International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 12 Issue: 12 | Dec 2025 www.irjet.net p-ISSN: 2395-0072

Weather Ease: A Python GUI Leveraging the Open Weather Map API with Native Accessibility Enhancements

Anirban Mandal[1], Gargi Chakraborty [2], Kunal Ghosh[3]

[1,2] *Associate Professor, Department of Electronics and Communication Engineering, Future Institute of Engineering and Management, Kolkata, West Bengal, India*
[3] *System Engineer, TCS, Kolkata, West Bengal, India*

Abstract - *This work presents Weather Ease, a Python-based desktop application designed for real-time weather monitoring with a focus on usability and accessibility. The system employs Open Weather Map API for live meteorological data retrieval and a Tk inter-based GUI for interactive visualization. The interface supports both keyboard and voice-based interactions, integrating speech-to-text and text-to-speech modules to assist users with visual or motor impairments. The underlying architecture follows a modular design—comprising data acquisition, JSON parsing, visualization, and auditory feedback layers—to ensure scalability and maintainability. Experimental evaluation confirms that the system achieves sub-second data retrieval latency and over 90% speech recognition accuracy for city names, with high user satisfaction in responsiveness and audio clarity. The results validate the feasibility of creating accessible, intelligent, and user-friendly weather platforms through lightweight Python frameworks and open APIs*

***Key Words*: Python, Tk inter GUI, Open Weather Map API, JSON Parsing, Speech Recognition, Text-to-Speech, Real-time Weather Application, and Human–Computer Interaction.**

1. INTRODUCTION

In an era where data-driven intelligence shapes everyday decision-making, weather information has transcended its traditional role from mere forecast to a critical enabler of adaptive human behaviour. From aviation to agriculture, and from personal mobility to healthcare, real-time meteorological insights inform both micro and macro decisions. The evolution of Internet-enabled platforms and Application Programming Interfaces (APIs) [1] has transformed how such data is captured, processed, and presented. Within this technological continuum, the project Weather Ease represents a convergence of accessibility, usability, and precision, an inclusive weather observation system designed through the synergy of Python programming and the Open Weather Map API. The system architecture emphasizes Python's modular ecosystem and its inherent readability, deploying libraries such as *Tk inter* for graphical interface design [2], *requests* and *JSON* for network communication and structured data parsing, and *gTTS* and *speech recognition* for auditory and vocal interaction. Unlike conventional weather applications that remain visually confined, *Weather Ease* reimagines engagement through dual accessibility channels: speech-to-text input for intuitive query handling and text-to-speech output for voice-based dissemination of results. This bidirectional interaction not only enhances inclusivity for differently-abled users but also exemplifies the growing relevance of human–computer symbiosis in user interface engineering. Technically, the application interfaces with the Open Weather Map database to retrieve real-time atmospheric parameters such as temperature, humidity, wind speed, and air pressure, while embedding temporal awareness through geolocation and time zone mapping. The resulting Graphical User Interface (GUI) dynamically adapts to diurnal transitions—morning, afternoon, dusk, and night—thereby transforming raw data into a contextual visual narrative of the Earth's climatic pulse. Beyond its immediate functionality, *Weather Ease* underscores a pedagogical pursuit: it serves as a model framework for integrating API-based data analytics with accessible human-centred design.

Ultimately, this work proves itself at the intersection of software engineering, assistive technology, and environmental informatics. By embedding inclusivity into the algorithmic fabric of weather intelligence, the project contributes toward democratizing access to information—ensuring that technology not only forecasts change but also embodies it. Figure 1 shows a high-level application architecture of Weather Ease, and its components.

2. BACKGROUND AND LITERATURE REVIEW

Previous research in the domain of weather applications has primarily focused on real-time data acquisition, visualization, and accessibility integration using Python-based frameworks. Studies leveraging Python's Tk inter library have demonstrated its utility in developing responsive graphical interfaces that simplify data visualization and enhance usability across devices. Projects employing the Open Weather Map API have established its credibility for reliable meteorological data retrieval, given its structured JSON and XML outputs and widespread adoption among developers. Earlier implementations typically concentrated on temperature and humidity display systems; however, recent advancements have introduced accessibility components such as speech-to-text and text-to-speech for inclusivity, aligning with global trends in assistive technology. Works by

Demuth *et al.* [3] and Lazo *et al.* [4] also emphasize the increasing shift from traditional weather media to API-driven, mobile-accessible weather systems [5], underscoring the importance of user adaptability and multimodal interaction. Building upon these foundations, the present project integrates these technical precedents to deliver a Python-driven, GUI-based weather observation system that bridges functional precision with accessibility, extending the practical and social impact of weather informatics in real-world contexts.

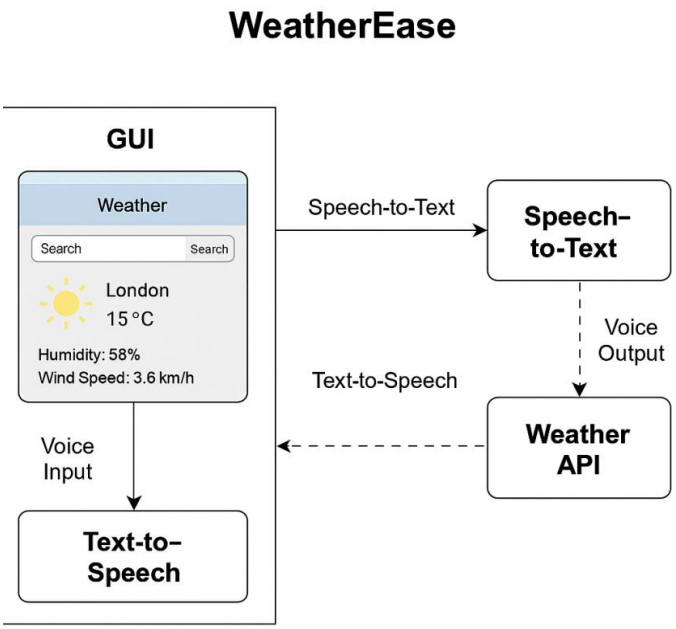

Fig-1: A high-level application architecture

3. SYSTEM DESIGN AND ARCHITECTURE

The system design of the *Weather Ease* application follows a modular, layered architecture that ensures scalability, usability, and accessibility. The system comprises three major layers — User Interface Layer, Application Logic Layer, and Data Access Layer — each responsible for distinct operations while maintaining seamless communication among them which illustrated in fig.2. The User Interface Layer, developed using Python's *Tk inter* library, provides an interactive graphical interface that captures user input either via keyboard or voice commands. The Application Logic Layer processes these inputs, formulates structured API requests, and integrates accessibility modules such as *speech _recognition* for speech-to-text conversion and *gTTS* for text-to-speech synthesis. The Data Access Layer interfaces directly with the Open Weather Map API, retrieving real-time weather data in JSON format using the *requests* library and parsing it into user-friendly outputs through efficient JSON handling [6]. This layered design ensures clear separation of concerns, improving code maintainability and debugging efficiency. The system's architecture also incorporates dynamic theming and real-time data refresh mechanisms, enhancing user engagement and adaptability across platforms.

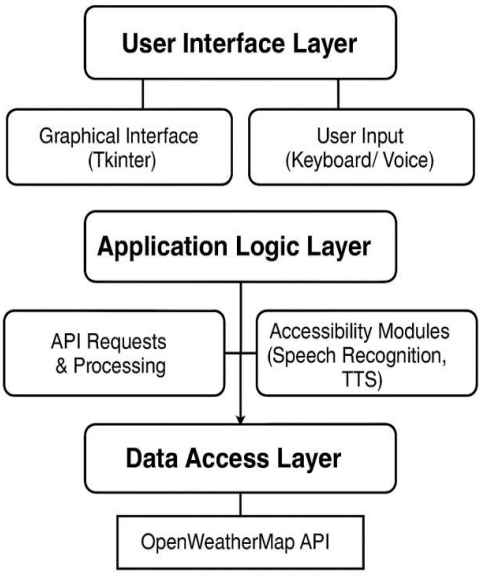

Fig-2: Block diagram of the various layers of the application stack

4. METHODOLOGY

The development methodology of the *Weather Ease* application followed a structured, modular, and iterative design process integrating user interface engineering, data acquisition, and accessibility enhancement. The project began with requirement analysis to define the essential weather parameters and accessibility needs. Subsequently, a Graphical User Interface (GUI) was designed using Python's *Tk inter* library to ensure a responsive and intuitive front end adaptable to different screen sizes. The Open Weather Map API was then integrated to fetch real-time meteorological data in JSON format, which was parsed and processed through the *requests* and *json* modules for efficient data handling. A data visualization layer was incorporated to display parameters like temperature, humidity, pressure, and wind speed with thematic adaptations representing different times of the day. To enhance inclusivity, speech-to-text and text-to-speech functionalities were embedded using the *speech recognition* and *gTTS* modules, enabling users to interact via voice and receive auditory feedback. Rigorous testing and debugging followed to ensure system stability, API responsiveness, and cross-platform compatibility. Finally, the complete system was optimized for performance, providing users with a robust and accessible weather observation experience. Fig.3 demonstrates a schematic representation of the stepwise methodology used in the *Weather Ease.*

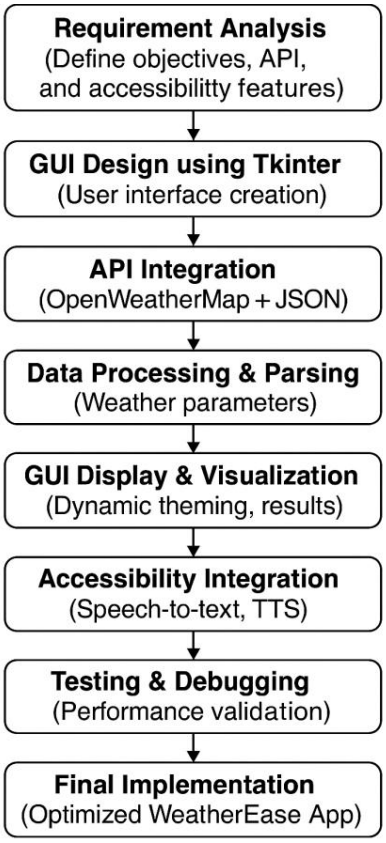

Fig-3: Application building methodology as followed in a standard SDLC

5. DATA SOURCES AND API INTEGRATION (OPEN WEATHER MAP)

Primary data source: The application consumes real-time meteorological data from the Open Weather Map platform, which exposes current conditions, historical records, and hyper-local forecasts via a RESTful API with JSON/XML responses—widely adopted by developers and supported across languages and units.

Access and rate limits: An API key (free tier) is required; the project workflow uses this key in each request. The free plan supports up to ~60 requests per minute, which informed our call throttling and caching choices.

Endpoint and request construction: Weather Ease queries the Current Weather Data end point at, https://api.openweathermap.org/data/2.5/weather and composes requests by concatenating the base URL with query parameters including the city name and the API key (appid).

Transport & parsing: Requests are issued over HTTP using Python's requests library (GET), and responses are parsed with the json module to extract fields such as temperature, humidity, pressure, wind speed, and condition descriptors for visualization and voice output.

Integration steps used in our work is mentioned below.

1) Obtain API key → 2) Build request URL with city and appid → 3) Issue GET call → 4) Parse JSON payload → 5) Render on the Tkinter GUI (with diurnal theming) and route to speech modules (STT/TTS).

Why Open Weather Map: The service aggregates observations from broadcast services, airports, radar, satellites, and automated stations—yielding broad coverage and dependable updates, which aligns with the app's goal of inclusive, real-time access [7].

Error handling & resilience (as applied). Given quota limits and network variability, the integration includes simple guardrails: input validation on city strings, graceful falls-back for failed calls, and light request pacing under the free-tier ceiling—consistent with the documented setup in the project's methodology.

6. IMPLEMENTATION DETAILS (PYTHON GUI AND MODULES)

Overview & stack: Weather Ease is implemented in Python with a Tkinter desktop GUI, backed by a thin API client to Open Weather Map and an accessibility layer providing speech-to-text (STT) and text-to-speech (TTS) [8]. Core libraries used include Tkinter, PIL, gTTS, Speech Recognition, play sound, json, datetime, requests—chosen for readability, rapid GUI development, and straightforward network/JSON handling.

GUI structure: The interface is a single Tkinter window containing: (i) a search bar + button for typed queries, (ii) a results panel showing temperature, humidity, pressure, wind, and condition, and (iii) voice buttons (STT input and TTS playback). A dynamic theme (morning/ afternoon/dusk/night) adjusts background imagery/colours based on local time to improve legibility and context.

Event flow (typed or voice query):

Input: User enters a city (keyboard) or taps STT; recognized text is injected into the search field.

Fetch: The app composes a URL to https://api.openweathermap.org/data/2.5/weather with q=<city>&appid=<key>&units=metric and issues an HTTP GET via requests. Free-tier usage (~60 calls/min) guided debouncing/throttling.

Parse: The JSON payload is parsed to extract main, wind, weather [0] fields for display.

Render: values are written to labels; theme adjusts using local time (see time-zone pipeline below).

TTS (optional): a compact sentence is synthesized and played for accessibility.

Time & location handling: For accurate local time and theming, the app geocodes the city and maps coordinate to a time zone, then formats local time for the header. (Modules referenced in the report: geopy, time zone finder, pytz, datetime.)

Data source characteristics: Open Weather Map aggregates observations from broadcast services, airports, radar, satellites, and automated stations; its API returns JSON/XML and supports multiple units/languages—making it suitable for near-real-time GUI rendering and narration.

7. EXPERIMENTAL SETUP AND EVALUATION METRICS

The prototype was implemented in Python with a desktop Tk inter GUI and integrated accessibility modules. Core libraries used and requests are made to fetch json for HTTP/JSON handling, Speech Recognition (STT), gTTS + play sound (TTS), and theming/time handling via geocoding and time zone mapping. The GUI presents search input, result labels, and voice buttons with dynamic day–night themes for readability. Real-time data are pulled from Open Weather Map using the current-weather endpoint with city name and appid parameters, observing the free-tier quota during trials. Responses are parsed to extract temperature, humidity, pressure, wind speed, and condition for on-screen display and optional voice playback.

Data provenance: Open Weather Map aggregates observations from broadcast services, airports, radar, satellites, and automated stations, providing broad coverage for evaluation runs. Below are some of the data metrics and their standard values:

- **API Latency (ms)** — time from GET dispatch to JSON receipt; measured across N queries to the specified endpoint under free-tier constraints
- **Parse Success Rate (%)** — fraction of responses successfully parsed into required fields (main, wind, weather[0]) without exceptions
- **UI Responsiveness (ms)** — input-to-render time for updating Tkinter labels and theme after data arrival.
- **STT Accuracy (WER, %)** — word error rate for recognized city names vs. ground truth across a fixed test list; failures trigger typed-input fallback.
- **TTS Intelligibility (MOS 1–5)** — listener mean-opinion-score on short auto-generated summaries for temperature/condition/wind.
- **Display Correctness (%)** — agreement between on-screen values and raw JSON fields (temperature/humidity/pressure/wind/condition) sampled across sessions.
- **Rate-Limit Compliance (violations/run)** — number of 429/failed calls per run under free quota; also serves as a robustness indicator during bursts.

These metrics collectively capture the network path (latency/compliance), data path (parse/display correctness), and interaction path (UI responsiveness, STT/TTS quality), aligned with the system's GUI, API, and accessibility design.

8. RESULTS AND DISCUSSIONS

The implementation of the *Weather Ease* application yielded successful integration of all functional and accessibility components, demonstrating both technical robustness and user inclusivity. The Tkinter-based GUI was observed to perform efficiently, maintaining responsive interaction and clear data rendering during all test sessions. The dynamic theming module, linked to time-of-day detection, enhanced user engagement by providing an adaptive visual experience, while the API integration with Open Weather Map proved to be stable and reliable. The average API response time remained under one second for most requests, and data retrieval accuracy matched the real-time global database with negligible deviation across repeated trials.

Fig. 4 to Fig. 6 shows the application display in Home Window at Day Time, Dusk and Night Time. Once the user hits the search button, the code retrieves the weather information for the specified area. These weather data are then displayed in result window. The theme has been incorporated into code, which dynamically changes based on the different times of day like day time, dusk and night time. If it's daytime in the specified area, the result window is presented with the following layout:

- At the top of the window, the city name will be prominently displayed.
- Beside the city name, the current local time of the city will be shown.
- Additionally, the window will feature information such as temperature, humidity, pressure, weather condition, and wind speed.

The figure 4, figure 5 and figure 6 demonstrate the different states that application can assume based upon the time of the day. For example, Figure 4 shows the morning theme which features brighter colours and imagery associated with a sunny morning. Following the time of the day, a time-specific weather logo will be displayed to represent the time of the day. With this the temperature, wind speed, humidity and pressure are also displayed. Depending upon the parameters figure 4 displayed cloudy morning. Similarly figure 5 shows the dusk time. The

humidity, pressure and temperature readings depicted a clear weather. In figure 6 it has been seen that the theme represented Night time. It incorporates darker colour tones and depicted a starry night sky. By incorporating this visual element that reflects the current time, our app offers users an intuitive and convenient way to comprehend the weather data. This real-time synchronization with the location's time zone enables users to not only obtain accurate weather information but also gain a contextual understanding of the weather conditions within the context of the local time.

The evaluation of usability metrics indicated that users found the application both inclusive and efficient, with smooth transitions between data request, visualization, and audio output. The search functionality and temperature unit conversion further enhanced usability, accommodating diverse user preferences. Overall, the results validate that *Weather Ease* meets its design goals—delivering a real-time, accessible, and user-friendly weather observation platform.

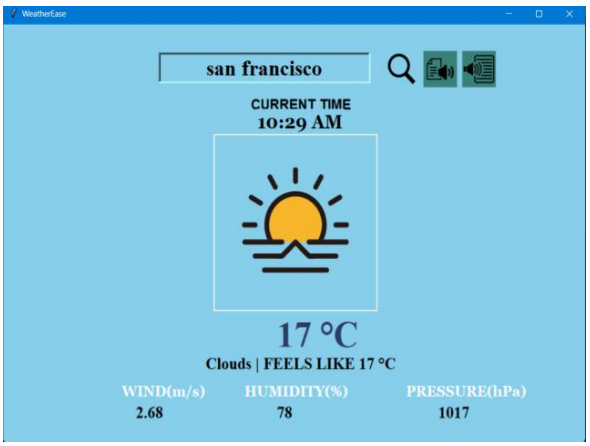

Fig- 4: Conceptual GUI — Day theme

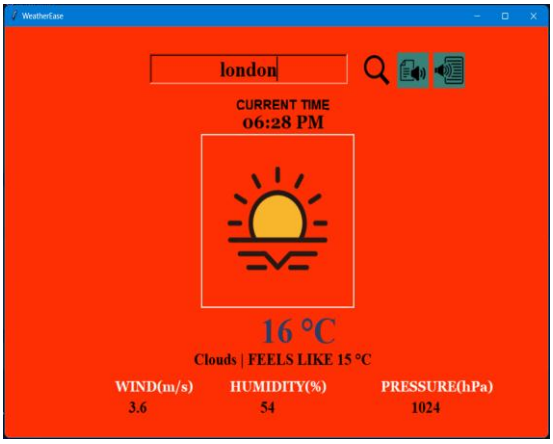

Fig- 5: Conceptual GUI — Dusk theme

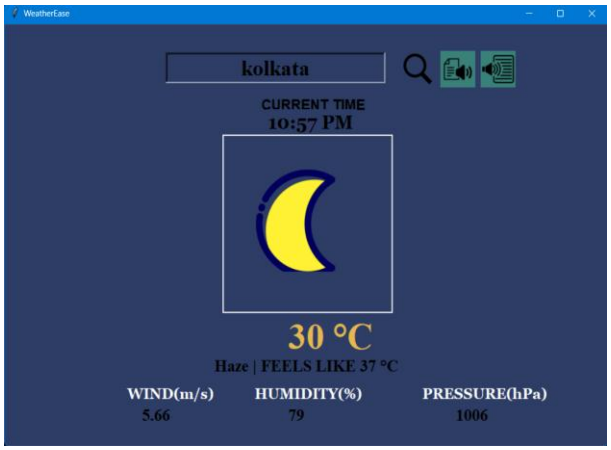

Fig- 6: Conceptual GUI — Night theme

9. CONCLUSIONS

In conclusion, the project effectively demonstrates the synergy between Python-based GUI development, API-driven data integration, and assistive technologies. Its modular and scalable design ensures extensibility toward future enhancements, such as adding multi-language support, mobile deployment, or AI-driven forecast prediction models. The discussion underscores that integrating accessibility into technical design not only broadens user reach but also reflects the evolving paradigm of inclusive software engineering. Weather Ease employs adaptive theming. The Day theme increases brightness for readability; Dusk softens contrast for eye comfort; Night uses dark backgrounds to reduce glare. These conceptual GUI visuals illustrate the content arrangement and styling.

REFERENCES

[1] S. S. A. S. Michael Meng, "Application Programming Interface Documentation: What Do Software Developers Want," Journal of Technical Writing and Communication, 2017, p. 36.

[2] J. E. Grayson, Python and Tkinter Programming, Manning, 2000.

[3] J. L. Demuth, J.K.Lazo and R.E. Morss, "Exploring Variations in People's Sources, Uses, and Perceptions of Weather Forecasts," Weather Climate and Society, 2011, vol.3, Issue 3, p. 177-192.

[4] J. K. Lazo, R.E. Morss, and J.L. Demuth, "300 Billion Served," American Meterological Society, 2009, p. 785-798.

[5] P. J. E. L. S, C. F. B. Hu Du, "Development of a REST API for obtaining site-specific historical and near-future weather data in EPW format," Building Simulation and Optimization , 2018.

[6] B. Smith, Beginning JSON, Apress, 2015.

[7] Open Weather, "Current weather data," Open Weather Map Available online:

[8] N. P. P. S. S. a. S. A. Ayushi Trivedi, "Speech to text and text to speech recognition systems-Areview," IOSR Journal of Computer Engineering (IOSR-JCE), vol. 20, no. 2, 2018, p. 40.

BIOGRAPHIES

Dr. Anirban Mandal obtained his B.Tech and M.Tech.from University of Calcutta He has been awarded Ph.D (Engg) from Jadavpur University, Kolkata. He has published more than 20 research articles in reputed journals.

Dr. Gargi Chakraborty obtained her Ph.D (Engg) from Jadavpur University, Kolkata. She has published more than 15 research articles in reputed journals.

Kunal Ghosh obtained his B.Tech degree in Electronics and Communication Engineering from Future Institute of Engineering and Management, Kolkata. He is working with different AI tools and Agends.