# Project Summary of Weather App

## What My App Does:

Stay informed and prepared with the **Weather App**, a sleek and intuitive desktop application that delivers real-time weather updates for cities across the globe. Designed for speed, simplicity, and a visually engaging experience, this app puts accurate weather data at your fingertips—powered by the trusted Open-Meteo API.

Whether you're planning your day or checking conditions in another city, the Weather App makes it effortless with built-in autocomplete, live forecasts, and intelligent caching to boost performance.

Perfect for everyday users, travelers, or anyone who wants quick and reliable weather updates in a modern interface.

## App Features (with Screenshots):

* **Live Weather Lookup**: Enter a city name to retrieve current temperature, wind speed, and weather conditions.
* **Autocomplete Suggestions**: As you type a city name, the app suggests popular cities to speed up input.
* **Data Caching**: Weather data is cached for 1 hour to reduce API calls and improve performance for repeated queries.
* **API Integration**: The app uses the Open-Meteo geocoding and forecast APIs to fetch accurate location and weather data.
* **User Guidance**: A built-in help menu provides a quick README guide for users, including troubleshooting tips.
* **Clean UI**: Designed with a fixed-size window, styled fonts, and optional background imagery for a visually appealing experience.

This app is a great demonstration of combining API usage, local caching, and GUI design in a Python project

Screenshot 1: City entered – “Tokyo”

A screenshot of a computer

AI-generated content may be incorrect.

This is the Autocomplete suggestion: when the user start to enter, the autocomplete feature will suggest the possible entries of city.

Screenshot 2: Display – “Temperature: 19.5°C | Wind Speed: 54 km/h | Weather: Partly Cloudy”

A screenshot of a weather application

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.Click ? icon, the separate Readme window shall be prompted.

## How I Used AI:

I used: -

**1. AI Code Assistants for Rapid Development**

* **ChatGPT or Perplexity:**  
  I entered the requirements or code into these AI assistants and ask for:
  + Code generation (e.g., “Write a Tkinter app that shows weather for a city”)
  + Code review, e.g., “How can I improve this code?”
  + Feature suggestions, e.g., “Add autocomplete to the city input”
  + Debugging help, e.g., “Why does this error occur?”
  + Documentation generation, e.g., “Draft or improve the internal program documentation and a README for this app”
* **Iterative improvement or development:** 
  + Paste the initial python code or describe your idea.
  + Ask for step-by-step improvements, new features, or error fixes.
  + Iterate: Try the suggestions, then ask follow-up questions as needed.

**2. Use AI-Powered IDE Extensions based on the course material, I**

* **GitHub Copilot:**
  + Integrates with VS Code
  + Offers real-time code completions, docstrings, and suggestions as you type.
* **Tabnine or Qodo.:**
  + Suggest code, fix bugs, and help with documentation.

**3. I used AI to generate images for Weather App, but “not-so-success” integrated with python and thus I decided to do next time**

* AI image generators thru DALL·E which I learned from course material**.**

**4. Use AI for Testing and Documentation**

* generate test cases for the functions.
* generate docstrings and comments for clarity.
* generate a README or help content by describing the app’s features.

## What I Learned and What Was Challenging:

1. I learned :-
   * how to build a modern Python GUI application using Tkinter, including advanced features like autocomplete for city names, integrating a background image, and displaying results in a user-friendly, centered format.
   * the practical experience with error handling, caching, and using external APIs (Open-Meteo) to fetch real-time weather data.
   * how to use type hints, docstrings, and modular code to improve readability and maintainability.
   * how to leverage AI tools for code generation, debugging, and documentation, which accelerated my development process and helped me solve problems more efficiently.
2. The challenges are: -
   * Handling file paths and image loading on Windows required careful attention to escape characters and library updates, e.g., Pillow’s Image.LANCZOS replacing Image.ANTIALIAS.
   * Ensuring compatibility with different Python versions, especially regarding type hints and third-party package imports, was sometimes tricky.
   * Integrating a professional background image and keeping the UI readable, e.g., centering text, adjusting label transparency, took experimentation.
   * Managing dependencies and resolving errors like missing packages, e.g. dateutil, Pillow, but not limited to, or import issues was a key part of the process.
   * Implementing robust error handling and making the app user-friendly for both technical and non-technical users required several iterations and feedback cycles

## One Thing I Am Proud Of:

Within short period of time, the Weather App was created and is usable thru AI-based tools.

## One Thing I Would Improve:

One thing I would improve is enhancing the user interface further by incorporating more advanced UI frameworks like PyQt or Kivy to provide a richer, more responsive, and visually appealing experience beyond what Tkinter offers. This would allow for smoother layouts, better styling options, and potentially cross-platform consistency, addressing some limitations encountered with Tkinter’s basic widgets and styling capabilities. Additionally, I would refine AI prompt engineering to generate even more precise and context-aware code suggestions, improving development efficiency and output quality.

**Link:**

https://docs.google.com/document/d/1KCZjqA6wRVAxZ5kccBf3dm-0VxeZURm-/edit?usp=drive\_link&ouid=101253427602294653421&rtpof=true&sd=true