1. develop a weather prediction quadratic equation using waterfall model. explain five phases for these model.

Weather modeling involves creating systems that predict weather conditions based on various inputs such as temperature, humidity, and wind speed.we'll explain the five main phases of the Waterfall Model in the context of developing a weather prediction quadratic equation.

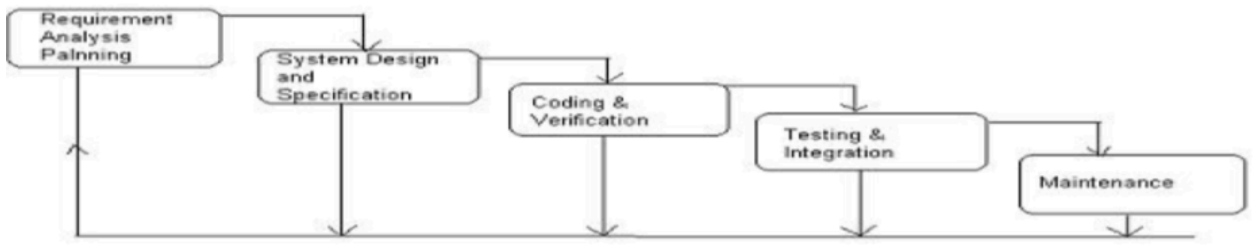

Requirement Analysis Palnning
System Design and Specification
Coding & Verification
Testing & Integration
Maintenance

Figure: The Waterfall Model

1. Requirements Gathering and Analysis :
Objective: Define the project's scope, understand the equation, and determine the necessary inputs and outputs.

- Input: Temperature, humidity, wind speed, time
- Output: Weather prediction (e.g., Sunny, Cloudy, Rainy, Stormy)
- User Requirements**:** Identify how users will use predictions (e.g., daily forecasts, real-time updates).
- Constraints and Assumptions: Define any assumptions (e.g., linear relationship of humidity) or constraints (e.g., limited data availability).

Key Questions:

1. What format should the input data (time, humidity, wind speed) be in?
2. What specific weather variable does WWW represent?
3. What is the time range for prediction (hourly, daily, or weekly)?
4. Are there any constraints on historical data access or use?

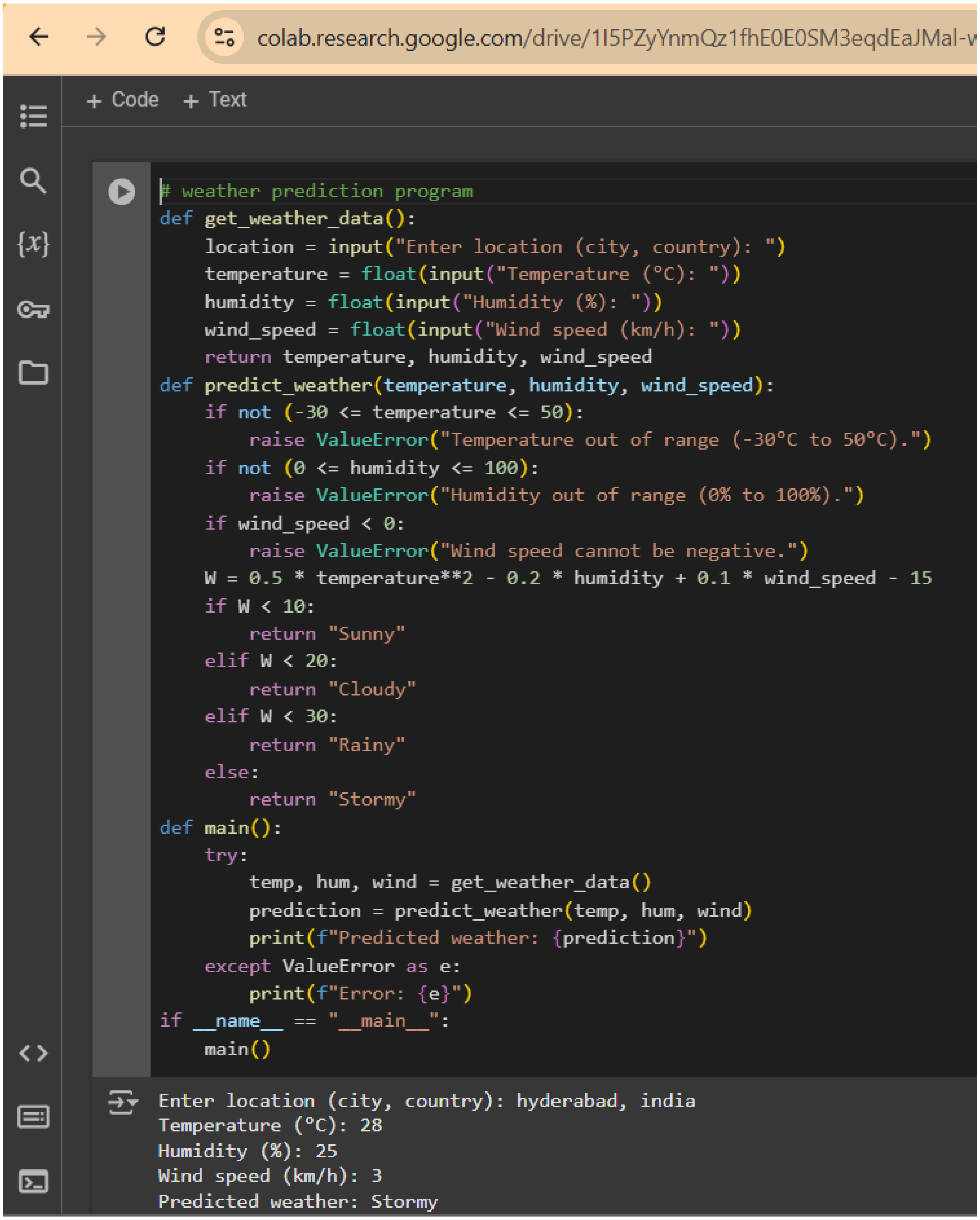
2. System Design:

- Create the architecture:
  - Input module: Reads data from the user or file.
  - Processing module: Calculates the weather index.
  - Output module: Displays predictions.
- Specify algorithms for calculations (e.g., w = 0.5 t^2 - 0.2 h + 0.1 w - 15).
- **Key questions :**

1. What tools and libraries will be used to implement the equation (e.g., Python NumPy)?
2. How will the system handle outliers or incomplete input data?
3. What user interface will be used (e.g., web app, desktop application)?
4. How will the output (WWW) be visualized for better interpretation?
5. What is the expected system workflow for input-processing-output?

3 . implementation :

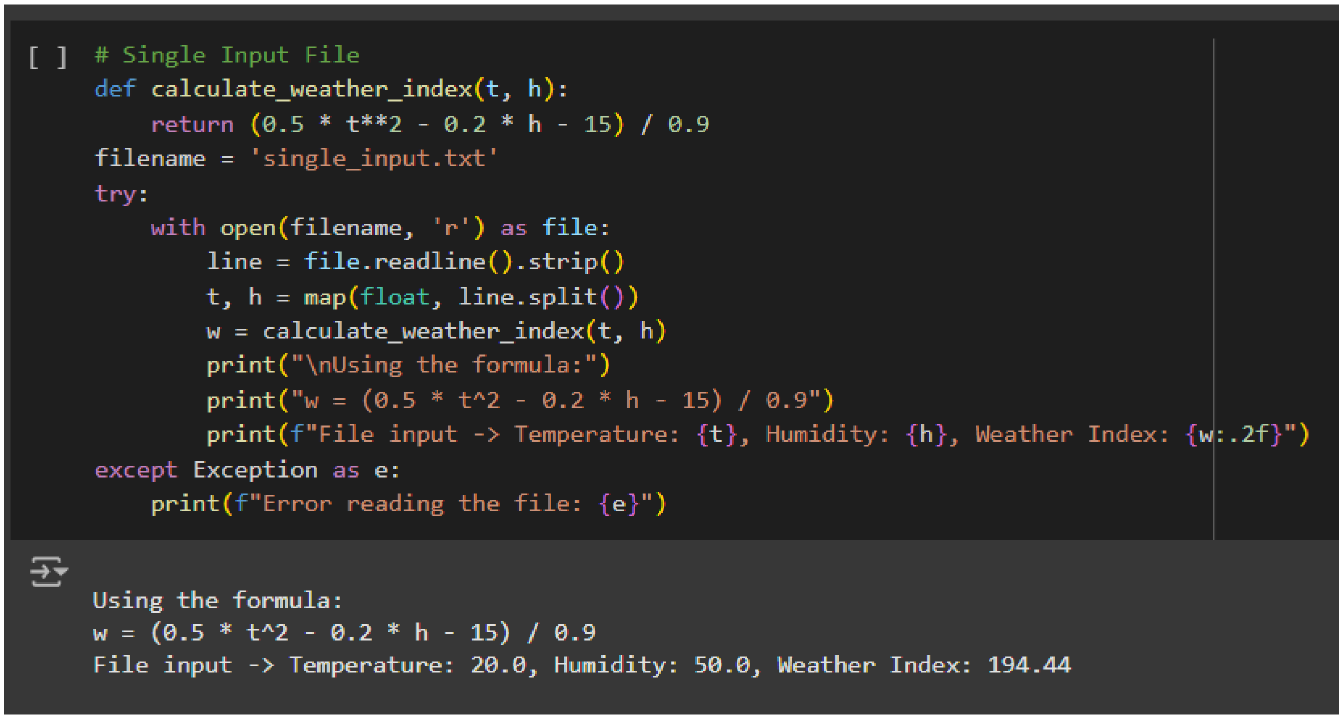
- **Code the modules :**



colab.research.google.com/drive/1I5PZyYnmQz1fhE0E0SM3eqdEaJMal-v

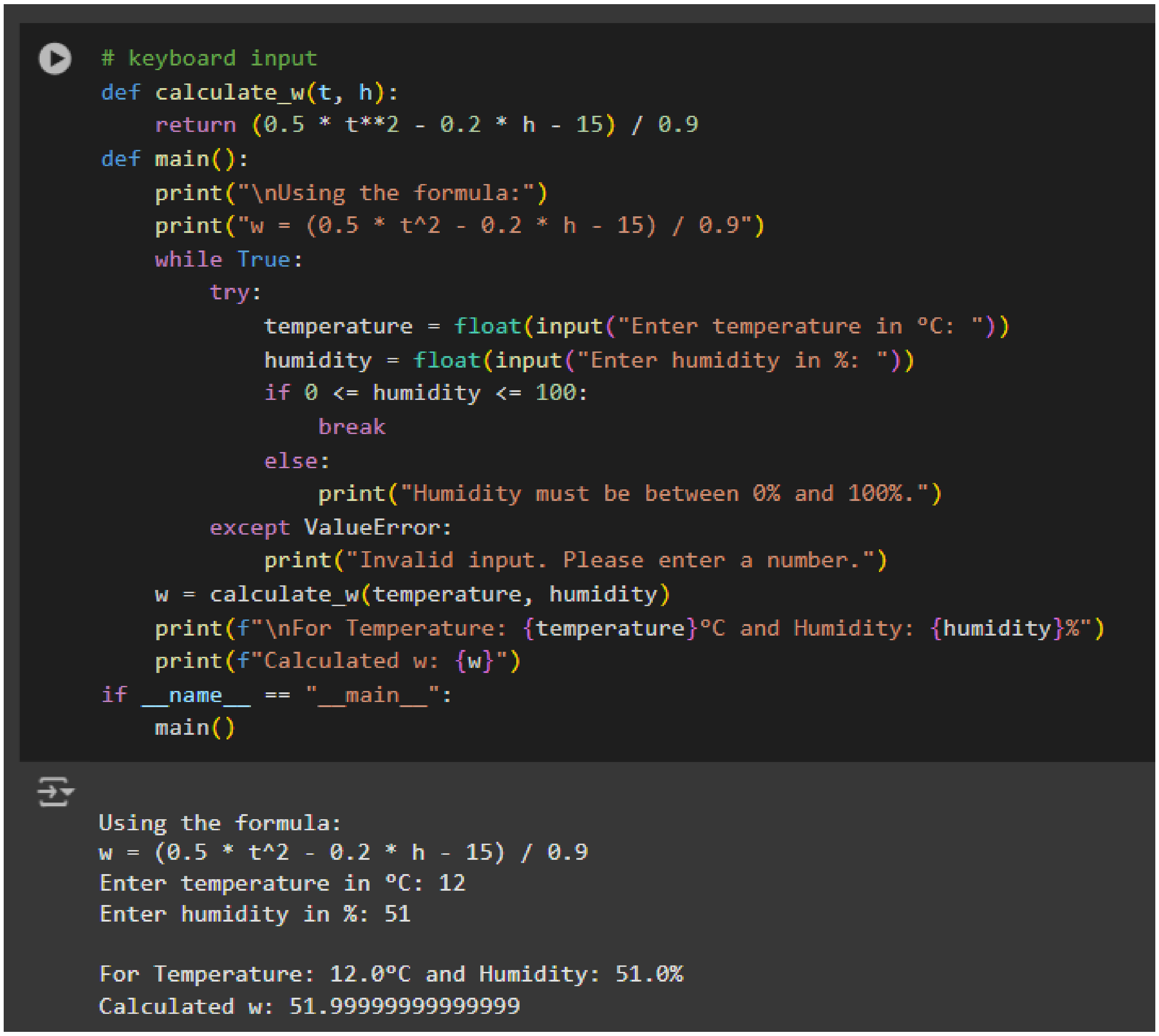
```python
# weather prediction program
def get_weather_data():
    location = input("Enter location (city, country): ")
    temperature = float(input("Temperature (°C): "))
    humidity = float(input("Humidity (%): "))
    wind_speed = float(input("Wind speed (km/h): "))
    return temperature, humidity, wind_speed
def predict_weather(temperature, humidity, wind_speed):
    if not (-30 <= temperature <= 50):
        raise ValueError("Temperature out of range (-30°C to 50°C).")
    if not (0 <= humidity <= 100):
        raise ValueError("Humidity out of range (0% to 100%).")
    if wind_speed < 0:
        raise ValueError("Wind speed cannot be negative.")
    W = 0.5 * temperature**2 - 0.2 * humidity + 0.1 * wind_speed - 15
    if W < 10:
        return "Sunny"
    elif W < 20:
        return "Cloudy"
    elif W < 30:
        return "Rainy"
    else:
        return "Stormy"
def main():
    try:
        temp, hum, wind = get_weather_data()
        prediction = predict_weather(temp, hum, wind)
        print(f"Predicted weather: {prediction}")
    except ValueError as e:
        print(f"Error: {e}")
if __name__ == "__main__":
    main()
```

```
Enter location (city, country): hyderabad, india
Temperature (°C): 28
Humidity (%): 25
Wind speed (km/h): 3
Predicted weather: Stormy
```



```
[ ] # Single Input File
    def calculate_weather_index(t, h):
        return (0.5 * t**2 - 0.2 * h - 15) / 0.9
    filename = 'single_input.txt'
    try:
        with open(filename, 'r') as file:
            line = file.readline().strip()
            t, h = map(float, line.split())
            w = calculate_weather_index(t, h)
            print("\nUsing the formula:")
            print("w = (0.5 * t^2 - 0.2 * h - 15) / 0.9")
            print(f"File input -> Temperature: {t}, Humidity: {h}, Weather Index: {w:.2f}")
    except Exception as e:
        print(f"Error reading the file: {e}")
```

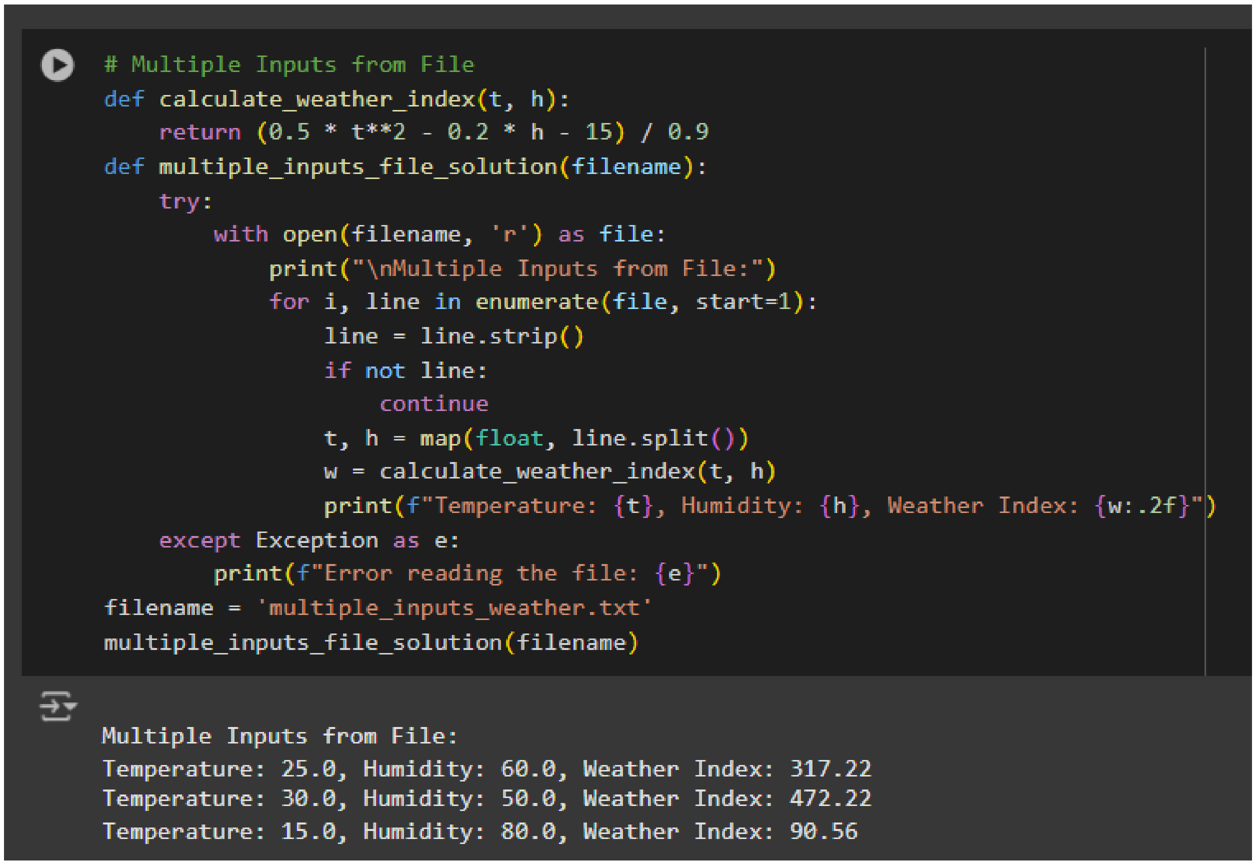
```
Using the formula:
w = (0.5 * t^2 - 0.2 * h - 15) / 0.9
File input -> Temperature: 20.0, Humidity: 50.0, Weather Index: 194.44
```


```python
# keyboard input
def calculate_w(t, h):
    return (0.5 * t**2 - 0.2 * h - 15) / 0.9
def main():
    print("\nUsing the formula:")
    print("w = (0.5 * t^2 - 0.2 * h - 15) / 0.9")
    while True:
        try:
            temperature = float(input("Enter temperature in °C: "))
            humidity = float(input("Enter humidity in %: "))
            if 0 <= humidity <= 100:
                break
            else:
                print("Humidity must be between 0% and 100%.")
        except ValueError:
            print("Invalid input. Please enter a number.")
    w = calculate_w(temperature, humidity)
    print(f"\nFor Temperature: {temperature}°C and Humidity: {humidity}%")
    print(f"Calculated w: {w}")
if __name__ == "__main__":
    main()
```

```

Using the formula:
w = (0.5 * t^2 - 0.2 * h - 15) / 0.9
Enter temperature in °C: 12
Enter humidity in %: 51

For Temperature: 12.0°C and Humidity: 51.0%
Calculated w: 51.99999999999999
```



```python
# Multiple Inputs from File
def calculate_weather_index(t, h):
    return (0.5 * t**2 - 0.2 * h - 15) / 0.9
def multiple_inputs_file_solution(filename):
    try:
        with open(filename, 'r') as file:
            print("\nMultiple Inputs from File:")
            for i, line in enumerate(file, start=1):
                line = line.strip()
                if not line:
                    continue
                t, h = map(float, line.split())
                w = calculate_weather_index(t, h)
                print(f"Temperature: {t}, Humidity: {h}, Weather Index: {w:.2f}")
    except Exception as e:
        print(f"Error reading the file: {e}")
filename = 'multiple_inputs_weather.txt'
multiple_inputs_file_solution(filename)
```

```
Multiple Inputs from File:
Temperature: 25.0, Humidity: 60.0, Weather Index: 317.22
Temperature: 30.0, Humidity: 50.0, Weather Index: 472.22
Temperature: 15.0, Humidity: 80.0, Weather Index: 90.56
```

Key Questions:

1. How will the quadratic equation be implemented in code?
2. How will historical weather data be imported and processed?
3. How will the system ensure the accuracy of the computation?

4 . Testing and verification :

Key Points:

1. **Accuracy Testing:** Use test data to verify W values are computed correctly.
2. **Stress Testing:** Test the system with large datasets to check scalability.
3. **Edge Case Testing:** Ensure the system handles extreme or unexpected values of t,h,w
4. **Comparison Testing:** Compare system predictions with actual historical weather data.
5. **Bug Identification:** Identify and resolve any coding or logical errors

Key questions :

1. How accurate is the system's prediction (WWW) compared to actual weather data?

2. What metrics (e.g., RMSE, MSE) will be used to evaluate prediction accuracy?

3. How will the system handle extreme input values (e.g., high wind speeds)?

5. Deployment and Maintenance :

Objective: Deploy the system for end-users and maintain its functionality over time.

Key Points:

1. System Deployment: Host the system on a platform (e.g., web server, mobile app) for user access.
2. Monitoring: Continuously monitor system performance and prediction accuracy.
3. Feedback Mechanism: Collect user feedback for system improvements.
4. Updates: Periodically update the system with new data or improved algorithms.
5. Documentation: Maintain detailed documentation for system usage and troubleshooting.

Key questions :

1. Where will the system be deployed (web app, desktop app, or cloud)?
2. How will the system ensure consistent performance after deployment?
3. How will users provide feedback to improve the system?
4. What maintenance schedule will be followed for updates and bug fixes?

Advantages :

- Clear structure.
- Well-defined milestones.
- Easy to manage progress.

Disadvantages :

- Inflexible to changes.
- Issues in earlier phases may propagate to later stages.

2 . Water moduling using iteration model :

The **Iteration Model** is a development process where the system is built step-by-step through repeated cycles, called iterations. Each iteration delivers an improved version of the system based on feedback and evolving requirements. Below is a simple explanation of how weather modeling can be developed using the iteration model.

Steps in Iterative Weather Modeling :

1. Planning the First Iteration

- Identify basic requirements for the weather model:
    - Collect input data such as temperature and humidity.
    - Use a simple formula to calculate a weather index.
    - Predict basic weather conditions like "Sunny" or "Rainy."
- Set goals for the first iteration to create a basic prototype.

2. Build and Test the Prototype

- **Build**:
    - Develop a simple system where users input temperature and humidity.
    - Implement a basic formula to calculate the weather index.
    - Output predictions like "Sunny" or "Rainy."
- **Test**:
    - Validate the prototype with sample data to ensure functionality.

3. Feedback and Refinement

- Collect feedback from users or testers:
    - Are the predictions accurate?
    - Is the system easy to use?
- Identify areas for improvement:
    - Add more variables, like wind speed.
    - Refine prediction categories (e.g., add "Cloudy" or "Stormy").

4. Plan the Next Iteration

- Based on feedback, plan enhancements for the next iteration:
    - Update the formula to include wind speed.
    - Improve the user interface for easier input and output.

    - Expand prediction categories for more detailed results.

5. Repeat the Process

- Build the improved version of the system.
- Test it with new scenarios and gather feedback.
- Continue refining and improving through multiple iterations until the final system meets all requirements.

Example of Iterative Development :

Iteration 1: Basic Prototype

- **Input**: Temperature and Humidity.
- **Formula**: Basic weather index calculation.
- **Output**: Predictions like "Sunny" or "Rainy."

Iteration 2: Enhanced Model

- **Input**: Add Wind Speed to Temperature and Humidity.
- **Formula**: Refined weather index using all three variables.
- **Output**: Expanded predictions such as "Sunny," "Cloudy," or "Rainy."

Iteration 3: Advanced Features

- **Enhancements**:
    - Integrate real-time weather data from APIs.
    - Add predictions like "Stormy" or "Snowy."
    - Implement a graphical interface for better usability.

Advantages of Using the Iteration Model :

1. **Flexibility**: Allows changes and improvements after each iteration.
2. **Feedback Integration**: Ensures the model evolves based on user needs.
3. **Risk Reduction**: Problems are identified and fixed early in the process.
4. **Progressive Development**: Delivers a functional product early, which gets better over time.

Conclusion :

Using the Iteration Model for weather modeling ensures that the system is developed step-by-step, improving accuracy and functionality with each cycle. This approach is ideal for projects where requirements may evolve, and constant feedback is necessary to refine the model.