Aptech®
Unleash your potential

PROJECT DOCUMENTATION

WEATHER NINJA

WEATHER NINJA
Aptech Garden Center, Karachi.

CONTENTS

Acknowledgement

I would like to express our heartfelt gratitude to TechWiz for providing us with this incredible opportunity to work on the Weather Ninja project. Your trust and support have been instrumental in enhancing our skills and gaining valuable hands-on experience in IoT solutions development. This opportunity has allowed us to explore new technologies, deepen my knowledge, and contribute meaningfully to the field of weather monitoring systems.
Thank you, TechWiz, for your guidance. and encouragement throughout this journey

Project Analysis

Weather Ninja aims to utilize IoT technology to create an advanced weather monitoring system. By collecting real-time weather data using various sensors and displaying the results on an LCD, it enables users to make informed decisions about weather patterns. The system is designed to be used in different locations to provide precise weather insights and notifications.

Problem Statement

Traditional weather forecasting methods are no longer sufficient to handle the increasing demand for accurate, real-time weather data. Human observation and basic instruments provide limited insights, which is why there is a need for a more intelligent and adaptive weather monitoring system. The challenge is to collect, process, and present real-time weather data in a reliable and scalable way, using IoT technology to improve accuracy and user accessibility.

Project Requirments

· **Sensors for data collection:** DHT11 (Temperature and Humidity) and Gauge pressure sensor.

· **Data Storage**: Collected data should be stored on a cloud platform for future access and analysis.

· **Hardware**: Raspberry Pi, Jumper Wires, Power Supply, LCD display.

· **Connectivity:** The system will utilize Ethernet or Wi-Fi for cloud storage and communication.

· **Alerts:** The system will send weather condition alerts to users via email or message.

Functional Requirments

- **Data Collection:** Real-time collection of weather data using IoT sensors.
- **Temperature and Humidity Monitoring:** The system will track and report temperature and humidity using DHT11 sensors.
- **Atmospheric Pressure Monitoring:** The system uses a gauge pressure sensor to record atmospheric pressure.
- **Data Storage:** Data collected by Raspberry Pi will be stored in the cloud for long-term access and analysis.
- **Visualization:** Weather data will be presented in a graphical format on an LCD screen.
- **Alerting:** Users will receive weather updates through email or messages.

Non-Functional Requirments

- **Compatibility:** The system should be compatible across multiple platforms and browsers.
- **Accuracy:** Sensors must deliver high-resolution, real-time data for reliable weather forecasts.
- **Reliability:** The system should provide consistent performance with minimal downtime.
- **Performance:** The system should be able to handle real-time data and provide quick updates to users.
- **Maintainability:** The system should be easy to update and maintain.
- **User-Friendliness:** Data visualizations should be clear and easy to interpret.

Interference Requirments

· **Hardware Interface:**

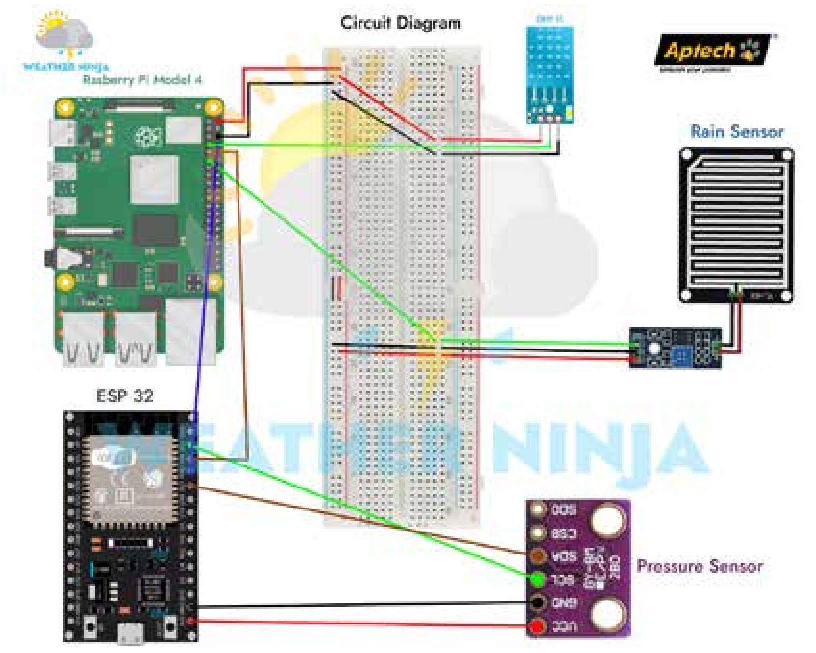
· Raspberry Pi for data collection and computation.

· DHT11 sensor for temperature and humidity.

· Gauge pressure sensor for atmospheric pressure data.

· LCD for displaying data.

· Wi-Fi/Ethernet for internet connectivity.



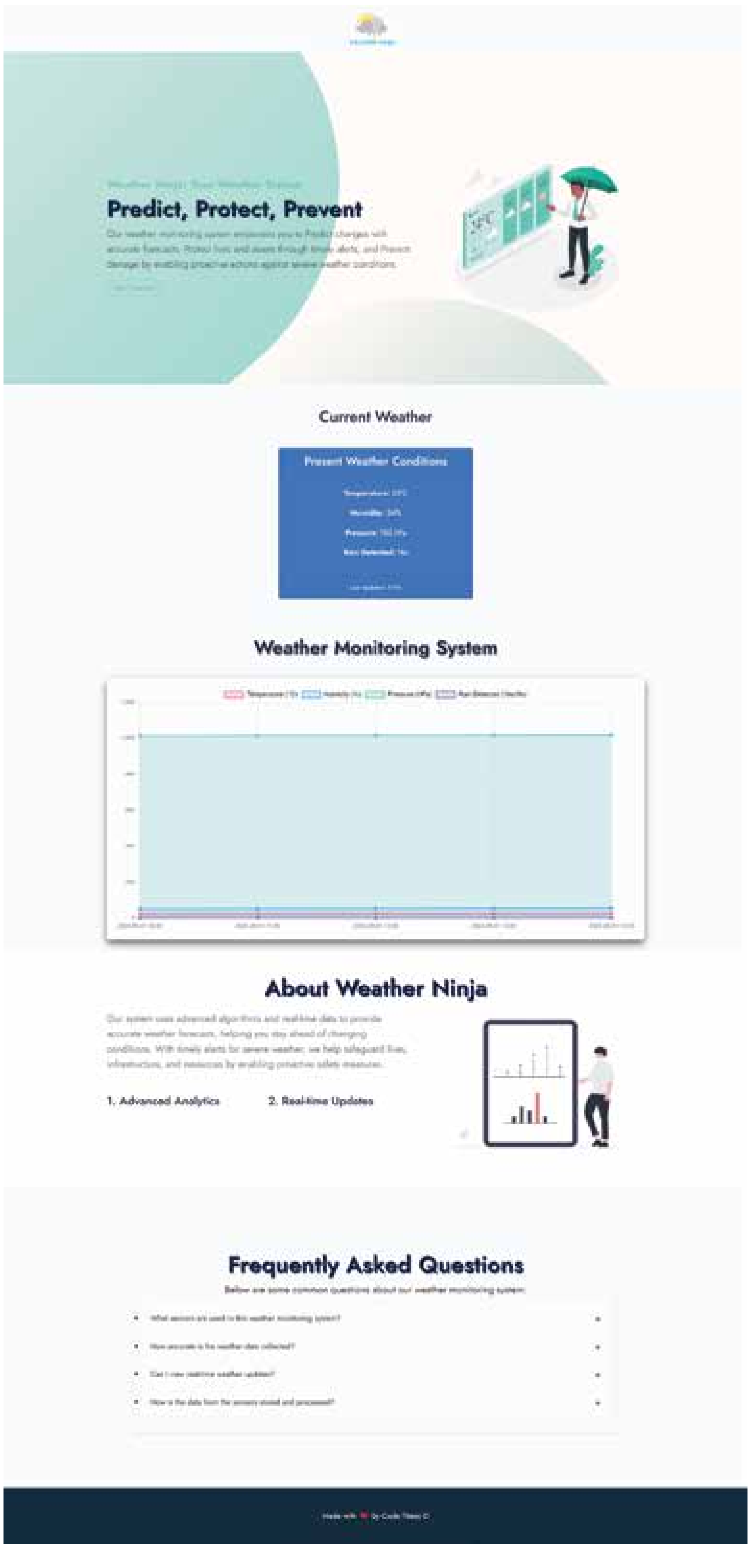
· **Software Interface:**

· Frontend: HTML5 for the web application.

· Backend: Python with frameworks like Flask or Django.

· Datastore: JSON or TXT file-based system.

· Operating System: Raspbian OS.



[illegible]

Predict, Protect, Prevent

Our weather monitoring system empowers you to Predict changes with accurate forecasts, Protect lives and assets through timely alerts, and Prevent damage by enabling proactive actions against severe weather conditions.

[illegible]

Current Weather

Present Weather Conditions

Temperature: [illegible]

Humidity: [illegible]

Pressure: [illegible]

Rain Detected: [illegible]

[illegible]

Weather Monitoring System

About Weather Ninja

Our system uses advanced algorithms and real-time data to provide accurate weather forecasts, helping you stay ahead of changing conditions. With timely alerts for severe weather, we help safeguard lives, infrastructure, and resources by enabling proactive safety measures.

1. Advanced Analytics

2. Real-time Updates

Frequently Asked Questions

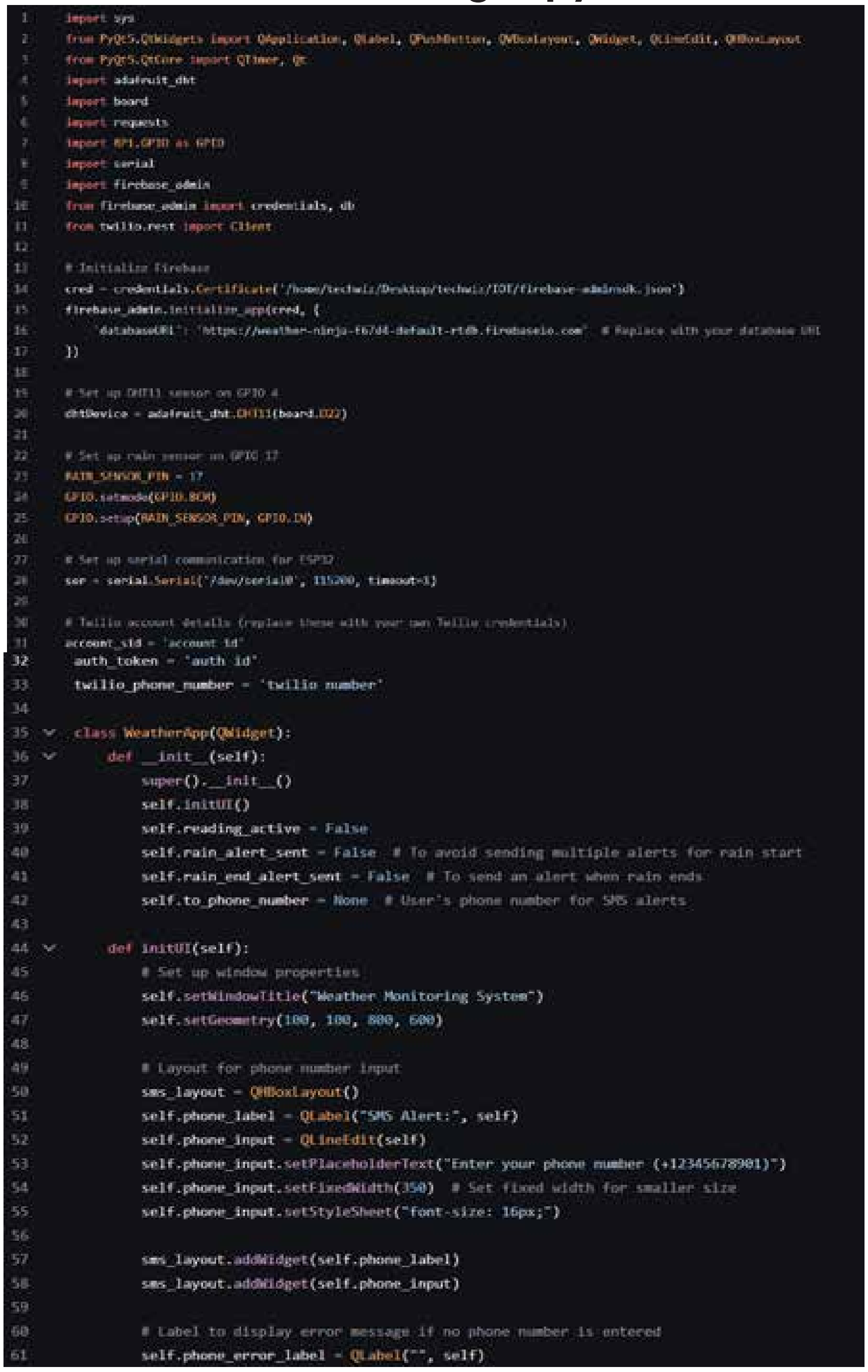
Below are some common questions about our weather monitoring system:

- What sensors are used in this weather monitoring system?
- How accurate is the weather data collected?
- Can I view real-time weather updates?
- How is the data from the sensors stored and processed?

Made with ♥ by Code Titans ©

Source Code

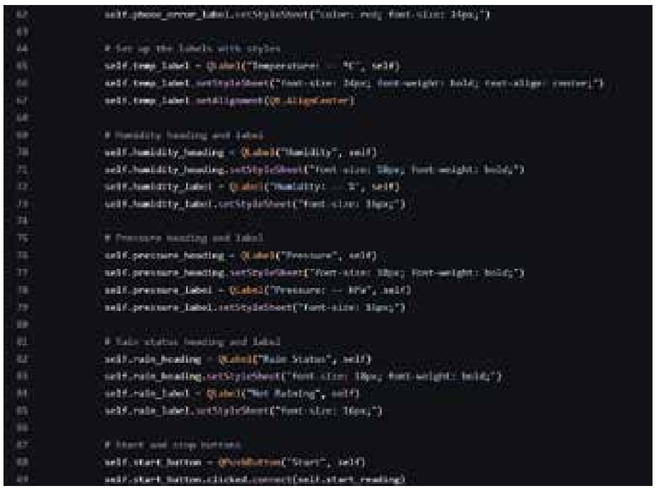
weather_gui.py



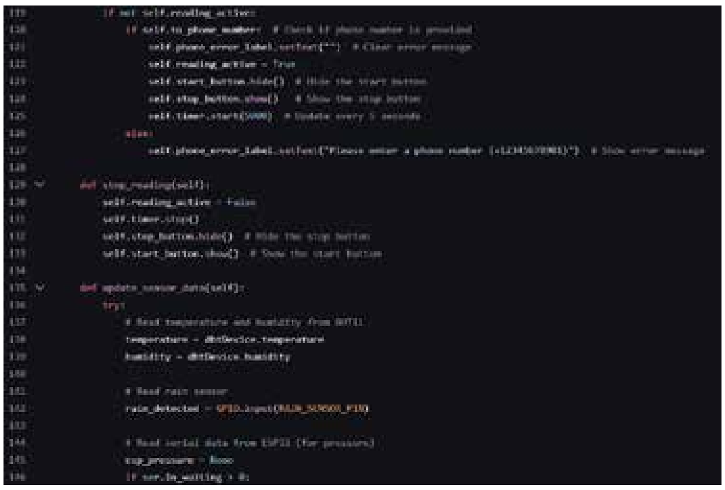
```
import sys
from PyQt5.QtWidgets import QApplication, QLabel, QPushButton, QVBoxLayout, QWidget, QLineEdit, QHBoxLayout
from PyQt5.QtCore import QTimer, Qt
import adafruit_dht
import board
import requests
import RPi.GPIO as GPIO
import serial
import firebase_admin
from firebase_admin import credentials, db
from twilio.rest import Client

# Initialize Firebase
cred = credentials.Certificate('/home/techwiz/Desktop/techwiz/IOT/firebase-adminsdk.json')
firebase_admin.initialize_app(cred, {
    'databaseURL': 'https://weather-ninja-f67d4-default-rtdb.firebaseio.com'  # Replace with your database URL
})

# Set up DHT11 sensor on GPIO 4
dhtDevice = adafruit_dht.DHT11(board.D22)

# Set up rain sensor on GPIO 17
RAIN_SENSOR_PIN = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(RAIN_SENSOR_PIN, GPIO.IN)

# Set up serial communication for ESP32
ser = serial.Serial('/dev/serial0', 115200, timeout=1)

# Twilio account details (replace these with your own Twilio credentials)
account_sid = 'account id'
 auth_token = 'auth id'
 twilio_phone_number = 'twilio number'

 class WeatherApp(QWidget):
     def __init__(self):
         super().__init__()
         self.initUI()
         self.reading_active = False
         self.rain_alert_sent = False  # To avoid sending multiple alerts for rain start
         self.rain_end_alert_sent = False  # To send an alert when rain ends
         self.to_phone_number = None  # User's phone number for SMS alerts

     def initUI(self):
         # Set up window properties
         self.setWindowTitle("Weather Monitoring System")
         self.setGeometry(100, 100, 800, 600)

         # Layout for phone number input
         sms_layout = QHBoxLayout()
         self.phone_label = QLabel("SMS Alert:", self)
         self.phone_input = QLineEdit(self)
         self.phone_input.setPlaceholderText("Enter your phone number (+12345678901)")
         self.phone_input.setFixedWidth(350)  # Set fixed width for smaller size
         self.phone_input.setStyleSheet("font-size: 16px;")

         sms_layout.addWidget(self.phone_label)
         sms_layout.addWidget(self.phone_input)

         # Label to display error message if no phone number is entered
         self.phone_error_label = QLabel("", self)
```

WEATHER NINJA

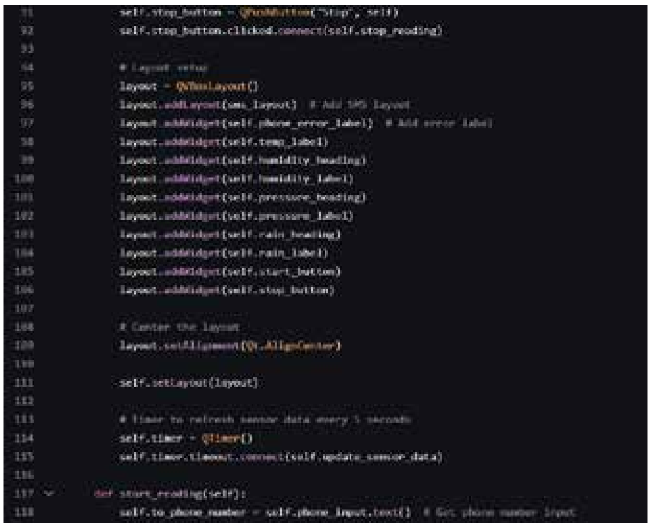
Aptech Garden Center, Karachi.



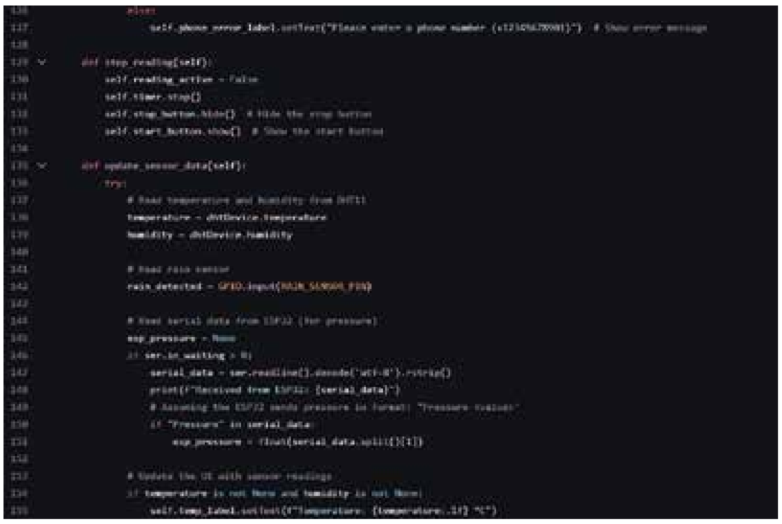
```
        self.phone_error_label.setStyleSheet("color: red; font-size: 14px;")

        # Set up the labels with styles
        self.temp_label = QLabel("Temperature: -- °C", self)
        self.temp_label.setStyleSheet("font-size: 24px; font-weight: bold; text-align: center;")
        self.temp_label.setAlignment(Qt.AlignCenter)

        # Humidity heading and label
        self.humidity_heading = QLabel("Humidity", self)
        self.humidity_heading.setStyleSheet("font-size: 18px; font-weight: bold;")
        self.humidity_label = QLabel("Humidity: -- %", self)
        self.humidity_label.setStyleSheet("font-size: 16px;")

        # Pressure heading and label
        self.pressure_heading = QLabel("Pressure", self)
        self.pressure_heading.setStyleSheet("font-size: 18px; font-weight: bold;")
        self.pressure_label = QLabel("Pressure: -- hPa", self)
        self.pressure_label.setStyleSheet("font-size: 16px;")

        # Rain status heading and label
        self.rain_heading = QLabel("Rain Status", self)
        self.rain_heading.setStyleSheet("font-size: 18px; font-weight: bold;")
        self.rain_label = QLabel("Not Raining", self)
        self.rain_label.setStyleSheet("font-size: 16px;")

        # Start and stop buttons
        self.start_button = QPushButton("Start", self)
        self.start_button.clicked.connect(self.start_reading)
```



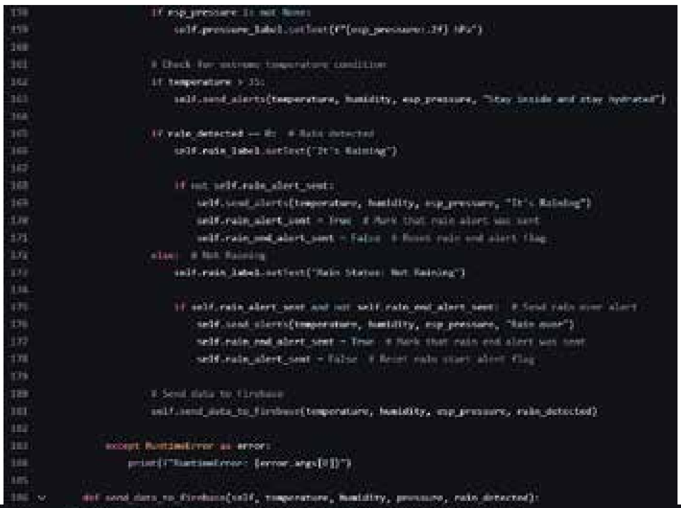
```
        if not self.reading_active:
            if self.to_phone_number:  # Check if phone number is provided
                self.phone_error_label.setText("")  # Clear error message
                self.reading_active = True
                self.start_button.hide()  # Hide the start button
                self.stop_button.show()   # Show the stop button
                self.timer.start(5000)  # Update every 5 seconds
            else:
                self.phone_error_label.setText("Please enter a phone number (+12345678901)")  # Show error message

    def stop_reading(self):
        self.reading_active = False
        self.timer.stop()
        self.stop_button.hide()  # Hide the stop button
        self.start_button.show()  # Show the start button

    def update_sensor_data(self):
        try:
            # Read temperature and humidity from DHT11
            temperature = dhtDevice.temperature
            humidity = dhtDevice.humidity

            # Read rain sensor
            rain_detected = GPIO.input(RAIN_SENSOR_PIN)

            # Read serial data from ESP32 (for pressure)
            esp_pressure = None
            if ser.in_waiting > 0:
```



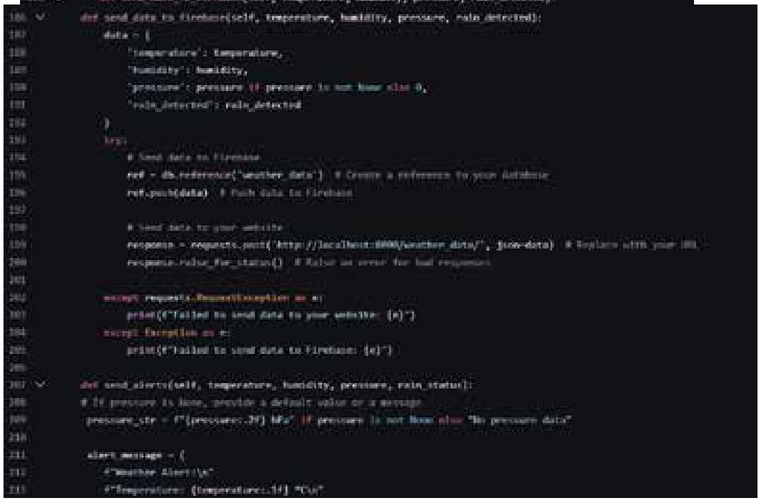
```
        self.stop_button = QPushButton("Stop", self)
        self.stop_button.clicked.connect(self.stop_reading)

        # Layout setup
        layout = QVBoxLayout()
        layout.addLayout(sms_layout)  # Add SMS layout
        layout.addWidget(self.phone_error_label)  # Add error label
        layout.addWidget(self.temp_label)
        layout.addWidget(self.humidity_heading)
        layout.addWidget(self.humidity_label)
        layout.addWidget(self.pressure_heading)
        layout.addWidget(self.pressure_label)
        layout.addWidget(self.rain_heading)
        layout.addWidget(self.rain_label)
        layout.addWidget(self.start_button)
        layout.addWidget(self.stop_button)

        # Center the layout
        layout.setAlignment(Qt.AlignCenter)

        self.setLayout(layout)

        # Timer to refresh sensor data every 5 seconds
        self.timer = QTimer()
        self.timer.timeout.connect(self.update_sensor_data)

    def start_reading(self):
        self.to_phone_number = self.phone_input.text()  # Get phone number input
```



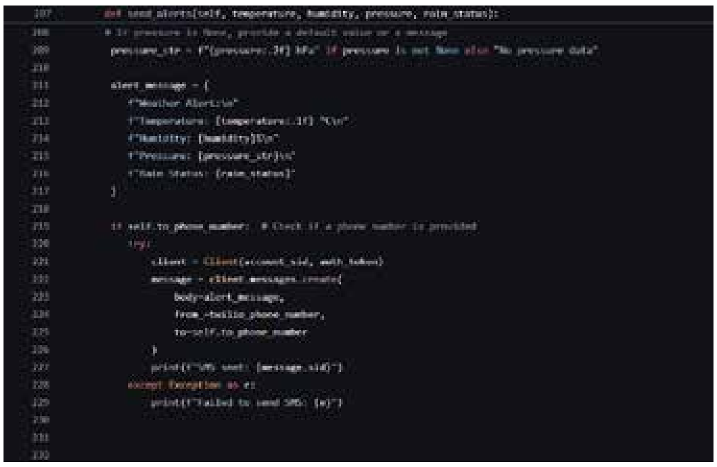
```
            else:
                self.phone_error_label.setText("Please enter a phone number (+12345678901)")  # Show error message

    def stop_reading(self):
        self.reading_active = False
        self.timer.stop()
        self.stop_button.hide()  # Hide the stop button
        self.start_button.show()  # Show the start button

    def update_sensor_data(self):
        try:
            # Read temperature and humidity from DHT11
            temperature = dhtDevice.temperature
            humidity = dhtDevice.humidity

            # Read rain sensor
            rain_detected = GPIO.input(RAIN_SENSOR_PIN)

            # Read serial data from ESP32 (for pressure)
            esp_pressure = None
            if ser.in_waiting > 0:
                serial_data = ser.readline().decode('utf-8').rstrip()
                print(f"Received from ESP32: {serial_data}")
                # Assuming the ESP32 sends pressure in format: "Pressure <value>"
                if "Pressure" in serial_data:
                    esp_pressure = float(serial_data.split()[1])

            # Update the UI with sensor readings
            if temperature is not None and humidity is not None:
                self.temp_label.setText(f"Temperature: {temperature:.1f} °C")
```



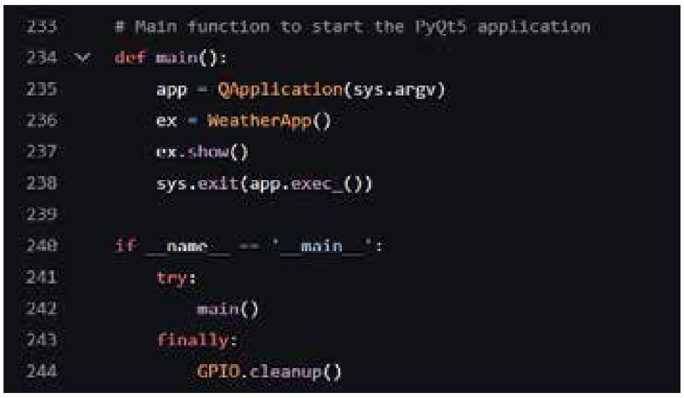
```python
                if esp_pressure is not None:
                    self.pressure_label.setText(f"{esp_pressure:.2f} hPa")

                # Check for extreme temperature condition
                if temperature > 35:
                    self.send_alerts(temperature, humidity, esp_pressure, "Stay inside and stay hydrated")

                if rain_detected == 0:  # Rain detected
                    self.rain_label.setText("It's Raining")

                    if not self.rain_alert_sent:
                        self.send_alerts(temperature, humidity, esp_pressure, "It's Raining")
                        self.rain_alert_sent = True  # Mark that rain alert was sent
                        self.rain_end_alert_sent = False  # Reset rain end alert flag
                else:  # Not Raining
                    self.rain_label.setText('Rain Status: Not Raining')

                    if self.rain_alert_sent and not self.rain_end_alert_sent:  # Send rain over alert
                        self.send_alerts(temperature, humidity, esp_pressure, "Rain over")
                        self.rain_end_alert_sent = True  # Mark that rain end alert was sent
                        self.rain_alert_sent = False  # Reset rain start alert flag

                # Send data to Firebase
                self.send_data_to_firebase(temperature, humidity, esp_pressure, rain_detected)

        except RuntimeError as error:
            print(f"RuntimeError: {error.args[0]}")

    def send_data_to_firebase(self, temperature, humidity, pressure, rain_detected):
```



```python
    def send_data_to_firebase(self, temperature, humidity, pressure, rain_detected):
        data = {
            'temperature': temperature,
            'humidity': humidity,
            'pressure': pressure if pressure is not None else 0,
            'rain_detected': rain_detected
        }
        try:
            # Send data to Firebase
            ref = db.reference('weather_data')  # Create a reference to your database
            ref.push(data)  # Push data to Firebase

            # Send data to your website
            response = requests.post('http://localhost:8000/weather_data/', json=data)  # Replace with your URL
            response.raise_for_status()  # Raise an error for bad responses

        except requests.RequestException as e:
            print(f"Failed to send data to your website: {e}")
        except Exception as e:
            print(f"Failed to send data to Firebase: {e}")

    def send_alerts(self, temperature, humidity, pressure, rain_status):
    # If pressure is None, provide a default value or a message
     pressure_str = f"{pressure:.2f} hPa" if pressure is not None else "No pressure data"

     alert_message = (
        f"Weather Alert:\n"
        f"Temperature: {temperature:.1f} °C\n"
```



```python
    def send_alerts(self, temperature, humidity, pressure, rain_status):
    # If pressure is None, provide a default value or a message
     pressure_str = f"{pressure:.2f} hPa" if pressure is not None else "No pressure data"

     alert_message = (
        f"Weather Alert:\n"
        f"Temperature: {temperature:.1f} °C\n"
        f"Humidity: {humidity}%\n"
        f"Pressure: {pressure_str}\n"
        f"Rain Status: {rain_status}"
     )

     if self.to_phone_number:  # Check if a phone number is provided
        try:
            client = Client(account_sid, auth_token)
            message = client.messages.create(
                body=alert_message,
                from_=twilio_phone_number,
                to=self.to_phone_number
            )
            print(f"SMS sent: {message.sid}")
        except Exception as e:
            print(f"Failed to send SMS: {e}")
```



```python
# Main function to start the PyQt5 application
def main():
    app = QApplication(sys.argv)
    ex = WeatherApp()
    ex.show()
    sys.exit(app.exec_())

if __name__ == '__main__':
    try:
        main()
    finally:
        GPIO.cleanup()
```

Task Sheet

		Title	Date of preparation of Activity Plan			
No.	Task		Start Date	Actual Days	Team-Mate Name	Status
01.	Research	Weather Ninja	18 - SEP - 2024	6 - DAYS -	All	Done
02.	UI				Shahmeer	Done
03.	Hardware				Amna Mustafa Mohammad Shayan	Done
04.	Backend				All	Done
05.	User Guide & Documentation				Mohammad Shayan Shahmeer	Done