Weather-Now app documentation

Introduction

Project Overview:

The Weather App is a simple and useful web application built with the Django framework. Its main purpose is to display real-time weather details for any city or country entered by the user. The app connects with a third-party Weather API to fetch live data such as temperature, humidity, wind speed, and weather conditions.
The idea behind this project was to explore how Django can work with external APIs and how data from an API can be displayed on a website dynamically. The app focuses on providing accurate results in a clean and easy-to-use layout, without using any JavaScript.

Objectives:

The main goals of this project are:

To create a web application that provides real-time weather updates.
To practice Django's backend features and learn API integration.
To design a simple, responsive layout using HTML and CSS.
To understand how to handle API responses in Django views.

Scope:

This project is useful for anyone who wants to quickly check live weather information for different cities around the world. In the future, the app can be improved by adding more features like weather forecasts, automatic location detection, or user profiles for saving favorite cities.

Page 2: System Design and Implementation

2.1 System Design

The app follows Django's Model-View-Template (MVT) structure. The frontend is designed with HTML and CSS, while the backend uses Django's views to communicate with the Weather API. The data is processed and passed to the template to be displayed neatly on the webpage.

Model: Not heavily used, since data comes directly from the API instead of a database.

View: Handles logic, makes API requests, and processes responses.

Template: Displays the fetched weather data in a readable format.

Working Process

The user opens the website and enters a city or country name.
Django takes that input and sends a request to the weather API.
The API responds with live data in JSON format.
Django processes that data and sends it to the HTML template.
The webpage then displays details like temperature, humidity, and weather condition.

Tools and Technologies Used

Frontend: HTML, CSS
Backend: Python (Django Framework)
API Used: OpenWeatherMap API (or any similar weather service)
Development Tools: Visual Studio Code, Git, Postman

Key Features

Real-time weather information using Django API integration.

Displays temperature, humidity, and general conditions.
Simple and clean interface made with only HTML and CSS.
Error handling for incorrect or missing city names.

Testing, Results, and Conclusion

Results

The app successfully retrieves and displays real-time weather details for any valid city name entered by the user. It shows the temperature in Celsius, humidity percentage, weather condition (e.g., cloudy, sunny, rainy), and wind speed. The interface is simple, mobile-friendly, and works smoothly across browsers.

Testing

Several test cases were performed to check the accuracy and reliability of the application.

Test Scenarios:

When a valid city is entered---correct weather data is shown.
When an invalid city name is entered---error message is displayed.
When API is not reachable--- user is informed that data can't be loaded.
All tests were successful, and the app performed as expected in different situations.

Challenges

During development, a few challenges were faced:

Handling API response errors when users entered invalid inputs.
Formatting JSON data to make it look clean on the webpage.
Managing API keys and ensuring they remain secure.

Conclusion

This project was a great learning experience for understanding how APIs can be connected with Django applications. The Weather App not only displays live data effectively but also shows how backend logic and template rendering work together. It's a small but complete project that demonstrates practical web development skills in Python and Django.