INT216 CAPSTONE

Project on

Project Title: Weather Interface Report

Submitted by:

Name: Kedhareswer Naidu

Registration Number: 12110626

Roll Number: RK21HCA02

Section: K21HC

Course Code: INT216

Mentor: Sir Ved Prakash Chaubey

School of Computer Science and Engineering

DECLARATION

I, hereby declare that the project work entitled ("Weather Interface") is an authentic record of our work carried out as requirements of the Capstone Project for the award of a B.Tech degree in CSE AI and ML from Lovely Professional University, Phagwara, under the guidance of "Sir Ved Prakash Chaubey" during August to November 2022. All the information furnished in this capstone project report is genuine and based on our intensive work.

Name of Student:

Registration Number:

Date:

CERTIFICATE

This is to certify that the declaration statement made by the student is correct to the best of my knowledge and belief. They have completed this Capstone Project under my guidance and supervision. The present work is the result of their original investigation, effort, and study. No part of the work has ever been submitted for any other degree at any University. The Capstone Project is fit for the submission and partial fulfillment of the conditions for the award of a B.Tech degree in CSE AI and ML from Lovely Professional University, Phagwara.

Signature and Name of the Mentor

Designation

School of Computer Science and Engineering,
Lovely Professional University,
Phagwara, Punjab.

Date:

Table of Contents

S.No.	Title	Page number
1	Declaration	
2	Certificate	
3	Abstract	
4	Introduction	
5	Objective	
6	Description	
7	Code	
8	Code Output	
9	Scope of Project	
10	Future Development of Project	
11	Conclusion	

Abstract

→ This code is about Python GUI application for a weather app. It allows the user to enter a location and select a unit (either Celsius or Fahrenheit) and retrieves weather data from the OpenWeatherMap API based on the user input. The weather data is displayed in a table with four categories: temperature, humidity, wind speed, and description. If the location entered is not found, an error message is displayed in the table.

→ The GUI itself consists of a background image (a map), labels and entry fields for the location and unit selection, a button to retrieve weather data, and the weather data table. The user inputs their desired location and unit, then clicks the "Get Weather" button to retrieve the weather data from the API. The weather data is then displayed in the table below.

→ Overall, this code demonstrates how to use the requests and json modules to retrieve and parse data from an API, and how to create a basic GUI application using the tkinter and ttk modules.

Introduction

→ The code implements a simple weather application that allows users to retrieve current weather information for a specific location. The app is built using Python and uses the tkinter library to create a graphical user interface (GUI) for the user to interact with. The GUI consists of a background image and labels and entry fields for the user to input their desired location and temperature unit (either Celsius or Fahrenheit).

→ The OpenWeatherMap API is used to retrieve the weather information for the specified location. The requests and json libraries in Python are used to make an API request and parse the JSON data returned by the API. Once the weather data is retrieved, it is displayed in a table format using the ttk library in tkinter. The table displays the temperature, humidity, wind speed, and a brief description of the weather conditions for the specified location.

→ Overall, the code provides a simple yet effective way for users to retrieve current weather information for a specific location. The application is user-friendly and easy to use, making it a useful tool for anyone interested in checking the weather.

Objective

→ The objective of this code is to create a graphical user interface (GUI) for a weather application that allows users to retrieve weather information for a specific location. The GUI consists of several elements, including a background image, labels, and entry fields for location input and unit selection, a button to retrieve weather data, and a table to display the weather data.

→ The GUI is created using the tkinter module, which provides a set of tools for creating graphical user interfaces in Python. The GUI consists of a main window, which is defined using the Tk () class, and several widgets, such as labels, buttons, and entry fields, which are defined using various tkinter classes, such as Label, Button, and Entry. The background image is loaded using the PhotoImage class and displayed using the Label widget. The table is created using the ttk module, which provides a set of enhanced widgets over the standard tkinter widgets and is displayed using the Treeview class. Overall, the code provides a functional and user-friendly interface for retrieving and displaying weather information.

Description

→ This code is a Python script that creates a graphical user interface (GUI) for a weather app that retrieves and dislays weather data for a user-specified location using data from the OpenWeatherMap API. The app uses the tkinter library to create the GUI.

→ The code starts by importing the required libraries: requests, json, and tkinter. The tkinter library is imported as tk, and the ttk sub-library is imported separately to create the table widget for displaying the weather data. A global variable is defined to hold the background image for the app.

→ The function set_up_gui() is then defined, which sets up the GUI for the weather app. It creates a tkinter window with dimensions 700x400 and sets the title to "Weather App". It loads the background image from a file and creates a label widget to display the image in the window. It also creates a welcome message and labels and entry fields for location input and unit selection.

→ The function get_weather() is defined to retrieve weather data from the OpenWeatherMap API based on the user's input for location and unit selection. It first gets the location input from the user and constructs a URL for the API request based on the input location and unit selection. It then makes an API request to the OpenWeatherMap

API using the requests library and the constructed URL. The API response is parsed as JSON data using the json library. The relevant weather information is then extracted from the JSON data and used to update the table widget created earlier.

→ Finally, the function set_up_gui() creates a button widget to retrieve weather data when clicked and a table widget to display the weather data. The mainloop() function of the tkinter library is called to run the GUI and wait for user input. When the user clicks the "Get Weather" button, the get_weather() function is called to retrieve and display the weather data for the specified location. If the location is not found, an error message is displayed in the table widget.

Code

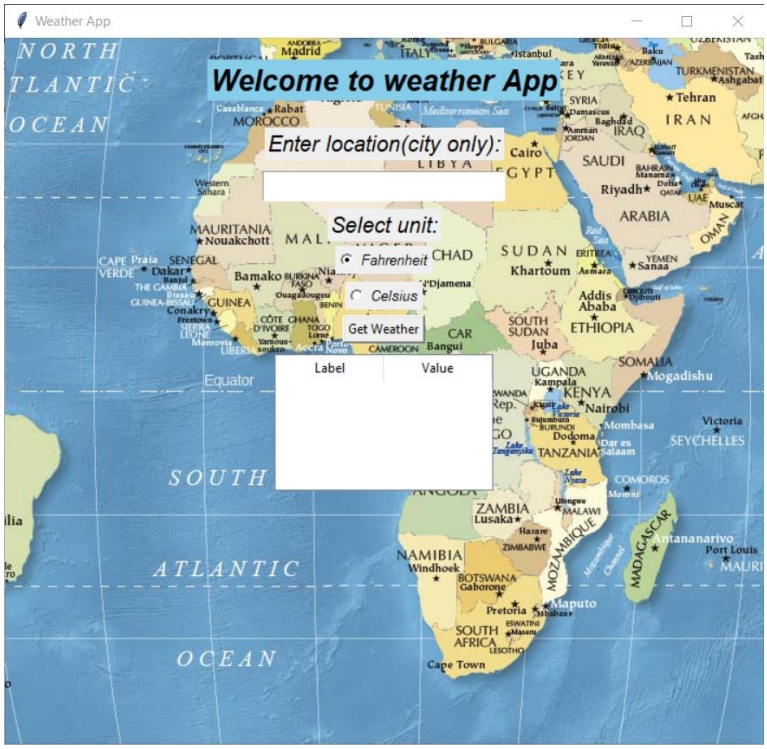
```
import requests

import json

import tkinter as tk

from tkinter import ttk


# Set up background image

bg_image = None


def set_up_gui():

    global bg_image


    # Set up the GUI

    root = tk.Tk()

    root.geometry("700x400")

    root.title("Weather App")


    # Load the background image

    bg_image = tk.PhotoImage(file = "map4.png")


    # Create a label to display the background image

    bg_label = tk.Label(root, image=bg_image)

    bg_label.place(x=0, y=0, relwidth=1, relheight=1)
```

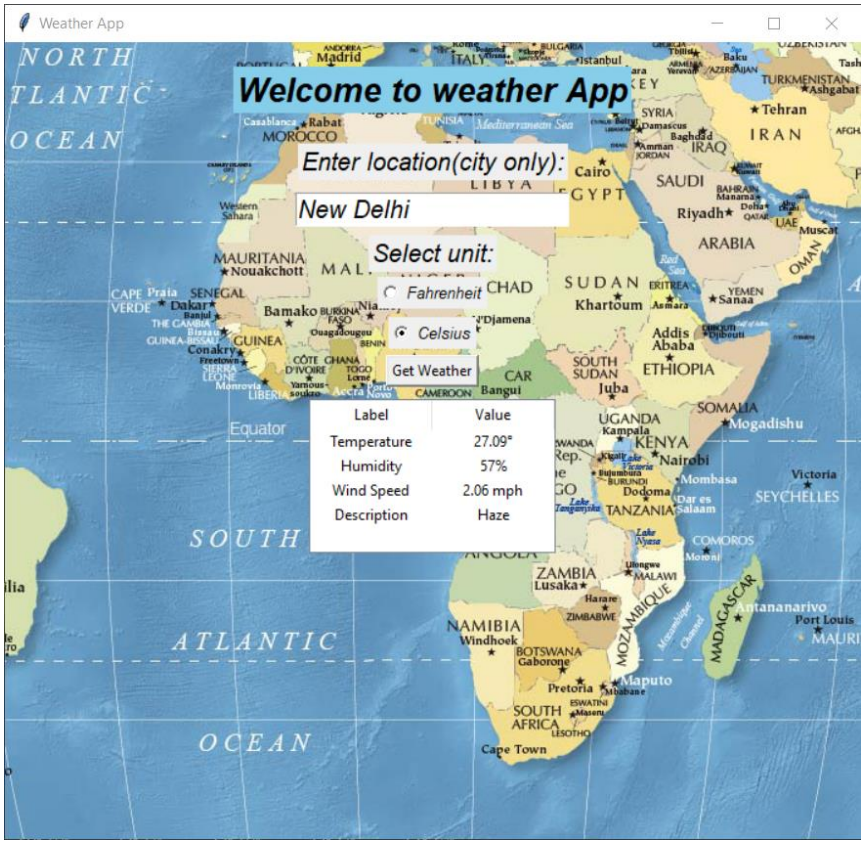
```
    location_label = tk.Label(root, text="Welcome to weather App", font=("Roffe",20,
"italic","bold"),bg="Sky blue")
    location_label.pack(pady=20)

    # Create labels and entry fields for location input and unit selection
    location_label = tk.Label(root, text="Enter location(city only):",
font=("Roffe",15,"italic"), anchor='center')
    location_label.pack(pady=5)
    location_entry = tk.Entry(root, font=("Roffe",15,"italic"))
    location_entry.pack(pady=5)

    unit_label = tk.Label(root, text="Select unit:", font=("Roffe",15,"italic"))
    unit_label.pack(pady=2)
    unit_var = tk.StringVar()
    unit_var.set("imperial")
    unit_imperial = tk.Radiobutton(root, text="Fahrenheit", variable=unit_var,
value="imperial", font=("Roffe",10,"italic"))
    unit_metric = tk.Radiobutton(root, text="Celsius", variable=unit_var, value="metric",
font=("Roffe",10,"italic"))
    unit_imperial.pack(pady=2)
    unit_metric.pack(pady=5)

    # Create a function to retrieve weather data from the API and update the table
```

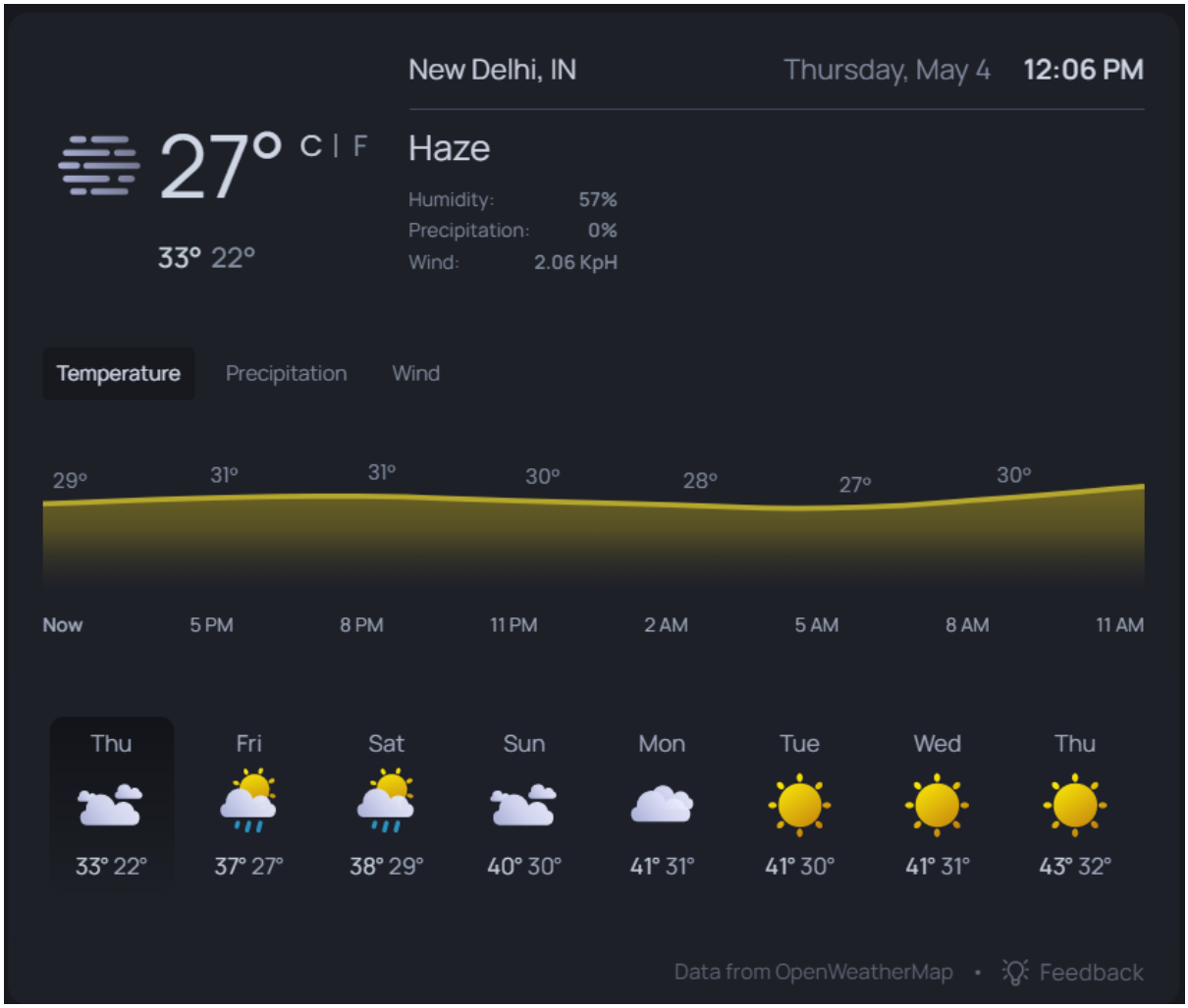
```
    def get_weather():
        # Get the location input from the user
        location = location_entry.get()

        # Make an API request to OpenWeatherMap
        url =
f"http://api.openweathermap.org/data/2.5/weather?q={location}&units={unit_var.get()}
&appid=4a613c2e952661de99c0f360ed97376c"
        response = requests.get(url)

        # Parse the JSON data from the response
        data = json.loads(response.text)

        # Check if the API call was successful
        if data["cod"] != "404":
            # Extract the relevant weather information
            temp = data["main"]["temp"]
            humidity = data["main"]["humidity"]
            wind_speed = data["wind"]["speed"]
            description = data["weather"][0]["description"]

            # Update the weather data table
            table.delete(*table.get_children())
            table.insert("", 0, values=("Temperature", f"{temp}°"))
```

```
            table.insert("", 1, values=("Humidity", f"{humidity}%"))
            table.insert("", 2, values=("Wind Speed", f"{wind_speed} mph"))
            table.insert("", 3, values=("Description", f"{description.capitalize()}"))
        else:
            # Display an error message if the location was not found
            table.delete(*table.get_children())
            table.insert("", 0, values=("Error", "Location not found."))

    # Create a button to retrieve weather data when clicked
    button = tk.Button(root, text="Get Weather", command=get_weather)
    button.pack()

    # Create a table to display weather data
    table = ttk.Treeview(root, columns=("Label", "Value"), show="headings", height=5)
    table.heading("Label", text="Label")
    table.column("Label", width=100, anchor="center")
    table.heading("Value", text="Value")
    table.column("Value", width=100, anchor="center")
    table.pack(pady=10)
# Run the GUI
    root.mainloop()
set_up_gui()
```

Code Output


Weather App
Welcome to weather App
Enter location(city only):
Select unit:
Fahrenheit
Celsius
Get Weather
Label
Value


Weather App
Welcome to weather App
Enter location(city only):
New Delhi
Select unit:
Fahrenheit
Celsius
Get Weather
Label
Value
Temperature
27.09°
Humidity
57%
Wind Speed
2.06 mph
Description
Haze


New Delhi, IN
Thursday, May 4
12:06 PM
27° C | F
Haze
Humidity: 57%
Precipitation: 0%
Wind: 2.06 KpH
33° 22°
Temperature
Precipitation
Wind
29° 31° 31° 30° 28° 27° 30°
Now 5 PM 8 PM 11 PM 2 AM 5 AM 8 AM 11 AM
Thu 33° 22°
Fri 37° 27°
Sat 38° 29°
Sun 40° 30°
Mon 41° 31°
Tue 41° 30°
Wed 41° 31°
Thu 43° 32°
Data from OpenWeatherMap
Feedback

Scope of the Project

→ The scope of this project is to develop a weather app using the OpenWeatherMap API and Python's tkinter library for creating a graphical user interface (GUI). The user can enter a location (city only) and select a unit of temperature measurement (imperial or metric) to retrieve the current weather conditions for that location. The weather data includes temperature, humidity, wind speed, and a brief description.

→ The GUI is designed with a background image of a map and the weather data is displayed in a table using the ttk.Treeview widget. The user can click on the "Get Weather" button to retrieve and display the weather data for the entered location. If the entered location is not found, an error message is displayed in the table.

→ The scope of this project could be expanded by adding additional features such as a 7-day forecast for the entered location, the ability to save and view previously searched locations, and the option to select different weather data to display (such as precipitation, UV index, or air quality). Another possible expansion could be to add a feature to compare the weather conditions between multiple locations. Additionally, the GUI could be improved by adding more visual elements such as icons or graphs to display the weather data in a more engaging and informative way.

Future Development of the Project

There are several potential avenues for the future development of this weather app project:

Additional features:

1. The app could be expanded to include more features such as a 7-day forecast, the ability to save favorite locations, or even the option to receive weather alerts for severe weather conditions.
2. User interface improvements: While the current GUI is functional, there is room for improvement in terms of design and user experience. For example, the app could incorporate more visually appealing graphics or provide more detailed weather information in a user-friendly way.
3. Integration with other services: The app could be integrated with other services to provide additional functionality, such as the ability to plan outdoor activities based on weather conditions or to check flight status in case of weather-related delays or cancellations.

Overall, there are many directions in which this project could be expanded and improved upon to make it even more useful and user-friendly for individuals seeking accurate and up-to-date weather information.

Conclusion

In conclusion, the weather app project is a simple yet useful tool for checking the weather conditions of any location around the world. The app retrieves weather data from the OpenWeatherMap API and displays it in a user-friendly GUI. The user can input the location and select the preferred unit of measurement, and the app will display the current temperature, humidity, wind speed, and weather description for that location.

This project has great potential for further development, including the addition of more features such as weather forecasts, historical weather data, and alerts for severe weather conditions. It can also be improved by adding more intuitive user interface elements, such as a map-based interface for selecting locations, or voice-based input for hands-free use. Furthermore, the project can be integrated with other platforms and devices, such as mobile applications or smart home assistants, to provide users with more convenience and accessibility.

Overall, this weather app project is a great example of how programming can be used to solve real-world problems and provide useful tools for everyday use. With the continuous development of new technologies and the increasing demand for personalized and accessible information, this project has the potential to become a widely used and highly valuable tool for people all around the world.

THANK YOU