

WEATHER FORECASTING SYSTEM



A PROJECT REPORT

Submitted by

MOHAMED IBRAHIM F (2303811724321068)

in partial fulfillment of requirements for the award of the course

CGB1201 – JAVA PROGRAMMING

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by
AICTE, New Delhi)

SAMAYAPURAM – 621 112

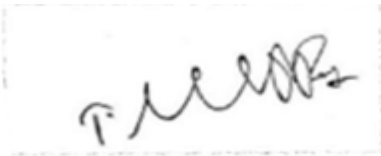
DECEMBER, 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on “**WEATHER FORECASTING SYSTEM**” is the bonafide work of **MOHAMED IBRAHIM F(2303811724321068)** who carried out the project work during the academic year 2024 - 2025 under my supervision.



Signature


Dr.T AVUDAIAPPAN M.E.,Ph.D.,
HEAD OF THE DEPARTMENT,
Department of Artificial Intelligence,
K. Ramakrishnan College of Technology,
Samayapuram, Trichy -621 112.




Signature

Mrs. S.GEETHA M.E.,
SUPERVISOR,
Department of Artificial Intelligence,
K. Ramakrishnan College of Technology,
Samayapuram, Trichy -621 112.

Submitted for the viva-voce examination held on 3.12.24



INTERNAL EXAMINER



EXTERNAL EXAMINER

DECLARATION

I declare that the project report on “**WEATHER FORECASTING SYSTEM**” is the result of original work done by me and best of my knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This project report is submitted on the partial fulfillment of the requirement of the award of the **CGB1201 – JAVA PROGRAMMING**.



SIGNATURE

MOHAMED IBRAHIM F

Place: Samayapuram

Date: 3/12/2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and indebtedness to our institution, **“K. Ramakrishnan College of Technology (Autonomous)”**, for providing us with the opportunity to do this project.

I extend our sincere acknowledgement and appreciation to the esteemed and honourable Chairman, **Dr. K. RAMAKRISHNAN, B.E.**, for having provided the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director, **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding our project and offering an adequate duration to complete it.

I would like to thank **Dr. N. VASUDEVAN, M.TECH., Ph.D.**, Principal, who gave the opportunity to frame the project to full satisfaction.

I thank **Dr.T.AVUDAIAPPAN, M.E.,Ph.D.**, Head of the Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for providing her encouragement in pursuing this project.

I wish to convey our profound and heartfelt gratitude to our esteemed project guide **Mrs.S.GEETHA M.E.**, Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for her incalculable suggestions, creativity, assistance and patience, which motivated us to carry out this project.

I render our sincere thanks to the Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards.

MISSION OF THE INSTITUTION

- Be a centre of excellence for technical education in emerging technologies by exceeding the needs of industry and society.
- Be an institute with world class research facilities.
- Be an institute nurturing talent and enhancing competency of students to transform them as all- round personalities respecting moral and ethical values.

VISION AND MISSION OF THE DEPARTMENT

To excel in education, innovation and research in Artificial Intelligence and Data Science to fulfill industrial demands and societal expectations.

Mission 1: To educate future engineers with solid fundamentals, continually improving teaching methods using modern tools.

Mission 2: To collaborate with industry and offer top-notch facilities in a conducive learning environment.

Mission 3: To foster skilled engineers and ethical innovation in AI and Data Science for global recognition and impactful research.

Mission 4: To tackle the societal challenge of producing capable professionals by instilling employability skills and human values.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

PEO 1: Compete on a global scale for a professional career in Artificial Intelligence and Data Science.

PEO 2: Provide industry-specific solutions for the society with effective communication and ethics.

PEO 3: Hone their professional skills through research and lifelong learning initiatives.

PROGRAM OUTCOMES

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- **PSO 1:** Capable of working on data-related methodologies and providing industry-focussed solutions.
- **PSO2:** Capable of analysing and providing a solution to a given real-world problem by designing an effective program.

ABSTRACT

The **Weather Forecasting System** is a comprehensive solution designed to automate the process of collecting, analyzing, and predicting weather conditions in real-time. It addresses challenges such as delayed weather updates, inaccurate predictions, and the lack of timely alerts for severe weather conditions, ensuring accurate forecasts and enhanced decision-making for users. Built using Java and utilizing AWT for the user interface, the system provides an intuitive and interactive platform for users to access weather updates, forecasts, and alerts seamlessly. The system operates through various modules, including weather data collection, processing and analysis, forecast generation, and alert notifications. It retrieves real-time weather data from APIs, processes it to identify patterns, generates accurate forecasts, and delivers timely alerts for extreme conditions such as storms or heavy rainfall. Core functionalities include providing city-specific forecasts, displaying current weather conditions, and notifying users about potential weather hazards. With its robust design, the Weather Forecasting System ensures reliability, scalability, and user-friendly interaction, making it a vital tool for informed planning and decision-making.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No.
	ABSTRACT	viii
1	INTRODUCTION	1
	1.1 INTRODUCTION	1
	1.2 OBJECTIVE	1
2	PROJECT METHODOLOGY	2
	2.1 PROPOSED WORK	2
	2.2 BLOCK DIAGRAM	2
3	JAVA PROGRAMMING CONCEPTS	3
	3.1 EVENT HANDLING	3
	3.2 APIS AND NETWORKING	4
4	MODULE DESCRIPTION	5
	4.1 WEATHER DATA COLLECTION	5
	4.2 DATA PROCESSING AND ANALYSIS	6
	4.3FORECAST GENERATION	7
	4.4 WEATHER ALERT	8
	4.5 USER INTERACTION	9
5	CONCLUSION	10
	REFERENCES	11
	APPENDICES	12
	Appendix A – Source code	12
	Appendix B – Screen shots	17

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The Weather Forecasting System is a Java-based application designed to provide real-time weather updates and forecasts, addressing the need for timely and accurate weather information. With features such as data collection, pattern analysis, weather forecasting, and alert notifications, the system ensures efficient processing and reliable results. The system incorporates a user-friendly GUI built using AWT, integrating core Java concepts to deliver an interactive and responsive experience. By leveraging APIs for real-time data retrieval and event-driven programming for dynamic interactions, the application offers a robust, scalable, and efficient solution. This tool is ideal for individuals and organizations seeking accurate and accessible weather forecasting to support better planning and preparedness.

1.2 OBJECTIVE

- **Accurate Predictions:** Provide precise, real-time weather forecasts.
- **Timely Alerts:** Deliver early warnings for severe weather conditions.
- **User Accessibility:** Develop an intuitive platform for general users and specialized industries.

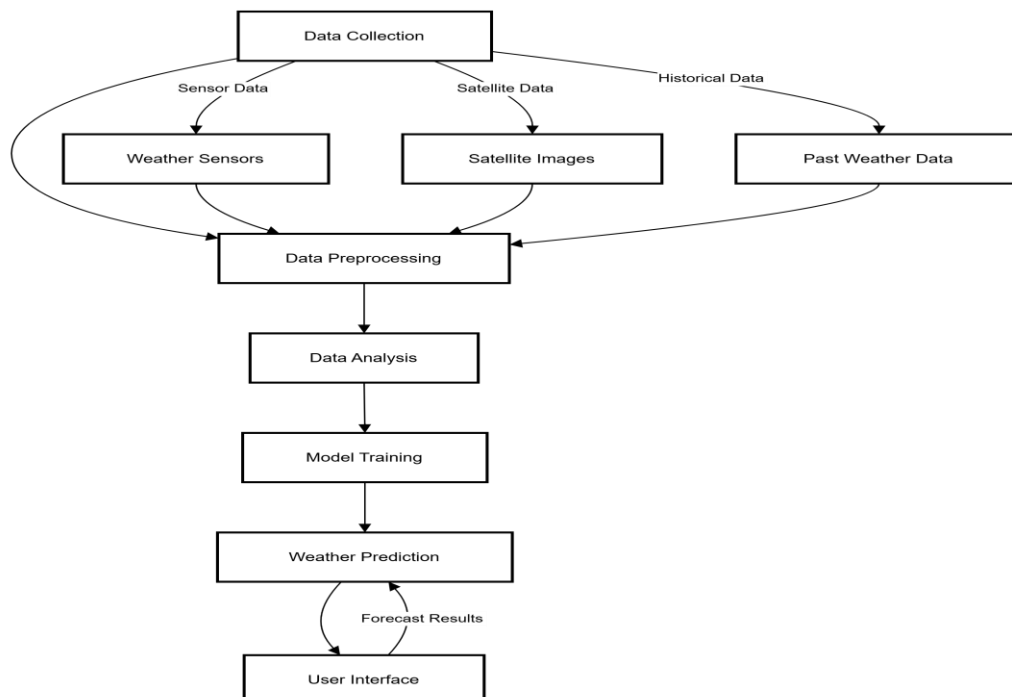
CHAPTER 2

PROJECT METHODOLOGY

2.1 PROPOSED WORK

The proposed **Weather Forecasting System** automates the process of retrieving, analyzing, and displaying real-time weather information using Java and AWT. It includes modules for weather data collection, processing, forecasting, alert generation, and user interaction. The system ensures accurate weather predictions, timely alerts for extreme conditions, and an interactive user interface for seamless operation. Designed for scalability and efficiency, it provides a robust platform for users to access reliable weather updates, make informed decisions, and plan effectively.

2.2 BLOCK DIAGRAM



CHAPTER 3

JAVA PROGRAMMING CONCEPTS

3.1. EVENT HANDLING

The *Weather Forecasting System* employs **event-driven programming** through the use of AWT (Abstract Window Toolkit) components like buttons, text fields, and labels. Specifically:

- **Components Used:**
 - **TextField:** Used for entering the city name.
 - **Button:** A "Get Forecast" button triggers the fetching of weather data.
 - **TextArea:** Displays the weather forecast results.
- **Implementation of Event Handling:**
 - The ActionListener interface is utilized to handle button click events.
 - When the "Get Forecast" button is clicked, an action is triggered, calling the `getWeatherForecast()` method.
 - This method fetches the weather data from the OpenWeatherMap API and processes it for display.
- **Benefits:**
 - Event handling allows the program to respond dynamically to user actions.
 - This approach ensures an interactive and user-friendly experience, where each action triggers a specific functionality in the system, such as fetching and displaying the weather forecast.

3.2. APIS AND NETWORKING

The *Weather Forecasting System* makes extensive use of APIs and networking concepts to fetch weather data from the OpenWeatherMap API.

- **Concept Overview:**
 - **HTTP Requests:** The program uses `HttpURLConnection` to send GET requests to the API's URL, including the city name and API key as parameters.
 - **Response Handling:** The system reads the response from the API using an `InputStream` and parses it for the required weather details such as temperature, humidity, and condition.
- **Implementation Details:**
 - A URL is constructed dynamically based on the user-provided city and the API key.
 - The program establishes a connection using `HttpURLConnection` and checks the response code to ensure successful data retrieval.
 - Data is read line-by-line using a `Scanner`, and the raw JSON response is processed for display.
- **Benefits:**
 - APIs enable the system to access real-time weather data without requiring a proprietary database or infrastructure.
 - Networking ensures the system can seamlessly connect to external data sources, enhancing its functionality and reliability.

CHAPTER 4

MODULE DESCRIPTION

4.1 WEATHER DATA COLLECTION

Purpose:

The Weather Data Collection Module is responsible for retrieving real-time weather information from reliable sources, such as APIs and weather sensors. It ensures accurate and up-to-date data is collected for further processing and analysis.

Description:

This module interacts with external APIs (e.g., OpenWeatherMap) to fetch weather details such as temperature, humidity, wind speed, and atmospheric conditions. The data is retrieved in a standardized format and prepared for processing. Error handling mechanisms ensure reliability by addressing connectivity or data retrieval issues. For example, the module sends an HTTP request with the city name and processes the response to extract weather parameters.

Key Functions:

1. **Fetch Real-Time Data:** Retrieve current weather data from APIs or connected sensors based on user inputs such as city name.
2. **Validate Responses:** Ensure the data received from external sources is complete and error-free.
3. **Store Data Temporarily:** Save the fetched data for further analysis and pattern recognition.

4.2 DATA PROCESSING AND ANALYSIS

Purpose:

The Data Processing and Analysis Module analyzes raw weather data to identify trends, patterns, and anomalies, providing a foundation for generating accurate weather forecasts.

Description:

This module processes the collected weather data to clean, organize, and structure it for meaningful analysis. It identifies weather trends using statistical techniques and detects anomalies like sudden temperature spikes. The module also facilitates pattern recognition to predict upcoming weather conditions. For instance, it computes averages, identifies significant weather changes, and correlates various weather parameters to ensure accurate results.

Key Functions:

1. **Data Cleaning:** Filter out incomplete or erroneous data from the collection phase.
2. **Pattern Analysis:** Recognize weather trends such as rising temperatures or recurring storms.
3. **Data Structuring:** Organize data into formats suitable for forecasting and user display.

4.3 FORECAST GENERATION

Purpose:

The Forecast Generation Module is responsible for generating accurate weather forecasts based on the processed and analyzed data.

Description:

This module leverages weather patterns and historical data to predict future weather conditions. It calculates expected temperatures, precipitation levels, and other key parameters using predictive algorithms. The forecasts are prepared in a user-readable format, providing valuable insights into upcoming weather scenarios. For example, based on rising humidity and cloud patterns, the module predicts the likelihood of rain.

Key Functions:

1. **Predict Weather Conditions:** Generate forecasts for temperature, humidity, and precipitation.
2. **Calculate Future Trends:** Estimate changes in weather conditions over a specified time period.
3. **Provide Detailed Results:** Present weather predictions in a format suitable for display to users.

4.4 WEATHER ALERT

Purpose:

The Weather Alert Module provides timely notifications for severe weather conditions, enabling users to take proactive measures for safety and planning.

Description:

This module monitors processed weather data for extreme conditions such as storms, heavy rainfall, or heatwaves. When thresholds are met, it generates alerts and notifies users through the system interface. For example, if high wind speeds or low temperatures are detected, the module issues appropriate warnings.

Key Functions:

1. **Monitor Weather Parameters:** Continuously evaluate conditions for potential hazards.
2. **Generate Alerts:** Create notifications for severe weather events based on pre-defined thresholds.
3. **Notify Users:** Display alerts directly on the user interface for immediate attention.

4.5 USER INTERACTION

Purpose:

The User Interaction Module ensures a seamless experience for users by providing an intuitive interface to input queries, view forecasts, and receive weather alerts.

Description:

This module manages the graphical user interface (GUI), built using AWT components, to enable smooth interactions. Users can input city names, view weather forecasts, and receive alerts through the interface. The module handles user actions, such as button clicks, and displays relevant data dynamically. For example, when a user enters a city name and clicks "Get Forecast," the module retrieves and displays the forecasted data.

Key Functions:

1. **User Input Handling:** Allow users to input city names or locations for weather information.
2. **Display Results:** Show forecasts, current weather conditions, and alerts in a clear and organized format.
3. **Dynamic Updates:** Update the interface based on real-time user actions and system responses.

CHAPTER 5

CONCLUSION

The Weather Forecasting System built using Java provides an efficient and accurate solution for obtaining real-time weather updates and forecasts. By leveraging Object-Oriented Programming (OOP) principles, the system ensures modularity, scalability, and ease of maintenance. The Data Collection module integrates with APIs to fetch real-time weather data, while the Data Processing and Pattern Analysis modules efficiently handle the parsing and analysis of weather patterns, enabling reliable forecasting.

The Forecast Display module presents the results in an intuitive and user-friendly interface, enhancing accessibility and usability for end-users. Additionally, the Alert Notification feature ensures timely dissemination of critical weather updates, promoting safety and preparedness.

In conclusion, the Weather Forecasting System effectively demonstrates the power of Java programming in creating practical, scalable, and user-centric applications that address real-world needs.

REFERENCES:

a) **GeeksforGeeks - Java Programming for Weather Applications**
[GeeksforGeeks Java](#)

- Provides comprehensive examples and tutorials on Java programming concepts like networking, JSON parsing, and GUI development.

b) **Head First Java by Kathy Sierra and Bert Bates**

- A beginner-friendly resource that lays a strong foundation in Java programming, covering OOP concepts and GUI development, essential for building the Weather Forecasting System.

c) **Java: The Complete Reference by Herbert Schildt**

- Offers detailed explanations of Java features, including AWT for GUI and HttpURLConnection for networking, both of which are extensively used in the Weather Forecasting System.

d) **Java Programming Tutorials - YouTube Channel by thenewboston**
[thenewboston Java Tutorials](#)

- Provides easy-to-follow video tutorials on Java programming, covering key concepts like GUI design, event handling, and API integration.

e) **Oracle Java Documentation**
[Oracle Java Documentation](#)

- The official Java API documentation, offering in-depth details about AWT, networking libraries, and best practices for Java programming.

APPENDICES

APPENDIX A – SOURCE CODE

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.Scanner;

public class WeatherForecast extends Frame {
    private TextField cityInput;
    private TextArea resultArea;
    private final String apiKey = "036939265f38b7943c9d6acc5a5db0b0"; // Your
    OpenWeatherMap API key

    public WeatherForecast() {
        setTitle("Weather Forecasting System");
        setSize(400, 300);
        setLayout(new FlowLayout());

        Label cityLabel = new Label("Enter City:");
        cityInput = new TextField(20);
        Button forecastButton = new Button("Get Forecast");
        resultArea = new TextArea(10, 30);

        add(cityLabel);
        add(cityInput);
```

```

add(forecastButton);
add(resultArea);

forecastButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        getWeatherForecast();
    }
});

addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
        dispose();
    }
});

setVisible(true);
}

private void getWeatherForecast() {
    String city = cityInput.getText();
    if (city.isEmpty()) {
        resultArea.setText("Please enter a city name.");
        return;
    }

    try {
        String response = fetchWeatherData(city);
        if (response != null) {
            parseAndDisplayWeatherData(response);
        }
    }
}

```

```

    } else {
        resultArea.setText("Error fetching weather data. Please try again.");
    }
} catch (IOException e) {
    resultArea.setText("Error: " + e.getMessage());
}
}

```

```

private String fetchWeatherData(String city) throws IOException {
    String urlString = "http://api.openweathermap.org/data/2.5/weather?q=" + city +
"&appid=" + apiKey + "&units=metric";
    URL url = new URL(urlString);
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
    conn.setRequestMethod("GET");

    int responseCode = conn.getResponseCode();
    if (responseCode == 200) {
        Scanner scanner = new Scanner(url.openStream());
        StringBuilder response = new StringBuilder();
        while (scanner.hasNext()) {
            response.append(scanner.nextLine());
        }
        scanner.close();
        return response.toString();
    } else {
        InputStream errorStream = conn.getErrorStream();
        if (errorStream != null) {
            Scanner scanner = new Scanner(errorStream);
            StringBuilder errorResponse = new StringBuilder();

```

```

        while (scanner.hasNext()) {
            errorResponse.append(scanner.nextLine());
        }
        scanner.close();

        System.out.println("Error Response: " + errorResponse.toString()); //
Debugging output
    }
    return null;
}
}

```

```

private void parseAndDisplayWeatherData(String response) {
    String temperature = "N/A";
    String humidity = "N/A";
    String condition = "N/A";

    // Simple JSON parsing without using libraries
    try {
        int tempIndex = response.indexOf("\"temp\":") + 7;
        int tempEndIndex = response.indexOf(",", tempIndex);
        temperature = response.substring(tempIndex, tempEndIndex) + "°C";

        int humidityIndex = response.indexOf("\"humidity\":") + 11;
        int humidityEndIndex = response.indexOf(",", humidityIndex);
        humidity = response.substring(humidityIndex, humidityEndIndex) + "%";

        int conditionIndex = response.indexOf("\"main\": \"") + 8;
        int conditionEndIndex = response.indexOf("\"", conditionIndex);
        condition = response.substring(conditionIndex, conditionEndIndex);
    }
}

```



```

    } catch (Exception e) {
        resultArea.setText("Error parsing weather data.");
        return;
    }

    String forecast = "Weather in " + cityInput.getText() + ":\n" +
        "Temperature: " + temperature + "\n" +
        "Humidity: " + humidity + "\n" +
        "Condition: " + condition;

    resultArea.setText(forecast);
}

public static void main(String[] args) {
    new WeatherForecast();
}
}

```

APPENDIX B - SCREENSHOTS

