M602 Individual Final Project Weather Dashboard

Student: *Suraj Shah (GH1047720)*

Module: *M602 Computer Programming Summer 2025*

Date: *20 September*

1. Project Overview aim and scope

So this project is basically a Python weather dashboard thing. It pulls in current weather and forecasts for whatever cities you pick. Uses the OpenWeather APIs for that. I have integrated off some object-oriented stuff, handles files for saving user prefs and history, deals with errors pretty well, hooks into external APIs, and even has a basic GUI plus some visualization options.

2. Features implemented

- Search by city name. That means geocoding to get coordinates first, then hitting the One Call weather endpoint. From openweathermap.org
- Favorites management. You can save them, list them out, remove some. All stored in favorites.json.
- History logging too. Every request gets logged to history.csv.
- Visualization part. It plots a 5-day min max temperature trend using matplotlib.
- Simple GUI with Tkinter. Just to make it user-friendly, you know.
- Error handling covers detailed stuff. Like missing API key, network problems, geocoding fails, file I/O issues.

3. Design decisions and structure

- OOP setup with three main classes. WeatherAPI wraps the external API. CityManager handles persistence. WeatherApp acts as the controller. This way, unit testing gets easier, and extending it later isn't a hassle.
- APIs picked are OpenWeather One Call. That's the recommended one for mixing current hourly daily data. Plus Direct Geocoding for solid city to coordinates mapping. Sticking with One Call cuts down on API calls, simplifies handling the data. openweathermap.org plus one.
- File formats. Favorites in JSON because it's structured, easy to edit. History in CSV for basic audit and tracking.
- Exception handling uses custom ones. They wrap HTTP errors and file I/O problems. Gives clearer messages to the user.

4. Implementation details key snippets and rationale

- WeatherAPI.geocode for a city hits the OpenWeather Direct Geocoding endpoint. Gets lat lon reliably before calling One Call.

- WeatherAPI.get_current_and_daily takes lat lon. Calls One Call with exclude to skip unnecessary payloads. Requests units metric by default.
- CityManager makes sure files exist. Creates them automatically if they're missing. Stops those crash-on-missing-file errors.
- For plotting, pandas to_datetime handles dates robustly. Matplotlib because it's widely used, easy to toss into a demo.

5. Testing and validation

i. Manual testing scenarios.
ii. Valid city search. Leads to correct coordinates, displayed temperatures.
iii. Invalid city name. Gives a graceful error message. Like City X not found.
iv. API key missing. Shows an instructive error on setting OPENWEATHER_API_KEY.
v. File permission issues. Handled with CityFileError.
vi. Edge cases thought about. API downtime, rate limits that expose errors, empty favorites list.

6. Limitations and future improvements

i. Rate limiting and paid tier features. One Call 3.0 has subscription options for advanced data. For heavy use, you might need a higher tier. openweathermap.org
ii. GUI is pretty minimal right now. Could improve with a more polished layout, add icons, embed charts.
iii. Unit tests not included. Could add them using pytest to check parsing and file handling.
iv. Maybe add local caching of responses. That would limit API calls, allow offline display.

7. Academic integrity and originality

This implementation is my own original work. I mean, I consulted external docs for OpenWeather API endpoints and parameters. References are there. openweathermap.org plus two openweathermap.org plus two

8. How to run and demo notes

Steps to run, check the README. There is also the demo video, for convenient.

Final notes and citations

OpenWeather One Call Current endpoints and call format. https://openweathermap.org/.org plus one
How to obtain an API key signup. OpenWeather appid guide.