RAJARSHI GHOSH

rghosh4@gitam.in

Quadratic Weather Model

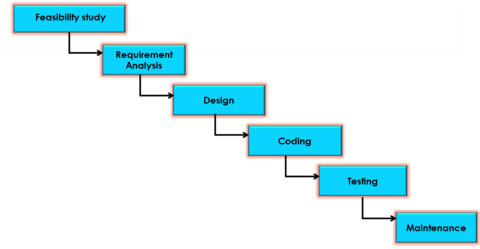
developing a model which calculates the weather temp of a place using different data points

SOFTWARE ENGINEERING LAB

ASSIGNMENT-2

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)

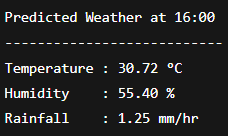
**Output Modes:**

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.

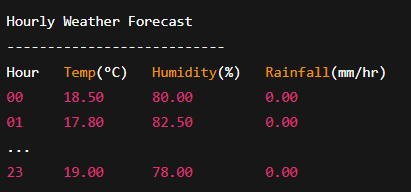
EXAMPLE:



**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:**

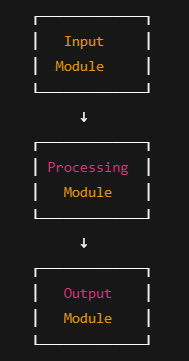
The system is divided into three main modules:

* **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**

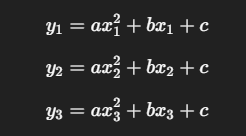
For each parameter, the system takes **three observed data points**:



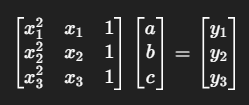
where x\_i is the hour of the day and y\_i is the observed value of the parameter at that hour.

**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**

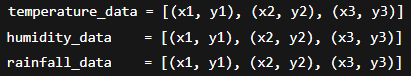
* Via user input or
* From .csv / .xlsx files.

**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**

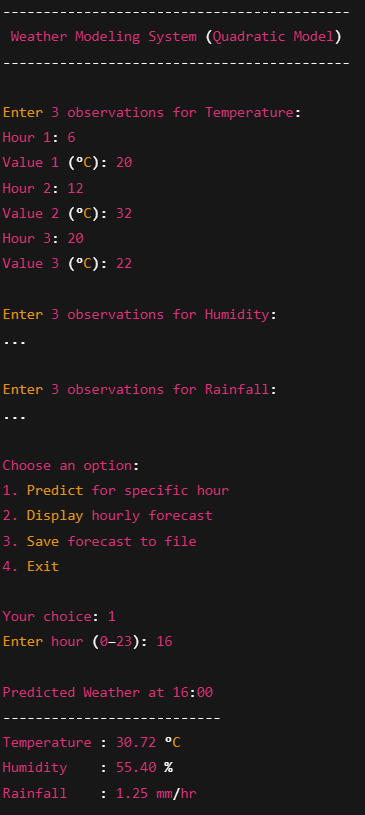
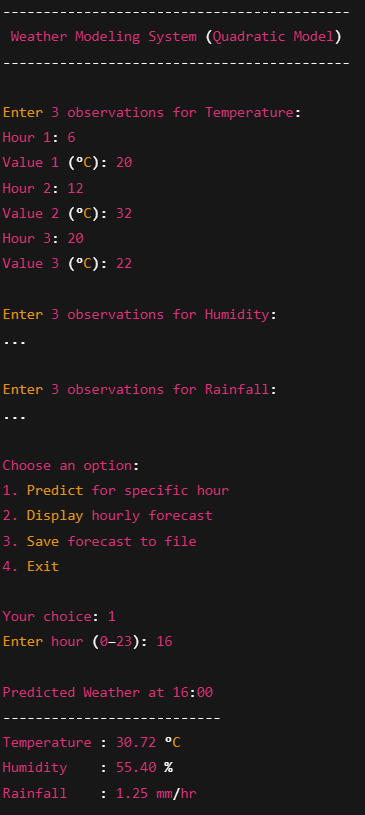
* Display the quadratic equations (coefficients) for each parameter.
* Display predicted values at user-requested hour.
* Optionally display a full-day (0–23) hourly forecast in a table.

**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