Project Report: Weatherbash Weather Application

Date: November 23, 2025 **Author:** Parth Gaur **Project Status:** Initial Development

1. Executive Summary

The **Weatherbash Weather App** is a proof-of-concept desktop application designed to retrieve and display current meteorological data for selected locations within India. Developed using Python's Tkinter library for the graphical user interface (GUI) and the requests library to interface with the OpenWeatherMap API, the application successfully demonstrates basic API consumption and data presentation in a desktop environment. The primary limitation is the hardcoded API key and the restricted list of queryable locations.

2. Project Overview and Technology

2.1. Purpose

The main objective of this project is to provide a simple, functional tool for users to quickly check essential weather metrics for major Indian states and territories.

2.2. Technology Stack

Component	Technology	Role
GUI	Python (Tkinter/ttk)	Provides the window, labels, buttons, and the Combobox dropdown.
Networking	Python (requests)	Handles the synchronous HTTP GET request to the external API.
Data Source	OpenWeatherMap API	Supplies real-time weather information in JSON format.
Language	Python 3.x	Core programming language.

3. Core Features Analysis

The application is structured to be straightforward and easy to use:

Feature	Description	Implementation Detail
Location Selection	A dropdown (ttk.Combobox) populated with a predefined list of Indian states and Union Territories (list_name).	Uses textvariable=city_name to capture user selection.
Data Retrieval	Triggered by the "Done" button, executing the data_get() function.	Uses a simple requests.get() call with a hardcoded API key.
Data Display	Four key meteorological	Dedicated Tkinter Labels

	parameters are displayed on the main window.	(W_label1, temp_label1, etc.) are dynamically updated using .config(text=...).
Temperature Conversion	Raw temperature data from the API (Kelvin) is converted to Celsius.	Performed directly within data_get() using the formula: K - 273.15.

4. Technical and Security Review

4.1. Data Handling

The data_get function effectively manages the entire data pipeline:

1. Retrieves the city name string.
2. Constructs the API URL.
3. Makes the API call.
4. Parses the JSON response.
5. Performs necessary calculations (Kelvin to Celsius).
6. Updates the GUI.

This single-function approach is suitable for a small-scale application but could be refactored for better modularity in a larger project (e.g., separating API logic from GUI updates).

4.2. API Key Security

As noted in the readme.md, the API key (3daf7fb98ac4f7b004cdd1a86118ca5e) is **hardcoded** into the data_get function. This presents a **significant security vulnerability** as the key is exposed in the source code.

5. Recommendations for Future Development

Based on standard development practices and the suggested improvements in the project's documentation, the following actions are recommended:

5.1. Security and Robustness (High Priority)

- **API Key Management:** The hardcoded API key must be moved out of the source code. It should be loaded from a system environment variable, configuration file, or a secure secrets management service.
- **Error Handling:** Implement try...except blocks around the requests.get() call and JSON parsing (data = requests.get(...).json()). This is critical to gracefully handle network errors, timeouts, or API responses indicating an invalid city name or bad status code.

5.2. User Experience and Scope (Medium Priority)

- **Input Flexibility:** Replace the fixed-list Combobox with a simple Entry widget to allow users to input *any* global city supported by the OpenWeatherMap API.
- **User Feedback:** Add a status bar or temporary label to inform the user about the process (e.g., "Fetching data...", "Error: City not found.").
- **Unit Options:** Provide a toggle button or dropdown to switch the temperature output between Celsius and Fahrenheit.

5.3. Code Quality (Low Priority)

- **Refactoring:** Separate the API logic (data fetching and processing) from the GUI manipulation logic to make the code easier to test and maintain.
- **Styling:** Enhance the GUI's aesthetics by utilizing Tkinter styling options or exploring alternative libraries like custom Tkinter themes or customtkinter.