Capstone Weather App

Project Report

(PROJECT TERM FEBRUARY - AUGUST 2024)

TITLED:

DEVELOPMENT OF WEATHER APP FOR UPDATE OF THE DAILY WEATHER CONDITION.

SUBMITTED BY:

ADEOLU ADELEKE PETER.

STUDENT ID: **IDEAS/24/97988**

UNDER THE SUPERVISION OF: **HAYAT S. T**

DEPARTMENT: **SOFTWARE ENGINEERING.**

BAZE UNIVERSITY, ABUJA, NIGERIA.

AUGUST 2024

ACKNOWLEDGEMENT

I want to extend my sincere appreciation to this lovely department of Software Engineering which provided the opportunity for me to fulfil and achieve my goal.

I am really grateful to **HAYAT S. T** and everyone that has contributed to the success of this program and for provided an opportunity to undertake this project and provided me with all the facilities.

I am highly grateful sir for this active support, valuable time and advice, whole-hearted guidance, sincere cooperation and pains-taking involvement during the study and in completing the said project within the time stipulated.

Lastly, I am very grateful to all those, particularly the various instructors, friends, who have been instrumental in creating a proper, healthy and conducive environment including new and fresh innovative ideas for me during the project. Without their help, it would have been extremely difficult for me to prepare the project in a time-bound framework.

ABSTRACT

This project was carried out on Weather App. This is the summary of the project, including the problem statement, methodology, results, and conclusions.

1. **Problem Statement**:

   Current weather apps lack simplicity and speed, making it challenging for users to quickly access accurate information for specific cities. There is a need for a streamlined web application that prioritizes user-friendly interfaces, delivering real-time, precise weather details for informed decision-making in travel, planning, and daily activities.

2. **Methodology**:

   - **Visual Studio Code**: The official Integrated Development Environment (IDE) for desktop app development.
     - Provides features that enhance productivity during app building.
   - **APIs**:
     - APIs facilitate communication between software components using predefined definitions and protocols.
     - In this case, the weather app uses an API to fetch weather data.
   - **Features**:
     - Earth Time-Lapse: Displays global weather changes through images.
     - Rainfall Predictions.

- Sunrise and Sunset Times.
- Wind Predictions.
- Humidity Updates.
- UV Weather Map.
- Climatic Conditions Map.
- Detailed Weather Forecast (weekly, monthly, daily, and hourly).

3. **Results**:
    - The app provides accurate weather information, allowing users to make informed decisions.
    - Users can conveniently access weather reports from home.
    - Detailed data for every country and city is available.
    - The app enhances environmental awareness and preparedness.
4. **Conclusions**:

    Weather forecast apps play a crucial role in our lives, impacting decision-making, outdoor activities, and overall awareness.

TABLE OF CONTENTS

List of Figures and Tables

6.1 Introduction

Recent developments in the provision of weather and climate information have created opportunities to better integrate scientific information into decision-making e.g. Adams et al., Citation2015;

Furthermore, in the context of a changing climate (IPCC, 2013) and the high exposure of developing countries to climate change risks (Hewitson, & Coauthors, Citation2014), it is important that long-term planning decisions assess future climate projections to help reduce risks and utilize opportunities.

The relevance of weather and climate information is largely dependent on the ability of scientists to provide information that is fit-for-purpose and produced in formats that can be integrated into decision-making processes. The relevance of weather and climate information is dictated by the nature of the risks being managed, the economic sector of focus, the region of interest, the governance structures within which decisions are made, and other context-specific realities. In Africa, managing weather and climate risks is intrinsically related to the sociocultural context, differential vulnerability and economic development pathways.

Weather data is not only necessary for researchers or scientists, ordinary people can be benefitted from it as well. People nowadays are feeling the necessity of weather data as well. There are a variety of weather mobile apps in Google Play and the App store.

Therefore, weather apps have become indispensable tools for users worldwide. They provide real-time weather updates, forecasts, and critical information for planning daily activities. As technology advances, weather forecasting accuracy has improved significantly, making these apps even more valuable.

6.2 PROBLEM STATEMENT

Current weather apps lack simplicity and speed, making it challenging for users to quickly access accurate information for specific cities. The basic issue with weather apps, is that many of them remove a crucial component of a good, reliable **forecast**.

Our weather-app ambivalence is a strange pull between feeling grateful for instant access to information and simultaneously navigating a sense of guilt and confusion about how the experience is also, somehow, dissatisfying—a bit like staring down Netflix's endless library and feeling as if there's nothing to watch. Weather apps aren't getting worse. In fact they're only getting more advanced, inputting more and more data and offering them to us to consume. Which, of course, might be why they feel worse.

6.3 OBJECTIVES

1. **User-Friendly Interface**: Develop an intuitive and visually appealing weather app that provides accurate weather details.
2. **Accurate Predictions**: Implement precise weather predictions, including rain forecasts, cloud cover, and wind speed.
3. **Hyper-Local Forecast**: Offer minute-by-minute weather forecasts based on the user's current location.
4. **Engaging Visuals**: Enhance user experience with stunning maps and engaging visuals.
5. **Smart Notifications**: Remind users to carry umbrellas or take precautions based on real-time weather conditions. To make a real time weather application that takes user's exact location and provides weather forecast for the day and upcoming days also. We also tried to design a simple but visual UI that provides comprehensive data. Also, the application provides

suggestions to users based on weather conditions. And lastly, user can search and access data for custom locations (string based).

6.4 SCOPE

The weather forecast app will focus on providing accurate weather data for specific cities, including temperature, humidity, wind speed, and descriptions. It will cater for users seeking timely information for daily planning.

6.5 SIGNIFICANCE

1. Real-Time Weather Updates

One of the most obvious yet vital benefits of weather apps is providing real-time weather updates. Users get instant access to current weather conditions, which is crucial for planning daily activities.

2. Advanced Weather Forecasting

Modern weather apps offer advanced forecasting, using sophisticated algorithms and data analysis. This feature allows users to view weather predictions for days, or even weeks ahead, aiding in long-term planning and decision-making for both personal and professional activities.

3. Severe Weather Alerts

Safety is a paramount concern, and weather apps contribute significantly by providing severe weather alerts.

4. Customization and Personalization

Weather apps offer a high degree of customization. Users can set up alerts for specific weather conditions, track weather in multiple locations, and personalize the app interface to suit their preferences.

5. Enhancing Travel Safety and Planning

For travelers, weather apps are indispensable tools. They provide valuable information for safe travel, helping in planning journeys and avoiding weather-related disruptions.

6. Agricultural and Environmental Benefits

Weather apps are incredibly beneficial for agriculture. Farmers can plan sowing, irrigation, and harvesting activities based on accurate weather predictions.

7. Educational Tool

Weather apps serve as excellent educational tools, especially for students and enthusiasts interested in meteorology.

8. Business and Economic Advantages

Weather apps hold significant benefits for businesses. Industries such as agriculture, aviation, and tourism rely heavily on weather forecasts for operational planning.

9. Enhancing Outdoor Activities

Finally, for outdoor enthusiasts, weather apps are essential. They provide the necessary information to plan activities such as hiking, fishing, or photography, ensuring that the weather doesn't spoil the fun.

6.6 METHODOLOGY OVERVIEW

1. Team Formation
2. Creating Project Synopsys
3. Requirement Gathering
4. Coding or Implementation
5. Testing
6. Project Presentation

Step 1: Team Formation Phase

Team formation is a crucial step in any project it significantly impact on this project. In this project as I will be exploring about the web application for weather app so will be going to require following skill sets.

1. **Front end Development (Html, CSS).**
2. **Back-end Development (JavaScript).**
3. **Tester**
4. **UI/UX Developer**

Step 2: Creating Project Synopsys

A project synopsis serves as a concise overview or summary of a proposed project, offering a brief but comprehensive insight into its objectives, scope, methodology, and expected outcomes.

2.1 Introduction | Project Synopsys for Weather Forecasting Project

Today's Weather app is a web application which will tell the users about the weather details of any particular city. The easy and Interactive User Interface will help our users to easily know about the temperature, wind speed, humidity and description about the weather.

2.1.1 Problem Statement

2.1.2 Proposed Solution for Weather Forecasting Project

2.1.3 Objective of the Project

3. Requirement Gathering:

a. Create a Software Requirement Specification (SRS) document.

b. Define functional and non-functional requirements for your app.

4.Coding or Implementation:

1. Set up a development environment.
2. Implement the weather forecasting application's front end using HTML, CSS, and JavaScript.

5. **Testing**:

1. Thoroughly test the application to ensure accuracy and reliability.

2. Consider automated testing and manual testing.

6. **Project Presentation**:
    1. Prepare a presentation showcasing the weather forecasting application.
    2. Highlight its features, user interface, and functionality

7. LITERATURE REVIEW

7.1 INTRODUCTION

Weather forecasting is the application of science and technology to predict the state of the atmosphere for a given location. Ancient weather forecasting methods usually relied on observed patterns of events, also termed pattern recognition. For example, it might be observed that if the sunset was particularly red, the following day often brought fair weather.

However, not all of these predictions prove reliable. Here this system will predict weather based on parameters such as temperature, humidity and wind. User will enter current temperature; humidity and wind, System will take this parameter and will predict weather (rainfall in inches) from previous data in database (dataset).

The role of the admin is to add previous weather data in database, so that system will calculate weather (estimated rainfall in degree Celsius) based on these data.

Weather forecasting system takes parameters such as temperature, humidity, and wind and will forecast weather based on previous record therefore this prediction

will prove reliable. This system can be used in Air Traffic, Marine, Agriculture, Forestry, Military, and Navy etc.

Keywords- Weather forecasting, temperature; short range, governments, tropical storms, challenging task, forecasting.

Dr. C. K. GOMATHY in journal of engineering, computing & architecture opined that:

Weather forecasting is the prediction of the state of the atmosphere for a given location using the application of science and technology. This includes temperature, rain, cloudiness, wind speed, and humidity. Weather warnings are a special kind of short-range forecast carried out for the protection of human life.

Danielle Wardinsky Hallows School of Communications, BYU Master of Arts said traditional television viewership of weather forecasts are on the decline, mobile weather applications are becoming the new media medium for weather information. In fact, the total number of app downloads in the weather market reached 69.5 million at the end of 2021 (Statista, 2021). As a result, understanding the characteristics of weather app users and what those users are looking for is pertinent to research regarding weather communication. While weather is an ongoing phenomenon sought to be understood by people for many decades, the uses and gratifications of engaging in weather communication are underresearched in literature. Thus, this study aimed to identify the characteristics of weather app users and their motivations for checking mobile weather apps.

Features Weather Forecast Project In Python Django:

- Time to time update weather
- Temperature Update
- Last 7 days data Predict
- change weather in every hour as according to weather changes.
- provide accurate data information about weather.
- user can search weather anytime and anywhere.
- any places data can be search and provide information as according to weather.
- help user to travel.
- help User to future plans for holidays.

- **7.2 RELEVANT THEORIES/MODELS**

1. ***Weather Research and Forecasting (WRF) Model:***

- The WRF Model, released in 2000, is widely used for numerical weather prediction.
- It serves both research and operational needs, offering a range of options and capabilities.
- The WRF Model has a centralized support effort but is driven by contributions from a global user base.

1. **Numerical weather Prediction (NWP):**
   a. NPW models simulate atmospheric process using mathematical equations.
   b. They divide the atmosphere into grid cells and predict changes over time
   c. Popular NWP models include the Global forecast System (GFS) and predict changes over time.
2. **Ensemble forecasting**:
   a. Ensemble models provide a range of possible outcomes, improving prediction confidence
   b. Assimilation techniques combine satellite observations with model predictions to improve accuracy.

2. **Climate model**:
    a. Climate models simulate long term climate patterns

They help understand global climate change and its impact on local weather.

1. **Machine Learning Approaches**:
    - Researchers have explored machine learning techniques to improve weather prediction accuracy.
    - Parameters like temperature, humidity, wind direction, and precipitation are considered.
    - Decision tree and random forest algorithms enhance accuracy.
2. **Big Data Analytics**:
    - Employing big data analytics in weather forecasting can address challenges related to traditional data management.
    - Systematic literature reviews have explored big data analytic approaches in weather forecasting.
3. **Data-Driven Weather Forecasting**:
    - Recent developments focus on data-driven models.

7.3 Related Work

1. **TempoQuest**:
    - TempoQuest offers the world's most advanced weather forecasting software.
    - Their product, **AceCAST**, leverages GPU (Graphics Processing Unit) technology to accelerate high-resolution regional weather models.
    - It provides better insights into weather intensity, exact timing, and precise location of weather impacts.
    - TempoQuest also develops **WSV3**, a versatile live weather tracking and monitoring tool.

2. **National Weather Service (NWS)**:
    - The NWS collaborates with various providers and services to enhance weather data and software.
    - Their resources include commercial, academic, and governmental products.

3. **Google DeepMind's GraphCast**:
    - Introduced in 2023, GraphCast uses AI algorithms trained on 40 years of historical data.
    - It aims to improve weather prediction accuracy by refining predictive models over time.
4. **Pangu-Weather by Huawei**:
    - Huawei unveiled Pangu-Weather, another AI-based weather forecasting system.
    - These systems represent the cutting edge of weather prediction technology

7.4 Gaps in Existing Work

1. **Limited Data Availability**:

- AI models rely heavily on historical data for accurate predictions.
- However, certain weather conditions may have limited data availability, affecting the model's performance.
- Data gaps in remote or less monitored regions can impact forecasting accuracy

2. **Gate information highways:**

a. Some countries restrict access to weather data due to security or geographical reasons.

b. Lack of global data sharing hinders comprehensive forecasting especially for crossing border weather events.

3. Technology and hardware differences:

a. Weather systems across different countries varying technologies and hardware standard
b. Inconsistent data collection methods and equipment can introduce biases and affect prediction quality.

4. **Geographical and Microclimates changes:**
   a. Weather app often struggle with microclimates and localized variations.
   b. Geographical features create unique weather patterns that standard model may overlook.
5. **Forecast Horizon Limitations:**
   a. While modern weather apps can predict up to 10 days ahead, long term forecasts remain challenging.
   b. The chaotic nature of the atmosphere limits predictability beyond a certain point.

7.5 NUMBER OF ARTICLES:

1. Weather Forcasting APP Project Report. IT essentials (RMIT University).
2. Weather Forecast Project. Last Updated: 14 Feb, 2024.
3. Capstone Project Report Final. PYTHON (Lovely Professional University).
4. Building a Complete Weather App from Scratch with HTML, CSS, and JavaScript: A Step-by-Step Guide by Odumosu Matthew
5. *Articles, videos, and documentation to help software engineers get the most from Visual Crossing Weather. By Visual Crossing Weather*

8. SYSTEM REQUIREMENTS

8.1 FUNCTIONAL REQUIREMENTS

8.1.1 SOFTWARE REQUIREMENTS

This software package is developed using html, CSS for frontend and JavaScript for the backend. Using Vs Code as a text editor and Google Chrome for the execution of our code.

Required Specifications for our Device:

- **Operating System**: Windows 7, 8, 9, 10.
- **Language**: Html, Css, Javascript.
- **API**: Openweathermap api

Hardware Requirements:

- **Processor**: Intel core i3 or above for a stable experience and fast retrieval of data.
- **Hard Disk**: 4GB and above
- **RAM**: 256 MB or more, recommended 2 GB for fast reading and writing capabilities which will result in better performance time.

8.2 Non-Functional Requirements:

8.2.1 Usability Requirements

- Our user interface should be interactive simple and easy to understand. The system should prompt for the user and administrator to login to the application for proper input criteria.

- Weather Application shall handle expected and non expected error in ways that prevent loss in information and long downtime period.

8.2.2 Security Requirements

- System should be using secure web forecasting API.
- Normal users can just read information about the weather but they cannot edit or modify anything except their personal and some other information if required.
- System will have different types of users and every user has access constraints.
- Proper user authentication should be provided.
- Personal details like location should not be stored for security purpose.

8.2.3 Availability Requirements

Availability requirements for a weather application using APIs are crucial to ensure that the service is consistently accessible and operational.

Here are key availability requirements:

- **Uptime Percentage:** Maintain a high level of service availability, such as 99.9% uptime. Users rely on the weather application for real-time information, so a high uptime percentage ensures that the service is consistently accessible.
- **Load Balancing:** Use load balancing to distribute incoming traffic across multiple servers or instances. Load balancing helps distribute the load evenly, preventing individual servers from becoming overwhelmed and improving overall system performance and availability.

- **Monitoring and Alerting:** Implement continuous monitoring of key metrics and set up alerting systems to notify administrators of any issues or anomalies. Proactive monitoring allows for the rapid identification and resolution of potential problems, minimizing downtime.
- **Scalability:** Design the system to be scalable, allowing for the seamless addition of resources during periods of increased demand. Scalability ensures that the application can handle varying levels of traffic and user activity without degradation in performance.
- **Backup and Recovery:** Regularly back up critical data and implement robust recovery procedures. In the event of data loss or system failures, a well-defined backup and recovery strategy ensures that the application can be restored quickly and efficiently.

8.2.3 Performance Requirements:

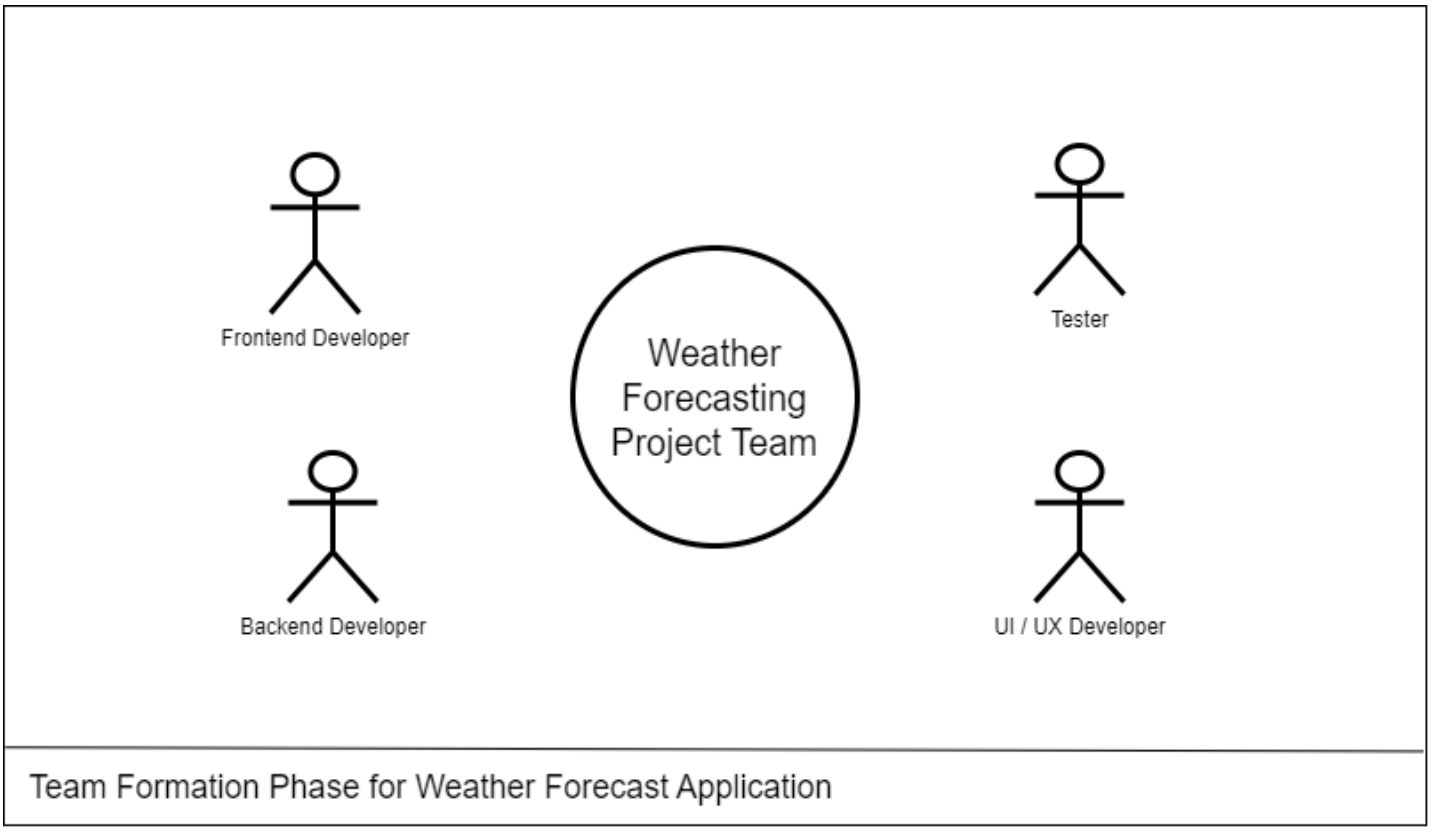
- The system shall accommodate high number of users simultaneously and users can check the weather of any location any number of times.
- Responses to view information shall take no longer than 5 seconds to appear on the screen.

8.2.4 Error Requirements:

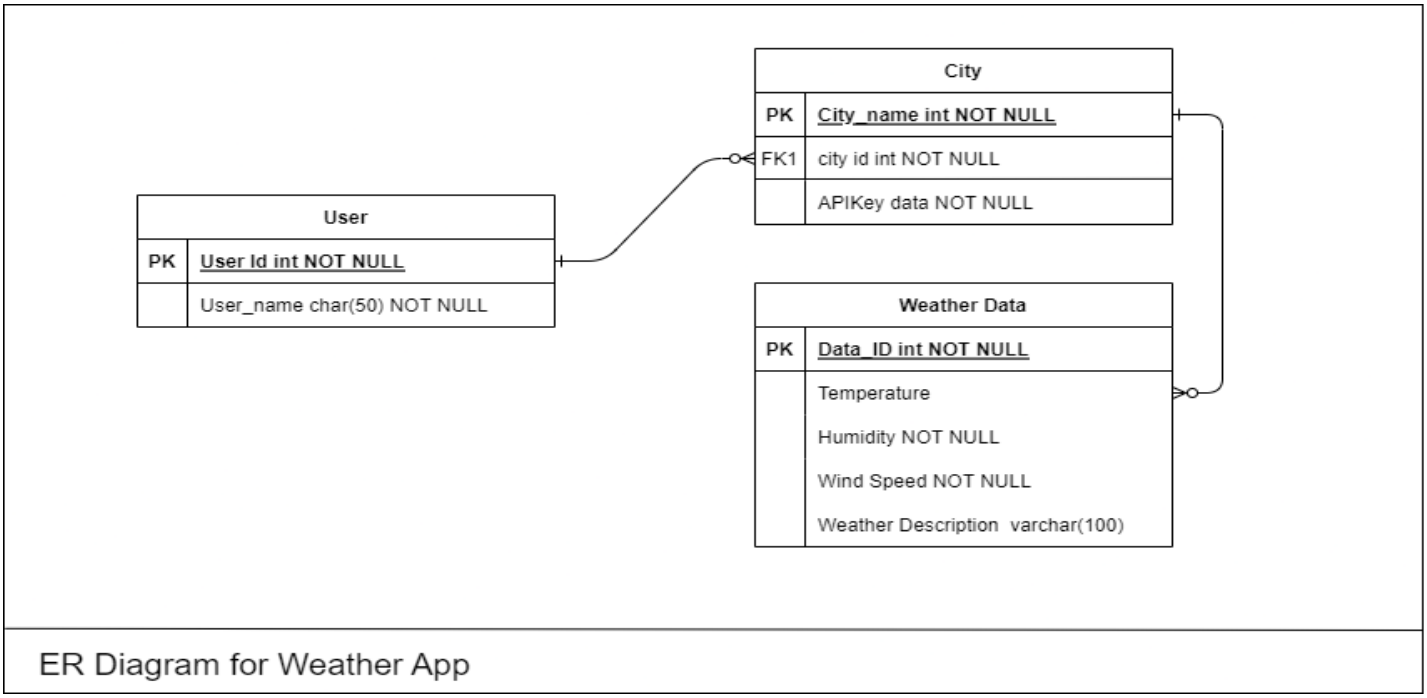
Weather app shall handle expected and non-expected errors in ways that prevent loss in information and long downtime period.

An alert should be shown if the API is not working properly which will improve the uptime of the service.

8.3 USE CASE DIAGRAMS



Team Formation Phase for Weather Forecast Application



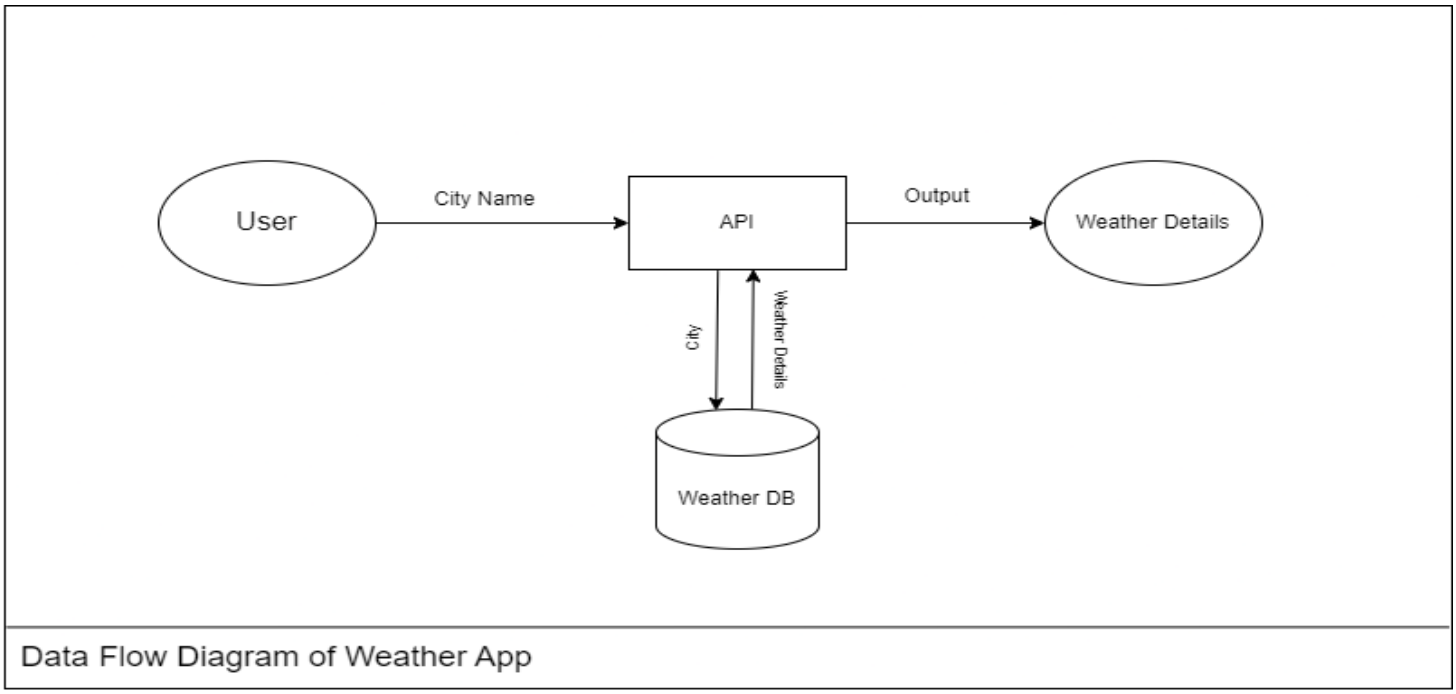
ER Diagram for Weather App

Entities:

- **User:** Attributes User Id (Primary Key)
- **City:** Attributes: City Name (Primary Key), API Key value.
- **Weather Details:** Attributes: Temperature, Wind Speed, Weather Description, Humidity.

Relation:

- **Enters:** User enters the city name in the application.
- **Returns:** API returns a list of weather details having temperature, wind speed humidity and weather details



Data Flow Diagram of Weather App

9. SYSTEM DESIGN

9.1 SYSTEM ARCHITECTURE

9.1.1 User Interface (UI):

- The UI is the front-end component that users interact with. It includes screens, buttons, input fields, and visual elements.
- Users can search for weather information, view forecasts, and customize preferences through the UI.

9.1.2 Client-Side Logic:

- This layer handles user interactions and communicates with the back end.

- It includes JavaScript (if web-based) or other client-side technologies.
- Responsibilities:
  - Collect user input (e.g., city name, coordinates).
  - Make API requests to fetch weather data.
  - Display weather information to users.

9.1.3 API Integration:

- The app interacts with external weather APIs (e.g., OpenWeatherMap, ClimaCell).
- API endpoints provide real-time weather data (current conditions, forecasts, historical data).
- The app sends requests to these APIs and processes responses.

9.1.4 Server-Side Logic (Back End):

- The back-end handles data processing, business logic, and communication with databases and APIs.
- Responsibilities:
  - Receive API requests from the client.
  - Validate input data.
  - Fetch weather data from external APIs.
  - Cache data to reduce API calls.
  - Format and preprocess data for the client.

9.1.5 **Database**:

- If the app stores user preferences, historical data, or cached weather information, it may use a database.
- Common databases include MySQL, PostgreSQL, or NoSQL databases like MongoDB.

9.1.6 **External APIs**:

- These APIs provide weather data (temperature, humidity, wind speed, etc.).
- The app integrates with them to retrieve accurate and up-to-date information.

9.1.7 **Caching Layer**:

- To improve performance, the app may cache frequently accessed data (e.g., recent weather conditions).
- Caching reduces the load on external APIs and speeds up response times.

9.1.8 **Third-Party Libraries and Frameworks**:

- Developers use libraries (e.g., Axios for HTTP requests) and frameworks (e.g., React, Angular, or Vue.js) to build the app efficiently.

9.1.9 **Deployment and Hosting**:

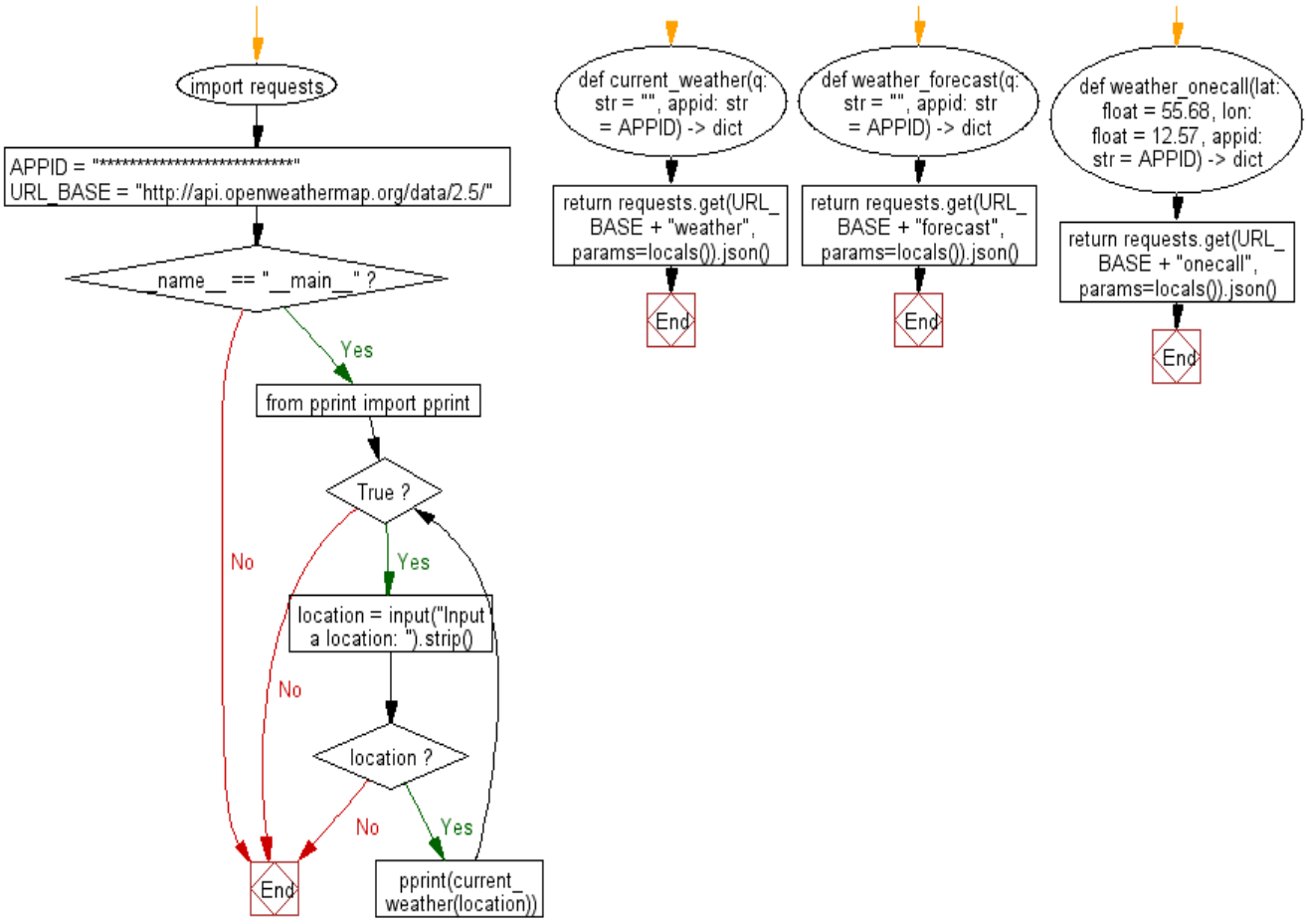
- The app needs a server to host the back end and serve the front end.
- Deployment platforms include cloud services (AWS, Azure, Google Cloud) or shared hosting providers.

9.1.10 Security Considerations:

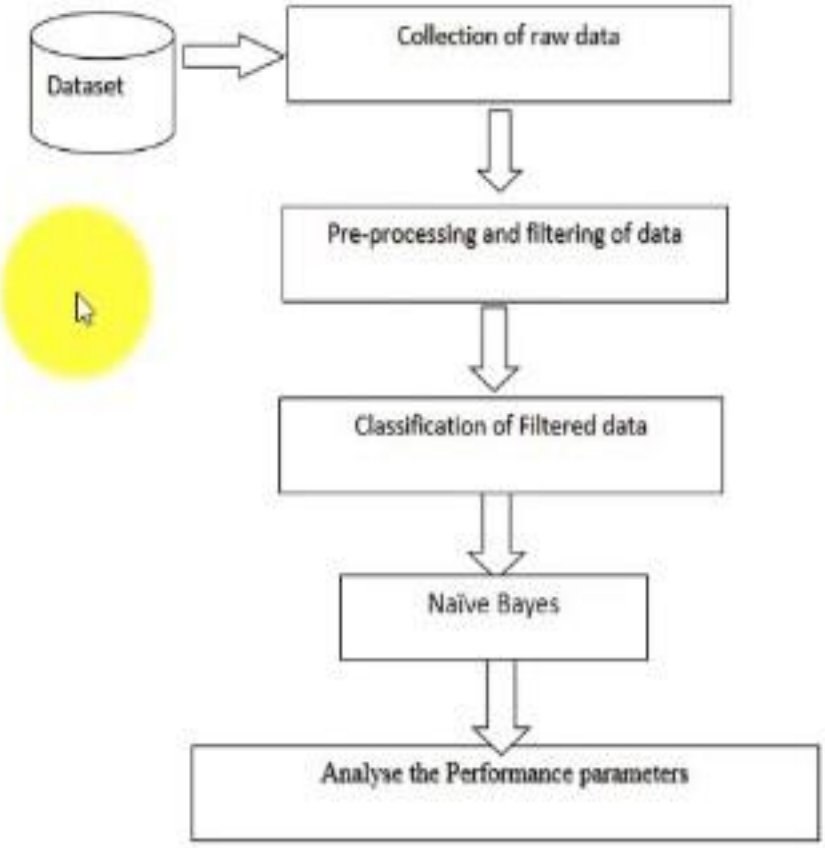
- Implement secure communication (HTTPS) between the client, server, and external APIs.
- Protect sensitive data (API keys, user preferences) using environment variables.
- Validate user input to prevent malicious requests.

9.2 DESIGN MODELS

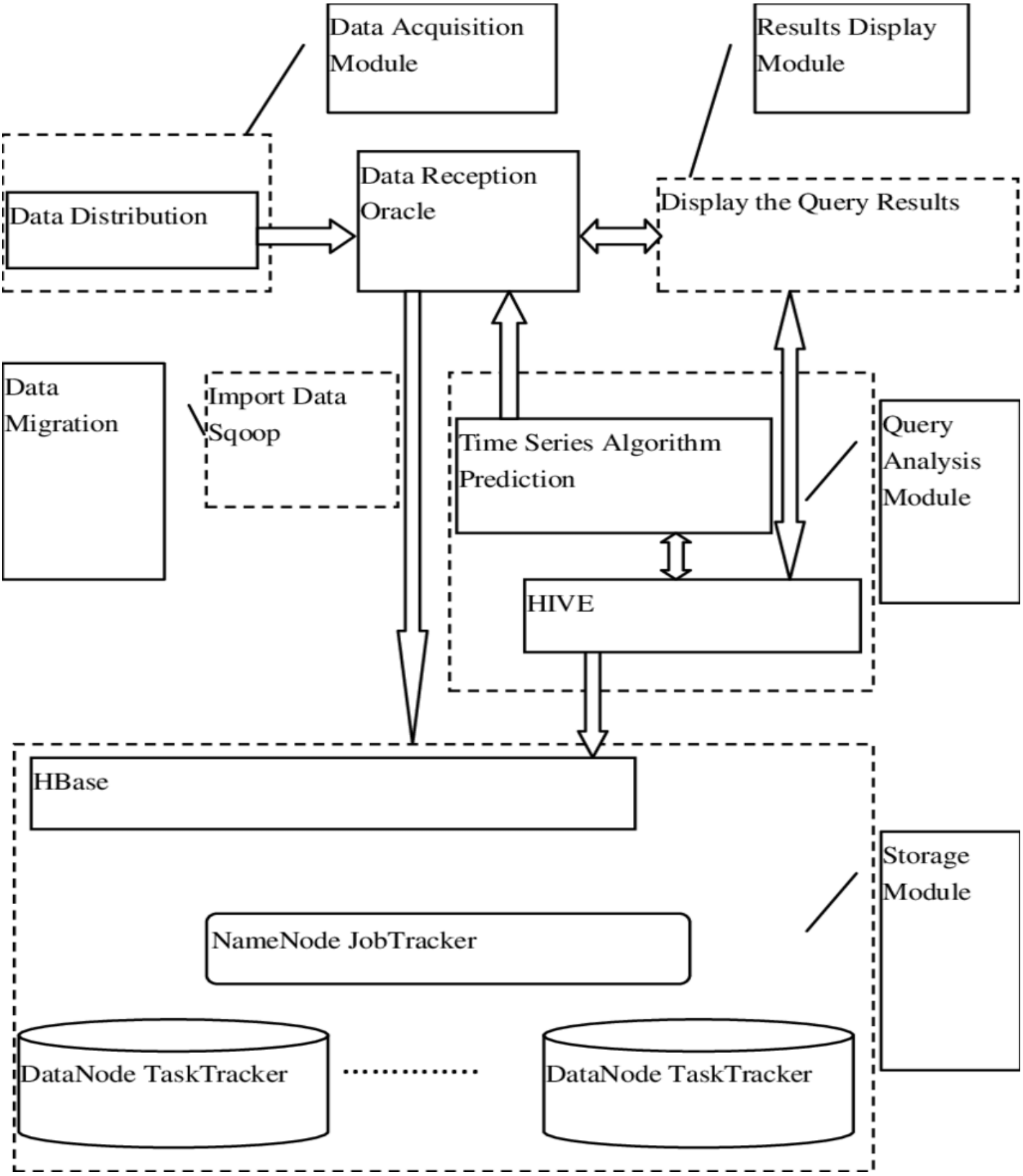
UML DIAGRAM



Flowchart


Dataset
Collection of raw data
Pre-processing and filtering of data
Classification of Filtered data
Naïve Bayes
Analyse the Performance parameters

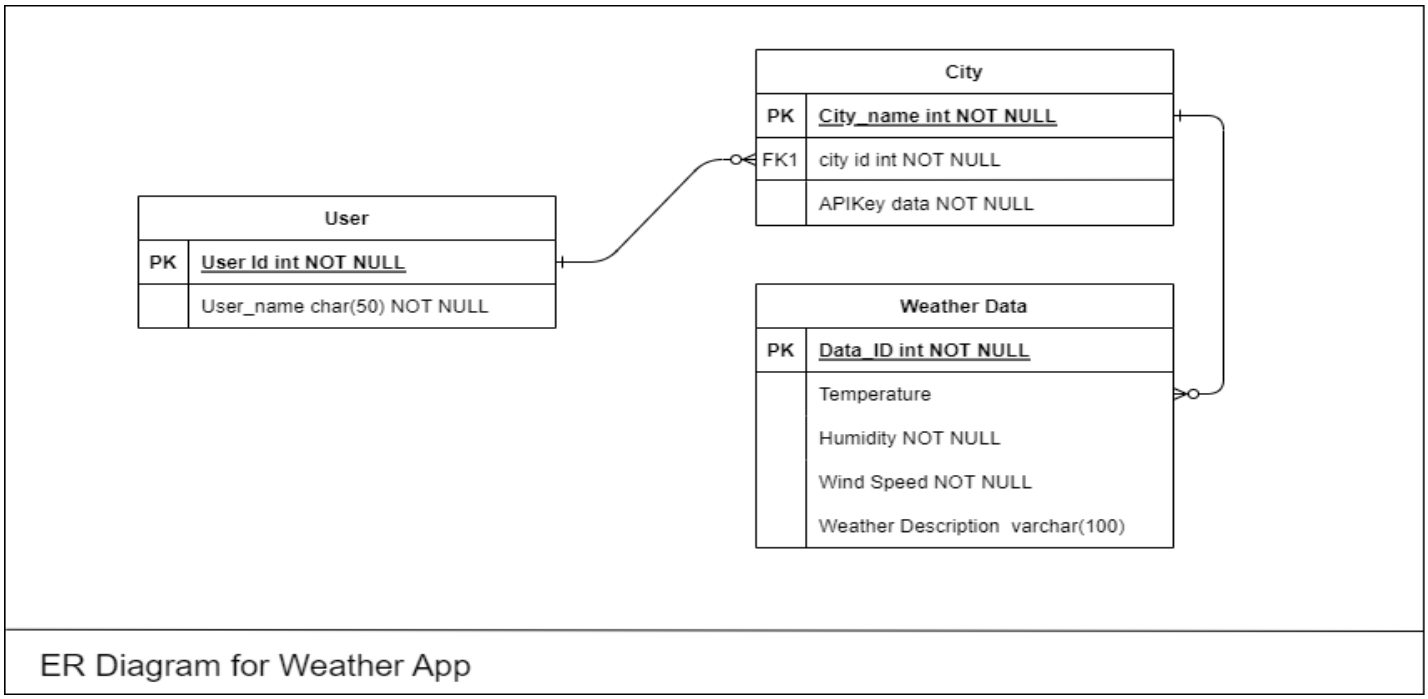
ACTIVITY DIAGRAM



9.3 DATABASE DESIGN:

What Is an ERD Diagram?

ERD is short for entity-relationship diagram; they are also known as ER diagrams. An ERD is a graphical representation of how to store data within a system; it shows the entities that hold data and the relationships between entities. It is essentially a graphical tool for database design. The main purpose of an ERD is to offer a clear overview of the structure of the data within a system, which can be helpful when designing or redesigning a database.



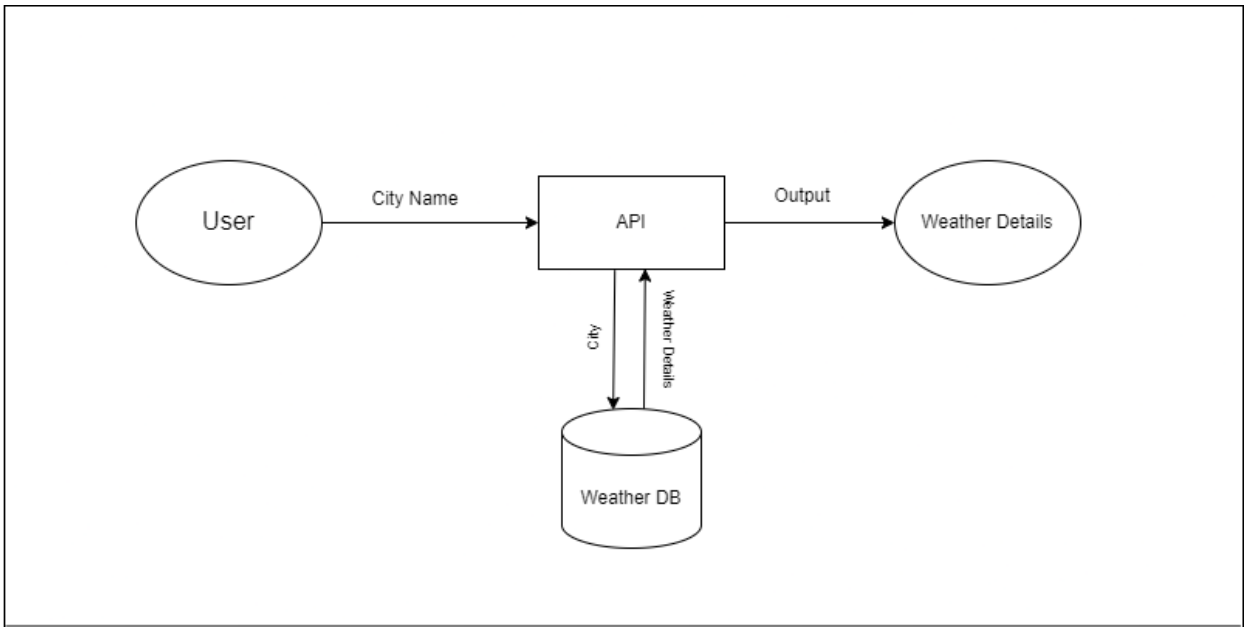
ER Diagram for Weather App

COMPONENTS OF AN ERD:

- Entities: These represent real-world objects or abstractions. For instance, in a weather forcast app database, entities could be Temperature, city, people/traveler
- Attributes: These are properties of an entity (e.g. degree of weather, name of location). Each attribute becomes a column in a database table.
- Relationships: Describe how entities interact (e.g., a tourist travelled to a location during wet season or winter). Relationships are represented as connecting lines between entities.

9.4 USER INTERFACE DESIGN:

The user interface for the task will have a site that will have the live feed alongside the data about the climate. This site will utilize html, CSS, JavaScript and API requests for site.



Data Flow Diagram of Weather App

9.5 TECHNOLOGY STACK:

1. VS Code: It's a platform where coding and all other activities are carried out.
2. Django: Django is a powerful Python web framework that simplifies building web applications. It provides features like URL routing, database models, authentication, and templating.
3. OpenWeatherMap API: To fetch weather data, where I will get API that provides weather information. OpenWeatherMap is a popular choice. You can sign up for a free API key on their website.
4. Bootstrap (optional): Bootstrap is a front-end framework that helps create responsive and visually appealing user interfaces. While not mandatory, it can enhance your app's design.
5. Requests Library: Is used to Request for the library to make HTTP requests to the OpenWeatherMap API and retrieve weather data.

10. IMPLEMENTATION

10.1 DEVELOPMENT ENVIRONMENT:

In this stage we are going to create the environment to build our project, we will install all required software and extensions for ease in the coding part.

Required Softwares:

- VsCode: Vs Code is a widely used text editor for development purpose.

- Google Chrome: You need to install a web browser to execute the html code. You can use any of your favourite web browser.

Extensions:

- Live Server: You can use live server extension because It enables you to right-click an HTML document, and it runs a server for you and opens a browser window with the file in it.

10.2 CODE STRUCTURE:

Below is the Code for Creating above page:

- script.js
- index.html
- style.css

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys

def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'weatherproject.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)

if __name__ == '__main__':
    main()

"""
Django settings for weatherproject project.
```

```
"""
Generated by 'django-admin startproject' using Django 5.0.6.

For more information on this file, see
https://docs.djangoproject.com/en/5.0/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/5.0/ref/settings/
"""

import os
from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-fgq1d(45t2u0-3vmem(n9mzd(5t3_xf52%qr*vw5*0u&$nw!6s'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [
    'weatherApp',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

```
ROOT_URLCONF = 'weatherproject.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'weatherproject.wsgi.application'

# Database
# https://docs.djangoproject.com/en/5.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}

# Password validation
# https://docs.djangoproject.com/en/5.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
```

```
]

# Internationalization
# https://docs.djangoproject.com/en/5.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/5.0/howto/static-files/

STATIC_URL = '/static/'
MEDIA_URL = '/images/'

STATICFILES_DIRS = (
    os.path.join(BASE_DIR, 'static'),
)

# Default primary key field type
# https://docs.djangoproject.com/en/5.0/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

from django.shortcuts import render

# Create your views here.
import urllib.request
import json

def index(request):
    if request.method == 'POST':
        city = request.POST['city']
        source =
urllib.request.urlopen('http://api.openweathermap.org/data/2.5/weather?q=' +
                                        city +
'&units=metric&appid=72d29d6a50742ee8626d89b977648e0b').read()
        list_of_data =  json.load(source)
        dataa = {
            "country_code": str(list_of_data['sys']['country']),
            "coordinate": str(list_of_data['coord']['lon']) * ','
            + str(list_of_data['coord']['lat']),
            "temp": str(list_of_data['main']['temp']) + 'deg_celsius',
            "pressure": str(list_of_data['main']['pressure']),
```

```
            "humidity": str(list_of_data['main']['humidity']),
            'main': str(list_of_data['weather']['main']),
            'description': str(list_of_data['weather']['description']),
            'icon': list_of_data['weather']['icon']
        }
        print(data)
    else:
        data = {}

        return render(request, 'main/index.html', data)
<DOCTYPE html>
<html lang="en">
{% load static %}

<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scales=1.0"/>
    <link rel="stylesheet" href="<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/bootswatch/5.3.3/cerulean/bootstrap.min.cs
s" integrity="sha512-
vaImhtQoaCXvevCM/UK+8ND/df0kiQWLtR65wgq0AWShEXLpWHMve0oGgU0q1gq1MbbyEuAhMMlqNmZd7s7wTQ=
=" crossorigin="anonymous" referrerpolicy="no-referrer"
    integrity="sha512-
dQLT/B7byn2LjN/DN4zeBKpwGVGqbidV0XiMRWQOL7TGrV7FK2FldkGG+DGMU+CQnMTcRZlUI7GMWtlj6akNew=
="
    crossorigin="anonymous" />
<!-- <link rel="stylesheet" href="(% static
'weatherproject/main/templates/static/style.css' %)" /> -->
 <title>Weather App </title>
 </head>

 <body>
    <div class="navbar navbar-expand-lg fixed-top navbar-dark bg-dark">
        <div class="container">
            <a href="../" class="navbar-brand">Weather App <spam style="color: rgb(39,
117, 161);"><strong> - Django
                    Framework</strong>
                </spam>spam> </a>
            <a href="https://openweathermap.org" class="navbar-tech">Openweather - Get
Your API </a>
        </div>
    </div>

<br /><br /> <br>
<div id="jumbotrom" class="jumbotro" style="text-align: center; margin-top:-s0px">
    hi class=display-s>Weather Desktop App </hi>
    <hs>Using python Language and Django Framework</hs>

```

```
    <img src="(% static 'images/weather.png' %)" class="image" style="width:100px;
margin-bottom:-50px; ">
    </div>

    <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
        <form method="post" class="col-md"">
            {% csrf_token %}
            <div class="input-group">
                <button type="submit" class="btn btn-primary">search</button>
            </div>
            </div>
            <from>
            </nav>
            <br> <br>
            <div class="row">
                {%if country code and coordinate and temp and pressure and humidity%}
                <div class="col d-flex justify-content-center" ">
                    <div class="card text-white bg-light mb-6">
                    <div clss=" card-body">
                        <h4><spam class="badge badge-primary">Country code :</spam>
{{country_code}}</h4>
                        <h4><spam class="badge badge-primary">Coordinates [X,Y]
:</spam> {{coordinate}}</h4>
                        <h4><spam class="badge badge-primary">Temperature in celsius
:</spam> {{temp}}</h4>
                        <h4><spam class="badge badge-primary">Humidity :</spam>
{humidity}}}</h4>
                        <h4><spam class="badge badge-primary">Forecast :</spam>
{{main}} <img
                            src="http://openweathermap.org/img/w/{{icon}}.png"
alt="image" style="width:70px"></h4>
                        <h4><spam class="badge badge=primary">Description : </spam>
{{description}}</h4>

                    </div>
                    {% endif %}
                </div>
            </body>

            </html>
```

10.3 KEY ALGORITHMS/MODULES:

1. Data Retrieval and Parsing:
    - API Integration: Connect to a weather API e.g OpenWeatherMap to retrieve real-time weather data and forecasts.
    - Data Parsing: Extract relevant information (temperature, humidity, wind speed, etc.) from the API response.
2. Geolocation Services:
    - Geocoding: Convert user-entered location (city name, ZIP code) into latitude and longitude coordinates.
    - Reverse Geocoding: Convert coordinates back to human-readable location names.
3. Forecasting Algorithms:
    - Numerical Weather Prediction (NWP): Use mathematical models (e.g., WRF, ECMWF) to simulate atmospheric processes and predict future weather conditions.
    - Statistical Methods: Techniques like autoregressive integrated moving average (ARIMA) or exponential smoothing for short-term forecasting.
4. Database Management:
    - Database Design: Store weather data efficiently (e.g., hourly, daily) in a database (NoSQL or relational).
    - Caching: Implement caching mechanisms to reduce API calls and improve performance.

5. User Interface (UI):
    - Responsive Design: Create an intuitive UI for users to input their location and view weather details.
    - Graphs and Charts: Display temperature trends, precipitation, and other data using visual elements.
6. Notifications and Alerts:
    - Severe Weather Alerts: Notify users about extreme weather conditions (storms, heatwaves, etc.).
    - Push Notifications: Send alerts directly to users' devices

10.4 CHALLENGES:

1. **Time limitation**

The deadlines set for the project is incredibly short and is one of the major challenges of being a software engineer. After all, it is a game of time. When engineers collaborate with several clients across various time zones, the process becomes considerably more difficult. These time restraints frequently cause development teams to work less productively, resulting in sub-quality products.

2. **The rapid advancement of technology**

Every technological innovation is a blessing. The rapid advancement of technology puts additional pressure on software development to take advantage of these trends when creating new software products to stand out from the crowd and obtain a competitive advantage. It is one of the major software engineering problems.

3. **Limited infrastructure/ resources**

The lack of resources or IT infrastructure to carry out projects successfully is another issue that most software development companies deal with and is among the major problems of software engineering. It could be caused by a lack of high-performance programming tools, strong computing platforms, ineffective data storage structures, or bad networks and connections. These obstacles lower software engineers' productivity and effectiveness, which affects the end result.

4. **Understanding the large and complex system requirements is difficult**

We are all aware that engineers do not communicate with clients directly because clients are contacted through the project manager or bidder procedure. As a result, engineers face challenges when dealing with customers who have personal ideas about the software they want to use because they rarely have direct contact with them. It is one of the key challenges in software engineering.

We all know that practically every development project requires pre-production and testing; therefore, the same problem arises when someone works on an unproven project. When working on a complicated project, managing your time and focusing on each component might be challenging.

5. **Validating and tracing requirements**

Project requirements that are constantly changing make it harder for software engineers to work. Before beginning the implementation phase, it is crucial to double-check the list of requirements. Additionally, both forward and backward tracing should be possible.

6. Validating and tracing requirements

Project requirements that are constantly changing make it harder for software engineers to work. Before beginning the implementation phase, it is crucial to double-check the list of requirements. Additionally, both forward and backward tracing should be possible.

11. TESTING AND EVALUATION

11.1 TESTING STRATEGY: The testing stage of the device or program is the post deployment process in which the application is used on the server, and in response to high traffic, its reliability and consistency are tested. Testing is the way we figure out our processes, apps and applications' mistakes and the troubleshoot.

Assistance in testing the product standard: Tests are conducted here to detect any lap hole in the app that could affect its proper functioning. Testing mostly aims at removing conflicts. The program runs quickly and efficiently after removing disputes. There are now a host of methodologies for analyzing goods manufactured in industries. There are some research instruments that are used in the field of science to get a much better outcome. Through using these methods, the flaws and loop holes of the product can be readily detected and certain errors and defects removed. Testing can be seen in the following

spheres:

1. **Unit Testing:**

    - Test individual modules or components of the system in isolation to ensure they function as intended by testing the codes

2. **Integration Testing:**

    - Verify that different modules and components of the today's weather application work together seamlessly.

    - Test data flow and interactions between various parts of the system.

3• **System Testing**: It is done when the entire project is completed and ready to be tested for the functioning of the system thus it targets the functionality of the application.

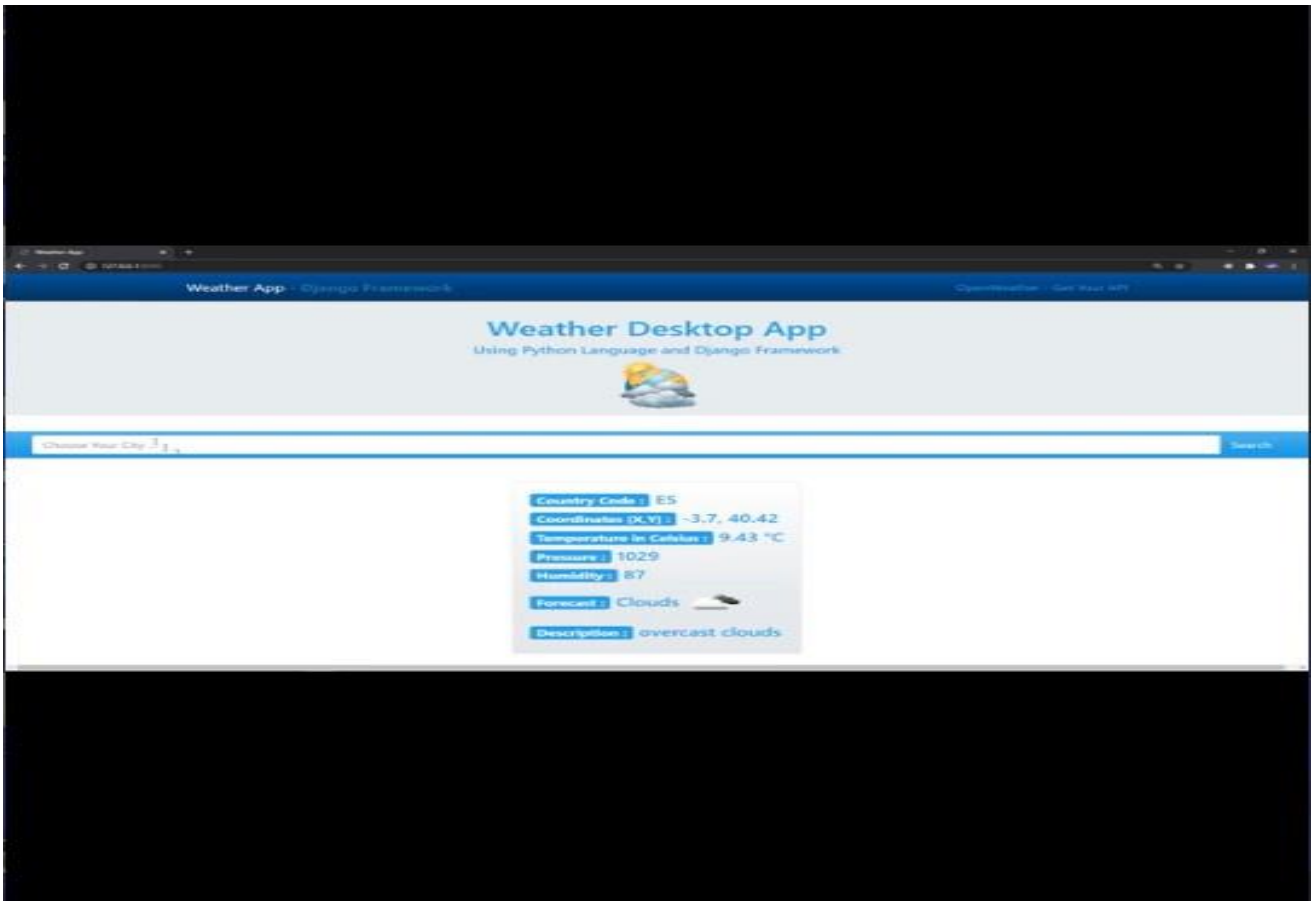
11.2 TEST CASES:

11.2.1 What is a Test Case?

A test case is the set of actions executed on a software application as part of the testing process to validate its features and functionality. In other words, it is a detailed description of a specific test that helps to execute a test successfully. Executing testing cases is essential to the software development process, as it helps ensure the application's quality and reliability.

A. Functional test case: Functionality Test Case: A functional testing case is used to find whether the function of the application's interface is in line with its users. In other words, with a functional testing case, you can identify the success or failure of the expected function of the software.

After the code had been written and run, this is the result so as to confirm the functionality of the app.



B. <u>User interface test case:</u> The User Interface (UI) testing case verifies the visuals and expected function of the Graphical User Interface. The testing and design teams are involved in writing user interface testing cases.

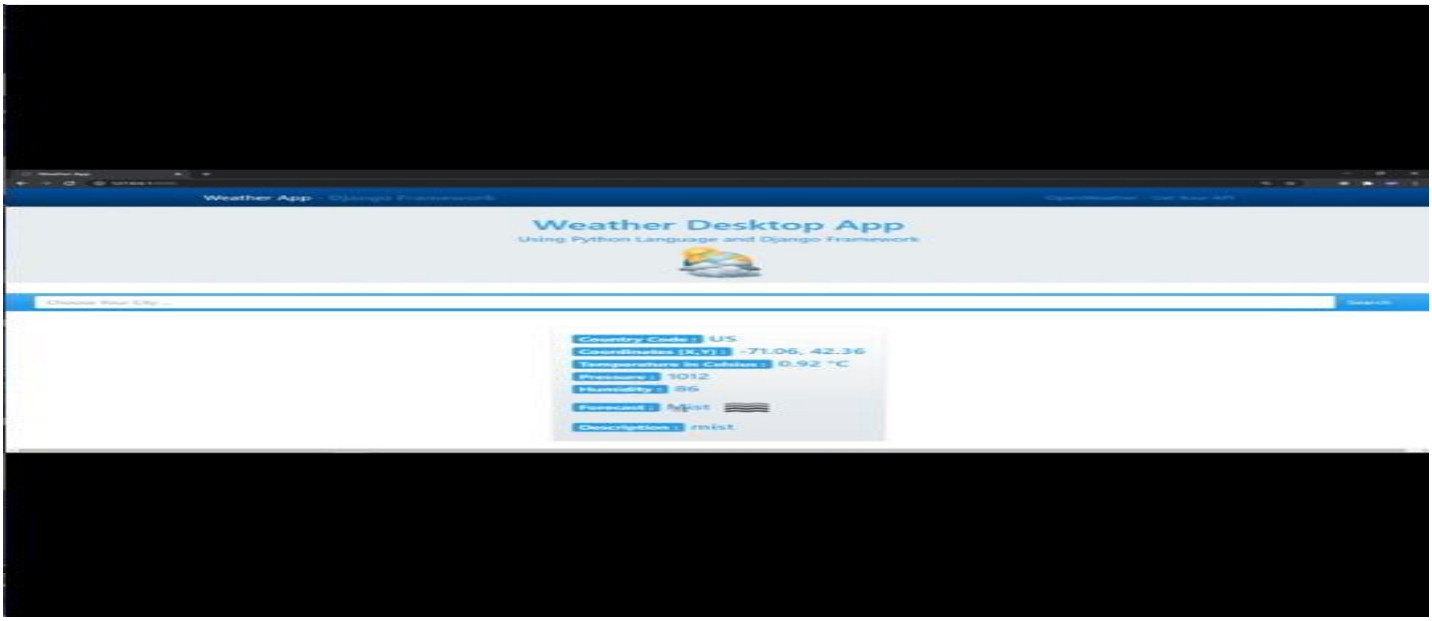
When this code which tested the visibility of the graphics was tested,

```
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/5.0/howto/static-files/

STATIC_URL = '/static/'
MEDIA_URL = '/images/'
```

```
STATICFILES_DIRS = (
    os.path.join(BASE_DIR, 'static'),
)
```

this is the result. It shows the Mist in the application.



C. <u>Performance test case:</u> Performance testing cases verify the functionality and response time of software applications.

The output of this code shows its better performance that will meet the purpose of its creation.

```
<DOCTYPE html>
<html lang="en">
{% load static %}
```

```
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scales=1.0"/>
    <link rel="stylesheet" href="<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bootswatch/5.3.3/cerulean/bootstrap.min.css" integrity="sha512-vaImhtQoaCXvevCM/UK+8ND/df0kiQWLtR65wgq0AWShEXLpWHMve0oGgU0q1gq1MbbyEuAhMMlqNmZd7s7wTQ==" crossorigin="anonymous" referrerpolicy="no-referrer"
    integrity="sha512-dQLT/B7byn2LjN/DN4zeBKpwGVGqbidV0XiMRWQOL7TGrV7FK2FldkGG+DGMU+CQnMTcRZlUI7GMWtlj6akNew=="
    crossorigin="anonymous" />
<!-- <link rel="stylesheet" href="(% static 'weatherproject/main/templates/static/style.css' %)" /> -->
 <title>Weather App </title>
 </head>

 <body>
    <div class="navbar navbar-expand-lg fixed-top navbar-dark bg-dark">
        <div class="container">
            <a href="../" class="navbar-brand">Weather App <spam style="color: rgb(39, 117, 161);"><strong> - Django
```

```
            Framework</strong>
          </spam>spam> </a>
        <a href="https://openweathermap.org" class="navbar-
tech">Openweather - Get Your API </a>
      </div>
   </div>


<br /><br /> <br>
<div id="jumbotrom" class="jumbotro" style="text-align: center; margin-
top:-s0px">
   hi class=display-s>Weather Desktop App </hi>
   <hs>Using python Language and Django Framework</hs>


   <img src="(% static 'images/weather.png' %)" class="image"
style="width:100px; margin-bottom:-50px; ">
   </div>


   <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
      <form method="post" class="col-md"">
         {% csrf_token %}
         <div class="input-group">
            <button type="submit" class="btn btn-
primary">search</button>
```

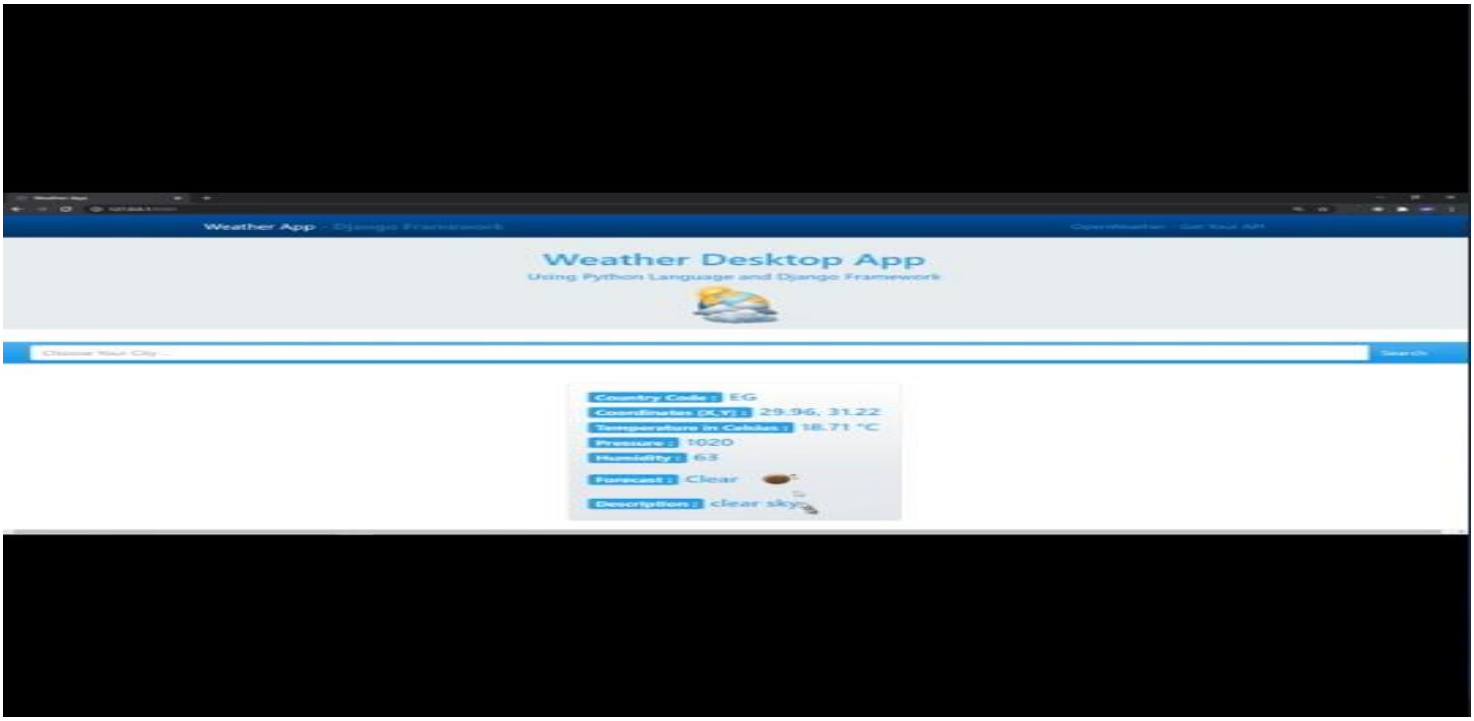
```
        </div>
        </div>
        <from>
        </nav>
        <br> <br>
        <div class="row">
          {%if country code and coordinate and temp and pressure and humidity%}
          <div class="col d-flex justify-content-center" >
            <div class="card text-white bg-light mb-6">
            <div clss=" card-body">
              <h4><spam class="badge badge-primary">Country code :</spam> {{country_code}}</h4>
              <h4><spam class="badge badge-primary">Coordinates [X,Y] :</spam> {{coordinate}}</h4>
              <h4><spam class="badge badge-primary">Temperature in celsius :</spam> {{temp}}</h4>
              <h4><spam class="badge badge-primary">Humidity :</spam> {{humidity}}</h4>
              <h4><spam class="badge badge-primary">Forecast :</spam> {{main}} <img
                  src="http://openweathermap.org/img/w/{{icon}}.png" alt="image" style="width:70px"></h4>
```

```
                <h4><spam class="badge badge=primary">Description :
</spam> {{description}}</h4>


            </div>
            {% endif %}
        </div>
    </body>

    </html>
```

This is the result:



11.3 Evaluation:

A. Performance: The system responded timely when it handles large amount of data under normal and peak load condition.

B. Usability: Feedback gathered from the end users gives efficient, effectiveness and update that should be done to the system.

Information was also gathered on how the system can support forcast under time pressure.

C. Functionality: The system provides accurate weather forcast as it gives alert, displays the humidity (visualization).

11.4 Limitations:

1. **Data Accuracy and Reliability**:
    - Weather predictions heavily rely on accurate data from various sources (satellites, weather stations, etc.). If the data is flawed or incomplete, the forecasts may be less reliable.
    - Historical data quality affects long-term predictions. Inaccuracies in historical records can propagate into future forecasts.
2. **Model Complexity and Computation Time**:
    - Sophisticated weather models involve complex mathematical equations and simulations. These models require significant computational resources and time.
    - Balancing model complexity with real-time predictions is a challenge.

3. **Local Variability and Microclimates**:
    - Weather conditions can vary significantly within a small geographic area. Models may struggle to capture microclimates accurately.
    - Urban heat islands, coastal effects, and local topography impact weather patterns.
4. **Short-Term vs. Long-Term Predictions**:
    - Short-term forecasts (hours to a few days) tend to be more accurate than long-term ones (weeks or months).
    - Long-term climate predictions face additional uncertainties due to natural variability and external factors (e.g., volcanic eruptions).
5. **Extreme Events and Rare Phenomena**:
    - Predicting rare events (e.g., tornadoes, hurricanes) remains challenging due to their infrequency and complex dynamics.
    - Extreme weather events can defy even the best models.
6. **User Expectations and Communication**:
    - Users often expect pinpoint accuracy, but weather forecasts inherently involve probabilities.
    - Communicating uncertainty effectively to users is crucial.

12. Deployment.

12.1 Deployment Environment: The system was deployed in local server (server operating system) windows environment via APIs.

12.2 Deployment Process: Firstly, the following tools are needed: VS code, Install python, Django.

This is the process as follows. Below, I'll outline the process, including setting up the project, integrating the OpenWeatherMap API, and deploying the app.

1. **Setting Up the Project.**
   - Install Django using pipenv:

mkdir the_weather_env

cd the_weather_env

pipenv install django

pipenv install requests

pipenv shell

   - **Create a new Django project:**

django-admin startproject the_weather

cd the_weather

python manage.py runserver

visit http://127.0.0.1:8000 in the browser to verify that Django is properly set up.

2. **Logging into the Admin Dashboard**
   Migrate the database with this code *python manage.py migrate.*

_ start the development server again: *python manage.py runserver*

Visit *and log in with the superuser credentials (create one using python manage.py createsuperuser if needed).*

3. **Creating the Weather App:**
   - Create a new app within your project: *python manage.py startapp weather*

Design the user interface using Bootstrap or another css framework.

4. Integrating OpenWeatherMap API:

- Sign up for an API key on the OpenWeatherMap website.

- In Django app, I created a form to input the city name.
- I use the Requests library to fetch weather data from the API based on the city name.
- Display the weather information on the app.

5. **Deploying the App:**
    - Choose a hosting platform.
    - Set up your environment variables (including the API key).
    - Deployed Django app to the chosen platform.

12.3 Configuration Management:

- Environment variables: Sensitive information like API Key was stored in environment variables. I used python OS module to access the variables within Django app.

Setting Module: A separate *setting.py*

Version Control:

Git: I used Git for version control.

12.4 Post-Deployment Monitoring:

After the implementation of the application, the testing took place. The app gets updated whenever the user faces a problem or have to be more flexible. The progress had been done by us.

- Monitor key metrics such as response time, error rates, CPU/memory usage, and database queries was set up. custom alerts for critical thresholds were also created.
- Logs and Error Tracking:

Python Module as structured logging was implemented. Issues will be promptly addressed.

. For health check, health endpoints were created to verify app availability.

. To ensure responsiveness, automated check was set up.

. Regular test will be taken on the app's scalability

13. Conclusion

13.1 Summary of Work

In conclusion, building a weather forcasting application involves several phases including team formation, requirement gathering, coding, testing and project presentation. Many aspects have been covered to develop this application so that it will provide users with real time weather details for the specific cities including temperature, wind, speed, humidity and weather descriptions. By integrating APIs like openweathermap, users are empowered to make informed decisions.

13.2 Contributions

As technology advances, the project will contribute immensely to the users. Few out of the contribution this project will bring to the users are as follow.

1. **Real-Time Weather Information**: Users can access up-to-date weather data for specific locations. This includes details such as temperature, humidity, wind speed, and precipitation forecasts.

Having accurate and timely information helps users plan their activities accordingly.

2. **Travel Planning**: Whether users are planning a weekend getaway or a business trip, a weather app assists them in making informed decisions. They can check the weather conditions at their destination and pack accordingly.
3. **Outdoor Activities**: Outdoor enthusiasts benefit from weather forecasts. Cyclists, hikers, runners, and other sports enthusiasts can plan their activities based on weather predictions. For example, they can avoid outdoor activities during heavy rain or extreme temperatures.
4. **Health and Safety**: Weather conditions impact health and safety. Users can receive alerts about severe weather events (storms, heatwaves, etc.) and take necessary precautions. For instance, they might stay indoors during a thunderstorm or avoid strenuous exercise during extreme heat.
5. **Agriculture and Farming**: Farmers rely on weather forecasts for crop management. Knowing when to plant, irrigate, or protect crops from frost is crucial. A weather app provides valuable insights for agricultural planning.
6. **Event Planning**: Event organizers can use weather data to plan outdoor events. They can choose suitable dates and times based on expected weather conditions. For instance, a music festival organizer might avoid scheduling an outdoor concert during heavy rainfall.
7. **Energy Efficiency**: Users interested in energy conservation can optimize their heating, cooling, and lighting systems based on

weather forecasts. For example, adjusting thermostat settings based on temperature predictions can lead to energy savings.

8. **Business Decisions**: Businesses in sectors like retail, construction, and logistics rely on weather information. Retailers stock seasonal products, construction companies plan outdoor projects, and logistics companies optimize routes based on weather conditions.

13.3 Future work

- Integration of future weather prediction using Machine learning technologies.
- Integration of any natural disaster prediction in the location.
- Integration of local time and international time in that location, we can add this using any api.
- We can add top weather headlines in the nearby locations as well.
- We can add last 10 days' time line of weather forecasting of the selected location as well.

14. REFERENCES

1. *Building a Complete Weather App from Scratch with HTML, CSS, and JavaScript: A Step-by-Step Guide by Odumosu Matthew*
2. *Articles, videos, and documentation to help software engineers get the most from Visual Crossing Weather. By Visual Crossing Weather.*
3. *Bek Brace Tutorial Class (Dec 22, 2020).*
4. DR.C.K.Gomathy , V.Geetha , S.Madhumitha , S.Sangeetha , R.Vishnupriya Article: A Secure With Efficient Data Transaction In Cloud Service, Published by International Journal of Advanced Research in Computer

Engineering & Technology (IJARCET) Volume 5 Issue 4, March 2016, ISSN: 2278 – 1323.

5. [2] Dr.C.K.Gomathy,C K Hemalatha, Article: A Study On Employee Safety And Health Management International Research Journal Of Engineering And Technology (Irjet)- Volume: 08 Issue: 04 | Apr 2021
6. Dr.C K Gomathy, Article: A Study on the Effect of Digital Literacy and information Management, IAETSD Journal For Advanced Research In Applied Sciences, Volume 7 Issue 3, P.No-51-57, ISSN NO: 2279-543X,Mar/2018

7. Dr.C K Gomathy, Article: An Effective Innovation Technology In Enhancing Teaching And Learning Of Knowledge Using Ict Methods, International Journal Of Contemporary Research In Computer Science And Technology (Ijcrcst) E-Issn: 2395-5325 Volume3, Issue 4,P.No-10-13, April '2017

8. Dr. C. K. GOMATHY1, U. SOMA RAJU CHOWDARY2, V. MANJUNATH REDDY3, Dr. V GEETHA4. 1, 4 -Assistant Professor, Department of CSE, SCSVMV University, Kanchipuram, Department of CSE, SCSVMV University, Kanchipuram. Article on **WEATHER FORECASTING APPLICATION USING PYTHON**

9. An Eye to the Sky: Describing Char o the Sky: (Describing Characteristics of Weather App Weather App Users Through Q Method) By Danielle Wardinsky Hallows Brigham Young University in 2022-03-31.

10. *Android Beginners Guide To Create A Weather Forecast App" on Udemy. This course teaches you how to create a weather forecast app using the OpenWeatherMap API and the Volley Library.*
11. *"Build a Full-Stack Weather App Under 60 Minutes". This tutorial walks you through creating a full-stack weather app using HTML, CSS, JavaScript, Node.js, and Express.js. It also integrates the OpenWeather API for real-time weather data.*
12. *"Design and Development of a Weather Forecasting Android Application". This book discusses the design and development of a weather forecasting app for Android, including unique features not found in other apps*

15. Appendices

SQL:

This is the sql code/tables for the weather forcast application created for daily weather condition.

```
-- MySQL dump 10.13  Distrib 8.0.38, for Win64 (x86_64)
--
-- Host: localhost    Database: weatherapp
-- ------------------------------------------------------
-- Server version       8.0.39

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!50503 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
```

```
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;


--
-- Table structure for table `cities`
--


DROP TABLE IF EXISTS `cities`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `cities` (
  `CityID` int NOT NULL,
  `CityName` varchar(255) DEFAULT NULL,
  `Country` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`CityID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;


--
-- Dumping data for table `cities`
--


LOCK TABLES `cities` WRITE;
```

```
/*!40000 ALTER TABLE `cities` DISABLE KEYS */;
INSERT INTO `cities` VALUES (1,'Seattle','United States'),(2,'Lagos','Nigeria'),(3,'Tokyo','Japan');
/*!40000 ALTER TABLE `cities` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Temporary view structure for view `vwcitydetails`
--

DROP TABLE IF EXISTS `vwcitydetails`;
/*!50001 DROP VIEW IF EXISTS `vwcitydetails`*/;
SET @saved_cs_client     = @@character_set_client;
/*!50503 SET character_set_client = utf8mb4 */;
/*!50001 CREATE VIEW `vwcitydetails` AS SELECT
 1 AS `CityID`,
 1 AS `CityName`,
 1 AS `Country`*/;
SET character_set_client = @saved_cs_client;

--
-- Temporary view structure for view `vwweatherdetails`
--
```

```
DROP TABLE IF EXISTS `vwweatherdetails`;
/*!50001 DROP VIEW IF EXISTS `vwweatherdetails`*/;
SET @saved_cs_client     = @@character_set_client;
/*!50503 SET character_set_client = utf8mb4 */;
/*!50001 CREATE VIEW `vwweatherdetails` AS SELECT
 1 AS `ConditionID`,
 1 AS `CityName`,
 1 AS `Date`,
 1 AS `Temperature`,
 1 AS `Humidity`,
 1 AS `WindSpeed`,
 1 AS `WeatherDescription`*/;
SET character_set_client = @saved_cs_client;


--
-- Table structure for table `weatherconditions`
--


DROP TABLE IF EXISTS `weatherconditions`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `weatherconditions` (
  `ConditionID` int NOT NULL,
  `CityID` int DEFAULT NULL,
```

```
  `Date` date DEFAULT NULL,
  `Temperature` decimal(5,2) DEFAULT NULL,
  `Humidity` decimal(5,2) DEFAULT NULL,
  `WindSpeed` decimal(5,2) DEFAULT NULL,
  `WeatherDescription` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`ConditionID`),
  KEY `FK_WeatherConditions_Cities` (`CityID`),
  CONSTRAINT `FK_WeatherConditions_Cities` FOREIGN KEY (`CityID`)
REFERENCES `cities` (`CityID`),
  CONSTRAINT `weatherconditions_ibfk_1` FOREIGN KEY (`CityID`) REFERENCES
`cities` (`CityID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;


--
-- Dumping data for table `weatherconditions`
--


LOCK TABLES `weatherconditions` WRITE;
/*!40000 ALTER TABLE `weatherconditions` DISABLE KEYS */;
INSERT INTO `weatherconditions` VALUES (1,1,'2024-08-
26',25.50,70.20,8.30,'Partly cloudy'),(2,2,'2024-08-
26',30.00,85.00,6.70,'Thunderstorms'),(3,3,'2024-08-
26',28.80,60.50,12.10,'Sunny');
/*!40000 ALTER TABLE `weatherconditions` ENABLE KEYS */;
```

```
UNLOCK TABLES;

--
-- Dumping events for database 'weatherapp'
--


--
-- Dumping routines for database 'weatherapp'
--


--
-- Final view structure for view `vwcitydetails`
--

/*!50001 DROP VIEW IF EXISTS `vwcitydetails`*/;
/*!50001 SET @saved_cs_client          = @@character_set_client */;
/*!50001 SET @saved_cs_results         = @@character_set_results */;
/*!50001 SET @saved_col_connection     = @@collation_connection */;
/*!50001 SET character_set_client      = utf8mb4 */;
/*!50001 SET character_set_results     = utf8mb4 */;
/*!50001 SET collation_connection      = utf8mb4_0900_ai_ci */;
/*!50001 CREATE ALGORITHM=UNDEFINED */
/*!50013 DEFINER=`root`@`localhost` SQL SECURITY DEFINER */
```

```
/*!50001 VIEW `vwcitydetails` AS select `c`.`CityID` AS `CityID`,`c`.`CityName` AS `CityName`,`c`.`Country` AS `Country` from `cities` `c` */;
/*!50001 SET character_set_client      = @saved_cs_client */;
/*!50001 SET character_set_results     = @saved_cs_results */;
/*!50001 SET collation_connection      = @saved_col_connection */;

--
-- Final view structure for view `vwweatherdetails`
--

/*!50001 DROP VIEW IF EXISTS `vwweatherdetails`*/;
/*!50001 SET @saved_cs_client          = @@character_set_client */;
/*!50001 SET @saved_cs_results         = @@character_set_results */;
/*!50001 SET @saved_col_connection     = @@collation_connection */;
/*!50001 SET character_set_client      = utf8mb4 */;
/*!50001 SET character_set_results     = utf8mb4 */;
/*!50001 SET collation_connection      = utf8mb4_0900_ai_ci */;
/*!50001 CREATE ALGORITHM=UNDEFINED */
/*!50013 DEFINER=`root`@`localhost` SQL SECURITY DEFINER */
/*!50001 VIEW `vwweatherdetails` AS select `wc`.`ConditionID` AS `ConditionID`,`c`.`CityName` AS `CityName`,`wc`.`Date` AS `Date`,`wc`.`Temperature` AS `Temperature`,`wc`.`Humidity` AS `Humidity`,`wc`.`WindSpeed` AS `WindSpeed`,`wc`.`WeatherDescription` AS `WeatherDescription` from (`weatherconditions` `wc` join `cities` `c` on((`wc`.`CityID` = `c`.`CityID`))) */;
```

```
/*!50001 SET character_set_client      = @saved_cs_client */;
/*!50001 SET character_set_results     = @saved_cs_results */;
/*!50001 SET collation_connection      = @saved_col_connection */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2024-08-30 14:38:17
```

1. *Source Code : https://github.com/BekBrace/Weather-A...*
2. *Weather API : https://openweathermap.org/api*
3. *Bootswatch : https://cdnjs.com/libraries/bootswatch*
4. *Django URL : https://www.djangoproject.com*
5. *DEV profile : https://dev.to/bekbrace*
6. *Github profile : https://github.com/BekBrace*