

# **Operational Technology Driven Weather Website Using Python**

MINOR PROJECT REPORT

By,

**K CHARANYA (RA2211030010304)**

**M ASEENA SULTHANA (RA2211030010317)**

Under the guidance of

**Dr. Varun Kumar**

*In partial fulfilment for the Course*

of

**21CSC203P – ADVANCED PROGRAMMING PRACTICE**

**in Networking and Communications**



- **FACULTY OF ENGINEERING AND TECHNOLOGY**  
**SCHOOL OF COMPUTING**  
**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**  
**KATTANKULATHUR**

**NOVEMBER 2023**

# **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

(Under Section 3 of UGC Act, 1956)

## **BONAFIDE CERTIFICATE**

Certified that this minor project report for the course **21CSC203P**

**ADVANCED PROGRAMMING PRACTICE** entitled in "**Operational Technology Driven Weather Website Using Python**" is the bonafide work of **K CHARANYA (RA2211030010304)** and **M ASEENA SULTHANA (RA2211030010317)** who carried out the work under my supervision.

### **SIGNATURE**

**Dr. Varun Kumar K A**

**Assistant Professor**

Dept of Networking and Communications

SRM Institute of Science and Technology

Kattankulathur

## **ABSTRACT**

In this project, we aim to design and implement a weather website using Python, combining back end logic, data retrieval from weather APIs, and front end development. This project aims to develop a weather website using Python, integrating backend logic with a web framework like Flask or Django, data retrieval from a weather API, and frontend technologies such as HTML, CSS, and JavaScript. The website will offer real-time weather updates and forecasts, with features like user authentication, geolocation services, and personalized preferences. The project emphasizes visualization and data presentation using charting libraries, ensuring a user-friendly interface across various devices. Deployment considerations include utilizing platforms like Heroku or AWS for scalability, and thorough testing and validation will be conducted to ensure the accuracy and reliability of weather data. The combination of Python's versatility and web development tools results in a comprehensive weather website providing an engaging and informative user experience.

## ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors. We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy**, for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal**, for bringing out novelty in all executions. We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman**, for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professors Dr. Vadivu. G , Professor, Department of Data Science and Business Systems and Dr. Sasikala. E Professor, Department of Data Science and Business Systems and Course Coordinators** for their constant encouragement and support.

We are highly thankful to our my Course project Faculty **Dr. Varun Kumar K A , Assistant Professor , Department of Networking and Communication**, for his assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our HOD **Dr. Annapurani K, Professor , Department of Networking and Commication** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project.

Above all, I thank the almighty for showering his blessings on me to complete my Course project.

## **TABLE OF CONTENTS**

<b>CHAPTER..NO</b>	<b>CONTENTS</b>	<b>PAGE..NO</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>6</b>
	1.1 Motivation	
	1.2 Objective	
	1.3 Problem Statement	
	1.4 Challenges	
<b>2</b>	<b>LITERATURE SURVEY</b>	<b>9</b>
<b>3</b>	<b>REQUIREMENT ANALYSIS</b>	<b>9</b>
<b>4</b>	<b>ARCHITECTURE &amp; DESIGN</b>	<b>12</b>
<b>5</b>	<b>IMPLEMENTATION</b>	<b>19</b>
<b>6</b>	<b>EXPERIMENT RESULTS &amp; ANALYSIS</b>	<b>24</b>
<b>7</b>	<b>CONCLUSION</b>	<b>26</b>
<b>8</b>	<b>REFERENCES</b>	<b>27</b>

# **1. INTRODUCTION**

The "Operational Technology Driven Weather Website Using Python" project introduces a comprehensive solution for delivering real-time and forecasted weather information through the fusion of operational technology (OT) and Python-based web development. Leveraging the capabilities of Python, this initiative integrates a robust backend, possibly utilizing frameworks such as Flask or Django, with advanced data retrieval from weather APIs.

The project emphasizes the seamless integration of operational technologies to enhance user experience, incorporating features like automated geolocation detection, personalized user preferences, and dynamic data visualization. By combining these technologies, the project aims to create a sophisticated weather website that not only provides accurate and up-to-date meteorological data but also ensures scalability, reliability, and a user-friendly interface. The integration of operational technology sets this project apart, allowing for efficient data handling, optimal user experience, and the seamless delivery of critical weather information.

## **1.1 Motivation**

The motivation behind the "Operational Technology Driven Weather Website Using Python" project stems from the increasing need for a technologically advanced and user-centric platform to access weather information. Traditional weather websites often lack seamless integration of operational technologies, leading to potential gaps in data accuracy, user engagement, and overall functionality. By leveraging Python for web development and integrating operational technology, this project aims to address these shortcomings. The motivation is grounded in the desire to create a weather website that not only delivers precise and real-time meteorological data but also optimizes user interaction through features like automated geolocation services and personalized preferences.

## 1.2 Objective:

The three main objectives of the "Operational Technology Driven Weather Website Using Python" project are:

**Seamless Integration of Operational Technology:** Incorporate operational technology into the website's framework to enhance data accuracy, processing, and presentation.

**Enhanced User Experience and Personalization:** Prioritize user experience by implementing features such as automated geolocation, personalized preferences, and a dynamic interface for a customized user experience.

**Scalability and Reliability:** Ensure the scalability and reliability of the website by deploying it on robust server platforms, utilizing cloud services, and implementing thorough testing processes to guarantee accurate and dependable weather data.

## 1.3 Problem Statment

To address the limitations in current weather platforms, the "Operational Technology Driven Weather Website Using Python" project seeks to seamlessly integrate operational technology, thus improving real-time data accuracy, user personalization, and scalability. Existing weather websites often fall short in providing a dynamic and tailored user experience, lacking essential features such as automated geolocation detection and personalized preferences. This project aims to overcome these challenges by leveraging the capabilities of Python to create a technologically advanced weather website. The goal is to optimize operational technology integration, ensuring precise real-time data, enhanced user experiences, and the scalability required to meet evolving user demands for accurate and personalized weather information.

## 1.4 Challenges

The "Operational Technology Driven Weather Website Using Python" project faces several challenges that need to be addressed for successful implementation:

1. **Integration Complexity:** Integrating operational technology seamlessly into the website's architecture poses a challenge, requiring a robust framework to handle diverse data sources, real-time updates, and ensure efficient processing.

2. **Data Accuracy and Reliability:** Ensuring the accuracy and reliability of weather data, especially in real-time scenarios, is a significant challenge. Addressing potential discrepancies from weather APIs and maintaining data integrity throughout the platform is crucial.

3. **User Privacy and Security:** Implementing personalized features, such as geolocation and user preferences, raises concerns about privacy and data security. Safeguarding user information and implementing secure authentication mechanisms is paramount.

4. **Scalability Issues:** Achieving scalability to handle varying levels of user traffic and data demands requires careful consideration. Deploying the website on scalable infrastructure and optimizing performance is crucial for a seamless user experience.

5. **Dynamic Data Visualization:** Implementing dynamic and visually appealing data visualization for weather trends and forecasts can be challenging. Balancing aesthetics with functionality and ensuring optimal performance is key.

6. **User Engagement and Experience:** Creating an engaging and user-friendly interface that caters to diverse user needs is challenging. Striking the right balance between feature-rich functionalities and a clean, intuitive design requires thoughtful consideration.



## 2. LITERATURE SURVEY

App Name	Features	Key Findings	References
1. <b>Windy.com</b>	<ul style="list-style-type: none"><li>- Interactive Maps</li><li>- Global Coverage</li><li>- Forecast Models</li></ul>	<ul style="list-style-type: none"><li>- Visual Appeal</li><li>- Global Weather</li></ul>	<a href="#">Windy: Wind map &amp; weather forecast</a>
2. <b>AccuWeather</b>	<ul style="list-style-type: none"><li>- MinuteCast</li><li>- Severe Weather Alert</li><li>- Radar &amp; Satellite Imagery</li></ul>	<ul style="list-style-type: none"><li>- Accurate Forecasts</li><li>- User Friendly Interface</li></ul>	<a href="#">India Current Weather   AccuWeather</a>
3. <b>Skymet</b>	<ul style="list-style-type: none"><li>- Localized Forecasting</li><li>- Agricultural Weather Services</li></ul>	<ul style="list-style-type: none"><li>- Regional Focus</li><li>- Agricultural Insights</li></ul>	<a href="#">Weather Forecast   Weather in India and World   Skymet Weather</a>

## REQUIREMENTS

### Development Environment:

#### 1. Python:

- Version: 3.x
- Description: The core programming language for backend development integration with Python-based libraries and frameworks.

#### 2. Web Framework:

- Choice between Flask or Django for backend development.
- Flask: Lightweight and modular.
- Django: Batteries-included framework with extensive built-in features.

#### 3. Database Management System:

- SQLite, PostgreSQL, or MySQL for data storage.
- SQLite for lightweight development.
- PostgreSQL or MySQL for scalable production environments.

#### **4. Frontend Technologies:**

- HTML, CSS, JavaScript for building the user interface.
- Optionally, consider using frontend frameworks like React, Vue.js, or Angular for dynamic UI components.

#### **5. Version Control:**

- Git for version control.
- GitHub, GitLab, or Bitbucket for repository hosting.

### **Weather Data Integration:**

#### **1. Weather API:**

- Choose a reliable weather API such as OpenWeatherMap, Weatherbit, or others for real-time weather data retrieval.

### **Data Visualization:**

#### **1. Charting Libraries:**

- Matplotlib, Plotly, or similar Python libraries for dynamic data visualization.

### **Deployment and Hosting:**

#### **1. Cloud Services:**

- AWS (Amazon Web Services), Heroku, or another cloud provider for scalable and reliable deployment.

#### **2. Web Server:**

- Gunicorn or uWSGI for deploying the Python web application.

## **Hardware Requirement**

#### **1. Computer:**

- A standard computer with a minimum of 8 GB RAM.
- Multi-core processor for faster compilation and testing.

#### **2. Storage:**

- Adequate free disk space for the development environment, libraries, and tools.

#### **3. Operating System:**

-Windows, macOS, or Linux-based operating system, depending on developer preference.

**4. Integrated Development Environment (IDE):**

-A Python IDE such as Pycharm , VS Studio etc.

**5. Network Connectivity :**

- Strong Network Connectivity to Connect with APIs.

**Server (if App Requires Server-Side Components):**

**1. Server:**

- A dedicated server or cloud hosting (e.g., AWS, Azure) for hosting server-side components, databases, and APIs.

**2. Database:**

- Database server (e.g., MySQL, PostgreSQL) for storing user data and other relevant information.

**3. Secure Connection:**

- SSL/TLS certificates for secure data transmission between the app and the server.

**Testing Devices:**

**1.Device Diversity:**

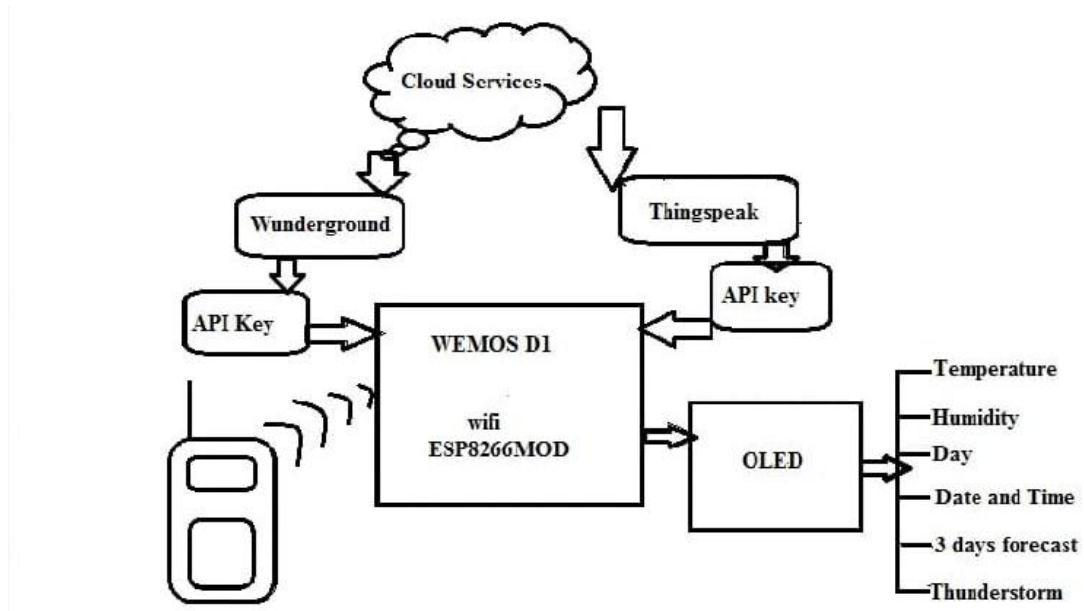
- Test the app on a variety of devices to ensure compatibility and optimal performance across different screen sizes, resolutions, and hardware specifications.

**2.Emulators:**

- Utilize emulators for testing on different Android or iOS versions and device configurations.

## ARCHITECTURE AND DESIGN

### Network Architecture

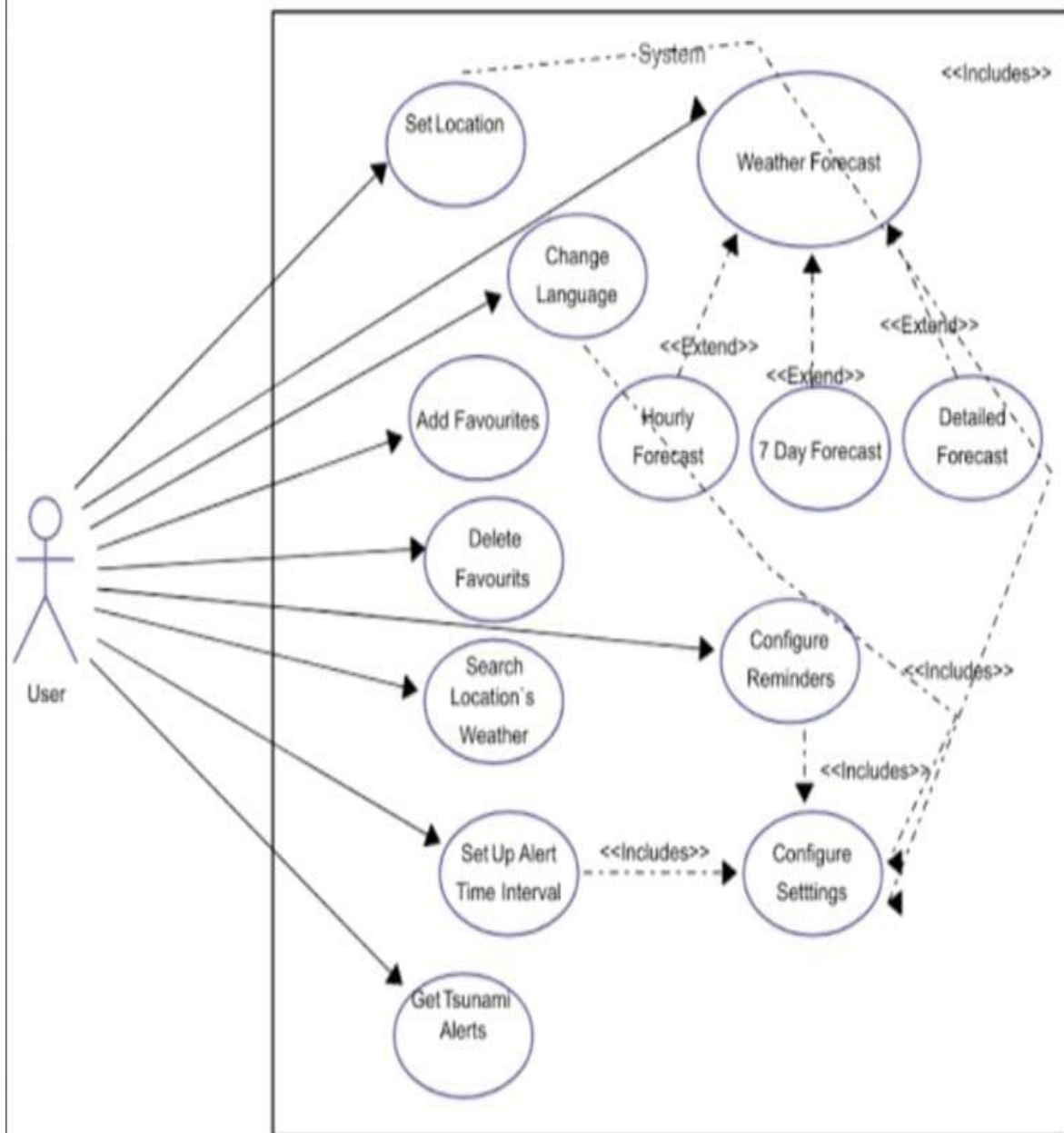


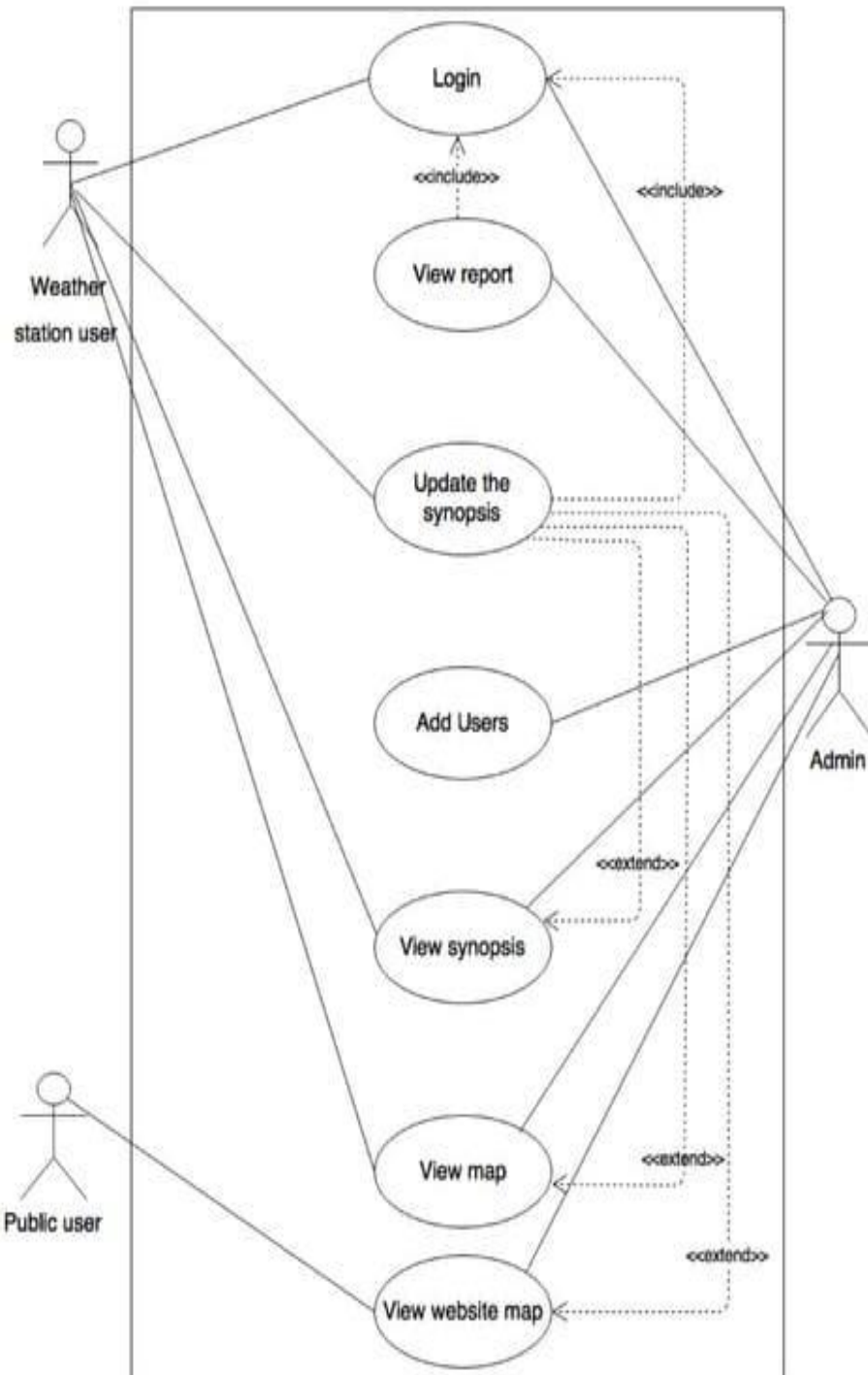
The architecture consists of three major networks:

- **User Network(s)**
- **API Key**
- **Cloud Service**

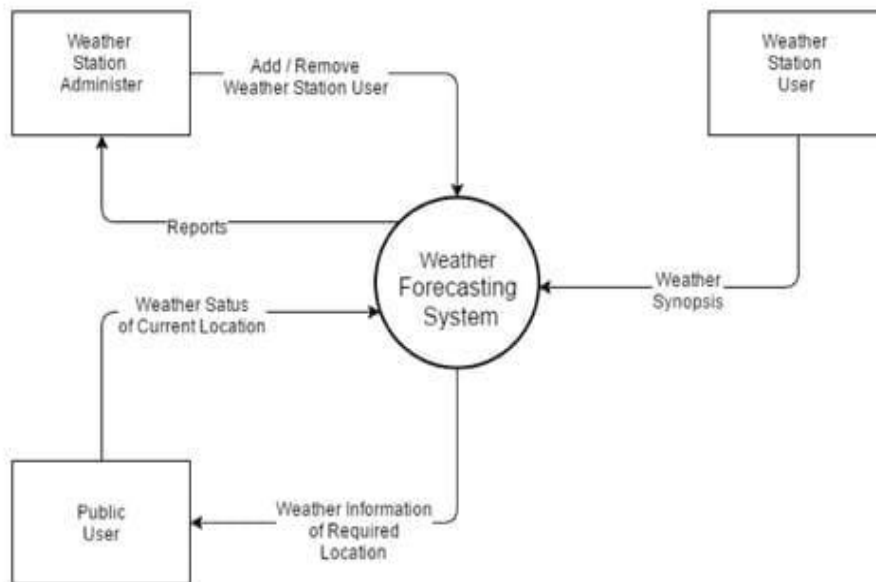
These networks are interconnected with each other with varying degrees (discussed in the implementation chapter).

## USE CASE DIAGRAM

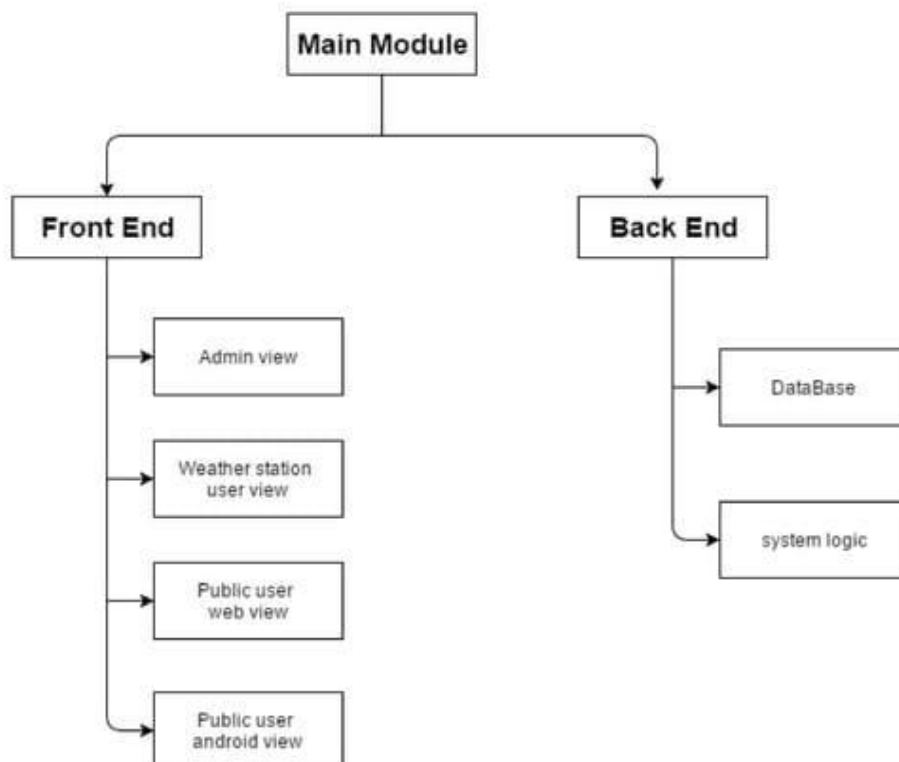




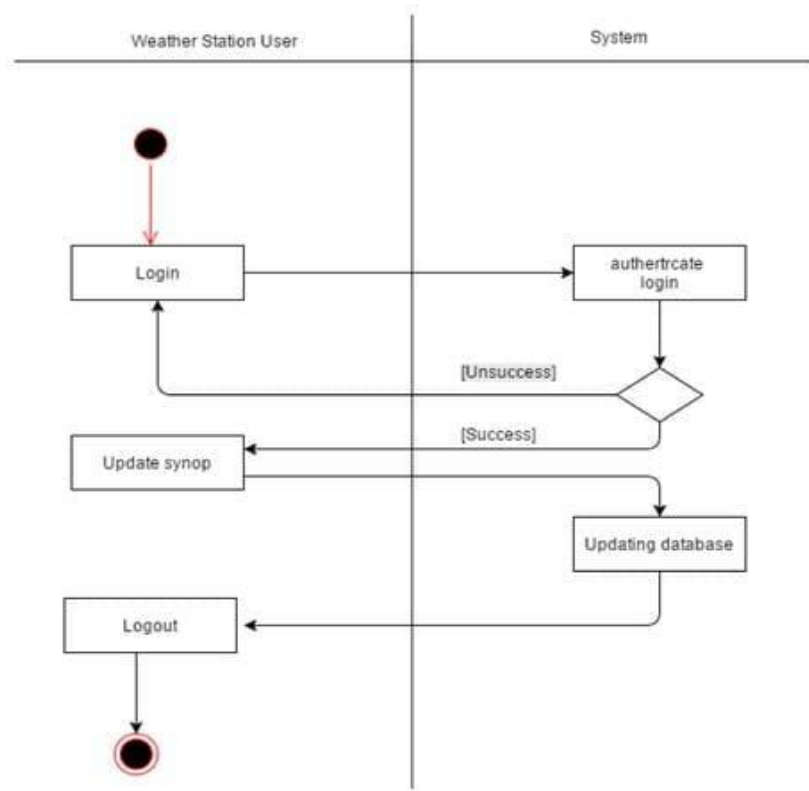
## SYSTEM DIAGRAM



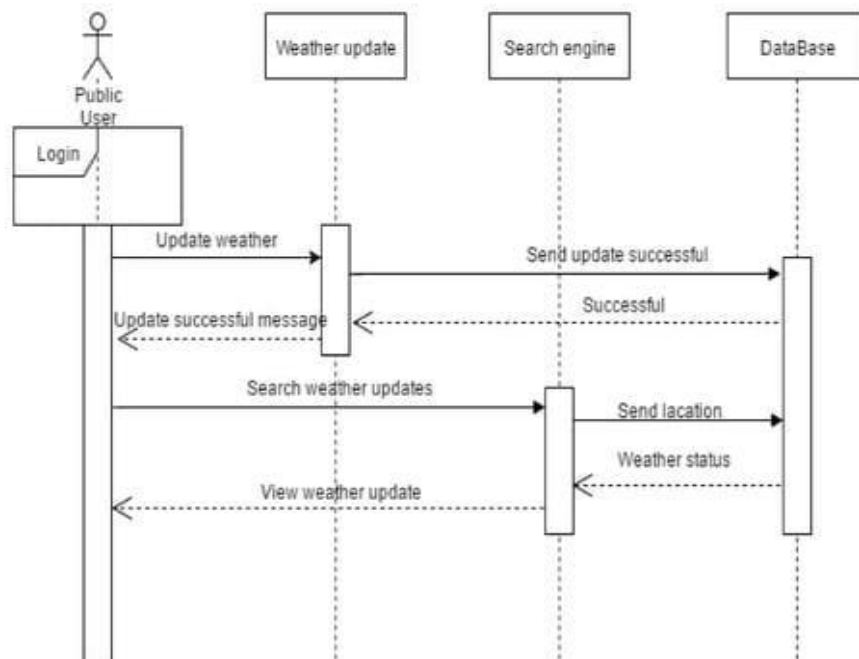
## TOP LEVEL DESIGN DIAGRAM



## ACTIVITY DIAGRAM

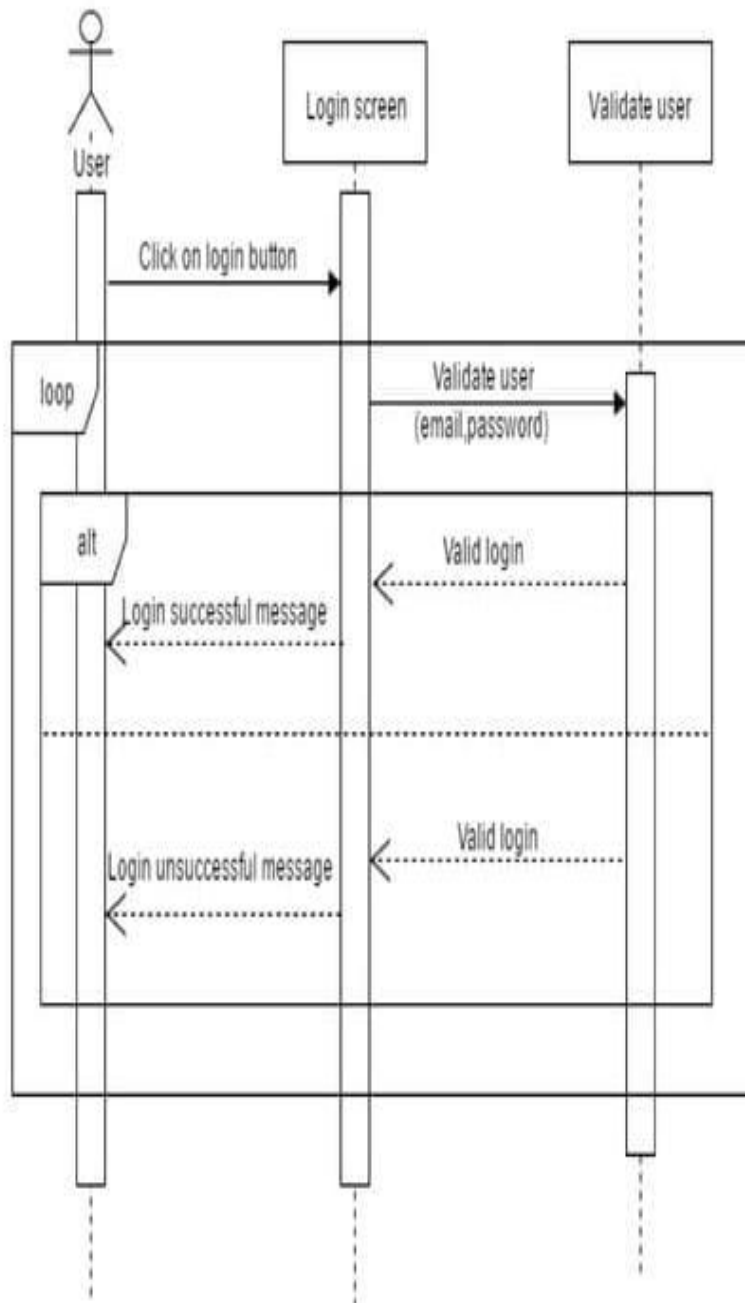


## SEQUENCE DIAGRAM

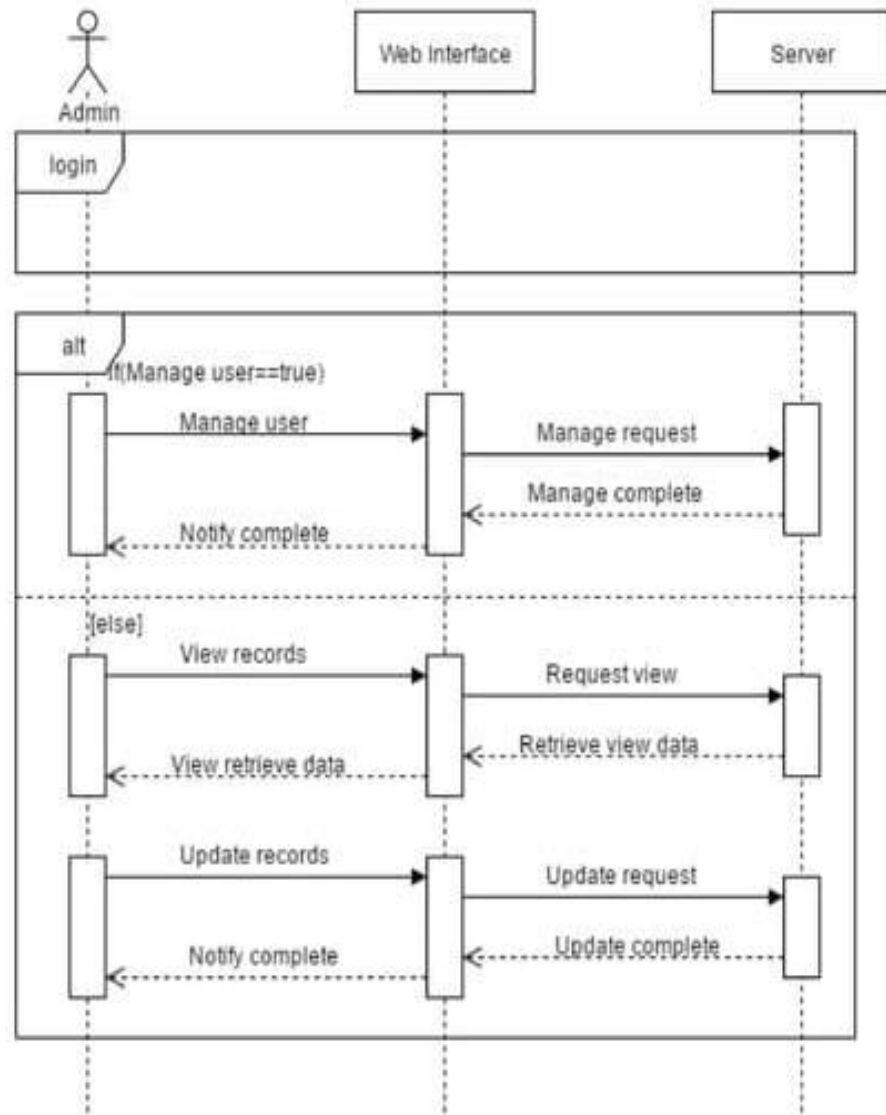




## SEQUENCE DIAGRAM FOR LOGIN



## SEQUENCE DIAGRAM FOR ADMIN



## IMPLEMENTATION

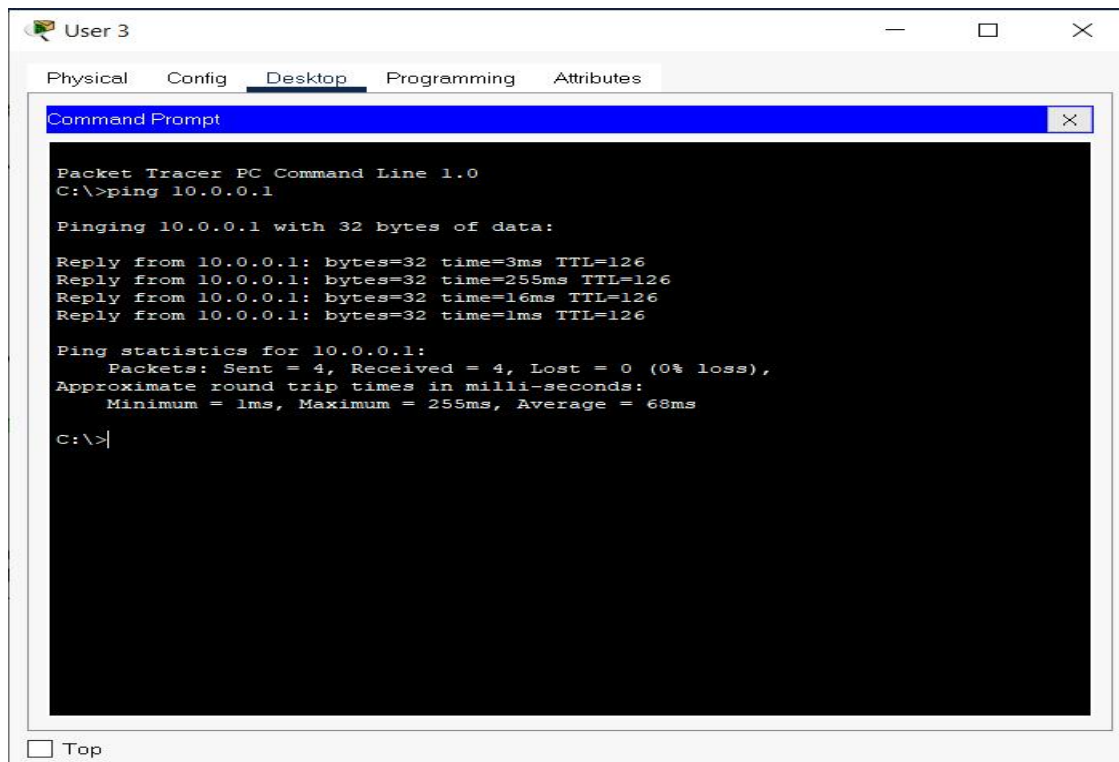
### Address Table :

The address table is as follows:

Device	Interface	Address
User Server	Fa0	172.16.0.2
Public Router	Fa0/0	172.16.0.1
	Fa1/0	192.16.0.1
	Se2/0	10.0.0.1
Computer	Fa0/0	192.16.0.2 to 192.16.0.7
Weather station Router	Se2/0	10.0.0.2
	Fa0/0	192.168.10.1
Public PC	Fa0/0	192.168.10.2 to 192.168.10.4

### Connection Check :

The network connections were checked by ping requests:



The screenshot shows a Packet Tracer PC Command Line window for 'User 3'. The window has tabs for Physical, Config, Desktop, Programming, and Attributes. The Desktop tab is active, displaying a Command Prompt window. The Command Prompt shows the following text:

```
Packet Tracer PC Command Line 1.0
C:\>ping 10.0.0.1

Pinging 10.0.0.1 with 32 bytes of data:

Reply from 10.0.0.1: bytes=32 time=3ms TTL=126
Reply from 10.0.0.1: bytes=32 time=255ms TTL=126
Reply from 10.0.0.1: bytes=32 time=16ms TTL=126
Reply from 10.0.0.1: bytes=32 time=1ms TTL=126

Ping statistics for 10.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 255ms, Average = 68ms

C:\>
```

At the bottom of the Command Prompt window, there is a checkbox labeled 'Top' which is currently unchecked.

### **SOURCE CODE :**

```
from tkinter import *
import requests
import json
from datetime import datetime

#Initialize Window

root =Tk()
root.geometry("400x400") #size of the window by default
root.resizable(0,0) #to make the window size fixed
#title of our window
root.title("Weather App - AskPython.com")


#Functions to fetch and display weather info
city_value = StringVar()

def time_format_for_location(utc_with_tz):
    local_time = datetime.utcfromtimestamp(utc_with_tz)
    return local_time.time()

city_value = StringVar()

def showWeather():
    #Enter you api key, copies from the OpenWeatherMap dashboard
    api_key = "eda2b2s6d#sd65f4de7c4b8" #sample API
```

```

# Get city name from user from the input field (later in the code)
city_name=city_value.get()

# API url
weather_url = 'http://api.openweathermap.org/data/2.5/weather?q=' + city_name +
'&appid='+api_key

# Get the response from fetched url
response = requests.get(weather_url)

# changing response from json to python readable
weather_info = response.json()

tfield.delete("1.0", "end") #to clear the text field for every new output

#as per API documentation, if the cod is 200, it means that weather data was
successfully fetched

if weather_info['cod'] == 200:
    kelvin = 273 # value of kelvin

#Storing the fetched values of weather of a city

temp = int(weather_info['main']['temp'] - kelvin)
#converting default kelvin value to Celcius
feels_like_temp = int(weather_info['main']['feels_like'] - kelvin)

```

```

pressure = weather_info['main']['pressure']
humidity = weather_info['main']['humidity']
wind_speed = weather_info['wind']['speed'] * 3.6
sunrise = weather_info['sys']['sunrise']
sunset = weather_info['sys']['sunset']
timezone = weather_info['timezone']
cloudy = weather_info['clouds']['all']
description = weather_info['weather'][0]['description']

sunrise_time = time_format_for_location(sunrise + timezone)
sunset_time = time_format_for_location(sunset + timezone)

```

**#assigning Values to our weather variable, to display as output**

```

weather = f"\nWeather of: {city_name}\nTemperature (Celsius): {temp}°\nFeels
like in (Celsius): {feels_like_temp}°\nPressure: {pressure} hPa\nHumidity:
{humidity}%\nSunrise at {sunrise_time} and Sunset at {sunset_time}\nCloud:
{cloudy}%\nInfo: {description}"

```

**else:**

```

weather = f"\n\tWeather for '{city_name}' not found!\n\tKindly Enter valid City
Name !!"

```

```

tfield.insert(INSERT, weather)

```

**#to insert or send value in our Text Field to display output**

**#-Frontend part of code - Interface**

```

city_head= Label(root, text = 'Enter City Name', font = 'Arial 12 bold').pack(pady=10)

#to generate label heading

```

```
inp_city = Entry(root, textvariable = city_value, width = 24, font='Arial 14 bold').pack()
```

```
Button(root, command = showWeather, text = "Check Weather", font="Arial 10",  
bg='lightblue', fg='black', activebackground="teal", padx=5, pady=5 ).pack(pady= 20)
```

```
#to show output
```

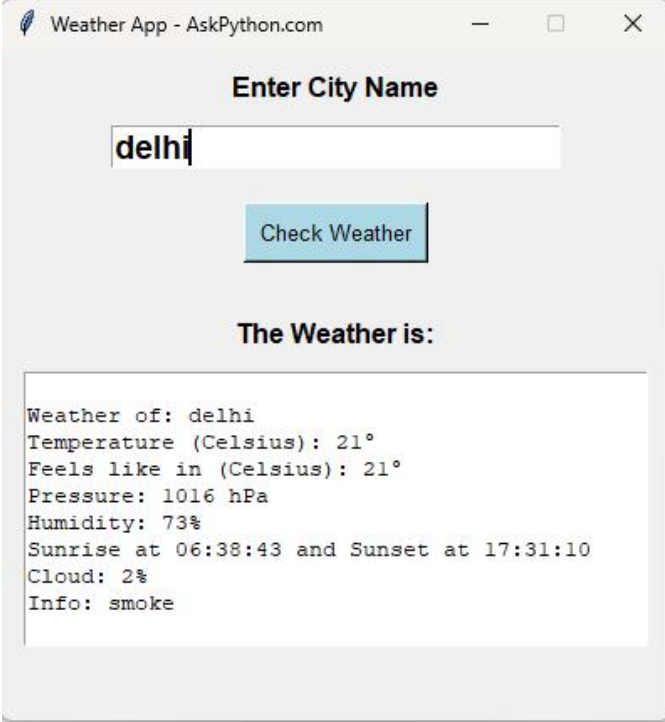
```
weather_now = Label(root, text = "The Weather is:", font = 'arial 12  
bold').pack(pady=10)
```

```
tfield = Text(root, width=46, height=10)  
tfield.pack()
```

```
root.mainloop()
```

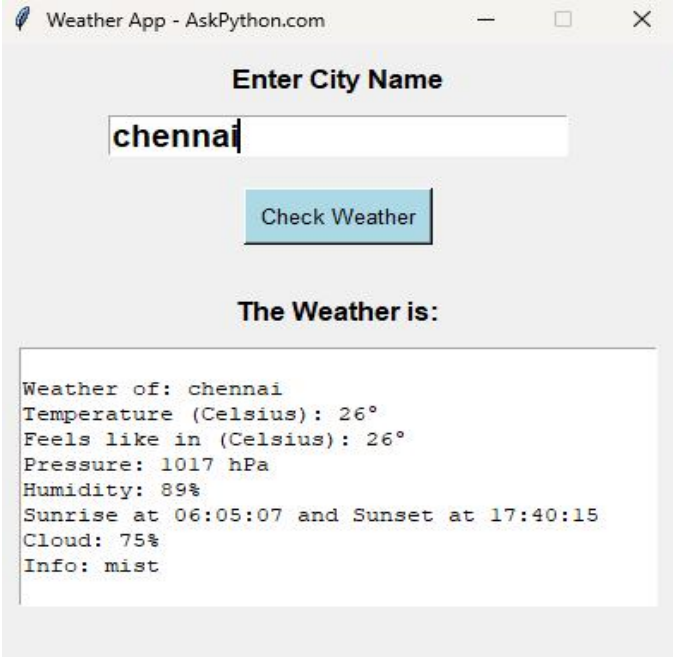
## RESULT AND DISCUSSION

### RESULT :



The screenshot shows a web browser window titled "Weather App - AskPython.com". The page has a light gray background. At the top, it says "Enter City Name". Below this is a text input field containing the word "delhi". Under the input field is a blue button with the text "Check Weather". Below the button, it says "The Weather is:". Underneath this is a white box containing the following text: "Weather of: delhi", "Temperature (Celsius): 21°", "Feels like in (Celsius): 21°", "Pressure: 1016 hPa", "Humidity: 73%", "Sunrise at 06:38:43 and Sunset at 17:31:10", "Cloud: 2%", and "Info: smoke".

Weather of: delhi  
Temperature (Celsius): 21°  
Feels like in (Celsius): 21°  
Pressure: 1016 hPa  
Humidity: 73%  
Sunrise at 06:38:43 and Sunset at 17:31:10  
Cloud: 2%  
Info: smoke



The screenshot shows a web browser window titled "Weather App - AskPython.com". The page has a light gray background. At the top, it says "Enter City Name". Below this is a text input field containing the word "chennai". Under the input field is a blue button with the text "Check Weather". Below the button, it says "The Weather is:". Underneath this is a white box containing the following text: "Weather of: chennai", "Temperature (Celsius): 26°", "Feels like in (Celsius): 26°", "Pressure: 1017 hPa", "Humidity: 89%", "Sunrise at 06:05:07 and Sunset at 17:40:15", "Cloud: 75%", and "Info: mist".

Weather of: chennai  
Temperature (Celsius): 26°  
Feels like in (Celsius): 26°  
Pressure: 1017 hPa  
Humidity: 89%  
Sunrise at 06:05:07 and Sunset at 17:40:15  
Cloud: 75%  
Info: mist



## **DISCUSSION :**

### **Integration of Operational Technology (OT):**

Discuss the significance of integrating OT into the project. How does this integration enhance the accuracy and efficiency of weather data processing? What operational challenges does it address.

### **Choice of Python and Web Framework:**

Explore the rationale behind selecting Python as the primary programming language and the chosen web framework (Flask or Django). How do these choices contribute to the project's development and scalability.

### **Real-Time Data Accuracy:**

Address the challenges and strategies involved in ensuring real-time accuracy in weather data. How does the project handle data validation, error handling, and maintain data integrity.

### **User Experience and Personalization:**

Discuss the importance of user experience in weather applications. How are features like automated geolocation, personalized preferences, and dynamic data visualization implemented to enhance user engagement.

### **Scalability and Deployment:**

Explore the considerations for scalability and the decision to deploy the website on a cloud infrastructure, specifically AWS. Discuss the benefits and challenges associated with this choice.

### **Data Visualization Techniques:**

Delve into the chosen data visualization techniques using Python libraries (e.g., Matplotlib, Plotly). How do these techniques contribute to presenting weather trends and forecasts in an intuitive manner.

### **Security Measures:**

Discuss the security measures implemented in the project, especially concerning user data and communication. How is SSL implemented, and what security protocols are in place.

**Testing and Quality Assurance:**

Explore the testing strategies employed in the project, including unit tests, integration tests, and validation processes. How does the project ensure the reliability and functionality of the system in diverse scenarios.

**Documentation and Knowledge Transfer:**

Discuss the importance of comprehensive documentation for developers and administrators. How does the project ensure transparency in implementation and facilitate knowledge transfer for future maintenance etc.

**Continuous Integration/Continuous Deployment (CI/CD):**

Explore the role of CI/CD in the project. How does automation contribute to the testing and deployment processes.

## CONCLUSION

In conclusion, the "Operational Technology Driven Weather Website Using Python" project represents a significant stride in leveraging technology to deliver an enhanced and operationally efficient weather information platform. The integration of operational technology, facilitated by Python and a chosen web framework (Flask or Django), lays the foundation for a robust backend capable of seamless data processing and real-time accuracy. The project prioritizes user experience through features like automated geolocation, personalized preferences, and dynamic data visualization, aiming to provide an engaging and user-friendly interface.

The decision to deploy the website on a cloud infrastructure, specifically AWS, underscores a commitment to scalability and reliability. This choice allows for optimal resource management, high availability, and adaptability to varying user demands. The security measures implemented, including SSL and rigorous testing processes, ensure the protection of user data and the overall integrity of the system.

Throughout the project's development, challenges were identified and effectively addressed, reflecting the team's commitment to delivering a robust solution. Comprehensive documentation and CI/CD practices further contribute to the project's transparency, maintainability, and potential for future enhancements.

## REFERENCES

- 1] [Weather App in Python | Tkinter - GUI - AskPython](#)
- 2] [UML Use Case Diagram - Javatpoint](#)
- 3] [Python Tutorial \(w3schools.com\)](#)
- 4] [Python Tkinter Tutorial - GeeksforGeeks](#)