```
#include <ESP8266WiFi.h>
#include "HX711.h"
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include "ThingSpeak.h"

// LCD Setup
LiquidCrystal_I2C lcd(0x27, 16, 2);

// WiFi Credentials
#define WIFI_SSID "water"
#define WIFI_PASSWORD "12345678"

// Telegram Bot Credentials
#define BOT_TOKEN "6997225072:AAHGEuOzUgqYD8ShJNQmb6nVKfZQUVSQ8uU"
#define CHAT_ID "1162650946"
const unsigned long BOT_MTBS = 1000;  // Mean time between scan messages

X509List cert(TELEGRAM_CERTIFICATE_ROOT);
WiFiClientSecure secured_client;
UniversalTelegramBot bot(BOT_TOKEN, secured_client);
unsigned long bot_lasttime;
WiFiClient client;

// Load Cell Setup
HX711 scale;
float weight;
float calibration_factor = -107325;  // Adjust this value as per calibration

unsigned long myChannelNumber = 2876802;
const char *myWriteAPIKey = "EXAES4VYPOY0A3WU";

// Weight Thresholds
const int underWeightThreshold = 300;
const int normalWeightThreshold = 600;
int count = 0;

void handleNewMessages(int numNewMessages) {
    Serial.print("handleNewMessages ");
    Serial.println(numNewMessages);

    for (int i = 0; i < numNewMessages; i++) {
        String text = bot.messages[i].text;
        String from_name = bot.messages[i].from_name;

        Serial.println("Message from: " + from_name + ", text: " + text);

        if (from_name.equals("")) {
            from_name = "Guest";
        }

        if (text.equals("/start")) {
```

```
            String welcome = "Welcome, " + from_name + ".\n";
            welcome += "This is a weight monitoring system.\n";
            welcome += "You will receive alerts if the weight is out of range.\n";
            bool success = bot.sendMessage(CHAT_ID, welcome, "");
              Serial.println(success ? "Sent welcome message" : "Failed to send welcome
message");
        }
    }
}

void setup() {
    Serial.begin(115200);
    scale.begin(D6,D5);
    lcd.init();
    lcd.backlight();
    lcd.clear();
    lcd.setCursor(6, 0);
    lcd.print("IoT");
    lcd.setCursor(1, 1);
    lcd.print("Weight Scale");
    delay(3000);
    lcd.clear();

    // WiFi Connection
    Serial.print("Connecting to WiFi SSID ");
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
    configTime(0, 0, "pool.ntp.org");  // Sync UTC time via NTP
    secured_client.setTrustAnchors(&cert);
    Serial.println(WIFI_SSID);

    while (WiFi.status() != WL_CONNECTED) {
        Serial.print(".");
        delay(500);
    }
    Serial.println("\nWiFi connected. IP address: " + WiFi.localIP().toString());

    lcd.setCursor(0, 1);
    lcd.print("WiFi Connected  ");
    delay(2000);
    lcd.clear();

    // Sync Time
    configTime(0, 0, "pool.ntp.org");
    time_t now = time(nullptr);
    while (now < 24 * 3600) {
        Serial.print(".");
        delay(100);
        now = time(nullptr);
    }
    Serial.println(now);

    // Send startup message
    bot.sendMessage(CHAT_ID, "Bot started up", "");
```

```
    // Initialize ThingSpeak
    ThingSpeak.begin(client);

    // Setup HX711 Load Cell
    scale.set_scale();
    scale.tare();  // Reset scale to 0
}

void loop() {
    if (millis() - bot_lasttime > BOT_MTBS) {
        int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
        if (numNewMessages) {
            Serial.println("Got response");
            handleNewMessages(numNewMessages);
        }
        bot_lasttime = millis();
    }

    // Read weight from HX711
    if (scale.is_ready()) {
        scale.set_scale(calibration_factor);
        weight = scale.get_units(5);

        lcd.setCursor(0, 0);
        lcd.print("Measured Weight");
        lcd.setCursor(0, 1);
        lcd.print(weight);
        lcd.print(" KG  ");
        delay(2000);
        lcd.clear();


        Serial.print("Weight: ");
        Serial.print(weight);
        Serial.println(" KG");

        // Send alerts based on weight thresholds
        if (weight > 0 && weight < underWeightThreshold ) {
                  String alertMessage = " Underweight Alert! \nMeasured Weight: " +
String(weight) + " KG.\nPlease check the load.";
            bool success = bot.sendMessage(CHAT_ID, alertMessage, "Markdown");
                  Serial.println(success ? "Sent Underweight Alert" : "Failed to send
Underweight Alert");
            lcd.setCursor(0, 1);
            lcd.print("Underweight Alert!");
        }
        else if (weight > normalWeightThreshold ) {
                   String alertMessage = " Overweight Alert! \nMeasured Weight: " +
String(weight) + " KG.\nOverload detected!";
            bool success = bot.sendMessage(CHAT_ID, alertMessage, "Markdown");
                  Serial.println(success ? "Sent Overweight Alert" : "Failed to send
Overweight Alert");
            lcd.setCursor(0, 1);
            lcd.print("Overweight Alert!");
```

```
        }

        delay(2000);
        lcd.clear();

        // Send data to ThingSpeak
        ThingSpeak.setField(1, weight);
        int response = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
        if (count == 15) {
                  response == 200 ? Serial.println("ThingSpeak update successful.") :
Serial.println("ThingSpeak update failed. HTTP error: " + String(response));
            count = 0;
        }

        // Reset scale for next measurement
        scale.tare();
    }

    count = (count + 1) % 15;  // Ensures alerts are sent periodically (~30 seconds)
    delay(100);
}
```