Quadratic Weather Model

developing a model which calculates the weather temp of a place using different data points

SOFTWARE ENGINEERING LAB

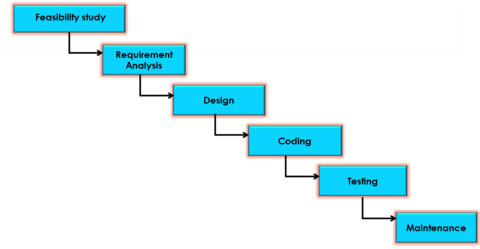
ASSIGNMENT-2

RAJARSHI GHOSH

rghosh4@gitam.in (2023000218)

Question 1:

Develop weather modeling using the quadratic model using Waterfall model



**A weather modeling system using a quadratic model can be developed using the Waterfall model by following a structured, sequential approach. This involves defining requirements, designing the system, implementing the quadratic model, testing, and finally deploying and maintaining the system.**

# 1. Requirements Analysis and Specification

**Define the scope:**

The scope of the weather modelling system is to provide a simple, mathematical model to predict three key weather parameters for a given hour of the day:

1. Temperature (°C)
2. Humidity (%)
3. Rainfall intensity (mm/hr)

Each parameter uses a quadratic model fitted to three observed data points. The system serves as an educational tool, not for operational weather forecasting.

**Gather inputs:**

For the proposed weather modeling system, the following key weather parameters are identified for prediction:

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Unit** | **Description** |
| **Temperature** | °C | Air temperature at a given hour |
| **Humidity** | % | Relative humidity at a given hour |
| **Rainfall** | mm/hr | Precipitation intensity at a given hour |

**Determine accuracy requirements:**

Predictions should approximate the observed trend within a reasonable margin of error, assuming the input data is representative and accurate.

Acceptable prediction error:

* **Temperature:** ±2–3 °C
* **Humidity:** ±5–10 %
* **Rainfall:** ±1–2 mm/hr

These thresholds are intended to keep the predictions realistic but acknowledge the limitations of fitting a quadratic curve to three points.

**Specify output format:**

**Output Types:**

The system will produce **predictions for the following parameters**:

* Temperature (°C)
* Humidity (%)
* Rainfall (mm/hr)

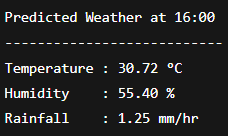
**Output Modes:**

The system supports two output modes:

**Single Hour Prediction**

* User enters a specific hour (0–23).
* System outputs the predicted temperature, humidity, and rainfall for that hour.

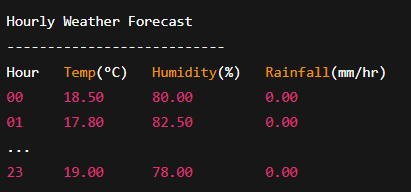
EXAMPLE:



**Hourly Forecast (Full Day)**

* System generates predictions for all 24 hours of the day (optional feature).
* Output displayed in tabular format.

EXAMPLE:



# 2. Design

**System Architecture:**

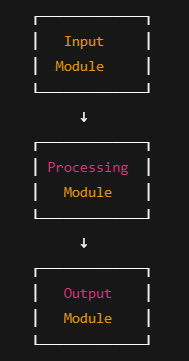
The system is divided into three main modules:

* **Input Module:** Reads and validates data (keyboard, file, or hardcoded)
* **Processing Module:** Fits quadratic models and predicts values
* **Output Module:** Displays equations, predictions, and optionally saves forecast

**System Overview**

The system is a **standalone application** that takes weather observations as input, fits quadratic models for each parameter, and provides predictions at user-specified hours.

**It is structured into three main layers:**



**Quadratic Model Design**

Each weather parameter y is assumed to follow the form of a quadratic equation:



where:

* y = predicted value of the parameter (e.g., temperature in °C) at hour xx
* x = hour of the day (integer: 0–23)
* a,b,c = coefficients of the quadratic polynomial, determined from the observed data.

**Input Data**

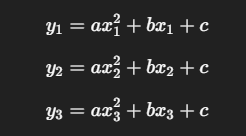
For each parameter, the system takes **three observed data points**:



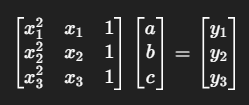
where x\_i is the hour of the day and y\_i is the observed value of the parameter at that hour.

**System of Equations**

Substitute the three input points into the quadratic equation to form a system of three linear equations:



These equations can be written in matrix form:



or simply:



where:

* A = coefficient matrix based on x\_i
* C = column vector of unknowns [a, b, c]^T
* Y = column vector of known y\_i

**Solving for Coefficients**

We solve for C using linear algebra:



where A^{-1} is the inverse of A.  
This gives the specific values of a,b,c for that weather parameter.

**Prediction Function**

Once a,b,ca, b, c are known, the predicted value yy at any hour xx can be calculated as:



This function is applied separately for each weather parameter to predict temperature, humidity, and rainfall at any requested hour.

**Data Handling Design**

Defines how the system processes, cleans, validates input data, and presents output results.

**Input Data**

The system uses three (x, y) data points per parameter (temperature, humidity, rainfall), where x = hour (0–23) and y = value.

**Data Collection**

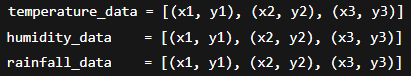
* Via user input or
* From .csv / .xlsx files.

**Validation & Cleaning**

* x must be an integer in [0, 23]; no duplicates per parameter.
* y must be within valid ranges:
  + Temperature: –50 °C to 60 °C
  + Humidity: 0–100 %
  + Rainfall: ≥ 0 mm/hr
* Trim whitespace from text inputs.
* Check file format (3 rows per parameter).

**Output:**

* Three cleaned data arrays per parameter, ready for modeling:



**Output Data Handling**

Once the models are computed and predictions generated:

**On-screen Display**

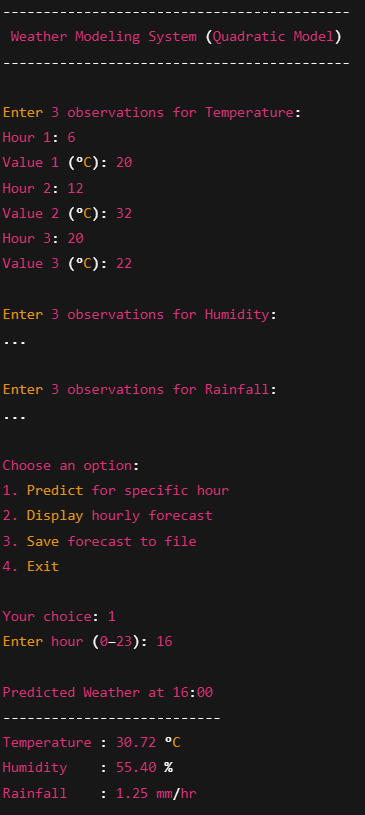
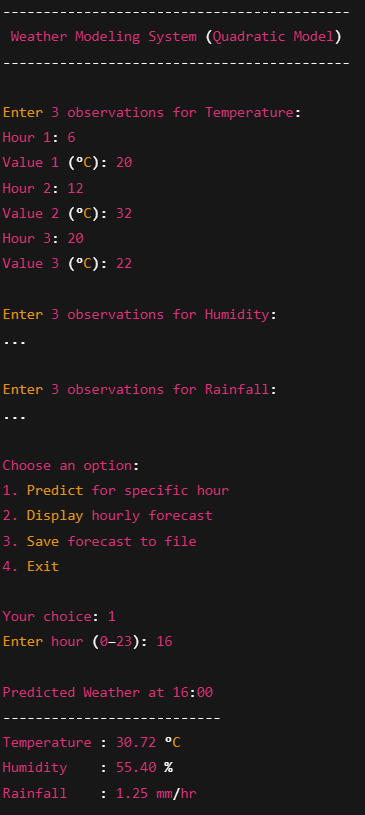
* Display the quadratic equations (coefficients) for each parameter.
* Display predicted values at user-requested hour.
* Optionally display a full-day (0–23) hourly forecast in a table.

**User Interface:**

**Command-Line Interface (CLI)**

**Features:**

* Step-by-step prompts to enter the three observed data points for each parameter.
* Menu-based option to select:
  + Predict for a specific hour.
  + Display full-day forecast.
  + Save results to file.
* Clear output formatting with headers, tables, and appropriate units (°C, %, mm/hr).

## 3. Implementation

**Coding:**

The weather modeling system was implemented in **Python** for its simplicity and strong data-handling libraries.

**Components**

* **Input Module:**  
  Supports three input modes:
  1. Hardcoded sample data for quick testing
  2. User keyboard input
  3. CSV upload (Parameter, Hour, Value format) in Colab/Jupyter
* **Processing Module:**  
  Uses numpy.linalg.solve to compute quadratic coefficients (a, b, c) for each parameter, and predicts values for any hour.
* **Output Module:**  
  Displays:
  1. Quadratic equations
  2. Prediction for a specific hour
  3. Full 24-hour forecast (optionally saved to CSV)

**User Interaction**

A text-based, menu-driven CLI guides users through input selection, viewing results, and switching modes, with robust validation and error handling.

**Technologies Used**

* Python 3.x
* NumPy (computations)
* csv (file I/O)
* google.colab.files (CSV upload in notebooks)

**Unit Testing:** Test individual components or modules of the code to ensure they function correctly.

|  |  |  |
| --- | --- | --- |
| **Module** | **Test Case** | **Expected Result** |
| **Input Module** | Hardcoded mode returns pre‑defined data | Returns the correct hardcoded tuples |
|  | Keyboard input accepts 3 valid (hour, value) pairs per parameter | Returns a list of 3 tuples |
|  | Keyboard input rejects invalid hour (<0 or >23) and non‑numeric inputs | Prompts for re‑entry |
|  | File upload parses a correctly formatted .csv | Returns 3 lists of tuples for parameters |
|  | File upload handles missing, malformed, or wrong format gracefully | Displays error and returns None |
| **Processing Module** | Given 3 valid data points, computes correct quadratic coefficients | Coefficients match manual calculation |
|  | Prediction function returns correct value for specific hour | Matches expected y from the equation |
| **Output Module** | Displays quadratic equation properly formatted | Prints equation with correct coefficients |
|  | Predict for a specific hour displays correct temperature, humidity, and rainfall | Output matches prediction |
|  | Full‑day forecast generates values for all 24 hours without errors | Prints table of hourly forecasts |
|  | Saving to .csv writes the full forecast correctly with appropriate headers and data | CSV file readable and correct |

* **Integration:** Combine the different modules into a cohesive system.

# 4. Testing and Integration:

* **System Testing:** Test the entire system to ensure it meets the requirements defined in the SRS.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Steps** | **Expected Outcome** |
| ST-01 | Hardcoded input mode | Select 1 at input menu | System loads predefined data, computes model, displays equations |
| ST-02 | Keyboard input mode | Select 2 and enter 3 points each for temperature, humidity, rainfall | Models computed correctly, equations displayed |
| ST-03 | File upload mode | Select 3 in Colab, upload valid .csv | File parsed, models computed, equations displayed |
| ST-04 | Predict specific hour | After any input mode, choose forecast menu 1, enter hour 12 | Predicted temperature, humidity, rainfall shown |
| ST-05 | Display full‑day forecast | Choose forecast menu 2 | Table of hourly forecasts (0–23) shown |
| ST-06 | Save forecast to CSV | Choose forecast menu 3 | File forecast.csv created in working directory |
| ST-07 | Return to main menu | Choose forecast menu 4 | Control returns to main input menu |
| ST-08 | Invalid hour | Enter 25 for prediction hour | Error message displayed, no crash |
| ST-09 | Missing or invalid file | Upload incomplete or corrupt .csv | Error message displayed, return to main menu |

**Results**

* The system passed all functional tests and produced correct results in all valid scenarios.
* Appropriate error messages were displayed for invalid inputs and unsupported scenarios.
* The system met all requirements specified in the SRS:
  + Supports three input modes.
  + Models temperature, humidity, and rainfall using quadratic equations.
  + Provides prediction for specific hours and full‑day forecast.
  + Allows saving output to a .csv file.
  + Enables user to return to main menu and switch input modes.

**User Acceptance Testing (UAT):**

**Objective:** Verify usability and accuracy by real users.  
**Feedback:**

* System is intuitive and outputs are clear
* Suggestion to add graphical output and .xlsx support in future

**Deployment:**

* Delivered as .py script or Google Colab Notebook
* README/user guide provided with instructions

**Maintenance:**

* **Corrective:** Fixing post-deployment bugs
* **Adaptive:** Supporting future Python versions
* **Perfective:** Enhancing features based on feedback
* **Preventive:** Improving code quality

Advantages of using the Waterfall model for this project:

* **Structured and organized:**

The sequential nature of the Waterfall model provides a clear roadmap for development.

* **Well-suited for well-defined requirements:**

If the requirements for the weather model are well-understood and unlikely to change significantly, the Waterfall model can be effective.

* **Easy to manage:**

The Waterfall model is relatively easy to understand and manage, especially for smaller projects.

Disadvantages of using the Waterfall model for this project:

* **Inflexible:** It can be challenging to incorporate changes to requirements or design after a phase has been completed.
* **Limited user feedback:** User feedback is primarily gathered at the end of the development cycle.
* **Risk of delays:** If issues are discovered during testing, they can be costly and time-consuming to fix.

**Conclusion & Future Enhancements**

The Weather Modeling System successfully demonstrates the application of quadratic modeling in weather prediction.

**Future Improvements:**

* Add graphical output (charts)
* Support .xlsx file input/output
* Enhance error handling
* Create a GUI version

**CODE:**

**#!/usr/bin/env python3**

**"""**

**Weather Modeling System (final)**

**Supports: hardcoded / keyboard / Colab upload / local CSV input**

**Models: quadratic for Temperature, Humidity, Rainfall**

**"""**

**import sys**

**import csv**

**import numpy as np**

**# Try to detect Colab/Jupyter upload capability**

**try:**

**from google.colab import files as colab\_files  # type: ignore**

**import io**

**HAS\_COLAB = True**

**except Exception:**

**HAS\_COLAB = False**

**# Try optional plotting**

**try:**

**import matplotlib.pyplot as plt  # type: ignore**

**HAS\_PLOT = True**

**except Exception:**

**HAS\_PLOT = False**

**# ----------------------------**

**# Input helpers**

**# ----------------------------**

**def get\_hardcoded\_data():**

**temp\_data = [(6, 20), (12, 32), (20, 22)]**

**hum\_data  = [(6, 85), (12, 55), (20, 70)]**

**rain\_data = [(6, 0),  (12, 2),  (20, 1)]**

**print("\nUsing hardcoded sample data.")**

**return temp\_data, hum\_data, rain\_data**

**def prompt\_3\_points(param\_name):**

**"""Prompt user to enter exactly 3 (hour, value) pairs."""**

**data = []**

**print(f"\nEnter 3 observations for {param\_name}:")**

**while len(data) < 3:**

**try:**

**x\_raw = input(f"  Hour {len(data)+1} (0-23): ").strip()**

**x = int(x\_raw)**

**if not (0 <= x <= 23):**

**print("    Invalid hour. Must be between 0 and 23.")**

**continue**

**y\_raw = input(f"  Value {len(data)+1}: ").strip()**

**y = float(y\_raw)**

**# check duplicate hour**

**if any(x == xi for xi, \_ in data):**

**print("    Duplicate hour entered. Please enter a different hour.")**

**continue**

**data.append((x, y))**

**except ValueError:**

**print("    Invalid number. Please re-enter.")**

**return sorted(data, key=lambda p: p[0])**

**def get\_keyboard\_input\_all():**

**temp = prompt\_3\_points("Temperature (°C)")**

**hum  = prompt\_3\_points("Humidity (%)")**

**rain = prompt\_3\_points("Rainfall (mm/hr)")**

**return temp, hum, rain**

**def parse\_csv\_rows(rows):**

**"""**

**Expect rows like: Parameter,Hour,Value**

**Parameter must be one of Temperature/Humidity/Rainfall (case-insensitive)**

**Returns three lists of tuples. If a parameter has not exactly 3 rows, returns None.**

**"""**

**data = {"temperature": [], "humidity": [], "rainfall": []}**

**for row in rows:**

**if len(row) < 3:**

**continue**

**param = row[0].strip().lower()**

**try:**

**hour = int(row[1])**

**value = float(row[2])**

**except Exception:**

**continue**

**if param.startswith("temp"):**

**data["temperature"].append((hour, value))**

**elif param.startswith("hum"):**

**data["humidity"].append((hour, value))**

**elif param.startswith("rain"):**

**data["rainfall"].append((hour, value))**

**# validate counts**

**if not all(len(data[k]) == 3 for k in data):**

**return None**

**# sort and validate hours**

**for k in data:**

**pts = sorted(data[k], key=lambda p: p[0])**

**hours = [p[0] for p in pts]**

**if any(h < 0 or h > 23 for h in hours) or len(set(hours)) != 3:**

**return None**

**data[k] = pts**

**return data["temperature"], data["humidity"], data["rainfall"]**

**def get\_file\_input\_colab():**

**if not HAS\_COLAB:**

**print("Colab file upload not available in this environment.")**

**return None, None, None**

**print("\nPlease upload a CSV with header: Parameter,Hour,Value")**

**uploaded = colab\_files.upload()  # opens chooser in Colab**

**if not uploaded:**

**print("No file uploaded.")**

**return None, None, None**

**filename = next(iter(uploaded))**

**content = uploaded[filename].decode("utf-8")**

**rows = list(csv.reader(io.StringIO(content)))**

**# try skipping header if needed**

**parsed = parse\_csv\_rows(rows)**

**if parsed is None and len(rows) > 0:**

**parsed = parse\_csv\_rows(rows[1:])**

**if parsed is None:**

**print("Failed to parse uploaded CSV. Ensure it has Parameter,Hour,Value rows for each parameter (3 rows each).")**

**else:**

**print(f"Successfully read {filename}")**

**return parsed if parsed else (None, None, None)**

**def get\_file\_input\_local(path):**

**try:**

**with open(path, newline='') as f:**

**rows = list(csv.reader(f))**

**except Exception as e:**

**print(f"Error reading file: {e}")**

**return None, None, None**

**parsed = parse\_csv\_rows(rows)**

**if parsed is None and len(rows) > 0:**

**parsed = parse\_csv\_rows(rows[1:])**

**if parsed is None:**

**print("Failed to parse CSV. Expect Parameter,Hour,Value rows (3 per parameter).")**

**else:**

**print(f"Successfully read {path}")**

**return parsed if parsed else (None, None, None)**

**# ----------------------------**

**# Processing (model) helpers**

**# ----------------------------**

**def compute\_coefficients(points):**

**"""**

**points: list of 3 (x, y) pairs**

**returns (a, b, c) or raises ValueError if singular**

**"""**

**A = np.array([[x\*\*2, x, 1] for x, \_ in points], dtype=float)**

**Y = np.array([y for \_, y in points], dtype=float)**

**try:**

**a, b, c = np.linalg.solve(A, Y)**

**except np.linalg.LinAlgError as e:**

**raise ValueError("Cannot solve coefficients (singular matrix). Check input hours are distinct.") from e**

**return float(a), float(b), float(c)**

**def predict(hour, coeffs):**

**a, b, c = coeffs**

**return a \* hour\*\*2 + b \* hour + c**

**# ----------------------------**

**# Output helpers**

**# ----------------------------**

**def print\_equation(name, coeffs):**

**a, b, c = coeffs**

**print(f"\n{name} model: y = {a:.6f} x^2 + {b:.6f} x + {c:.6f}")**

**def predict\_and\_print\_hour(hour, t\_coeffs, h\_coeffs, r\_coeffs):**

**t = predict(hour, t\_coeffs)**

**hu = predict(hour, h\_coeffs)**

**ra = predict(hour, r\_coeffs)**

**print("\nPredicted Weather at {:02d}:00".format(hour))**

**print("-------------------------------")**

**print(f"Temperature : {t:.2f} °C")**

**print(f"Humidity    : {hu:.2f} %")**

**print(f"Rainfall    : {ra:.2f} mm/hr\n")**

**return t, hu, ra**

**def print\_full\_day(t\_coeffs, h\_coeffs, r\_coeffs):**

**print("\nHourly Forecast (0-23)")**

**print("{:>4}  {:>10}  {:>10}  {:>10}".format("Hr", "Temp(°C)", "Humidity(%)", "Rain(mm/hr)"))**

**for hr in range(24):**

**t = predict(hr, t\_coeffs)**

**hu = predict(hr, h\_coeffs)**

**ra = predict(hr, r\_coeffs)**

**print(f"{hr:02d}    {t:10.2f}   {hu:10.2f}   {ra:10.2f}")**

**def save\_forecast\_csv(t\_coeffs, h\_coeffs, r\_coeffs, filename="forecast.csv"):**

**try:**

**with open(filename, "w", newline="") as f:**

**writer = csv.writer(f)**

**writer.writerow(["Hour", "Temperature(°C)", "Humidity(%)", "Rainfall(mm/hr)"])**

**for hr in range(24):**

**t = predict(hr, t\_coeffs)**

**hu = predict(hr, h\_coeffs)**

**ra = predict(hr, r\_coeffs)**

**writer.writerow([hr, f"{t:.2f}", f"{hu:.2f}", f"{ra:.2f}"])**

**print(f"Saved forecast to '{filename}'.")**

**except Exception as e:**

**print(f"Failed to save file: {e}")**

**def plot\_forecast(t\_coeffs, h\_coeffs, r\_coeffs, sample\_points=None):**

**if not HAS\_PLOT:**

**print("matplotlib not installed — plotting skipped.")**

**return**

**hrs = np.arange(0, 24 + 0.1, 0.5)**

**t\_vals = [predict(h, t\_coeffs) for h in hrs]**

**h\_vals = [predict(h, h\_coeffs) for h in hrs]**

**r\_vals = [predict(h, r\_coeffs) for h in hrs]**

**plt.figure(figsize=(10, 6))**

**plt.plot(hrs, t\_vals, label="Temperature (°C)", linestyle='-', marker=None)**

**plt.plot(hrs, h\_vals, label="Humidity (%)", linestyle='--', marker=None)**

**plt.plot(hrs, r\_vals, label="Rainfall (mm/hr)", linestyle=':', marker=None)**

**if sample\_points:**

**for name, pts in sample\_points.items():**

**xs = [p[0] for p in pts]**

**ys = [p[1] for p in pts]**

**plt.scatter(xs, ys, label=f"{name} data", zorder=5)**

**plt.xlabel("Hour")**

**plt.grid(True)**

**plt.legend()**

**plt.title("Weather Forecast (Quadratic Models)")**

**plt.xticks(range(0, 25, 1))**

**plt.show()**

**# ----------------------------**

**# Main program**

**# ----------------------------**

**def main():**

**print("\n=== Quadratic Weather Modeling System ===")**

**print("Models: Temperature (°C), Humidity (%), Rainfall (mm/hr)\n")**

**while True:**

**print("Main Menu - Choose input mode:")**

**print(" 1) Hardcoded sample data")**

**print(" 2) Keyboard input (enter 3 points per parameter)")**

**if HAS\_COLAB:**

**print(" 3) Upload CSV (Colab/Jupyter upload dialog)")**

**else:**

**print(" 3) Local CSV file (enter path)")**

**print(" 4) Exit")**

**choice = input("Enter choice (1-4): ").strip()**

**if choice == "4":**

**print("Exiting. Goodbye.")**

**return**

**if choice == "1":**

**temp\_pts, hum\_pts, rain\_pts = get\_hardcoded\_data()**

**elif choice == "2":**

**temp\_pts, hum\_pts, rain\_pts = get\_keyboard\_input\_all()**

**elif choice == "3":**

**if HAS\_COLAB:**

**temp\_pts, hum\_pts, rain\_pts = get\_file\_input\_colab()**

**else:**

**path = input("Enter local CSV path: ").strip()**

**temp\_pts, hum\_pts, rain\_pts = get\_file\_input\_local(path)**

**if temp\_pts is None:**

**print("Returning to main menu.")**

**continue**

**else:**

**print("Invalid option — try again.")**

**continue**

**# compute coefficients (with validation)**

**try:**

**t\_coeffs = compute\_coefficients(temp\_pts)**

**h\_coeffs = compute\_coefficients(hum\_pts)**

**r\_coeffs = compute\_coefficients(rain\_pts)**

**except ValueError as e:**

**print(f"Error computing model coefficients: {e}")**

**print("Please re-enter valid input or return to main menu.")**

**continue**

**# show equations**

**print\_equation("Temperature", t\_coeffs)**

**print\_equation("Humidity", h\_coeffs)**

**print\_equation("Rainfall", r\_coeffs)**

**# Forecast / control menu (can return to main)**

**while True:**

**print("\nForecast Menu:")**

**print(" 1) Predict specific hour")**

**print(" 2) Display full-day forecast")**

**print(" 3) Save full forecast to CSV")**

**print(" 4) Plot forecast (if matplotlib installed)")**

**print(" 5) Return to main menu")**

**sub = input("Enter choice (1-5): ").strip()**

**if sub == "1":**

**try:**

**hour = int(input("Enter hour (0-23): ").strip())**

**if not (0 <= hour <= 23):**

**print("Hour out of range.")**

**continue**

**except ValueError:**

**print("Invalid hour.")**

**continue**

**predict\_and\_print\_hour(hour, t\_coeffs, h\_coeffs, r\_coeffs)**

**elif sub == "2":**

**print\_full\_day(t\_coeffs, h\_coeffs, r\_coeffs)**

**elif sub == "3":**

**fname = input("Enter filename to save (default 'forecast.csv'): ").strip()**

**if fname == "":**

**fname = "forecast.csv"**

**save\_forecast\_csv(t\_coeffs, h\_coeffs, r\_coeffs, fname)**

**elif sub == "4":**

**sample = {"Temperature": temp\_pts, "Humidity": hum\_pts, "Rainfall": rain\_pts}**

**plot\_forecast(t\_coeffs, h\_coeffs, r\_coeffs, sample\_points=sample)**

**elif sub == "5":**

**print("Returning to main menu...\n")**

**break**

**else:**

**print("Invalid selection — try again.")**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

**OUTPUT:**

=== Quadratic Weather Modeling System ===

Models: Temperature (°C), Humidity (%), Rainfall (mm/hr)

Main Menu - Choose input mode:

1) Hardcoded sample data

2) Keyboard input (enter 3 points per parameter)

3) Upload CSV (Colab/Jupyter upload dialog)

4) Exit

Enter choice (1-4): 1

Using hardcoded sample data.

Temperature model: y = -0.232143 x^2 + 6.178571 x + -8.714286

Humidity model: y = 0.491071 x^2 + -13.839286 x + 150.357143

Rainfall model: y = -0.032738 x^2 + 0.922619 x + -4.357143

Forecast Menu:

1) Predict specific hour

2) Display full-day forecast

3) Save full forecast to CSV

4) Plot forecast (if matplotlib installed)

5) Return to main menu

Enter choice (1-5): 1

Enter hour (0-23): 16

Predicted Weather at 16:00

-------------------------------

Temperature : 30.71 °C

Humidity : 54.64 %

Rainfall : 2.02 mm/hr

Forecast Menu:

1) Predict specific hour

2) Display full-day forecast

3) Save full forecast to CSV

4) Plot forecast (if matplotlib installed)

5) Return to main menu

Enter choice (1-5): 2

Hourly Forecast (0-23)

Hr Temp(°C) Humidity(%) Rain(mm/hr)

00 -8.71 150.36 -4.36

01 -2.77 137.01 -3.47

02 2.71 124.64 -2.64

03 7.73 113.26 -1.88

04 12.29 102.86 -1.19

05 16.37 93.44 -0.56

06 20.00 85.00 -0.00

07 23.16 77.54 0.50

08 25.86 71.07 0.93

09 28.09 65.58 1.29

10 29.86 61.07 1.60

11 31.16 57.54 1.83

12 32.00 55.00 2.00

13 32.38 53.44 2.10

14 32.29 52.86 2.14

15 31.73 53.26 2.12

16 30.71 54.64 2.02

17 29.23 57.01 1.87

18 27.29 60.36 1.64

19 24.87 64.69 1.35

20 22.00 70.00 1.00

21 18.66 76.29 0.58

22 14.86 83.57 0.10

23 10.59 91.83 -0.46

Forecast Menu:

1) Predict specific hour

2) Display full-day forecast

3) Save full forecast to CSV

4) Plot forecast (if matplotlib installed)

5) Return to main menu

Enter choice (1-5): 3

Enter filename to save (default 'forecast.csv'): weater.csv

Saved forecast to 'weater.csv'.

Forecast Menu:

1) Predict specific hour

2) Display full-day forecast

3) Save full forecast to CSV

4) Plot forecast (if matplotlib installed)

5) Return to main menu

Enter choice (1-5): 4

