# Weather Predictor

**A PROJECT REPORT**

*Submitted by*

***Sandeep Mehra-23BCS13199***
***Samarth Chawla-23BCS13228***
***Vikesh Kharti-23BCs13231***

***in partial fulfillment for the award of the degree of***

## BACHELOR OF ENGINEERING

**IN**

COMPUTER SCIENCE & ENGINEERING

**Chandigarh University**

Month, 2024

# TABLE OF CONTENTS

# CHAPTER 1. INTRODUCTION

## 1.1. Introduction to Project

Weather prediction has long been a subject of scientific study, and with the rise of digital technology and machine learning, more sophisticated techniques have been introduced. However, the fundamental approach of predicting weather based on historical data remains effective. This project aims to develop a weather prediction system using a simple but powerful method—Dynamic Programming (DP). The primary objective of the project is to forecast future weather conditions such as temperature and precipitation using past weather data. The system integrates with the Meteostat API, which provides reliable historical weather data for the past 10 years. Through dynamic programming, we seek to optimize the prediction process, ensuring both computational efficiency and accuracy.

Additionally, the project has been implemented as a web-based application, making it more accessible to users. The web interface allows users to input the number of days for which they wish to forecast weather, and the system will provide predictions for temperature and precipitation. This web-based model eliminates the need for users to run complex algorithms on their local machines, offering them a simple, intuitive experience.

## 1.2. Identification of Problem

Traditional weather forecasting methods, such as numerical weather prediction (NWP), often require significant computational resources and extensive models. These models, while accurate, may not be easily accessible to users who are looking for quick, short-term predictions. Additionally, machine learning-based approaches, though highly effective, often require large datasets for training and fine-tuning. These approaches also necessitate substantial computing power, especially for larger-scale predictions.

The problem that this project addresses is the need for a lightweight, accessible, and accurate method of weather prediction that can work with historical data and be easily deployed on the web. Furthermore, the solution should be optimized for short-term predictions, leveraging dynamic programming to ensure that the system runs efficiently, even with relatively large data sets.

# CHAPTER 2. BACKGROUND STUDY

## 2.1. Existing Solutions

Several approaches have been used to predict weather patterns, with the most common being numerical weather prediction models and machine learning techniques. These solutions, while highly accurate, often come with significant trade-offs in terms of computational complexity and the required amount of training data.

1. **Numerical Weather Prediction Models**: These models rely on solving complex mathematical equations that simulate atmospheric processes. They require high computational power and are typically used by weather organizations and governments for long-term forecasting. However, they are not ideal for personal use due to their complexity and resource requirements.
2. **Machine Learning-based Models**: With the advent of machine learning, more efficient forecasting methods have been developed. Techniques such as the ARIMA model (AutoRegressive Integrated Moving Average) and LSTM (Long Short-Term Memory) networks have been used to predict time-series data like temperature and precipitation. These models can learn patterns from large datasets and make accurate forecasts. However, machine learning requires substantial amounts of historical data, and tuning these models to ensure accurate predictions requires considerable expertise.

## 2.2. Problem Definition

The primary problem addressed by this project is the need for an efficient and straightforward weather prediction system that can predict weather conditions for the near future using historical data. The challenge lies in implementing a system that balances prediction accuracy, computational efficiency, and ease of use. Traditional methods like numerical weather prediction are too complex for such a task, while machine learning models may require extensive data preprocessing and significant computational resources.

## 2.3. Goals/Objectives

The objectives of this project are as follows:

- **Design a Weather Prediction System**: Using dynamic programming to forecast future weather conditions such as temperature and precipitation, based on historical data.
- **Use of Historical Weather Data**: Integrating the Meteostat API for accessing reliable and comprehensive historical weather data.
- **Web-Based Interface**: Developing an intuitive web interface for users to interact with the prediction system. The system will allow users to specify the number of days they want to predict and display the results.
- **Efficient Implementation**: The dynamic programming algorithm should optimize predictions and ensure that the system runs efficiently without requiring excessive computational resources.
- **User Experience**: The web interface will be designed to be user-friendly, providing users with clear and visually appealing predictions.

# CHAPTER 3. DESIGN FLOW/PROCESS

## 3.1. Evaluation & Selection of Specifications/Features

The core feature of the weather prediction system is its ability to predict temperature and precipitation using historical data. Key specifications considered for the project include:

- **Accuracy**: The prediction model should provide reasonable accuracy for short-term weather forecasts.
- **Efficiency**: The dynamic programming model must be optimized for performance, allowing it to run efficiently even with larger datasets.
- **Web Interface**: The application should have a smooth and intuitive user interface that allows users to input parameters and view predictions easily.

## 3.2. Analysis of Features and Finalization Subject to Constraints

After evaluating different algorithms, dynamic programming was selected for this project because of its optimization capabilities. The primary challenge was to ensure that the system would work with the large amount of historical data provided by the Meteostat API. This involved selecting a prediction model that was simple, yet capable of producing accurate results. Given the available data, dynamic programming was chosen as it can efficiently compute predictions while maintaining a manageable level of computational complexity.

## 3.3. Design Flow

The overall design flow for the project is as follows:

1. **Data Collection**: The first step is collecting the historical weather data from the Meteostat API. This data is stored in a CSV format.
2. **Data Processing**: The collected data is then parsed and processed into a format that can be used for prediction. This involves extracting necessary attributes such as temperature and precipitation for each day.
3. **Dynamic Programming Algorithm**: The next step involves applying the dynamic programming technique to make predictions based on the historical data. The algorithm processes the data and forecasts future weather conditions.
4. **Web Interface**: After implementing the algorithm, a web interface is created using HTML, CSS, and JavaScript. This interface allows users to input the number of days for prediction and displays the results in a clear and visually appealing manner.
5. **Testing and Optimization**: Finally, the system is tested for accuracy and performance. The results are validated using real-world weather data, and any necessary optimizations are made.

# CHAPTER 4. RESULTS ANALYSIS AND VALIDATION

## 4.1. Implementation of Solution

The solution was implemented in two major phases:

- **Phase 1**: The dynamic programming algorithm was first implemented in C++, which is well-suited for handling large datasets and ensuring efficient computation. This phase involved parsing the historical data, processing it, and applying the dynamic programming technique to make weather predictions.
- **Phase 2**: The C++ code was then translated into JavaScript for web deployment. JavaScript was chosen because it is supported by all major web browsers and can be executed client-side, allowing the system to run on the user's device without the need for a server.

The system was validated by inputting various values and comparing the predicted temperatures and precipitation levels against real-world weather data from the past. The predictions were reasonably accurate, and the system was able to make forecasts for multiple days, depending on user input.

---

# CHAPTER 5. CONCLUSION AND FUTURE WORK

## 5.1. Conclusion

In conclusion, this project successfully demonstrates a lightweight and efficient method for weather prediction using dynamic programming. By integrating historical weather data from the Meteostat API and implementing the prediction algorithm in both C++ and JavaScript, the project ensures that the system is accessible to users through a web-based interface. The solution is capable of making reasonably accurate short-term weather predictions, offering an easy-to-use and efficient tool for weather forecasting.

## 5.2. Future Work

While the system is functional, there are several areas where future work could improve the accuracy and capabilities of the project:

- **Expansion to Other Variables**: Future versions of the system could incorporate additional weather variables such as humidity, wind speed, or cloud cover to enhance prediction accuracy.
- **Integration with Machine Learning**: Although dynamic programming is efficient for short-term predictions, machine learning models like ARIMA or LSTM networks could be explored for more accurate, long-term weather predictions.
- **Real-Time Data Integration**: Future work could also include integrating real-time weather data into the system, allowing for live weather updates and predictions based on the most current data available.