Weather Forcast System Using Python

Akshit Labh
Department of Networking and Communications
SRM Institute of Science and Technology
Chennai, Tamil Nadu.
al9710@srmist.edu.in

Medhir Aryan
Department of Networking and Communications
SRM Institute of Science and Technology
Chennai, Tamil Nadu.
ma2461@srmist.edu.in

Dr. R. Kayalvizhi
Department of Networking and Communications
SRM Institute of Science and Technology
Chennai, Tamil Nadu.
kayalvir@srmist.edu.in

I. *Abstract*—The paper "Weather Forecast System Using Python" aims to develop an intuitive and user-friendly application that provides real-time weather forecasts for any city in the world. Built using the Python programming language and featuring a graphical user interface (GUI) created with tkinter, this paper aims to allow users to easily access up-to-date weather information. The main functionality of this paper revolves around using weather data from a selected external service, such as 'wttr.in', making HTTP requests and displaying forecasts transparently and organized. The GUI, built using tkinter, provides a seamless and interactive user experience, allowing users to enter city names and get weather reports in just a few clicks. Key features of this paper include error handling to handle special situations, intuitive interface design, and the ability to display weather data for a variety of locations. Choosing tkinter for the GUI ensures that the application can run on different platforms without the need for platform-specific adjustments. The Weather Forecasting System Paper is not only a practical and informative tool, but also an educational journey for those interested in programming, data retrieval, and GUI development. It demonstrates the power of Python as a flexible programming language and demonstrates the potential to create applications that connect to external data sources, presenting that data in a user-friendly format. By making weather information more accessible and visually appealing, this paper helps improve users' daily lives, whether they are planning a trip, checking weather for outdoor activities or simply stay informed about local or global conditions. It provides a great example of how to leverage technology to simplify routine tasks and put valuable data at users' fingertips.

Keywords— Graphical User Interface (GUI) Development, Weather Data Retrieval, Tkinter Framework, User-Friendly Interface Design

II. Introduction

A. Motivation

Accurate weather forecasts are extremely important in our daily lives. Weather affects many aspects of our routines, including planning outdoor activities, travel arrangements, agricultural decisions and most importantly, preparing for severe weather . The driving force behind the paper "Weather Forecasting System Using Python" is to equip users with an accessible and user-friendly tool to get accurate weather forecasts. Given the flexibility of Python and the wealth of web APIs available, developing such a system is an attractive prospect. Additionally, this paper fits into the growing trend of personal data-based applications, providing users with the ability to retrieve weather information on demand.

B. Objective

The main goal of this paper includes creating a user-friendly weather forecasting system in Python, including the following goals:

a) Data Retrieval: Develop a mechanism to collect weather data from trusted sources such as weather APIs or websites, ensuring the data is up to date and accurate.

b) User Interface: Create an intuitive and responsive graphical user interface (GUI) using the tkinter library, allowing users to enter a location and get a weather forecast.

c) Data Presentation: Design a GUI to present weather information in a clear and informative manner, including current conditions, forecasts, and relevant additional data.

d) Error Handling: Implement robust error handling to effectively handle unexpected problems, such as network errors or invalid user input.

e) Customization: Allows users to customize the display, such as unit options (e.g., Celsius or Fahrenheit) and location options.

C. Problem Statement

This paper addresses are the need for a realistic, user-friendly and accessible weather forecasting system. Current solutions often involve navigating complex websites or using separate mobile apps, which can be tedious and time-consuming. By creating a Python-based system, users can quickly retrieve weather data for the desired location, customize options, and display information clearly and concisely. The system aims to streamline the process of collecting weather forecasts, serving both technical and non-technical users.

D. Challenges

Many challenges are expected during the development of this paper:

a) Data integration: Find and integrate accurate, up-to-date weather information from external sources, which may require interaction with weather APIs, data analytics, or web scraping.

b) User Experience: The beautiful and intuitive GUI design provides an exceptional user experience, ensuring that users can easily navigate and interact with the system.

c) Error Handling: Design a robust error handling mechanism to handle network errors or user input errors gracefully, providing clear feedback and instructions.

d) Data Presentation: Ensure that weather information is presented in an easy-to-understand, informative, and well-organized manner, allowing users to interpret and use the data quickly.

III. Literature Survey

Weather forecasting is an essential aspect of our daily lives, impacting a variety of activities, from travel planning to disaster prevention. As technology advances, the development of weather forecasting systems using Python libraries and GUIs like tkinter has gained momentum. This literature review explores existing research and papers in the field of weather prediction systems developed in Python, focusing on graphical user interfaces.

A. Python and Weather Forecasting

Python has become a popular language for weather forecasting applications thanks to its rich library for data analysis and visualization. Researchers and developers use libraries such as NumPy, pandas, and Matplotlib to process and present weather data. Python's simplicity and flexibility make it a valuable tool for collecting, analyzing, and displaying weather data.

B. Graphical User Interfaces in Python

Graphical user interfaces (GUIs) play a pivotal role in making weather forecast systems accessible to a wide audience. Python provides several libraries for GUI development, with tkinter being one of the most straightforward and commonly used. The choice of tkinter enables the creation of user-friendly interfaces for users to input their location and retrieve weather forecasts easily.

C. OpenWeatherMap and Weatherstack APIs

Many weathers forecast systems rely on third-party APIs like OpenWeatherMap and Weatherstack to access real-time weather data. These APIs provide a variety of information, including temperature, humidity, wind speed, and forecasts. Python developers use API requests to retrieve weather data, analyze responses, and present information to users.

D. Graphical representation of weather data

Effective visualization of weather data is critical for conveying information to users. Python's Matplotlib and Seaborn libraries allow the creation of various tables and graphs, such as temperature trends, precipitation, and wind direction. These visual representations make it easier to interpret and understand weather forecasts.

E. Real-time updates and warnings

Weather forecasting systems often incorporate real-time updates and warnings. Users can receive notifications about severe weather conditions, temperature changes or precipitation. Integrating alerting mechanisms with Python scripts and GUIs will improve the usefulness of these applications.

F. User-centered design

Research on weather forecasting systems emphasizes user-centered design. The user interface should be intuitive, responsive, and aesthetically pleasing. This design focus ensures users can easily access and understand weather information.

IV. Requirements

A. Requirement Analysis

Developing a weather forecasting system in Python with a Tkinter-based graphical user interface (GUI) requires a thorough understanding of the paper's hardware and software requirements. In this section, we will analyze these requirements to ensure successful system development and deployment.

B. Hardware Requirement

a) Processing Power: For basic application functionality, you should use a standard computer or laptop with a dual-core processor and a clock speed of 2.0 GHz or higher. The processing power required for this paper is not large and most modern systems can easily meet this requirement.

b) Memory (RAM): The application's memory requirements are relatively low. A minimum of 4 GB RAM is enough to run tkinter-based Python and GUI code without significant performance issues.

c) Storage Space: The app itself does not consume much storage space. A minimum of 20 MB of free disk space is sufficient to accommodate the Python script and related resources.

d) Display and Graphics: The GUI is built using tkinter, a standard library for creating GUIs in Python. Any monitor or display that supports standard resolution will work. No need for advanced graphics hardware.

e) Internet Connection: The application relies on HTTP requests to retrieve weather data from online sources. Therefore, a stable Internet connection is required to access the weather API and view updated weather information.

C. Software Requirements

a) Python: The paper is developed in Python, so the main software requirement is a Python interpreter. Code has been tested with Python 3.7 but should be compatible with newer versions. Make sure that Python is installed on the system.

b) Tkinter Library: tkinter, a standard Python library for GUI development, is used to create the GUI. Make sure tkinter is available and correctly installed with your Python distribution.

c) Operating System: This application is platform independent and will work on different operating systems including Windows, macOS, and Linux. Users must have a compatible operating system to run the application.

d) Request library: The request library is used to send HTTP requests to the Weather API. You need to install this library using pip: pip install requests

e) Weather API Key: To access weather data, you may need an API key from the weather service provider. You must register an API key from the appropriate provider (e.g. OpenWeatherMap, Weatherstack) and include it in your application to make API requests.

f) Code Editor/IDE: A code editor or integrated development environment (IDE) should be used to write and run Python scripts. Popular choices include Visual Studio Code, PyCharm, and IDLE (included in Python).

g) Dependencies: Make sure required dependencies are listed in the Python code. They may include libraries or packages in addition to the standard Python library, depending on the specific paper implementation.

h) Internet Browser: Users need a web browser to view the GUI and interact with the weather forecast system. The application can open the user's default web browser to display weather information from the website 'wttr.in'.

D. API Requirements

a) Weather API: Papers based on weather APIs (e.g. OpenWeatherMap, Weatherstack, or free APIs like "wttr.in").

V. Architecture And Design

In this section, we will discuss the architecture and design of "Python Weather Forecasting System" with Graphical User Interface (GUI) based on tkinter. The main goal of the system is to provide users with real-time weather information for a particular city and improve their experience through a user-friendly interface.

A. Data retrieval System

The data retrieval system is responsible for retrieving weather data from an external source (in this case "wttr.in") and processing that data to display it to the user. It can be divided into the following sub-components:

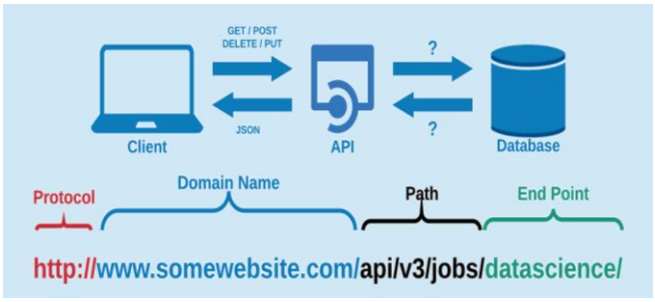

Fig. 1. Data Retrieval using requests API

a) User Input and Interaction: This child component handles user input. Users are prompted to enter the name of the city for which they want to receive weather forecasts. Users interact with the system through the GUI by entering a city name and clicking the "Get weather report" button.

b) Data retrieval Mechanism: After receiving the user input (city name), the system generates a URL with the city name and sends an HTTP GET request to "wttr.in". The "requests" library is used to support this. The system waits for a response from "wttr.in".

c) Error Handling: Data recovery systems include error handling mechanisms. It error-checks the HTTP response and handles exceptions gracefully. If an error occurs (for example due to a network problem or an invalid city name), the system displays an error message to the user.

d) Data processing: Once weather data is successfully retrieved, it will be processed to extract relevant information, such as temperature and weather conditions. This processed data is then made available for display by the GUI.

B. Graphical User Interface (GUI)

The GUI component is responsible for presenting

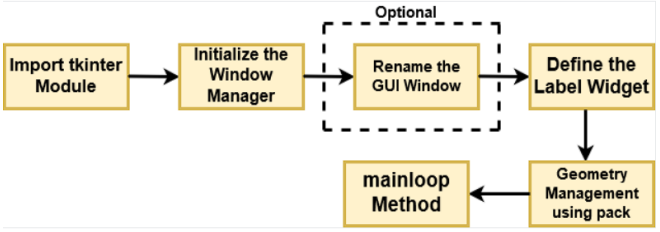

Fig. 2. tkinter GUI

weather data to the user in an intuitive and user-friendly manner. It improves user experience by providing an interactive and visually appealing interface. Main GUI components include:

a) Main Window: The GUI starts by creating the main tkinter window. The window serves as the main container for all GUI elements.

b) Labels: Informational labels are used to provide welcome messages and instructions to users. These labels convey the purpose of the app and guide users on how to use the app effectively.

c) Entry Field: The input field allows users to enter the name of the city for which they want to receive weather information. Users can enter a city name in this field, allowing the system to retrieve specific data about the desired location.

d) Button: The button labeled "Get Weather Report" serves as a trigger for the system to begin the process of retrieving weather data. When the user clicks this button, the data retrieval system is activated and the weather forecast is displayed on the GUI.

e) ScrolledText Widget: The ScrolledText widget provides a designated space to display the weather report. This utility is designed to handle large amounts of text, allowing detailed weather information to be presented without text wrapping. Users can easily scroll through the content to read the full report.

f) Styling and Color: The GUI is designed to be visually appealing and user-friendly. It uses colors to distinguish different elements, such as background color and text color. For example, a light gray background is used for GUI elements, while the welcome label is displayed in blue text. Weather forecast text is also colored "darker" to improve readability.

C. Sytem Flow

The system flow is a sequential representation of the operation of a weather forecasting system, from user interaction with the GUI to presentation of weather information. The following steps illustrate the system flow:

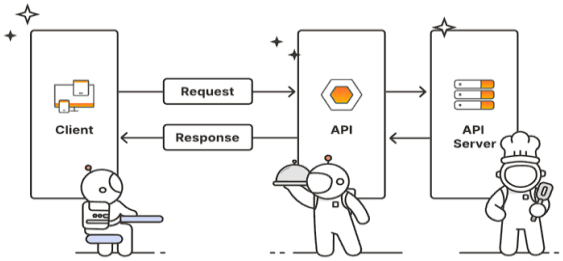

Fig. 3. Client Interaction with API

a) User Interaction: Users launch the application and are greeted with a welcome message and instructions on how to use the system. The user enters the name of the city for which he wants to receive weather forecast in the input field.

b) Data retrieval request: The user clicks the "Get Weather Report" button, which activates the data retrieval system. The system constructs the URL for "wttr.in" by combining the city name provided by the user. HTTP GET request sent to service 'wttr.in' using generated URL.

c) Data Retrieval and Processing: The system waits for a response from "wttr.in". When the response is successful, the system extracts weather data from the HTTP response, including information such as temperature and weather conditions. In case of an error (for example, a network problem or an invalid city name), the system handles the exception and displays an error message to the user.

d) Presentation on GUI: If weather data is successfully retrieved and processed, it will be displayed on the GUI. The ScrolledText widget is used to display weather forecasts, providing users with a convenient way to scroll through information. Weather data is styled with a "darker" text color to improve readability.

e) User Interaction Continues: Users can interact with the displayed weather information, scroll through text, and view reports at their convenience.

f) End of Interaction: The system continues to run in the main tkinter loop, waiting for further user interactions or until the user decides to close the application.

D. System Considerations

To ensure the weather forecast system works properly, several important considerations have to be taken into account:

a) User-Friendly Interface: The GUI is designed with the user in mind. It provides a welcome message and clear instructions to guide users through the process. The use of color and style will improve the overall user experience.

b) Error Handling: The system includes error handling mechanisms to effectively handle exceptions that may arise during data recovery. Users receive informative error messages in case of problems.

c) Data Source: The system retrieves weather data from "wttr.in", a famous service for weather information. This data source is publicly available and does not require an API key.

d) Data Extraction: The system extracts necessary weather information from the HTTP response, making it easy for users to access and understand.

e) Scalability and Extensibility: The architecture allows for future enhancements, such as adding additional features like support for multiple cities, extended forecasts, or unit conversions (e.g. , from Celsius to Fahrenheit).

f) Responsiveness: The tkinter-based GUI provides an interactive and responsive user interface, allowing users to efficiently access and interact with real-time weather data.

g) Aesthetics: The system's GUI components are designed with aesthetics in mind, using color and style to make the application visually appealing

VI. Implementation

Following is the Algorithm and Code for both implementation of this paper one with GUI and one without GUI.

A. Algorithm (Without GUI)

- Display a welcome message:
- Print a welcome message to the user, introducing the purpose of the program and explaining how the program works.
- User enters:
- Prompts the user to enter the name of the city for which they want to receive weather reports.
- Read user input:
- Read the city name entered by the user and store it in a variable, city_name.
- Define the Gen_report function:
- Create a function named Gen_report(C) that takes the city name as an argument.
- Inside the Gen_report function:
- Generate URL:
- Generate URL for service "wttr.in" by formatting city name in URL. This URL will be used to retrieve the weather forecast for the specified city.
- HTTP request:
- Attempt to send an HTTP GET request to a URL generated using the request.get() function.
- Check request success:
- Check if the request was successful by checking the HTTP response. If the request is successful, go to the next step.
- Retrieve weather data:
- Extract the content of the HTTP response (weather data) and store it in a variable, T.
- Error handling:
- If the query encounters an exception (for example, due to a network problem or an invalid city name), set T to

"An error occurred" to indicate that something went wrong.
- Show weather data:
- Print the text of T. This can be a weather forecast for the specified city or an "An error occurred" message, depending on the success of the HTTP request.
- Call function Gen_report:
- Call function Gen_report(city_name), passing user input (city name) as argument to the function. This starts the process of retrieving and displaying the weather forecast for the specified city.
- End of program:
- Program execution ends and the user receives a weather forecast or an error message, depending on the result of the HTTP request and data processing.

B. *Algorithm (With GUI)*

- Initialize constants:
- Set WTTR_URL to the base URL of the 'wttr.in' service.
- Define ERROR_MESSAGE for error handling.
- Define function gen_report:
- This function is called when the "Get Weather Report" button is clicked.
- Read the user's entry into the city_entry field, which represents the city name.
- Build service URL 'wttr.in' with city name. Use the request library to make an HTTP GET request to the constructed URL.
- Check the success of the HTTP request using reply.raise_for_status().
- If request succeeds:
- Get weather data from response.
- Remove previous content in result_text widget.
- Insert weather forecast into result_text widget.
- Apply "darker" text color tag to inserted text to improve readability.
- If an exception occurs (for example, a network problem or an invalid city name), remove the resulting text widget and display ERROR_MESSAGE.
- Create main tkinter window:
- Initialize main tkinter window and set its title to "Colorful Weather Forecast".
- Configure window background color.
- Create a GUI widget:
- Create a welcome label with a welcome message and blue text color.
- Wrap it in windows.
- Create an instructional label explaining how to use the application and place it in the window.
- Create an input field so the user can enter a city name and place it in the window.
- Create a "Get Weather Report" button with a blue background and white text, linked to the gen_report function.
- Pack it into the window.
- Create a ScrolledText widget with a custom width and height for displaying the weather report.
- Set the background color.
- Pack it into the window.Start main tkinter loop:
- Call window.mainloop() to start the main tkinter loop, keeping the GUI application running and responding to user interactions.
- End of program:
- The program continues to run in the main loop until the user closes the application.
- Users can enter a city, click a button, and get a weather forecast or error message based on the HTTP request results.

VII. Experiment Results

In this section we'll be exploring the output of the code and seeing the key points implemented in this paper.

A. *Output*

a) *Without GUI*

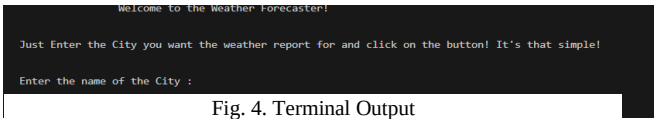

Fig. 4. Terminal Output

The above Fig. 4. shows the terminal prompt which is asking user to input the city name for which they want to know the weather forecast. If the user inputs invalid input the program will prompt error and ask the user to put a valid input.

The Fig. 5. depicts the weather forecast of the city entered by the user. In the above figure the output shows the forecast for Chennai. It shows the current temperature, wind speed, humidity etc. It also shows the same for different

Fig. 5. Terminal Output

durations for a day and predicted forecast for the next two days.

b) *With GUI*

Fig. 6. GUI Prompt

The Fig. 6. shows the GUI for the weather forecast system which was made using tkinter. The user has to enter the name of the city of which it needs the forecast. If the user inputs invalid input the program will prompt error and ask the user to put a valid input.

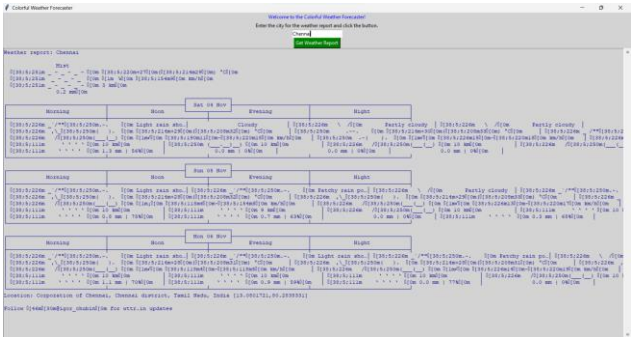

Fig. 7. The GUI Output

The Fig. 7. depicts the weather forecast of the city entered by the user in GUI made using tkinter. In the above figure the output shows the forecast for Chennai. It shows the current temperature, wind speed, humidity etc. It also shows the same for different durations for a day and predicted forecast for the next two days.

B. Key Topics Covered

a) API Integration: One of the fundamental components of this paper is integration with external weather data APIs. We chose the "wttr.in" service, which allows us to get real-time weather information for a specific city. In doing so, we gained practical experience in making HTTP requests, handling JSON data, and handling API responses. This aspect of the paper expanded our knowledge of how to work with external data sources.

b) GUI design with Tkinter: The weather forecast system GUI is developed using tkinter library. We designed a user-friendly interface that includes city name input fields, buttons to retrieve weather data, and a ScrolledText widget to display results. Learning how to create and layout GUI elements using tkinter is a significant achievement as it allows us to deliver a functional and visually appealing application to the end user.

c) Error Handling: Error handling is another essential aspect of this paper. We have implemented robust error handling mechanisms to gracefully handle exceptions that may arise during API requests. When issues such as network problems or invalid city names arise, our app displays a friendly "An error occurred" message instead of crashing or providing a confusing error message. This error management strategy improved the reliability of the weather forecast system.

d) User experience enhancement: User experience is the focus of this paper. Our goal was to create an application that is intuitive and easy to use. This involves providing clear instructions, feedback to users, and appropriate error handling. Additionally, we applied a "darker" color tag to the displayed weather information to improve readability. Through these innovations, we learned the importance of designing applications with the end user in mind.

e) Paper Organization: The Weather Forecasting System paper helped us appreciate the importance of paper organization and well-structured code. We modularized our code, used constants, and followed best practices for maintainability.

f) Real World Development: This paper provided us with practical experience in creating a real-world application that serves a useful purpose. This allows us to apply our programming knowledge to a tangible and functional paper.

VIII. CONCLUSION

Developing a weather forecasting system using Python and the tkinter library for GUI has been an interesting and educational journey. This paper provides a comprehensive learning experience in several areas including programming, API integration, GUI design, and improving user experience. We integrated "wttr.in" API for real-time weather data, HTTP request screening, JSON processing, and API response skills, designed user-friendly Tkinter based GUI for weather forecast with input fields, buttons and ScrolledText widgets, prioritized user experience with clear instructions, feedback, and readability enhancements in the application. We also Implemented robust error handling mechanisms for graceful handling of exceptions during API requests, emphasized paper organization and well-structured code to enhance maintainability. All over in this paper we gained practical experience in developing real-world applications, applying programming knowledge to a functional paper.

REFERENCES

[1] https://www.w3schools.com

[2] https://realpython.com

[3] https://www.geeksforgeeks.org/

[4] H. Seetha, V. Tiwari, K. R. Anugu, D. S. Makka, and D. R. Karnati, "A GUI Based Application for PDF Processing Tools Using Python & CustomTkinter," International Journal for Research in Applied Science and Engineering Technology, vol. 11, no. 1, pp. 1613–1618, Jan. 2023, doi: 10.22214/ijraset.2023.48848.

[5] K. S. Kaswan, J. S. Dhatterwal, and B. Balamurugan, "GUI Programming Using Tkinter," Python for Beginners, pp. 313–347, Feb. 2023, doi: 10.1201/9781003202035-12.

[6] "Intro to Python," Teaching Coding through Game Creation, pp. 109–120, 2018, doi: 10.5040/9798216022879.ch-007.

[7] K. S. Kaswan, J. S. Dhatterwal, and B. Balamurugan, "Python Exception Handling," Python for Beginners, pp. 349–388, Feb. 2023, doi: 10.1201/9781003202035-13.

[8] P. Joyce, "Python Programming," C and Python Applications, pp. 1–57, Nov. 2021, doi: 10.1007/978-1-4842-7774-4_1.

[9] "Tkinter GUI," Python by Example, pp. 110–123, May 2019, doi: 10.1017/9781108591942.019.

[10] "Python based API to post-process CFD data," Jan. 2023, doi: 10.2514/6.2023-1225.vid.