STTH3113 SENSOR-BASED SYSTEM (A)

SEMESTER 6 (A242)

Title:
Midterm Assignment

Prepared For:
Ahmad Hanis Bin Mohd Shabli

Prepared By:

Name	Matric Number
KHAIRUN NISAA BINTI DAWI CAHYONO	295071

Submission Date: 30th May 2025

YouTube Link: https://youtu.be/a_BGRbrELPE?si=3ErBUqz3IJOigFI4

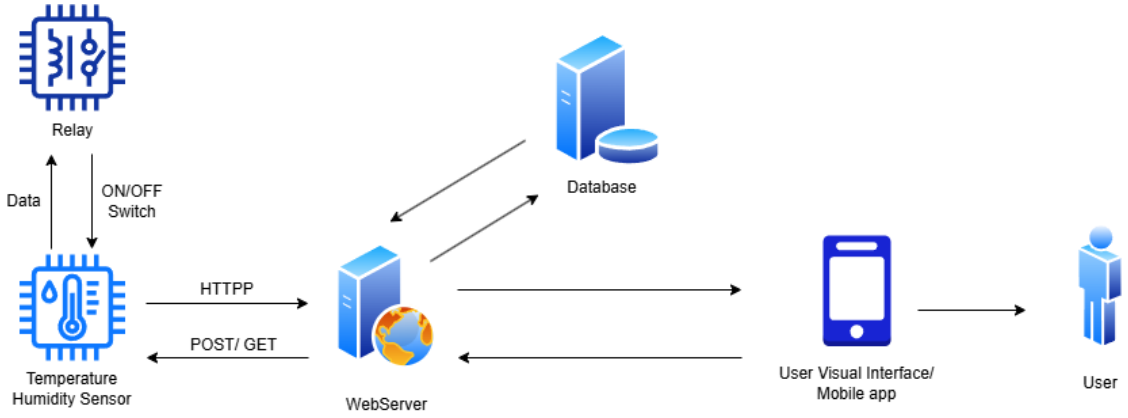
GitHub Link: nisaadawi/Midterm_STTHK3113_A242

Arduino folder: Midterm_STTHK3113_A242/DHT11_HTTP at main · nisaadawi/Midterm_STTHK3113_A242

Backend folder: Midterm_STTHK3113_A242/backend at main · nisaadawi/Midterm_STTHK3113_A242

Mobile App folder: Midterm_STTHK3113_A242/flutter_dht11_apps at main · nisaadawi/Midterm_STTHK3113_A242

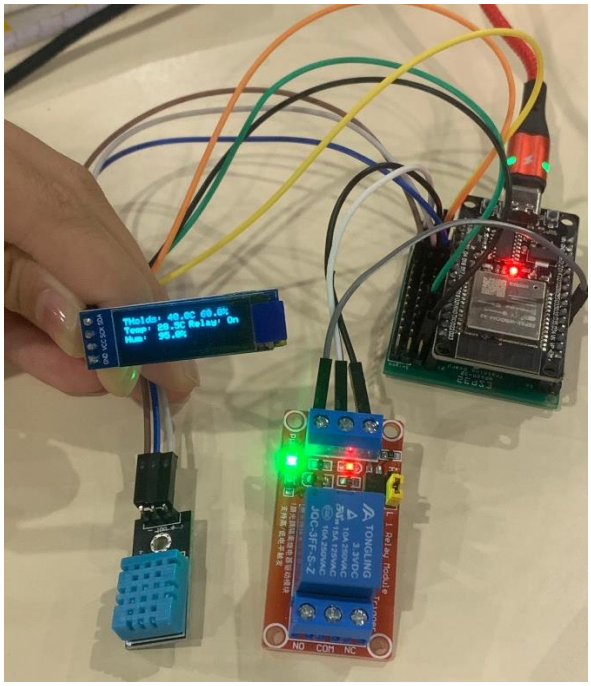
1. System Architecture Diagram



1. Temperature Humidity Sensor (DHT11)
    - Measure environmental data
    - Example output: "Temp:" 28°C, Humidity: 75%
2. Webserver (PHP)
    - Receive sensor data from HTTP request
    - Store data in Database
3. Database (PHPMyAdmin)
    - Stores: sensor reading, thresholds values and timestamp
4. User Interface
    - Web Dashboard for mobile app for monitoring

2. Setup Step

1. Sensor Setup



ESP32

- Main Controller

DHT 11

- +VE to 3V3
- OUT to GPIO 4
- -VE to GND

Relay

- DC+ to 3V3
- DC- to GND
- IN to GPIO 25

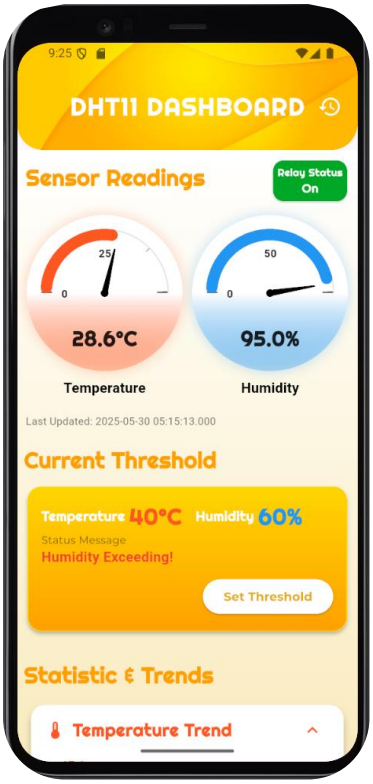
OLED

- VCC to 3V3
- GND to GND
- SCK to GPIO 22
- SDA to GPIO 21

Arduino

- Code the program

2. Web App



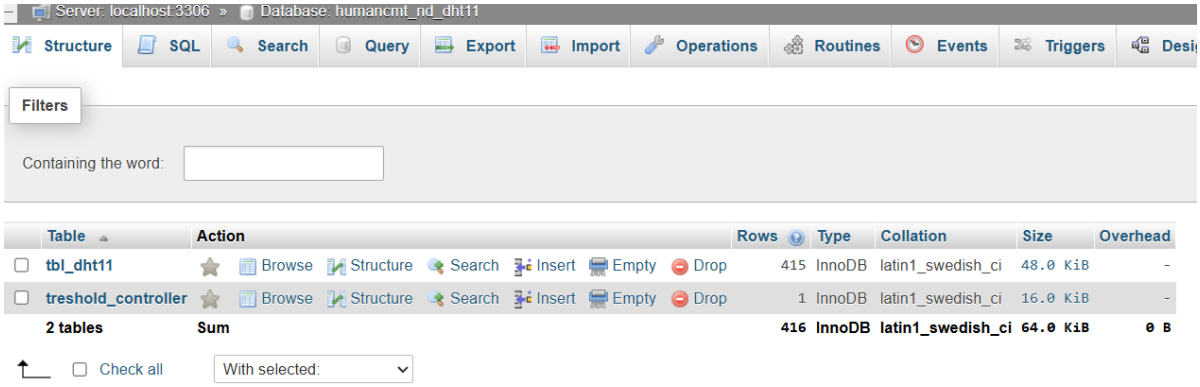
1. Use Flutter as UI
    - Main.dart
    - Myconfig.dart
    - Splash_screen.dart
    - Homepage.dart
    - dht_history_screen.dart

2. PHP as backend
    - Dbconfig.php
    - Dht11_backend.php
    - Fetch.php
    - Fetch_threshold.php
    - Update_threshold.php

3. Screenshots

Backend

I created 1 database with 2 table. 1 is table tbl_dht11 that used to save all the temperature and humidity value collected by the sensor every 10 seconds. While, the 2nd table threshold_controller which functions to save the value off temperature and humidity thresholds. User are also allowed to change/set new threshold



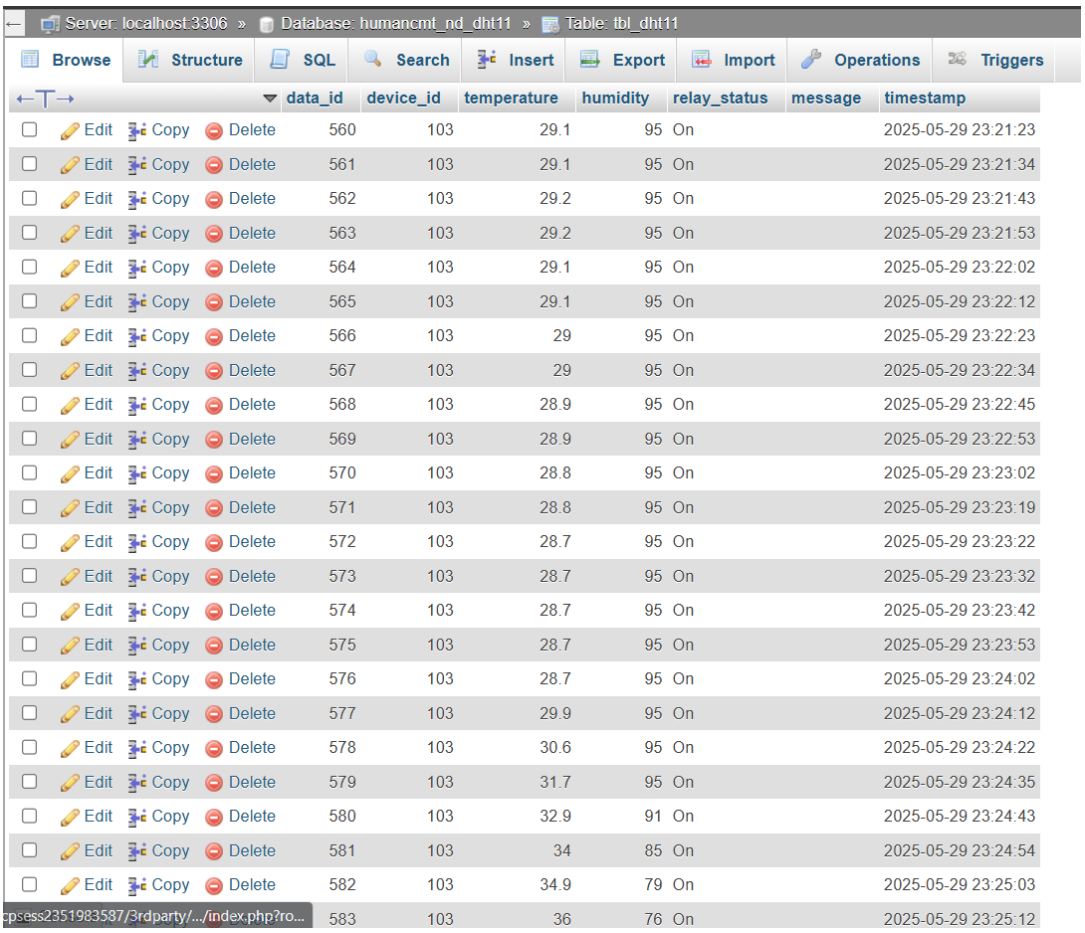


Figure. Table tbl_dht11

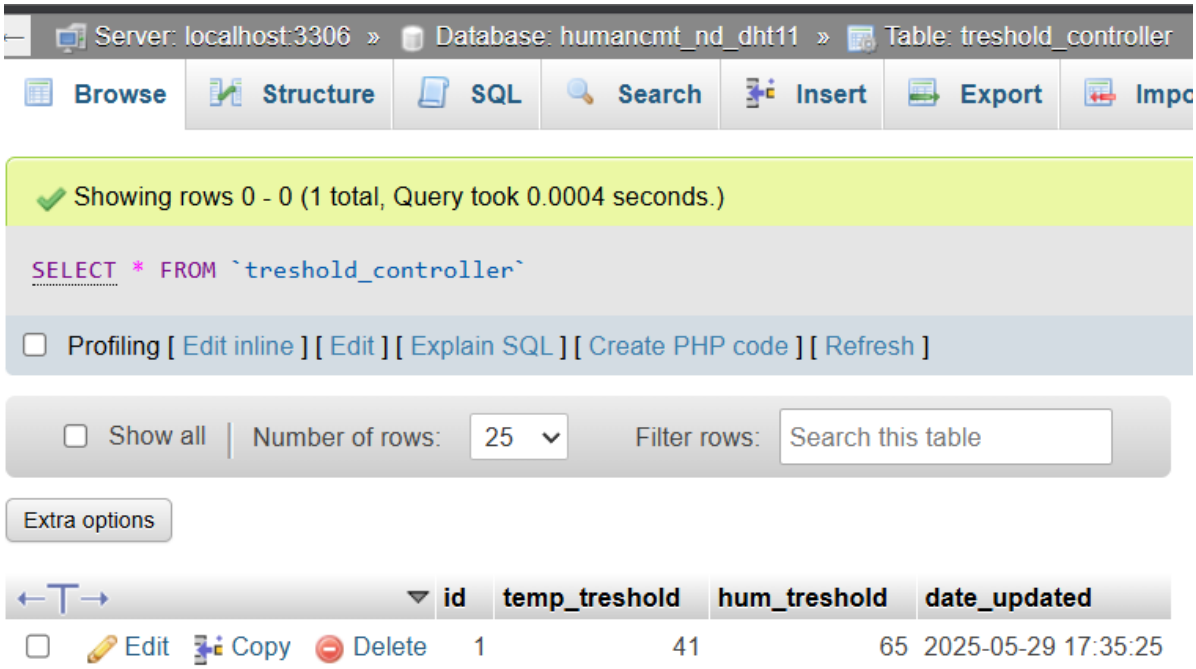


Figure. Table threshold_controller

Relay

Relay function to on/off the circuit. The relay in my project is off and close based on condition that triggers them. For example, if the temperature and humidity read by the sensor is greater than the threshold, it will show message that the value has exceed the threshold and the relay will on. To indicates that the relay is on, a red light appears. While if the temp and hum in normal condition, the relay will off and there is no red light. To make it easier for user, the web app has display the relay status on the top right of the screen.

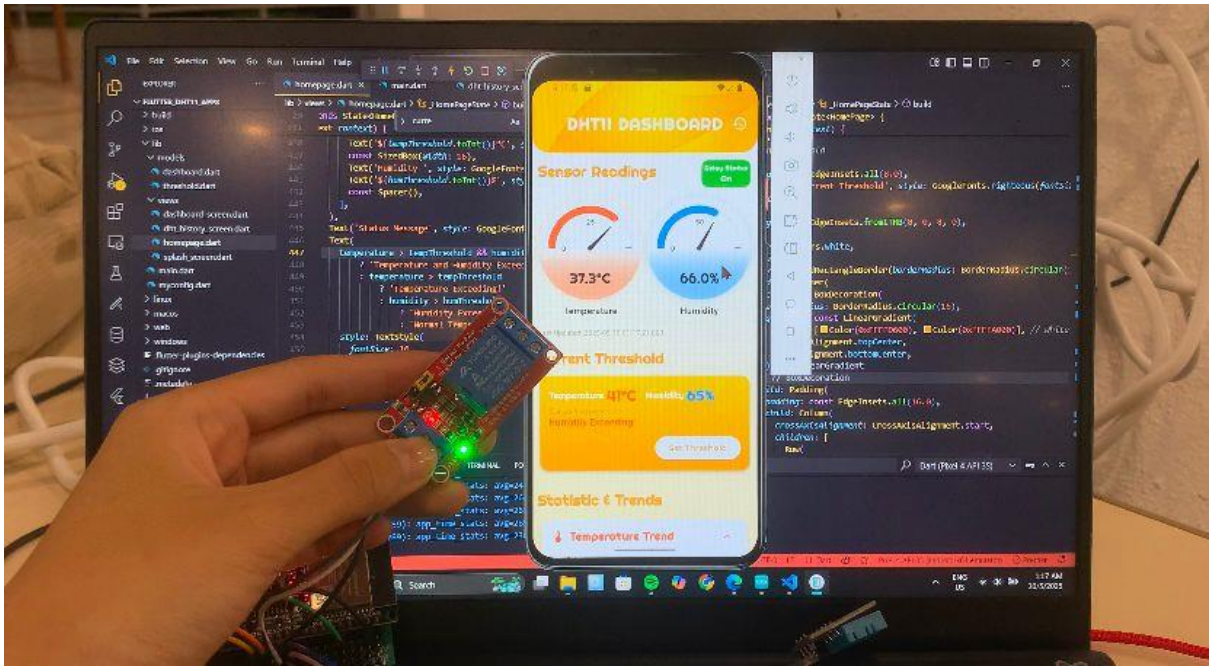


Figure. Relay was on and red light appear

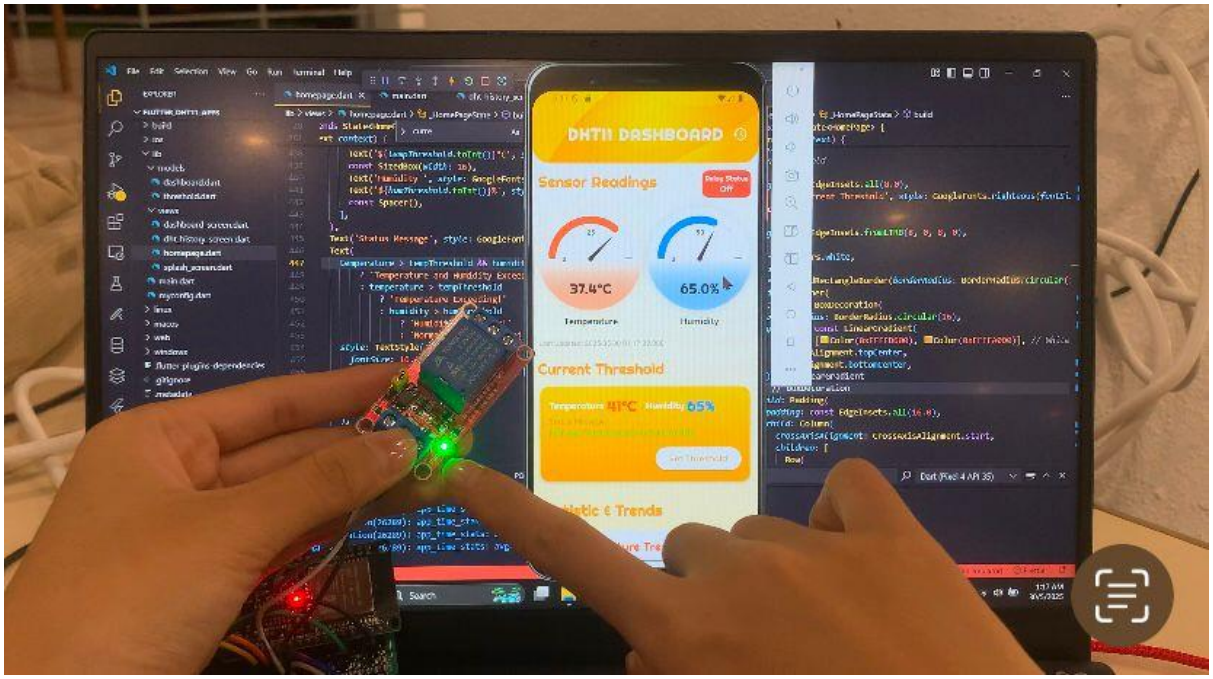


Figure: Relay is off and there are no red light

Application

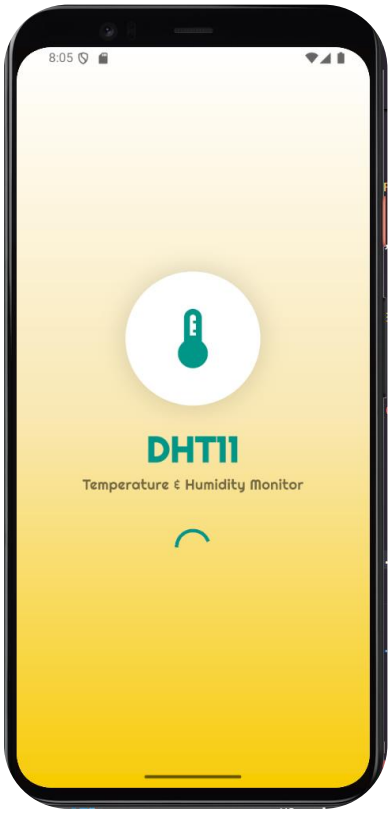


Figure. Splash screen of the system

(a.) (b.)

(c.) (d.)

Figure. The content of the system

Figure a. shows meter that repesent the data input form the sensor. It is responsive to evrytime changes and update every 10 second. The display of data also precise and accurate. Other than that is the ‘Current treshold’ card/section. It contains the current threshold value and messages that can alarm the user if the temprature and humidity readings are too high or low. Figure b. and c. shows the statistical and trends graphs for both temperature and humidity. The graphs are line graph and bar graph. These type of graph was selected because it was easier to understand by the user and user can see the trends more visible. In graph d, the details of every data is shown if you hover the cursor around the data and click it. A bubble message will pop up and show details of the input data including the value and date timestamp.

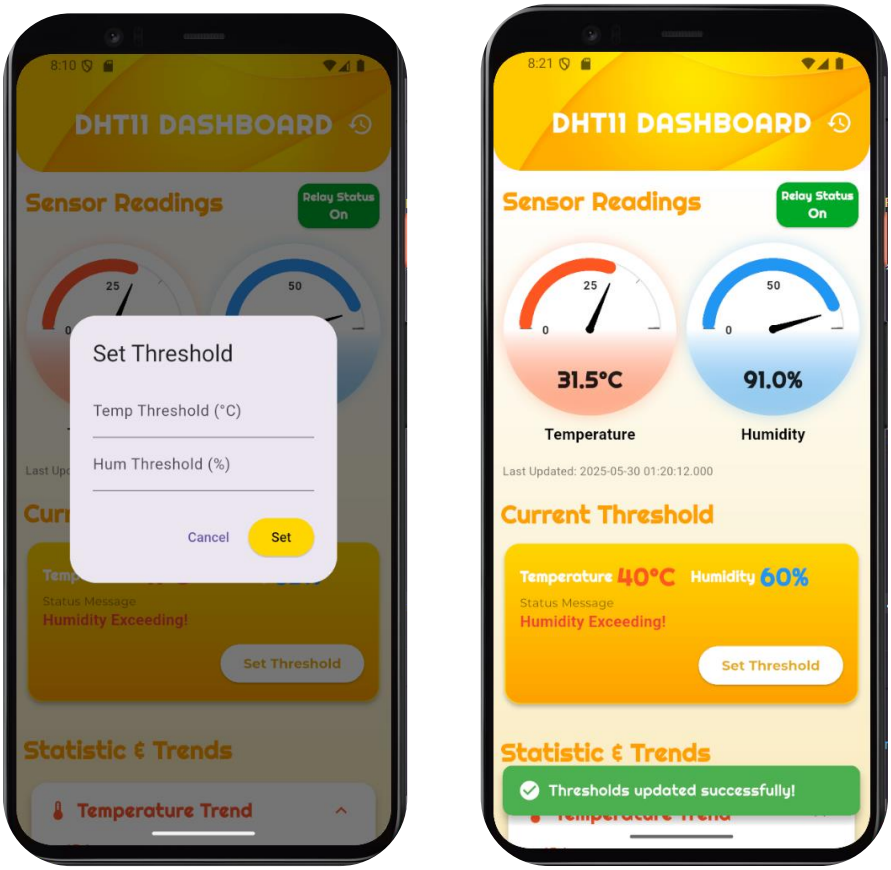


Figure. User set new treshold value

After user complete setting the new threshold, a succes meassge appear to indicates that the data has been saved and update in the database.

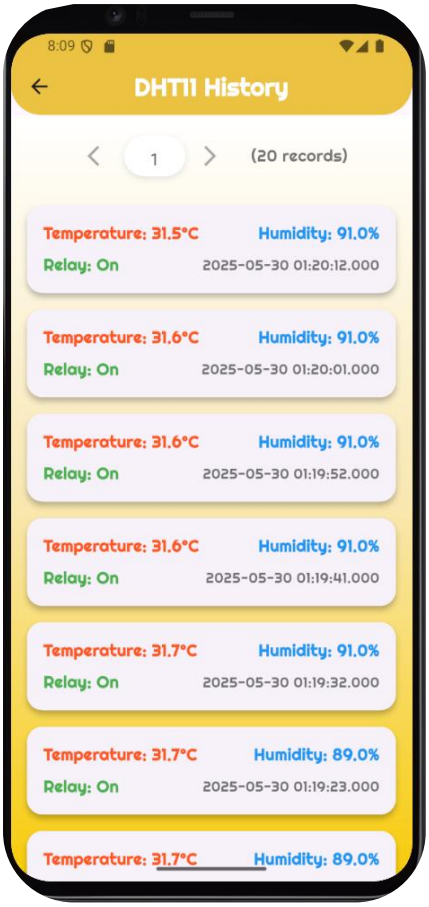


Figure. System shows the History data

4. Challenges and improvement

The Flutter environment setup was one huge hurdle, especially wrangling Gradle issues that snatched two hours away from actual development. The build process went on to fail while spewing cryptic errors-with the issue resolved through online forum consultation and a series of hit-or-miss attempts to solve it: updating Gradle, cleaning the cache, and who-knows-what till it just went away. Initial API requests from the ESP32 would probably time out, too, thanks to flaky Wi-Fi connectivity; however, that was sorted with a retry logic and error handling on the Arduino side.

To improve accessibility, the UI might adopt a more prominent font, intuitive icons, and color-coded alerts (red for high temperature) to appeal to all users across age groups. Voice-based alerts and simplified touch gestures (like swipe to refresh) could also bolster ease of use. Regarding reliability, offline data between syncs on the ESP32, coupled with a Wi-Fi strength indicator within the app, could go a long way in fending off connectivity concerns.