STTH3113 SENSOR-BASED SYSTEM (A)

SEMESTER 6 (A242)

Title:
Midterm Assignment

Prepared For:
Ahmad Hanis Bin Mohd Shabli

Prepared By:

Name	Matric Number
Shahidatul Hidayah binti Ahmad Faizal	295337

Submission Date: 30th May 2025

1.0 YouTube Link

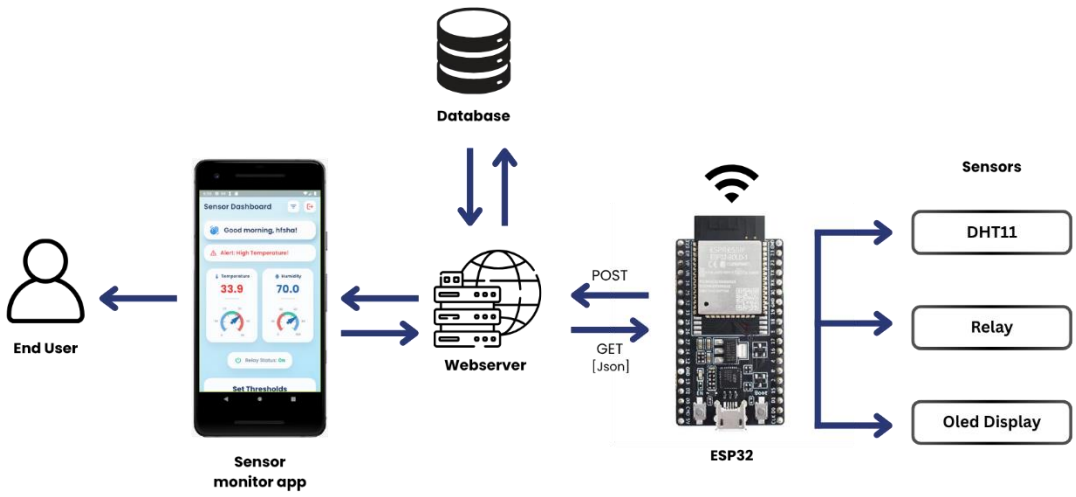
https://youtu.be/MClXsR-XRkg?si=BwqGN61WHNX8Jooh

2.0 GitHub Link

https://github.com/hfsha/Midterm_STTH3113_SensorMonitorApp

arduino	Delete arduino/midterm_dht11
backend	update backend
mobile_app/sensor_monitor_app	updated
report	Create temp
README.md	Initial commit

3.0 System architecture diagram



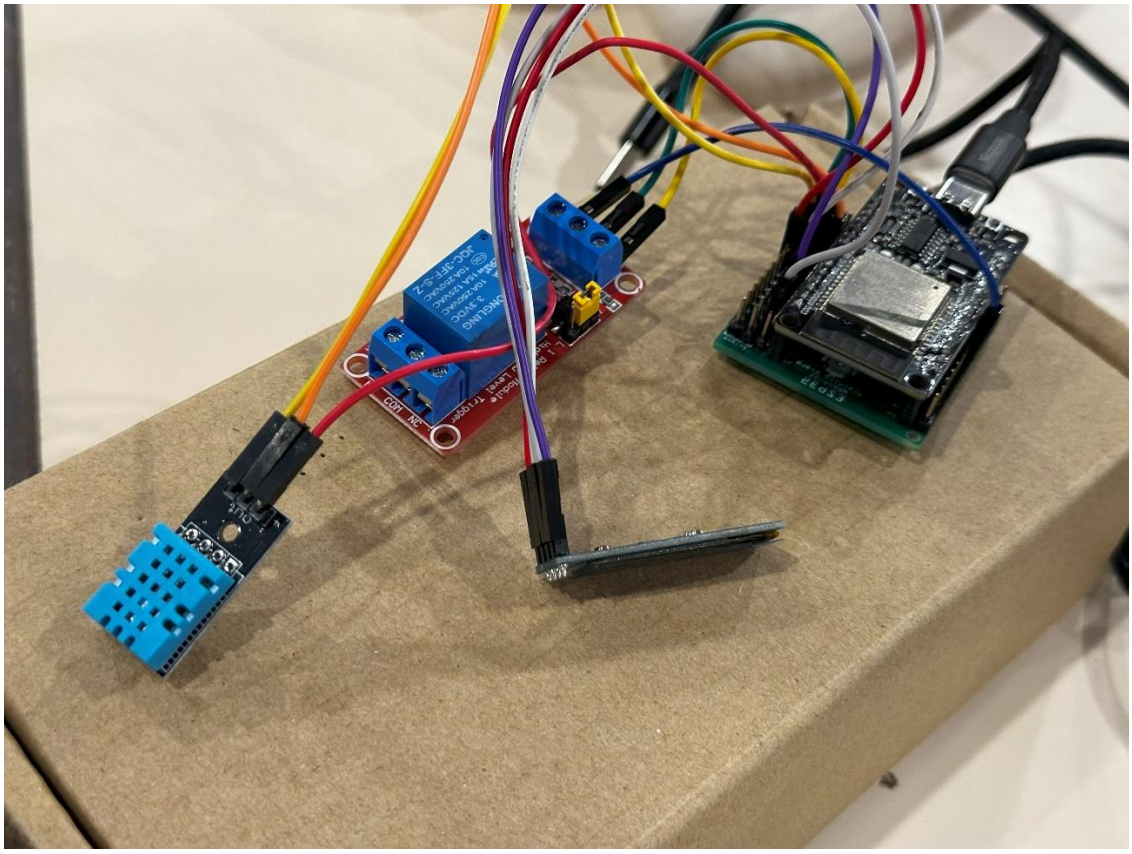
This system uses an **ESP32** to read data from a **DHT11 sensor** every 10 seconds and sends it to a backend server. The server stores the data in a **SQL database** and provides it to a **mobile app** via API.

These are the sensors I used for this midterm assignment:

- **Relay** is triggered if temperature or humidity exceeds user-set thresholds.
- **OLED Display** shows live values on the device. (e.g. Temperature, Humidity, Relay status)
- **Mobile App** (Flutter) shows real-time data and graphs, and allows setting thresholds (updated in the database).

Communication is done via **HTTP (POST/GET)**, enabling near real-time updates and control.

4.0 Setup steps



The connections are as below:

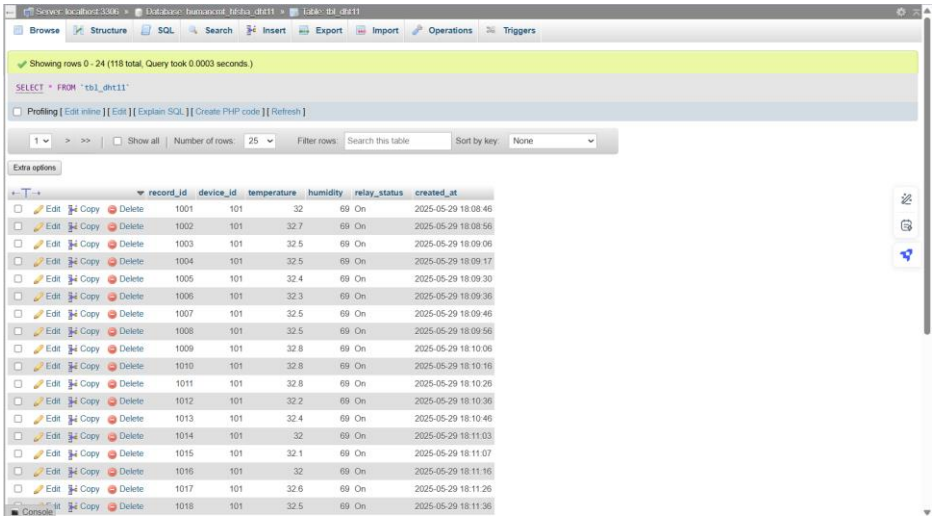
DHT11	OLED display	Relay
• VCC → 3.3V • GND → GND • Data → GPIO4	• VCC → 3.3V • GND → GND • SDA → GPIO21 • SCK → GPIO22	• VCC → 3.3V • GND → GND • IN → GPIO25

5.0 Screenshots

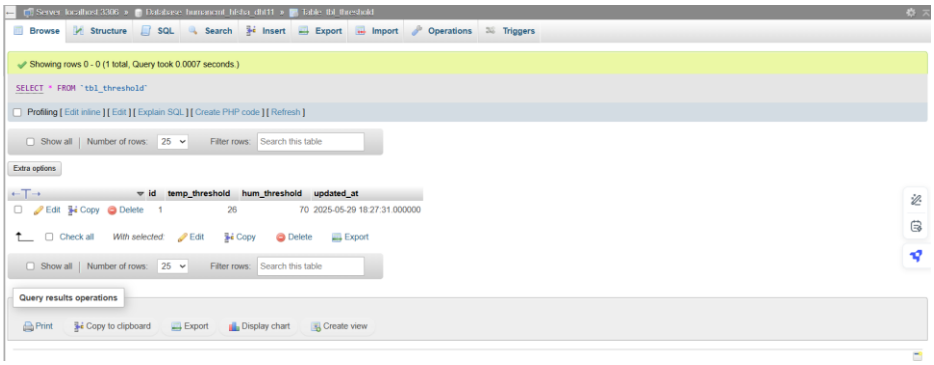
5.1 Backend

5.1.1 Database

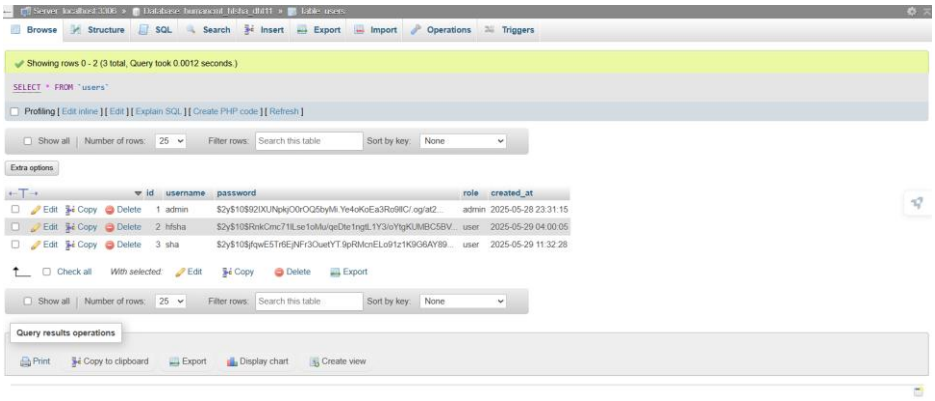
a. tbl_dht11



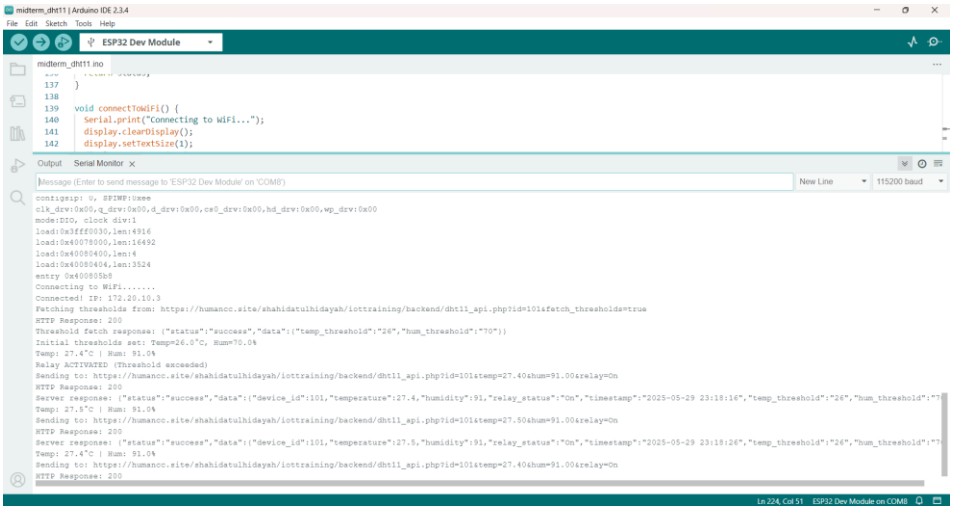
b. tbl_threshold



c. users

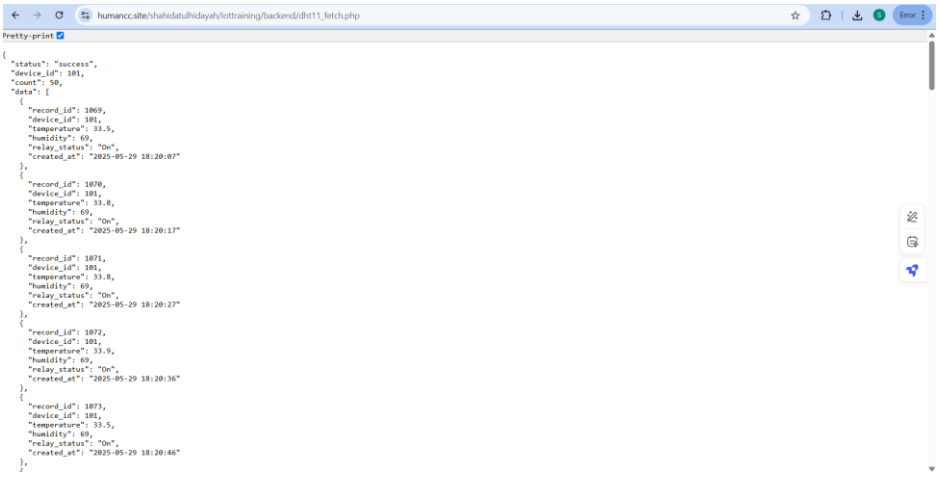


5.1.2 Arduino

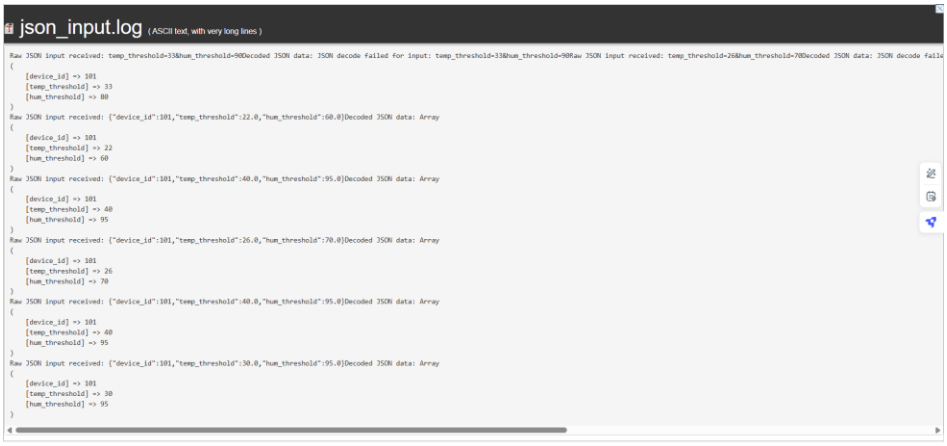


5.1.3 Backend

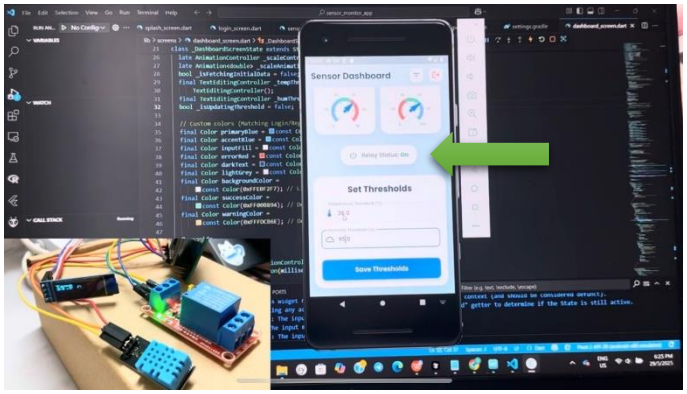
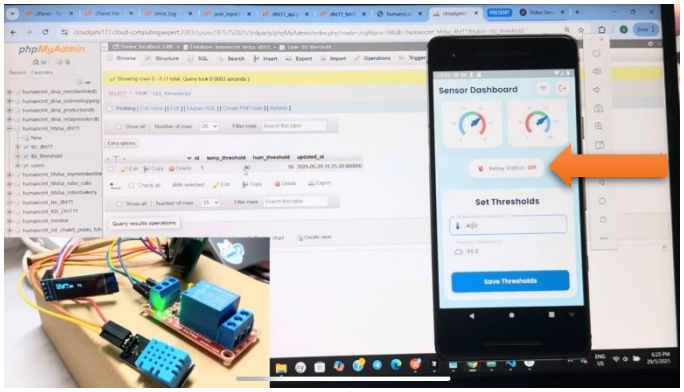
a. Browser



b. Log

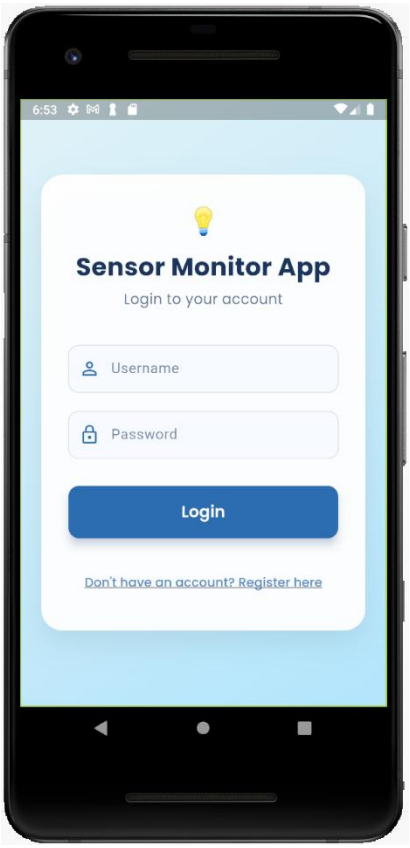


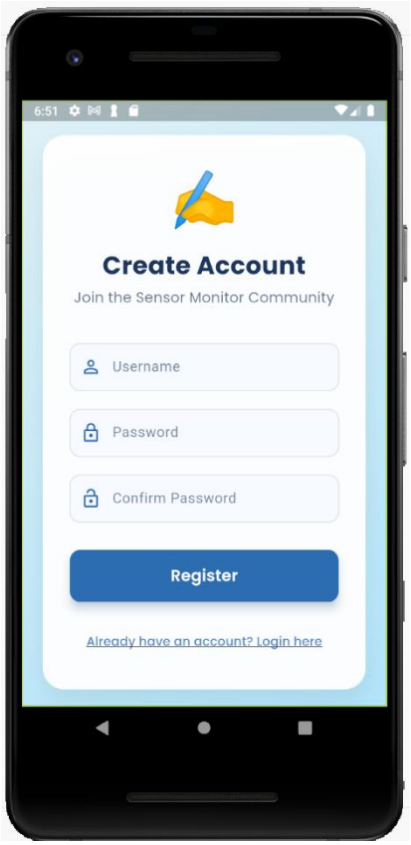
5.2 Relay

Relay Status	Diagram
On	
Off	

5.3 App

5.3.1 Login/ Sign up page

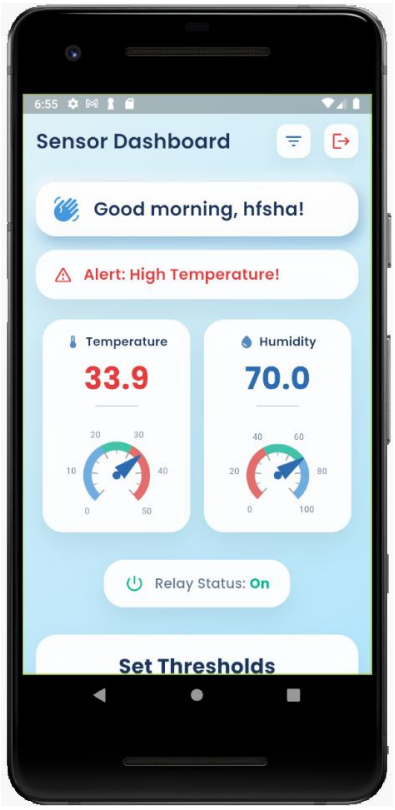


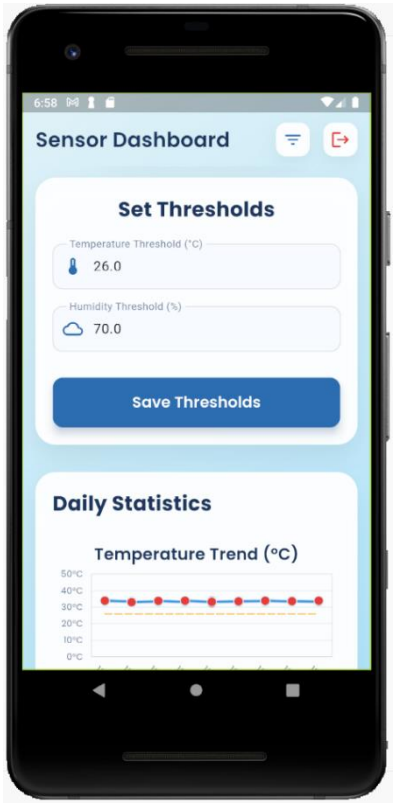


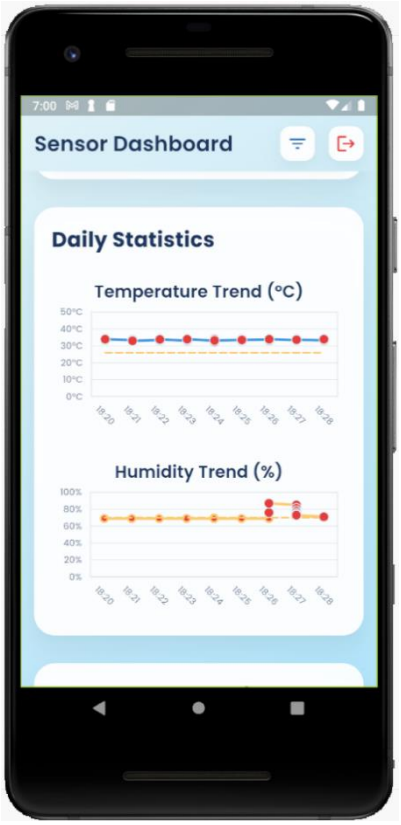
The mobile app features a clean and intuitive user interface starting with a login and registration system. The login screen allows existing users to securely access the app by entering their username and password. For new users, the registration screen provides a simple form to create an account by entering a username, password, and confirming the password.

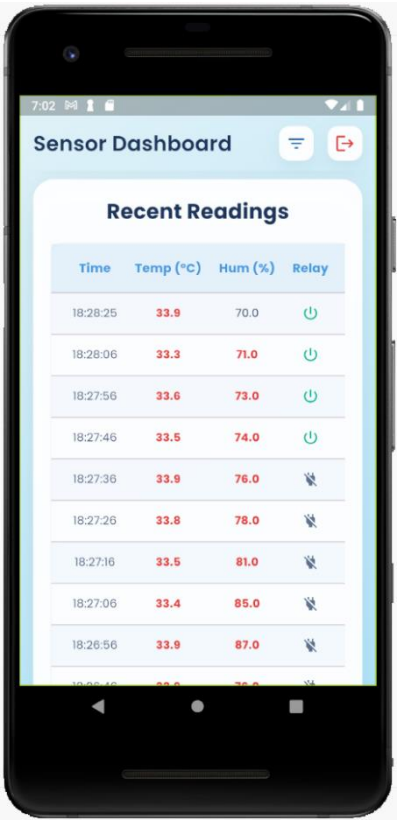
Both screens include navigation links to switch between login and registration. This user authentication process ensures that only authorized users can monitor sensor data and interact with the system features.

5.3.2 Dashboard









The Sensor Monitor App dashboard provides real-time environmental monitoring and control features. The main screen displays current temperature and humidity readings with color-coded alerts and a relay status indicator. Users can set custom threshold values for both temperature and humidity, which are stored and used to trigger the relay automatically. The dashboard also includes graphical trend charts for both temperature and humidity, helping users visualize changes over time. Additionally, a recent readings table displays timestamped logs of sensor data and relay status, allowing for easy tracking and verification of system activity.

6.0 Challenges and improvements

During the development of this project, I faced several challenges. One of the first challenges I faced was dealing with Gradle issues while building the mobile app in Flutter. Sometimes the app failed to compile due to outdated dependencies or version mismatches in the build.gradle files. I also encountered errors related to the Android SDK and emulator compatibility. To solve this, I updated the Gradle version and update java SDK to 17, synced the project again, and made sure all dependencies were using compatible versions. This process took some time around 2 hours and half and it was frustrating, but it helped me better understand how Flutter connects with native Android settings.

Another challenge I faced was connecting the backend to the database using dbconnect.php. At first, I kept getting connection errors because of incorrect database credentials and sometimes due to the server not running properly. I also forgot to enable MySQLi extension in XAMPP, which caused some confusion. After checking the dbconnect.php file line by line and making sure the host, username, password, and database name were correct, I managed to fix it. I also tested the connection separately before using it in the main API files to make sure everything worked smoothly.

On the mobile app side, it was a bit difficult to show real-time graph updates. I had to learn how to fetch data in intervals and refresh the UI without causing performance issues. Eventually, I used a timer to update the data every few seconds and made sure the graphs updated smoothly.

Lastly, styling the app to look clean and modern took extra time, but it helped improve user experience. If I had more time, I would add login token security, notification alerts, and maybe support for multiple sensors in one app such as Fan.