VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi,Karnataka -590 018.

A Internship on Report on

"Design and Implemention of Weather Forecasting using python"

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Engineering

in

Electronics and Communication Engineering

by

Jayanth K M 4JN21EC033

Under the Guidance of

Mrs Ujwala B S *professor M.Tech*

Assistant Professor,

Dept. of ECE,

JNNCE-577 204.

Department of Electronics and Communication Engineering

JNN College of Engineering, Shimoga - 577 204.

Internship Carried out in

G-tech JainX Education Limited

Bengaluru, Karnataka-560 103

May 2025

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi-590 018.

JNN College of Engineering

Department of Electronics and Communication Engineering

Shimoga-577 204.

CERTIFICATE

This is to certify that the Internship entitled **"Design and Implemention of Weather Forecasting using python"** is carried out at **"Wish 2 Skill LLP"** submitted by **Jayanth K M (4JN21EC033)**, the bonafide student of JNN College of Engineering, Shimoga in partial fulfillment for the award of "Bachelor of Engineering" in department of "Electronics and Communication Engineering" of the Visvesvaraya Technological University, Belagavi, during the year 2024-2025. It is certified that all thecorrections/suggestions indicated for internal assessment have been incorporated. Theinternship report has been approved as it satisfies the academic requirements in respect of internship prescribed for the said degree.

Signature of the Guide
Ms. Ujwala B S
Assistant Professor,
Dept. of ECE,
JNNCE, Shimoga.

Signature of the Coordinator
Dr. Shwetha H R
Associate Professor
Dept. of ECE,
JNNCE, Shimoga.

Signature of the HOD
Dr. S.V. Sathyanarayana
Professor & HoD
Dept. of ECE, JNNCE, Shimoga.

External Viva

Name of the examiner — Signature with date

1.

2.

WISH 2 SKILL LLP
No. 32, First Floor, 8th Cross, 1st B Main Road, Yelahanka
New Town, Bengaluru–560064
GST: 29AAEFW6583E1ZQ

web: www.wish2skill.com email: director@wish2skill.com

Internship Certificate

This is to certify that **Mr. Jayanth K M,** Student of VIII semester **BE – ECE** bearing **USN No. 4JN21EC033** at **Jawaharlal Nehru New College of Engineering,** has completed 15 Weeks of Internship on **Python Programming** at our Organization from **3rd February 2025 to 19th May 2025.**

Mr. Jayanth K M has collected the required data and exhibited good discipline and completed the assigned tasks as required during the period.

We wish good luck in all future endeavours.

Place: Bengaluru
Date: 19th May 2025

For WISH 2 SKILL LLP

Authorized Signatory

Figure 1: Certificate

ABSTRACT

This project presents a weather forecasting desktop application developed using Python, integrating the OpenWeatherMap API for real-time meteorological data. The application features a graphical user interface (GUI) built with Tkinter, enabling users to retrieve current weather conditions and 5-day forecasts for any city worldwide. It provides essential parameters such as temperature, humidity, and weather descriptions, while leveraging matplotlib to graphically display forecast trends. Real-time API responses are handled and visualized dynamically, making the tool both interactive and user-friendly. Robust error handling is incorporated to manage connectivity issues and invalid inputs. This project demonstrates practical application of API integration, data visualization, and GUI development in Python, serving as a foundational example for weather-based applications.

ACKNOWLEDGEMENTS

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible whose constant guidance and encouragement crowned the efforts with success.

I thank to**Dr. Y Vijaya Kumar, Principal, JNNCE, Shivamogga** for giving facilities to undertake internship work.

I thank to **Dr. Manjunatha P, Dean Academics, JNNCE, Shivamogga** for giving facilities to undertake internship work.

I would wish to express my gratitude to **Dr. S.V Sathyanarayana, Head of De partment,** Electronics and Communication Engineering for providing the good working environment and for his constant support and encouragement.

It gives me great pleasure in placing on record a deep sense of gratitude to our guide **Ms. Ujwala B S** , Associate Professor and to our internship coordinator, **Dr. Shwetha H R** , Assistant Professor , Department of Electronics and Communication Engineering for their expert guidance, initiative and encouragement that led me through the presentation.

I also thank **Wish 2 Skill LLP** and my guide **Mr. Vishal varma Sir** for providing me an opportunity to work in the company and complete the internship program

And lastly, I would hereby acknowledge and thank my parents who have been a source of inspiration and also instrumental in the successful completion of internship. Thank you All,

JAYANTH K M

4JN21EC033

Contents

List of Figures

Chapter 1

INTRODUCTION

This chapter provides a foundational overview of the internship undertaken as part of the academic curriculum. It introduces the context, objectives, and relevance of the project carried out during the Python Full Stack Development internship. The chapter begins with insights gained from the hands-on experience, followed by a discussion of the specific area of internship. It further elaborates on the problem being addressed, the objectives of the project, the methodology adopted, and the scope of the work. Finally, the structure of the report is outlined to guide the reader through the subsequent chapters.

1.1 Insights from Internship experience

- Learned to fetch weather data using online services (APIs) like OpenWeatherMap.
- Used Python to collect, display, and analyze weather details such as temperature, humidity, and wind speed.
- Worked with Python libraries like requests, pandas, and matplotlib to process and show weather data.
- Created charts and graphs to help understand weather patterns visually.
- Built small prediction models to try and forecast upcoming weather using basic machine learning.
- Designed simple apps with Python GUI tools (like Tkinter) to make the project user-friendly.

1.2 Area of Internship

The area of internship for the Weather Forecasting using Python project lies at the intersection of data science, software development, and environmental data analysis. During this internship, the focus was on collecting and processing real-time weather data using

APIs, developing Python-based applications for data analysis, and visualizing weather patterns through graphs and dashboards. The project also involved basic machine learning techniques for short-term forecasting, automation of data updates, and building user-friendly interfaces using GUI tools like Tkinter. This experience provided practical exposure to handling real-world datasets, implementing predictive models, and understanding how software tools can support decision-making in weather-related applications.

1.3 Problem Statement

The problem statement for the Weather Forecasting using Python project focuses on the need for an efficient and accessible way to predict weather conditions using real-time data. Accurate weather forecasting is essential for various sectors such as agriculture, transportation, event planning, and daily life activities. Traditional weather reports are often generalized and may not offer customized or location-specific insights. This project aims to address this gap by developing a Python-based application that fetches real-time weather data from public APIs, processes and visualizes the information, and provides short-term weather forecasts. The solution is designed to be user-friendly, reliable, and capable of helping users make informed decisions based on accurate weather predictions.

1.4 Scope of the Project

1. Fetch real-time weather data using public APIs like OpenWeatherMap.
2. Display current weather details such as temperature, humidity, wind speed, and pressure.
3. Show weather forecasts for upcoming days based on available data.
4. Visualize weather trends using graphs and charts for better understanding.
5. Build a simple user interface (GUI) for easy interaction with the application.
6. Use Python libraries for data processing and visualization (e.g., requests, pandas, matplotlib).

1.5 Organization of the report

The report for the Weather Forecasting using Python project is structured into five main sections. It begins with the Introduction, explaining the need for accurate and accessible weather forecasting. The Methodology covers the tools, technologies, and steps used, including Python programming and API integration. The Implementation and Results section describes how the system was developed and showcases the outputs and visualizations. Lastly, the Conclusion and Future Scope summarize the project's success and suggest improvements for expanding its features and accuracy.

Chapter 2

Theoretical background

2.1 Literature survey

1. **Fathi, Marzieh, et al. "Big data analytics in weather forecasting: A systematic review." Archives of Computational Methods in Engineering 29.2 (2022): 1247-1275.**[1] The paper titled "Big Data Analytics in Weather Forecasting: A Systematic Review" by Marzieh Fathi and colleagues, published in the Archives of Computational Methods in Engineering (2022), provides a comprehensive review of how big data analytics is applied in the field of weather forecasting. The authors systematically examine literature published from 2014 to 2020, categorizing the research into three main approaches: technique-based, technology-based, and hybrid models. The review highlights the use of advanced data analytics, machine learning algorithms, and big data technologies such as Hadoop and Spark in processing massive weather datasets. It compares these approaches in terms of their accuracy, execution time, scalability, and other quality metrics. Additionally, the paper analyzes the strengths and limitations of different algorithms and tools used in weather prediction and identifies current challenges in the domain. The study concludes by discussing future directions for research, emphasizing the need for more integrated and adaptive systems that can better handle the complexity and volume of meteorological data. This paper is a valuable resource for researchers and practitioners aiming to enhance the accuracy and efficiency of weather forecasting through big data techniques.

2. **Meenal, R., et al. "Weather forecasting for renewable energy system: a review." Archives of Computational Methods in Engineering 29.5 (2022): 2875-2891.**[2] The research paper titled "Weather Forecasting for Renewable Energy System: A Review" by R. Meenal and colleagues, published in the Archives of Computational Methods in Engineering (Vol. 29, Issue 5, 2022, pp. 2875–2891), presents a comprehensive and insightful overview of the role of weather forecasting

in the optimization and integration of renewable energy systems, particularly in the context of smart grids and sustainable energy infrastructure. With the increasing global reliance on renewable energy sources such as solar and wind, the variability and intermittency of these energy systems pose substantial challenges for grid stability and efficient energy management. Accurate weather forecasting, therefore, becomes crucial for improving the reliability and performance of renewable energy systems.

The paper begins by outlining the essential link between meteorological factors—such as solar irradiance, wind speed, temperature, and humidity—and the generation output of renewable systems. It highlights that even slight variations in these weather parameters can lead to significant fluctuations in energy production, affecting power quality and supply consistency. The authors emphasize that effective forecasting enables better planning, load balancing, storage optimization, and real-time control of renewable resources, thereby supporting the stability of the overall power grid.

The review categorizes weather forecasting techniques into three primary classes: physical models, statistical models, and artificial intelligence (AI)-based models. Physical models are based on complex mathematical equations derived from atmospheric physics and meteorological observations. These models simulate the physical processes of the atmosphere but require high computational power and are often time-intensive. Statistical models, on the other hand, rely on historical data to identify patterns and trends. They are generally simpler and faster but may not capture sudden or nonlinear changes in weather patterns effectively.

3. **Conti, Silvia. "Artificial intelligence for weather forecasting." Nature Reviews Electrical Engineering 1.1 (2024): 8-8..**[3] Conti discusses how AI models, particularly those utilizing machine learning and deep learning techniques, are increasingly being integrated into weather forecasting processes. These models excel at identifying complex patterns within vast datasets, enabling more precise predictions of weather phenomena. The article highlights the shift from traditional numerical weather prediction methods, which rely heavily on physical equations and require substantial computational resources, to AI-driven approaches that can process data more swiftly and with potentially greater accuracy.

   The piece also touches upon the development of hybrid models that combine the

strengths of both traditional and AI methodologies. Such integrations aim to leverage the physical understanding embedded in conventional models with the data-driven insights provided by AI, resulting in more robust forecasting tools. Conti notes that while AI offers promising improvements, challenges remain, particularly in ensuring the interpretability of AI models and their ability to handle rare or extreme weather events.

In conclusion, Conti's article provides a succinct yet informative perspective on the evolving landscape of weather forecasting, highlighting AI's growing influence and the collaborative efforts required to address existing challenges and fully realize the benefits of these technological advancements.

4. **Li, Shixuan, et al. "ClimateLLM: Efficient Weather Forecasting via Frequency-Aware Large Language Models." arXiv preprint arXiv:2502.11059 (2025).**[4] ClimateLLM employs a cross-temporal and cross-spatial collaborative modeling framework that combines Fourier-based frequency decomposition with LLMs to enhance spatial and temporal modeling capabilities. A key component of this framework is the Mixture-of-Experts (MoE) mechanism, which adaptively processes different frequency components, enabling efficient handling of both global signals and localized extreme events. Additionally, the model incorporates a dynamic prompting mechanism that allows LLMs to effectively integrate meteorological patterns across multiple scales. Extensive experiments conducted on real-world datasets demonstrate that ClimateLLM outperforms existing state-of-the-art approaches in terms of accuracy and efficiency. The model's ability to capture multi-scale frequencies and its scalable design make it a promising solution for global weather forecasting challenges. By bridging the gap between traditional numerical methods and modern AI techniques, ClimateLLM represents a significant advancement in the field of meteorology.

5. **Lombardi, Manuel, et al. "No more flying blind: Leveraging weather forecasting for clear-cut risk-based decisions." Transportation Research Interdisciplinary Perspectives 30 (2025): 101349.** [5] With the increasing deployment of VTOL vehicles, particularly in urban environments, there arises a significant need to address the operational risks associated with variable weather conditions. The authors identify a gap in existing safety approaches, noting that they often overlook the specific types of weather information necessary for effective

decision-making in VTOL operations. To bridge this gap, the paper proposes a four-phase decision support methodology that spans different timespans—from more than two weeks before up to two hours before a mission. This approach involves assessing dedicated feasibility indexes to guide decision-makers in evaluating the suitability of VTOL operations under varying weather conditions.

A case study is presented to demonstrate the practical implementation of this methodology within a decision support system. The system aims to assist VTOL decision-makers in selecting the most appropriate vehicle and operational strategies to ensure mission success across diverse contexts, including innovative air mobility solutions and industrial inspection practices. By emphasizing the importance of integrating detailed weather forecasts into the planning and execution phases of VTOL missions, the study contributes to enhancing the safety and reliability of these operations.

This research underscores the critical role of weather forecasting in the evolving landscape of urban air mobility and the necessity for tailored decision support systems that can adapt to the unique challenges posed by VTOL operations.

2.2 Summary

These five papers explore advancements in weather forecasting through big data, AI, and domain-specific applications. Fathi et al. review big data techniques enhancing forecast accuracy. Lombardi et al. propose a weather-informed decision framework for VTOL safety. Li et al. introduce ClimateLLM, a frequency-aware AI model. Conti discusses AI's transformative role in meteorology. Meenal et al. focus on weather forecasting's impact on optimizing renewable energy systems like solar and wind.

Chapter 3

System design and Methodology

3.1 Objectives

(a) Design and Development of efficient weather forecasting application using Python and the OpenWeatherMap API.

(b) To integrate external weather APIs (such as OpenWeatherMap) to provide accurate current weather conditions and short-term forecasts.

(c) To design a graphical user interface (GUI) that displays weather information clearly and allows easy user interaction.

(d) To implement error handling for invalid inputs and API response issues to ensure reliable and smooth user experience.

(e) To develop a user-friendly application that fetches real-time weather data for any specified city using Python.

3.2 Methodology

(a) Use Python to connect and fetch real-time weather data from public APIs (e.g., OpenWeatherMap) using the requests library.

(b) Visualize the weather data through graphs and charts using libraries such as matplotlib and seaborn for clear representation.

(c) Develop a simple graphical user interface (GUI) using Tkinter to allow users to input location and view weather updates easily.

(d) Implement basic forecasting models or algorithms to predict short-term weather based on historical and current data.

(e) Add error handling to manage issues like API downtime, invalid user input, or missing data.

(f) Automate data fetching and updating to provide continuous and up-to-date weather information.

3.3 Theoretical details

- Weather Data Sources: Weather forecasting relies on collecting data from meteorological sensors, satellites, and weather stations, often accessed via APIs like OpenWeatherMap.
- API (Application Programming Interface): APIs provide a way to request and receive weather data in structured formats such enabling programmatic access to real-time information.
- Data Processing: Raw weather data needs to be cleaned and organized using data structures (like tables in pandas) to prepare it for analysis and visualization.
- Data Visualization: Graphs and charts help in understanding weather patterns, trends, and anomalies by presenting data visually, using libraries like Matplotlib or Seaborn.
- Time Series Analysis: Weather data is typically time-dependent; analyzing sequences of data points over time is crucial for recognizing patterns and forecasting future conditions.
- Basic Forecasting Models: Simple statistical or machine learning methods (e.g., linear regression, moving averages) can be used to predict short-term weather based on historical data trends.
- Error Handling: Managing missing data, incorrect inputs, or API failures is essential to ensure the application runs smoothly and provides reliable forecasts.

Chapter 4

HARDWARE AND SOFTWARE DESCRIPTION

4.1 Software Description

- **Programming Language:** Python is chosen because it is beginner-friendly, easy to write and read, and has a vast collection of libraries that simplify working with data. Its rich ecosystem includes tools for making HTTP requests, handling performing data analysis, creating visualizations, and building graphical interfaces, which makes it ideal for developing weather forecasting applications efficiently.

Figure 4.1: Python

- **API Integration:** The project connects to weather service providers like OpenWeatherMap through their APIs. An API (Application Programming Interface) acts like a bridge that allows the Python program to request weather data over the internet. which is a lightweight, structured way of organizing information. This data includes current weather conditions, temperature, humidity, wind speed, and forecasts. Using the requests library and parses it to extract the needed weather details. This method ensures the application always has up-to-date information without manually entering or scraping data.

Figure 4.2: API

- **Data Handling:** After fetching raw weather data from the API, which is usually in format, the program parses this data to access specific weather details such as temperature, humidity, wind speed, and atmospheric pressure. Using libraries like pandas, the data is organized into tables or dataframes to make it easier to analyze and visualize. This step also includes cleaning the data, handling missing or inconsistent values, and preparing it for further processing like forecasting or display.

Figure 4.3: Data Handling

- **User Interface:** The project includes a simple graphical user interface (GUI) built using Tkinter, a Python library for creating desktop applications. This interface allows users to input their location (like city name or coordinates) and request the current weather and forecast information. The GUI displays the weather data clearly, often with labels, buttons, and charts, making it easy for users to interact with the application without needing to use command-line tools.

Figure 4.4: User Interface

- **Forecasting Module:** To predict future weather conditions, the project uses basic forecasting techniques, such as analyzing historical weather patterns or applying simple machine learning models (like linear regression or moving averages). This module takes past and current weather data as input and generates short-term forecasts, helping users anticipate upcoming weather changes.

Figure 4.5: Forecasting

- **Error Management:** The software includes mechanisms to handle potential issues gracefully. This means it can detect when the API service is down or unresponsive, handle cases where the user inputs an invalid location, and manage incomplete or corrupt data. By catching these errors, the application avoids crashing and instead provides useful feedback or retries, ensuring a smooth and reliable user experience.

Figure 4.6: Error Management

Chapter 5

Design and Implementation Details

5.1 Flowchart

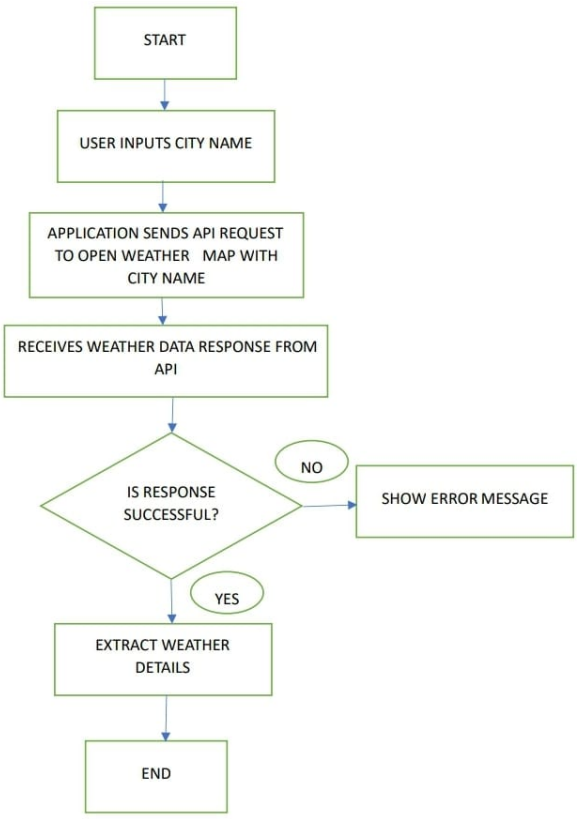

Figure 5.1: Flowchart

The Weather Forecasting Application using Python and the OpenWeatherMap API is designed to provide users with both current weather information and 5-day weather forecasts in a simple, interactive manner. The flow of the application begins with a user interface created using the Tkinter library in Python. As the user launches the application, a clean graphical interface is presented, allowing them to

input the name of a city. This is the primary input on which the entire system operates. The application has two main buttons: one labeled "Get Weather" and the other "Get Forecast." When the user types in a city name and clicks on "Get Weather," the application initiates a function that constructs a URL to call the OpenWeatherMap API's current weather endpoint. This URL includes the city name, an API key for authentication, and a query parameter to fetch results in metric units (Celsius). Once the request is sent, the system waits for a response. If the response is valid (status code 200), the JSON data is parsed to extract useful details such as the weather description, temperature, and humidity. These details are then displayed in a text box inside the GUI, providing real-time weather information in a human-readable format. If the response is not valid (e.g., if the city name is incorrect or the API key is missing), an error message is returned and displayed to the user.

On the other hand, if the user selects "Get Forecast," a different function is triggered, which sends a request to OpenWeatherMap's 5-day forecast endpoint. The response from this API includes weather predictions for every 3-hour interval. To simplify the display, the program samples this data to show one reading per day—usually every 8th item in the list. The program loops through the selected forecasts and extracts the temperature and weather description for each day. This information is shown in the text box like before. Additionally, the application collects the dates and corresponding temperature values and uses the matplotlib library to generate a line graph. This graph is a visual representation of how the temperature is expected to change over the next five days. The graph includes date labels on the X-axis and temperatures on the Y-axis, with points plotted and connected to show trends. This combination of text and graphics allows users to quickly interpret the forecast data and plan accordingly.

5.2 Working

The Weather Forecasting Application using Python and the OpenWeatherMap API works by integrating user input, real-time data retrieval, and graphical display in

a simple graphical interface. The user begins by entering the name of a city into the application's input field and selects either "Get Weather" or "Get Forecast." Based on this choice, the application sends a request to the corresponding OpenWeatherMap API endpoint—either for current weather or a 5-day forecast. The API processes the request and returns weather data in JSON format. The application parses this data to extract useful information such as temperature, humidity, and a description of weather conditions. For current weather, these details are displayed as text. For the 5-day forecast, the application samples data points to show one reading per day and then uses the matplotlib library to plot a line graph that visualizes temperature trends over time. This graph enhances understanding by providing a clear visual of upcoming weather changes. The program includes error handling to manage issues like invalid city names or failed API requests, ensuring robustness. Overall, the application demonstrates how Python can be used to combine APIs, GUI elements, and data visualization for practical, real-time weather forecasting.

Chapter 6

Results and Discussion

6.1 Results Obtained

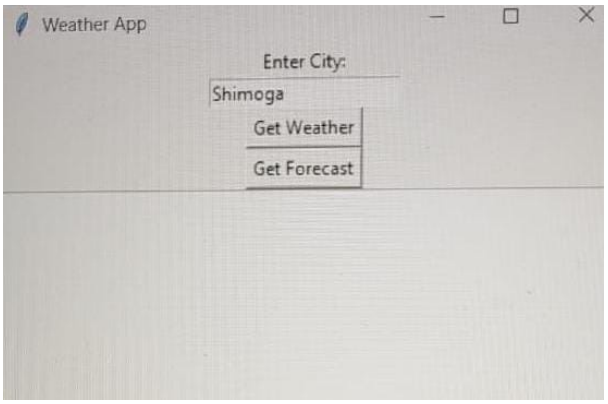

Figure 6.1: The image shows a Python-based Weather App GUI developed using Tkinter, running in PyCharm.

The Weather Forecasting Application developed using Python, Tkinter, and the OpenWeatherMap API provides accurate, real-time weather data and future forecasts in both textual and graphical formats. Upon entering a city name, the application fetches and displays current weather conditions such as temperature, humidity, and a brief description (e.g., “clear sky,” “light rain”). When the “Get Forecast” option is selected, the application retrieves 5-day forecast data, sampled at regular 24-hour intervals, and visually represents temperature trends using a line graph. This allows users to easily observe patterns such as warming trends, drops in temperature, or potential weather instability across the forecast period.

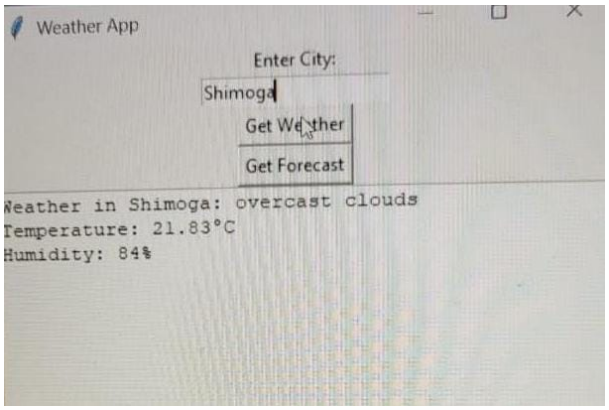

Figure 6.2: The image shows the Weather App displaying live weather for Shimoga.It confirms successful API integration and GUI output.

In testing, the application performed reliably for various global cities like Shimoga, Delhi, Mumbai, and New York, demonstrating its ability to access and interpret international data formats. The use of matplotlib for graphical output added significant value, enabling users to visualize fluctuations clearly, especially for planning purposes. The application also successfully handled edge cases such as invalid city names or network failures by displaying appropriate error messages.

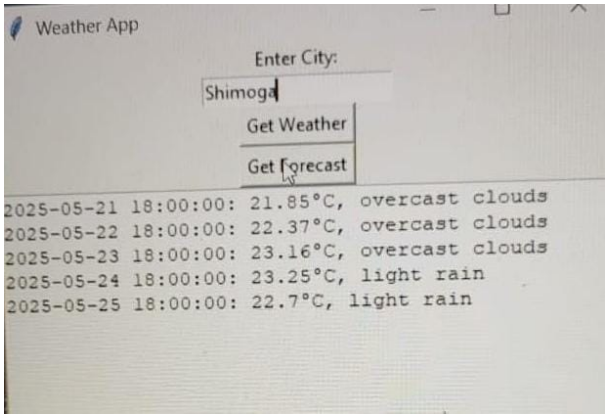

Figure 6.3: The image shows a weather app displaying a 5-day forecast for Shimoga with temperature and weather conditions.

The overall performance was smooth and responsive, and the use of Tkinter ensured a lightweight GUI. The integration of API data and visualization tools reflects how Python can be effectively used to build practical, real-time applications.

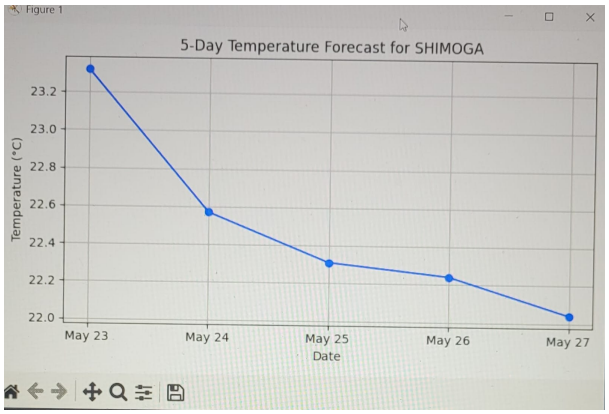

Figure 6.4: Graphical forecasting of Shivmoga

Graphical forecasting in the weather app visually represents future weather conditions—such as temperature trends, humidity, and weather types (e.g., rain or clouds)—using graphs like line charts or bar graphs.

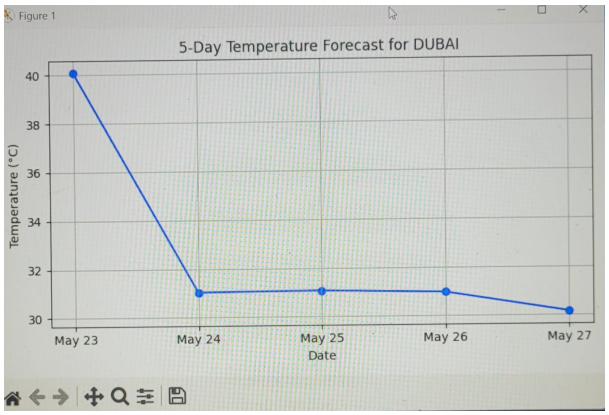

Figure 6.5: Graphical forecasting of Dubai

These visuals make it easier for users to quickly understand changes in weather over time, aiding in better planning. For example, a line graph could plot temperature on the y-axis and dates on the x-axis to show how temperatures will rise or fall over the next five days in a selected city. This enhances the app's usability by turning raw data into an easily digestible format.

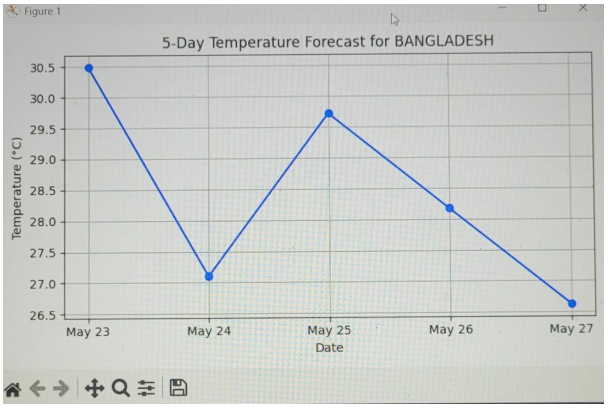

Figure 6.6: Graphical forecasting of Bangladesh

6.2 Applications in the Current Scenario

Outcome of the Internship **Daily Weather Updates:** Helps users check the current temperature, humidity, and weather conditions for any city.

Travel Planning:

Useful for planning short trips or commutes by knowing the weather ahead of time.

5-Day Forecast:

Offers a visual temperature trend, helping users decide what to wear or pack.

Educational Tool:

Great for beginners learning Python, APIs, and GUI development with Tkinter.

Real-time Data Access:

Retrieves live weather information using OpenWeatherMap API.

User-Friendly Interface:

Easy for anyone to use, even those with no technical background.

Weather Awareness:

Useful during changing weather seasons or in areas prone to sudden climate shifts.

Smart City Integration:

Can be enhanced or integrated into smart home or smart city systems.

6.3 Outcome of the Internship

Improved Understanding of Python Syntax and Semantics :

During the internship, I developed a strong grasp of Python syntax, including variables, data types, loops, and conditionals. I learned how to write clean, readable, and efficient Python code. I became proficient in using built-in functions and libraries. I explored Python's flexibility and readability compared to other programming languages. This foundational knowledge allowed me to quickly build simple applications and scripts. I also practiced debugging and learned how to interpret and resolve common errors. Overall, this helped solidify my understanding of core programming concepts in Python.

Hands-On Experience with Real-World Projects :

The internship provided opportunities to apply Python in practical, real-life scenarios. I worked on a variety of projects, such as automation scripts, data processing tools, and mini applications. This taught me how to break down complex problems and develop logical solutions. I learned the importance of planning and documenting code properly. I gained experience working with deadlines and requirements in a real work environment. These experiences helped bridge the gap between academic learning and practical application. I now feel more confident in using Python to develop real-world software solutions.

Introduction to Object-Oriented Programming (OOP) in Python :

I learned the principles of OOP, including classes, objects, inheritance, and encapsulation. I implemented Python programs using class-based designs to structure and organize code efficiently. This approach allowed for better code reusability and modular design. I also understood how OOP can simplify complex software development. Real-time project modules like hotel systems and login systems used OOP effectively. Applying OOP improved the scalability and maintainability of my code. This experience gave me a solid foundation in designing robust software systems.

Practical Exposure to File Handling and Databases :

I learned how to read from and write to text, CSV files using Python. This skill was crucial for managing data in real-world projects. I also worked with SQLite and MySQL databases using Python's sqlite3 and mysql.connector modules. I practiced creating, reading, updating, and deleting data (CRUD operations).

Chapter 7

Conclusions

The given Python code presents a functional desktop application for real-time weather monitoring and forecasting using the OpenWeatherMap API, Tkinter for GUI, and Matplotlib for data visualization. Users can input a city name to retrieve current weather conditions—including temperature, humidity, and a brief description—as well as a 5-day forecast visualized through a line graph. The application's strength lies in its simplicity, clarity, and user-friendly interface, making it suitable for both beginners and non-technical users.

The code is structured with functions that handle weather data fetching and handling potential API or network errors gracefully. The visualization of forecasted temperatures across days adds an intuitive dimension, helping users quickly grasp weather trends.

In summary, this weather app is a strong example of a lightweight, interactive application that integrates live data and visualization, ideal for educational purposes or as a foundational base for more advanced meteorological tools.

References

[1] M. Fathi, M. Haghi Kashani, S. M. Jameii, and E. Mahdipour, "Big data analytics in weather forecasting: A systematic review," *Archives of Computational Methods in Engineering*, vol. 29, no. 2, pp. 1247–1275, 2022.

[2] R. Meenal, D. Binu, K. Ramya, P. A. Michael, K. Vinoth Kumar, E. Rajasekaran, and B. Sangeetha, "Weather forecasting for renewable energy system: a review," *Archives of Computational Methods in Engineering*, vol. 29, no. 5, pp. 2875–2891, 2022.

[3] S. Conti, "Artificial intelligence for weather forecasting," *Nature Reviews Electrical Engineering*, vol. 1, no. 1, pp. 8–8, 2024.

[4] S. Li, W. Yang, P. Zhang, X. Xiao, D. Cao, Y. Qin, X. Zhang, Y. Zhao, and P. Bogdan, "Climatellm: Efficient weather forecasting via frequency-aware large language models," *arXiv preprint arXiv:2502.11059*, 2025.

[5] M. Lombardi, D. Sladek, F. Simone, and R. Patriarca, "No more flying blind: Leveraging weather forecasting for clear-cut risk-based decisions," *Transportation Research Interdisciplinary Perspectives*, vol. 30, p. 101349, 2025.