```cpp
#include <wiringPi.h>
#include <iostream>
#include <fstream>
#include <jsoncpp/json/json.h>
#include <thread>
#include <chrono>
#include <string>
using namespace std;
#define DHT_PIN 7             // GPIO pin for DHT11 sensor
#define CoolingS 0           // GPIO pin for relay
#define VentilationS 1
#define JSON_FILE "/var/www/html/data.json" // JSON file path in web
server directory

std::string CoolingS_status, VentilationS_status;

struct SensorData {
    float humidity;
    float temperature;
};

int readDHTSensor(float& humidity, float& temperature) {
    int data[5] = { 0, 0, 0, 0, 0 };

    pinMode(DHT_PIN, OUTPUT);
    digitalWrite(DHT_PIN, LOW);
    delay(18);
    digitalWrite(DHT_PIN, HIGH);
    delayMicroseconds(40);
    pinMode(DHT_PIN, INPUT);

    int pulseCount = 0;
    while (digitalRead(DHT_PIN) == LOW) {
        delayMicroseconds(2);
        pulseCount++;
        if (pulseCount > 10000)
            return -1;
    }

    pulseCount = 0;
    while (digitalRead(DHT_PIN) == HIGH) {
        delayMicroseconds(2);
        pulseCount++;
        if (pulseCount > 10000)
            return -1;
    }

    for (int i = 0; i < 40; i++) {
```

```
        pulseCount = 0;
        while (digitalRead(DHT_PIN) == LOW) {
            delayMicroseconds(2);
            pulseCount++;
            if (pulseCount > 10000)
                return -1;
        }

        pulseCount = 0;
        while (digitalRead(DHT_PIN) == HIGH) {
            delayMicroseconds(2);
            pulseCount++;
            if (pulseCount > 10000)
                return -1;
        }

        data[i / 8] <<= 1;
        if (pulseCount > 30)
            data[i / 8] |= 1;
    }

    if (data[4] == ((data[0] + data[1] + data[2] + data[3]) & 0xFF)) {
        humidity = static_cast<float>((data[0] << 8) + data[1]) * 0.1f;
        temperature = static_cast<float>(((data[2] & 0x7F) << 8) +
data[3]) * 0.1f;
        if (data[2] & 0x80)
            temperature *= -1;
        return 0;
    }

    return -1;
}

void Start_VentilationS() {
    digitalWrite(VentilationS, HIGH);
    std::cout << "Ventilation System turned ON" << std::endl;
    VentilationS_status = "Ventilation System turned ON";
}

void Start_CoolingS() {
    digitalWrite(CoolingS, HIGH);
    std::cout << "Cooling System turned ON" << std::endl;
    CoolingS_status = "Cooling System turned ON";
}

void VentilationS_Timer(int duration = 1) {
    std::this_thread::sleep_for(std::chrono::minutes(duration));
    Start_VentilationS();
```

```
}

void CoolingS_Timer(int duration = 1) {
    std::this_thread::sleep_for(std::chrono::minutes(duration));
    Start_CoolingS();
}

int main() {

    std::thread Ctimer;
    std::thread Vtimer;

    if (wiringPiSetup() == -1) {
        return 1;
    }

    if (wiringPiSetupGpio() == -1) {
        std::cout << "Failed to initialize wiringPi GPIO." <<
std::endl;
        return 1;
    }

    pinMode(CoolingS, OUTPUT);

    while (true) {
        SensorData SD;
        int result = readDHTSensor(SD.humidity, SD.temperature);

        if (result == 0) {
            std::cout << "Temperature: " << SD.temperature << "°C,
Humidity: " << SD.humidity << "%" << std::endl;

            if (SD.temperature > 25) {
                try {
                    Ctimer = std::thread(CoolingS_Timer, 1);  // Start
the timer thread with 1 minute duration
                    Ctimer.detach();  // Detach the timer thread,
allowing it to run independently
                }
                catch (const std::exception& e) {
                    std::cout << "Error starting the timer: " <<
e.what() << std::endl;
                }
            }
            else {

                if (Ctimer.joinable()) {
```

```
                    Ctimer.join();  // Stop the timer by waiting for it
to finish
                    std::cout << "Join Secccess" << std::endl;
                    digitalWrite(CoolingS, LOW);
                    std::cout << "Cooling System turned OFF" <<
std::endl;
                    CoolingS_status = "Cooling System turned OFF";
                }
                else {
                    digitalWrite(CoolingS, LOW);
                    std::cout << "Cooling System turned OFF" <<
std::endl;
                    std::cout << "Error stopping the timer: Timer
thread is not joinable." << std::endl;
                    CoolingS_status = "Cooling System turned OFF";
                }

            }

            if (SD.humidity > 60) {
                try {
                    Vtimer = std::thread(VentilationS_Timer, 1);  //
Start the timer thread with 1 minute duration
                    Vtimer.detach();  // Detach the timer thread,
allowing it to run independently
                }
                catch (const std::exception& e) {
                    std::cout << "Error starting the timer: " <<
e.what() << std::endl;
                }
            }
            else {
                if (Vtimer.joinable()) {
                    Vtimer.join();  // Stop the timer by waiting for it
to finish
                    std::cout << "Join Secccess" << std::endl;
                    digitalWrite(VentilationS, LOW);
                    std::cout << "Ventilation System turned OFF" <<
std::endl;
                    VentilationS_status = "Ventilation System turned
OFF";
                }
                else {
                    digitalWrite(VentilationS, LOW);
                    std::cout << "Ventilation System turned OFF" <<
std::endl;
                    std::cout << "Error stopping the timer: Timer
thread is not joinable." << std::endl;
```

```
                    VentilationS_status = "Ventilation System turned
OFF";
                }
            }

            // Create a JSON object to store the data
            Json::Value jsonData;
            jsonData["temperature"] = SD.temperature;
            jsonData["humidity"] = SD.humidity;
            jsonData["CoolingS_status"] = CoolingS_status;
            jsonData["VentilationS_status"] = VentilationS_status;

            // Open the JSON file for writing
            std::ofstream jsonFile(JSON_FILE);
            if (!jsonFile.is_open()) {
                std::cout << "Failed to open JSON file for writing." <<
std::endl;
                return 1;
            }

            // Write the JSON data to the file
            jsonFile << jsonData;
            jsonFile.close();
        }
        else {
            std::cout << "Failed to read data from DHT sensor" <<
std::endl;
        }

        delay(2000);
    }

    return 0;
}
//g++ -o main main.cpp -lwiringPi -ljsoncpp
//chmod o+r file_name
```