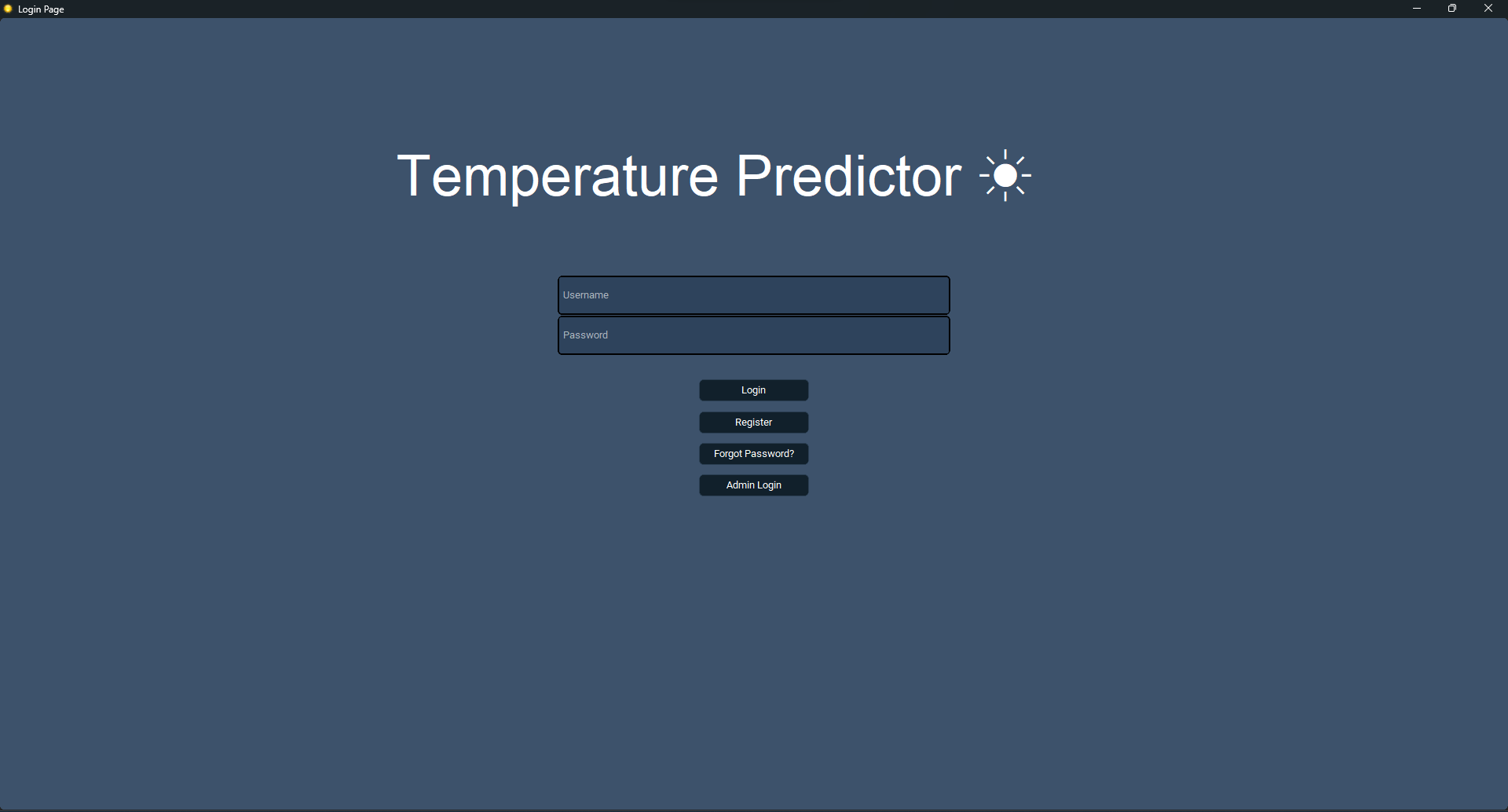
print("Done")

conn.commit()

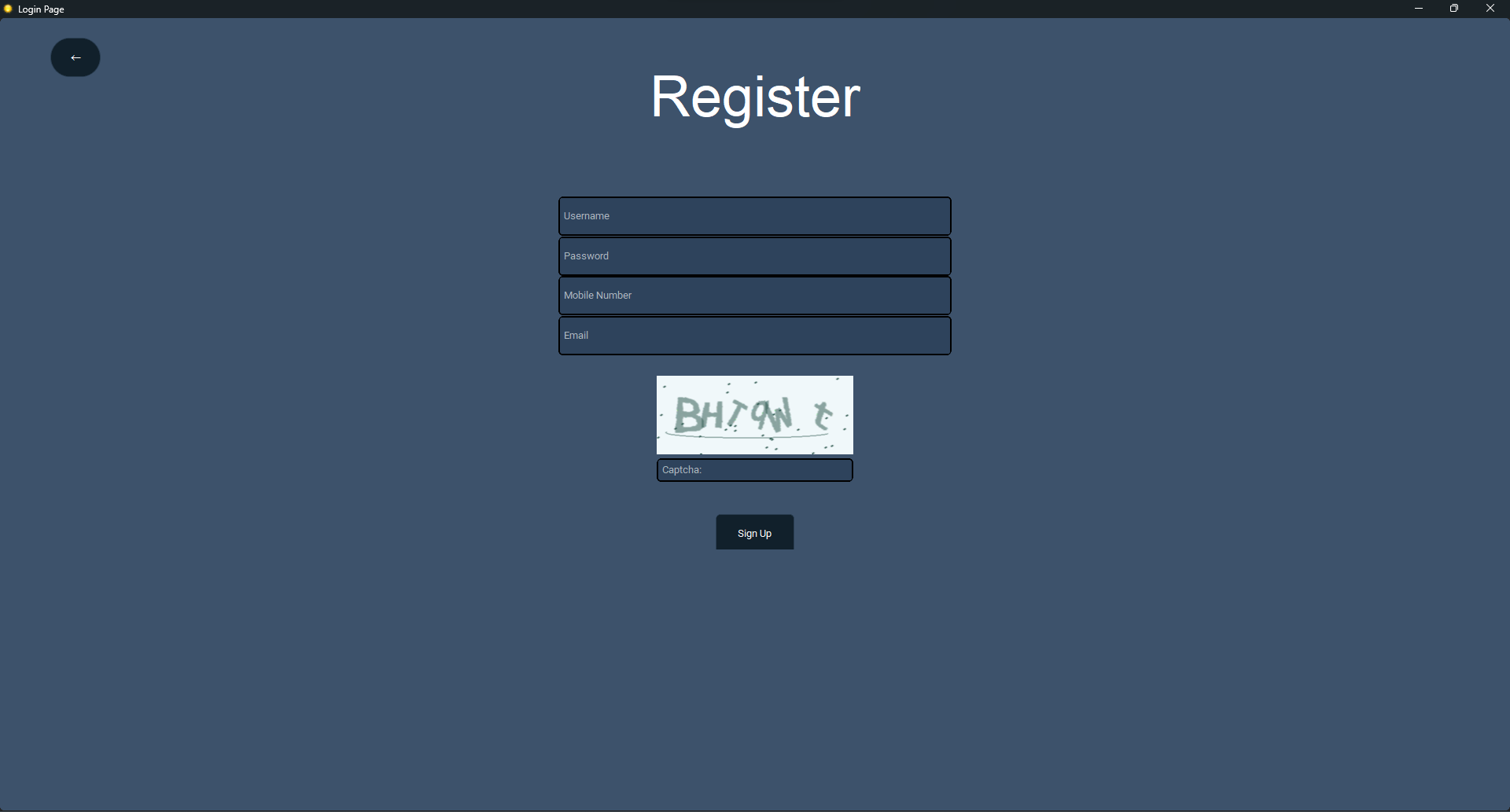
conn.close()

**Screen Shots of the Project**

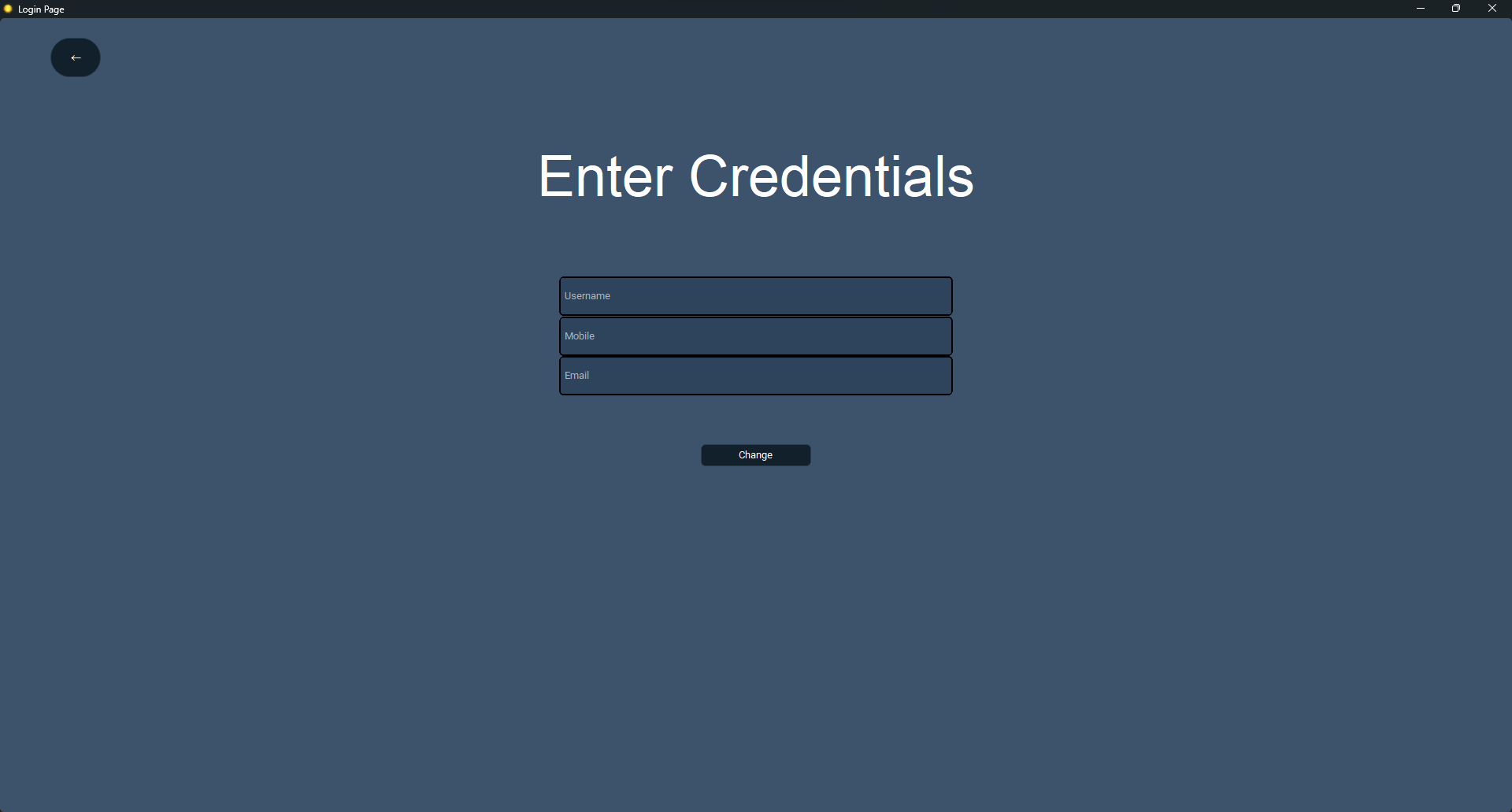
**Login Page:**

****

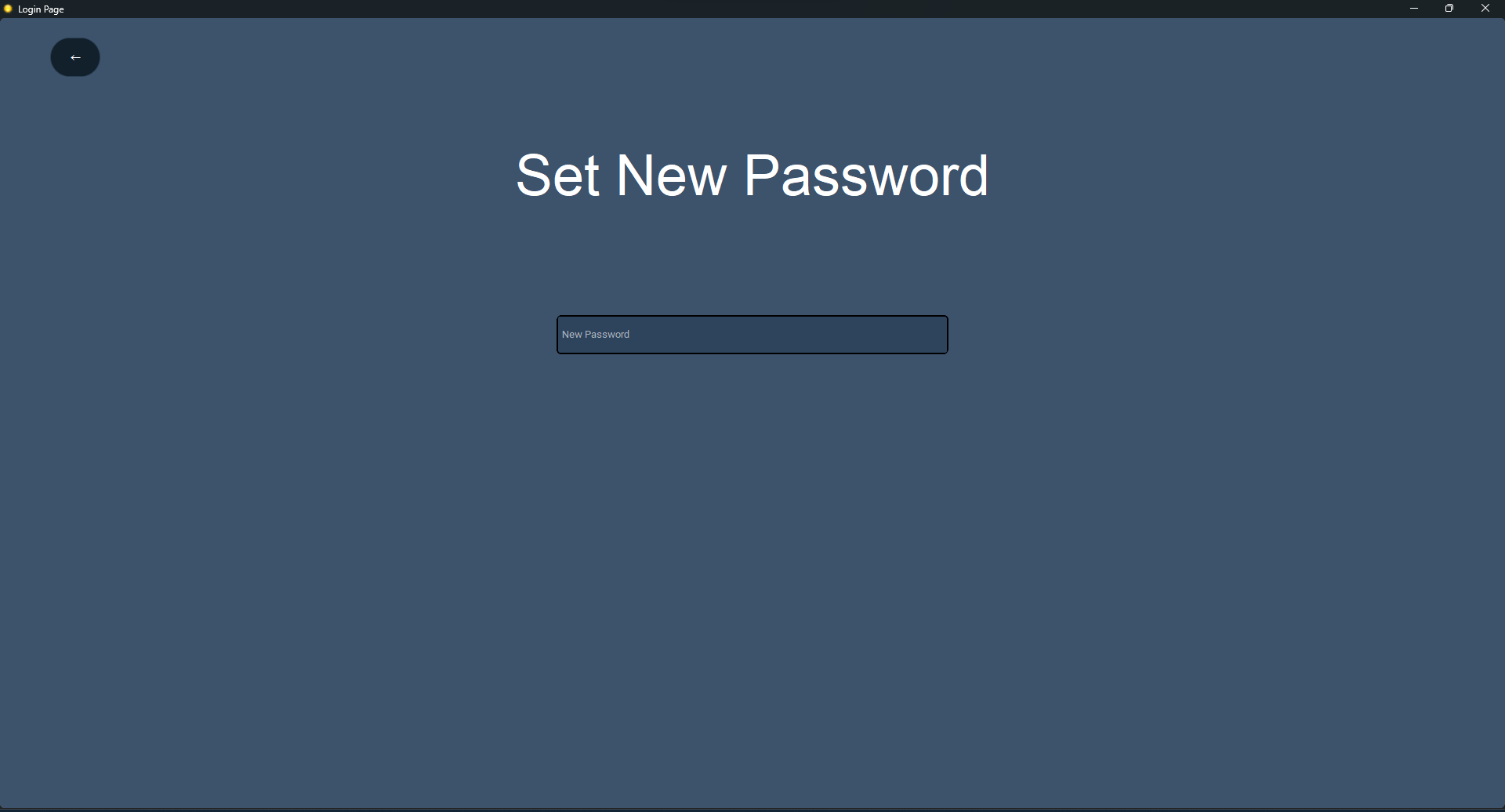
**Register Page:**

****

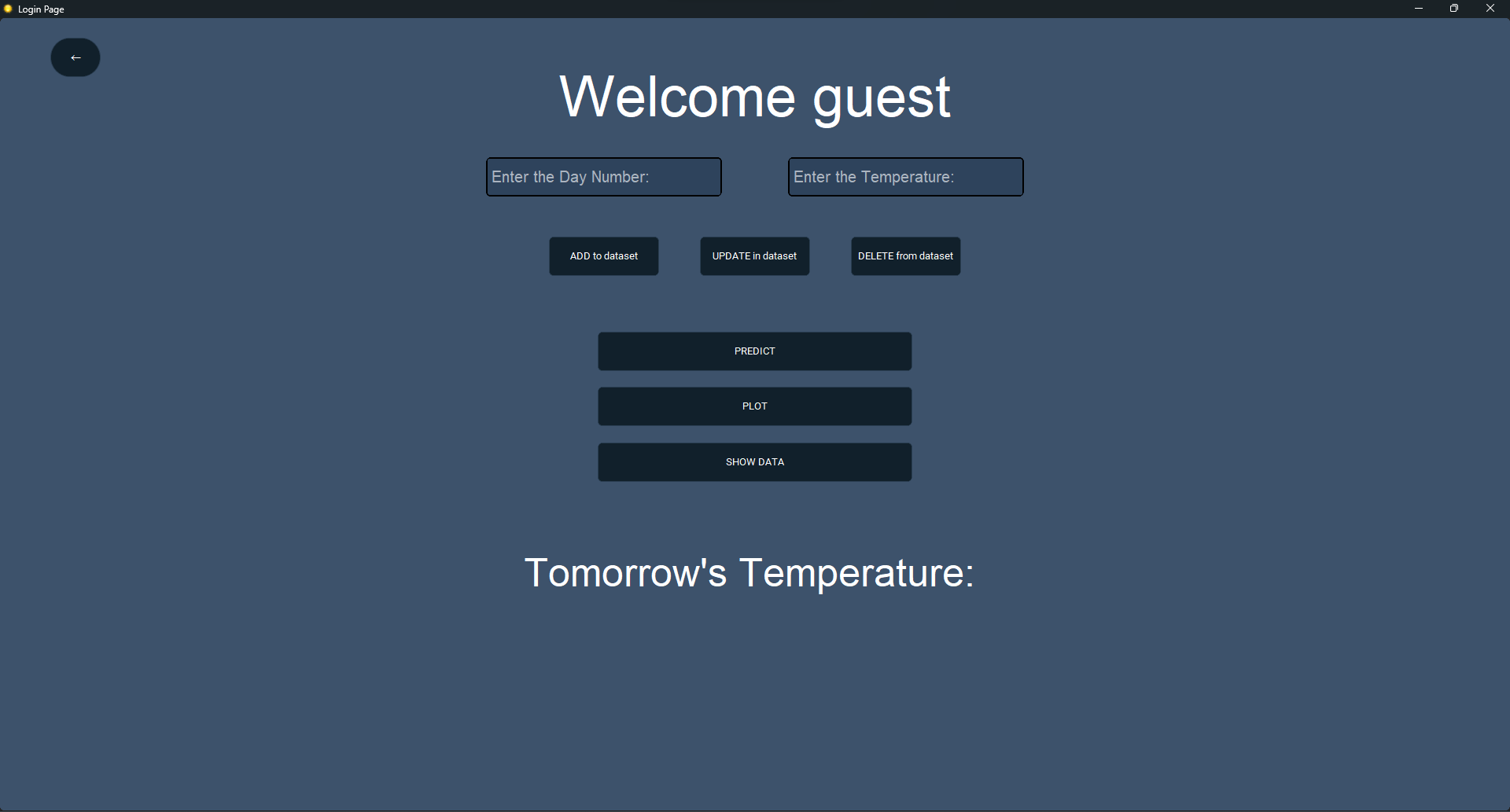
**Forgot Password Page:**

****

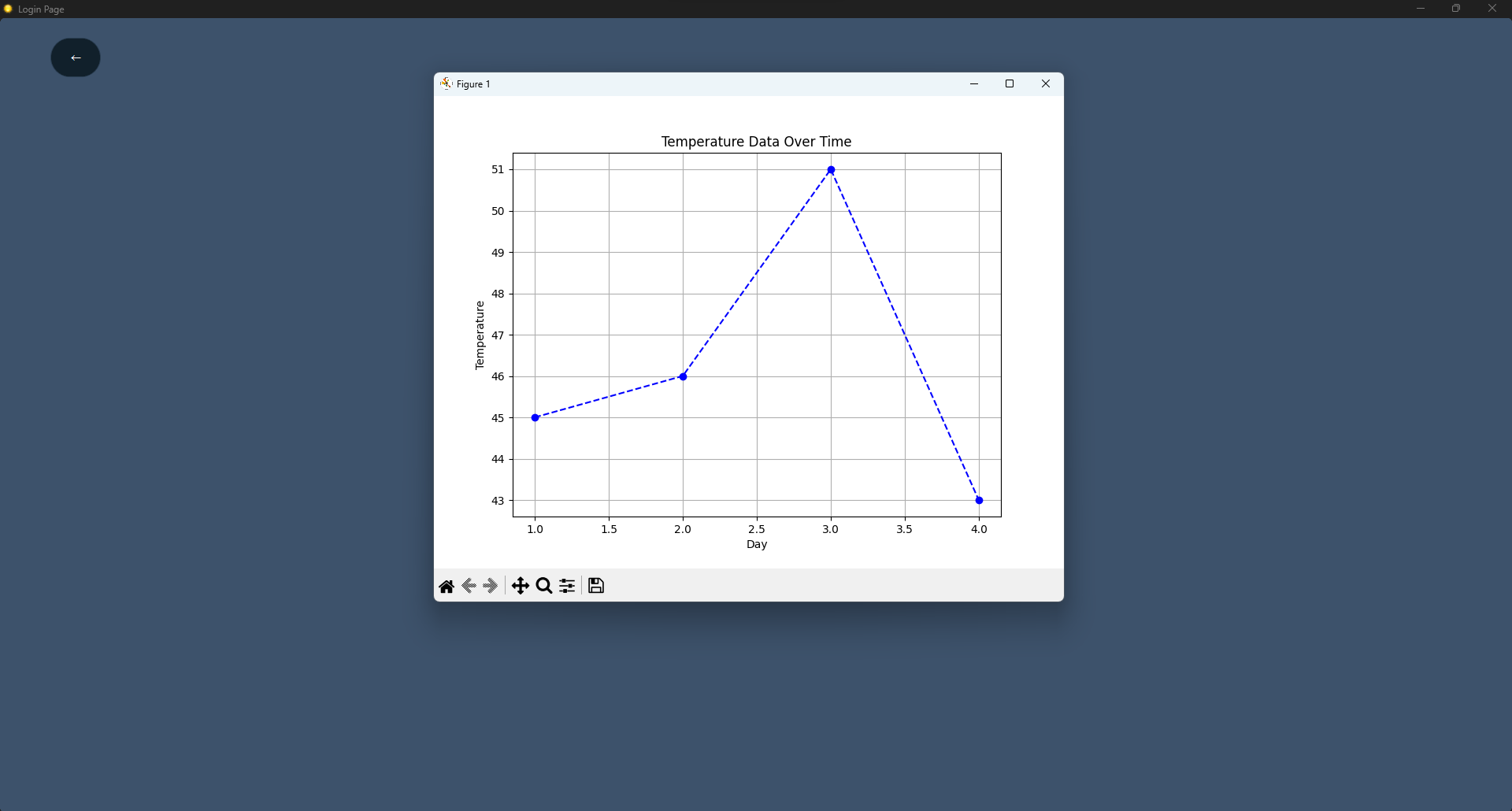
**Change Password Page:**

****

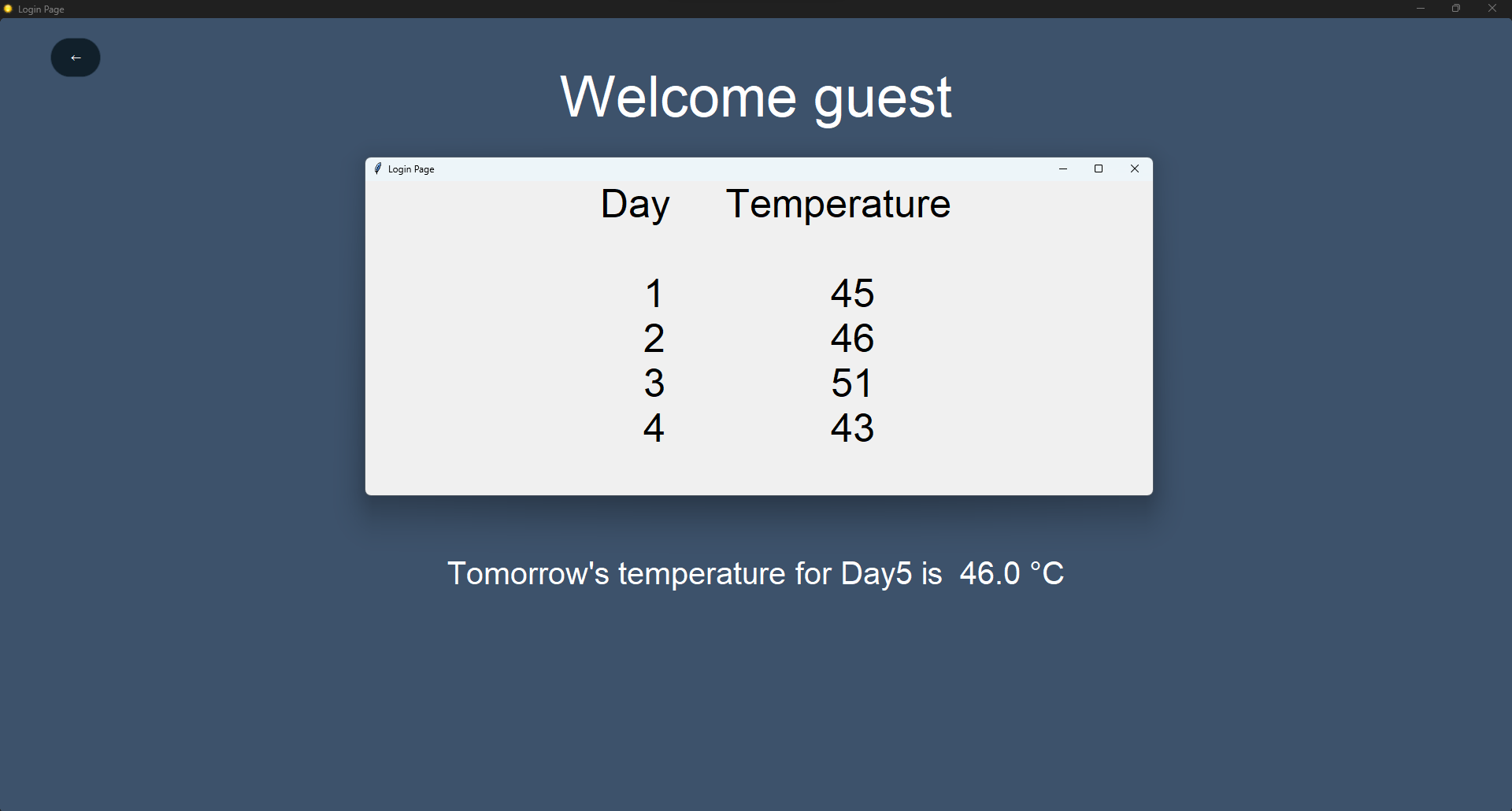
**Main Home Page:**

****

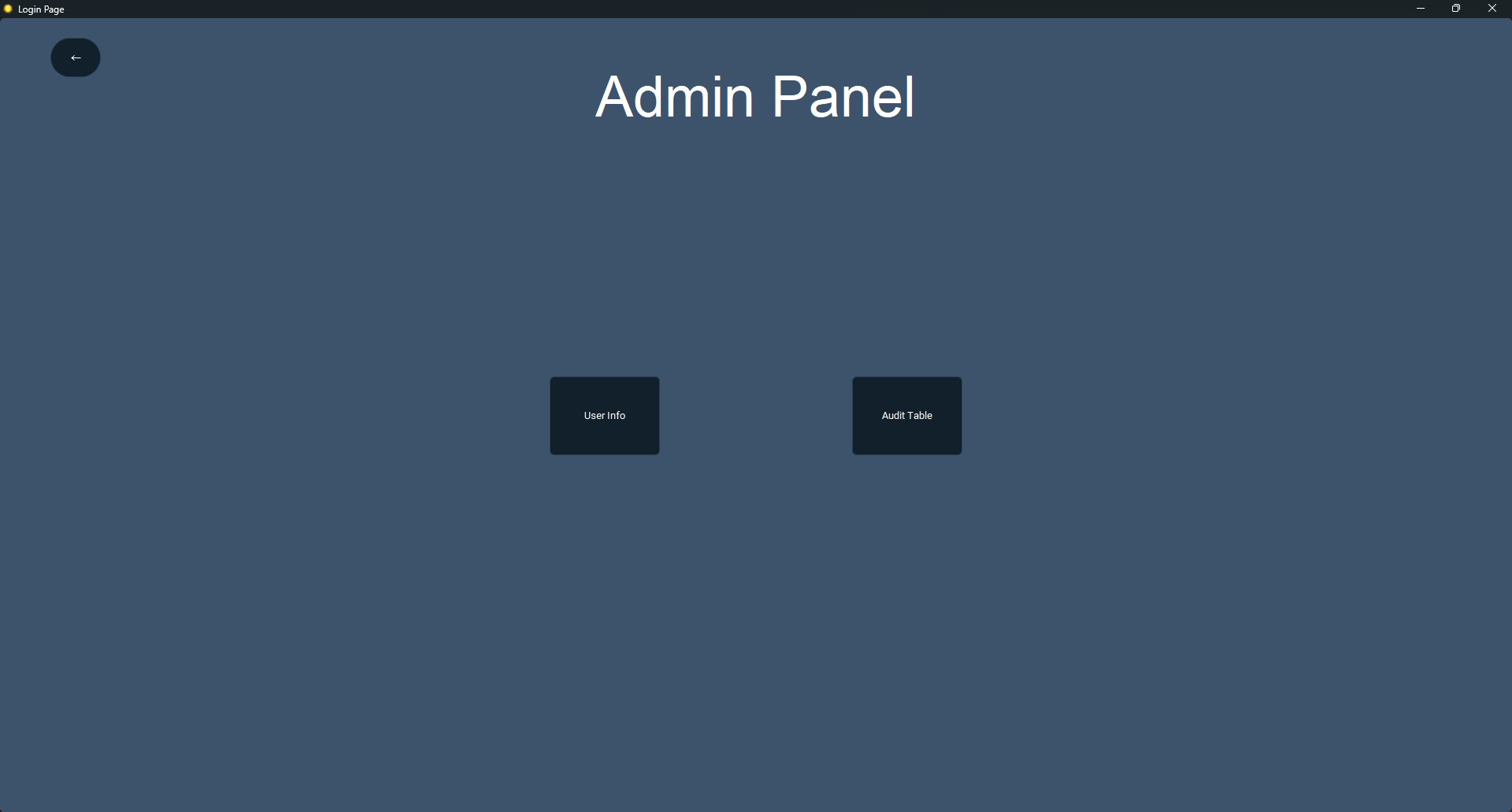
**Plotting Graph:**

****

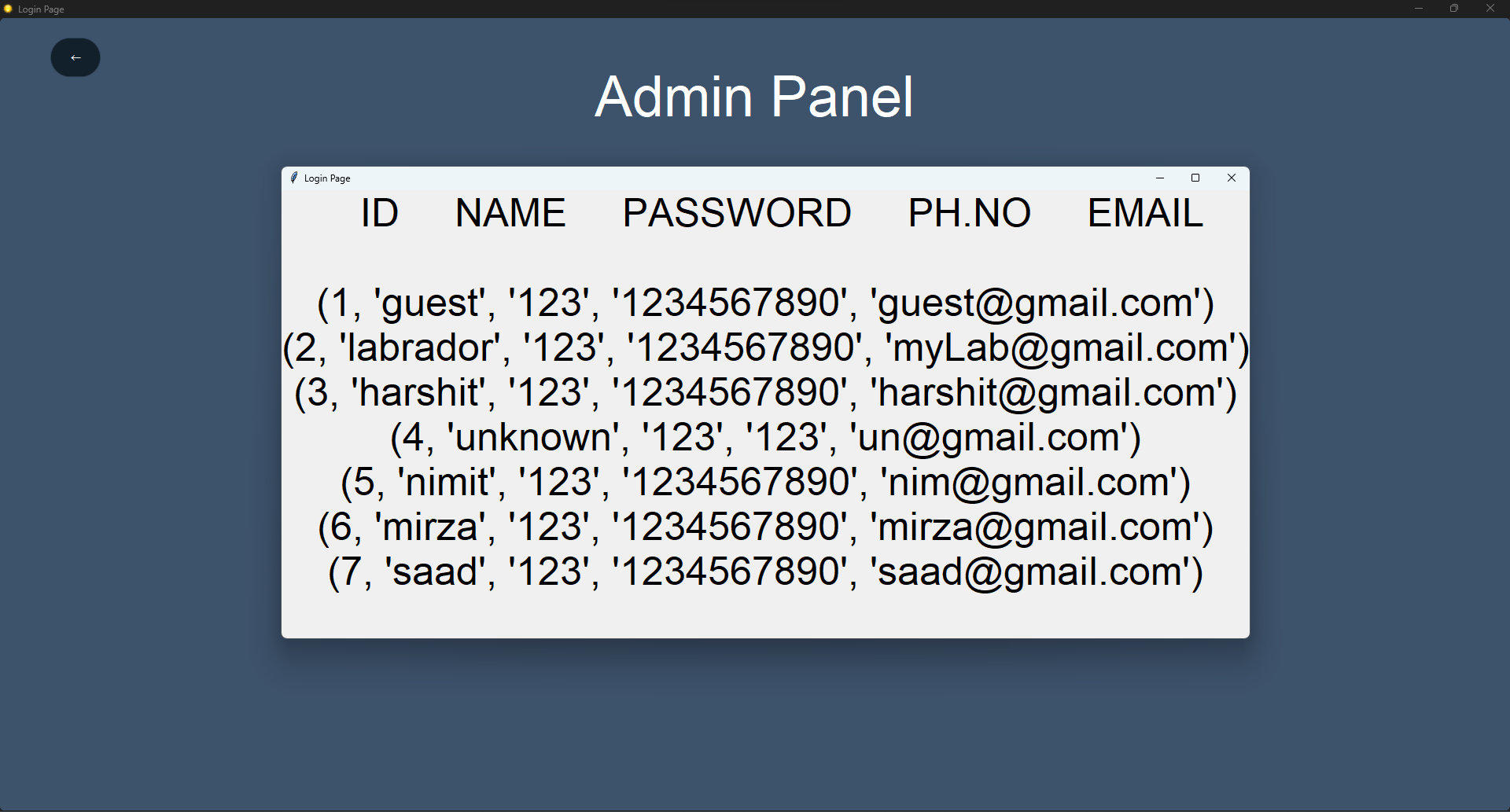
**Show Data Button:**

****

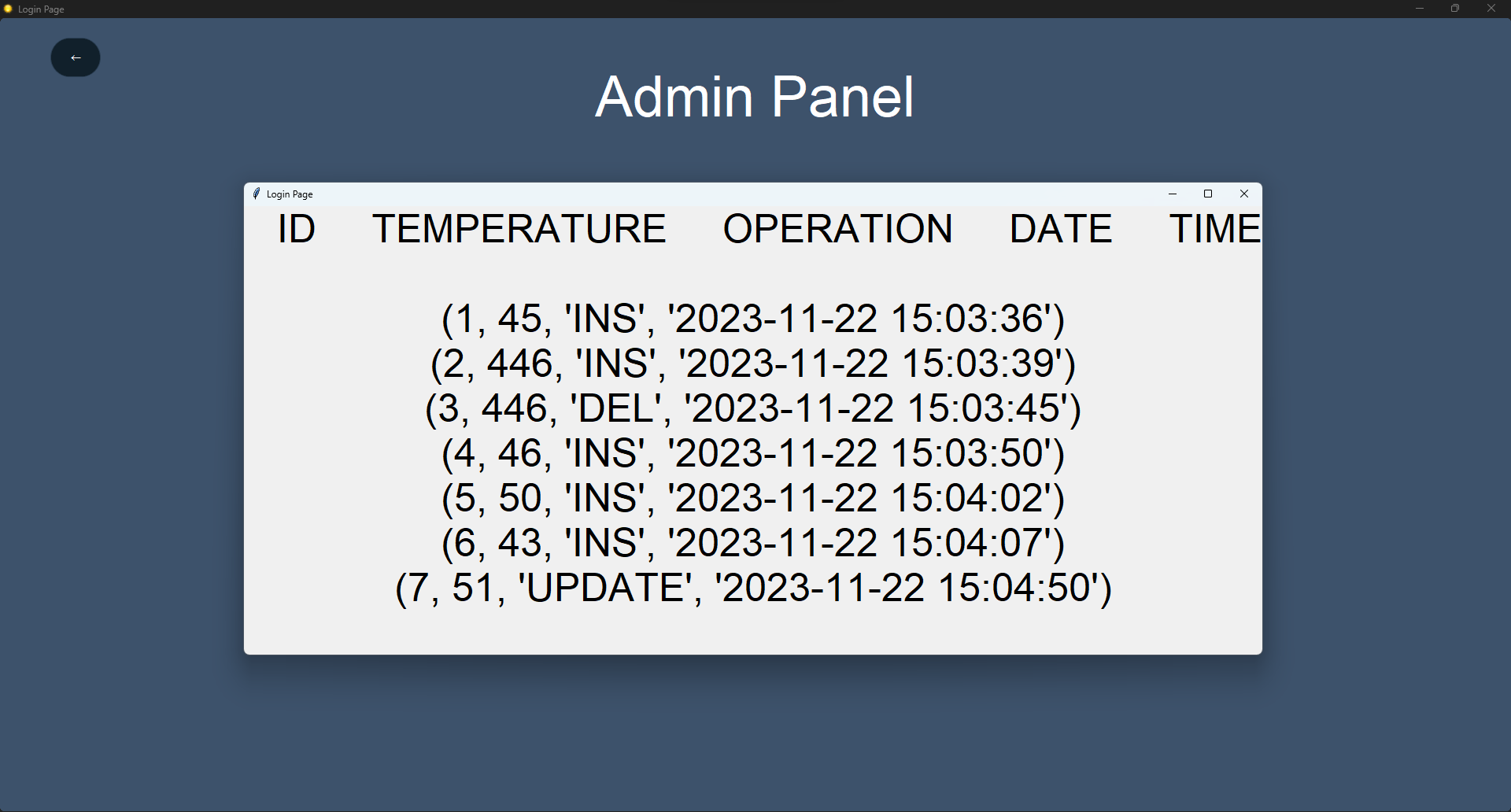
**Admin Panel:**

****

**All Users Info:**

****

**Audit Logs Info:**

****