**Thermo-Track: Smart HVAC Monitoring System**

(SD3A: Moses-Ailemen David Ibukunoluwa, Eyob Gebrekidan, Tanatswa Ben Musemburi)

# Project Description

**Thermo-Track is a smart monitoring system designed for educational institutions to optimize HVAC performance based on real-time occupancy and conditions.**

Instead of directly controlling the Building Management System (BMS), Thermo-Track uses sensor data to generate smart, prioritized recommendations for condition adjustments. The project has been scaled down to focus on the educational environment where the balance between occupant comfort, energy efficiency, and limited user control is most critical.

**Environment:** Educational Establishments

We target universities, secondary, and primary schools where occupants (students/teachers) have no direct control over room temperatures but are significantly impacted by poor conditions.

* **Goal:** To create a self-regulating environment where comfort issues are solved *before* they lead to complaints or manual intervention.
* **How:** By continuously monitoring room occupancy and feeding real-time adjustment suggestions to the central BMS, Thermo-Track ensures spaces are never too hot or cold when occupied, and no energy is wasted on empty rooms.
* **Result:** Those with control (Facilities Managers) can direct energy efficiently, while occupants (Lecturers/Students) enjoy consistent comfort without needing to intervene.

**Key Users**: Roles and Access

The system's full functionality is limited to staff responsible for the building's operation and maintenance.

* **Primary Users (Full Access):** **Facilities Managers** use the system to monitor data, set energy policies, manage equipment, and approve system recommendations.
* **Secondary User (Feedback Access):** **Lecturers/Teachers** act as key witnesses and bridges for student comfort. They are given a simple interface to provide real-time feedback (e.g., "Too Warm") or suggest temperature preferences, which inform the Facilities Manager's actions and future system policies.

# Personas and Use Cases

Given the refined scope, here are three tailored personas representing the key users, along with a specific use case for each.

**Persona 1:** Facilities Manager — Daniel Murphy

|  |  |
| --- | --- |
| **Detail** | **Description** |
| **Focus** | **Energy Efficiency & Cost Control** |
| **Context** | Daniel manages HVAC for the entire university campus. His core mandate is to cut the large energy bill while avoiding comfort-related staff/student complaints. |
| **Pain Point** | Rooms often retain climate control schedules from previous semesters, leading to AC running in empty lecture halls, resulting in high waste and no data to justify changes. |
| **Quote** | *"I need data that proves exactly how much money I'm saving by cooling an empty room, so I can secure more budget."* |
| **Top Task** | Identifying high-waste zones and setting campus-wide HVAC setback policies. |

**Use Case:** Automated Energy Setback

1. **Goal:** Reduce energy waste in rooms that are empty between scheduled classes.
2. **Scenario:** Daniel uses the Thermo-Track dashboard to define a policy: if occupancy is zero for 15 minutes in a scheduled-to-be-occupied room, the system recommends a temperature setback to the BMS.
3. **Action:** The system applies the setback. When occupancy returns (a student arrives early), the system instantly recommends returning to the comfort setpoint.
4. **Result:** Daniel downloads a monthly report showing a **reduction in energy** across non-utilized spaces, which he uses to justify the project's return on investment.

**Persona 2:** Room Host (Lecturer) Lucy Lue

|  |  |
| --- | --- |
| **Detail** | **Description** |
| **Focus** | **Immediate Comfort Feedback & Class Focus** |
| **Context** | Lucy teaches several classes a day. She is a secondary user with no access to the thermostat, but she's the first person to hear about comfort issues from students. |
| **Pain Point** | Students complain about the room being too cold or hot, but she has no way to fix it quickly. When she calls facilities, it takes hours for a technician to respond, disrupting her lesson. |
| **Quote** | *"I need a quick, simple way to tell the building something's wrong so I can get back to teaching."* |
| **Top Task** | Submitting comfort feedback during a lecture without disrupting the class. |

**Use Case:** Low-Friction Comfort Nudge

1. **Goal:** Quickly communicate a comfort issue to the facilities team without making a phone call or leaving the class.
2. **Scenario:** Ten minutes into a lecture, students indicate the room is getting too warm. Lucy opens the Thermo-Track mobile interface via a dedicated room QR code.
3. **Action:** She sees the live room temperature and taps the **"Too Warm"** button. The system registers this as a priority **comfort feedback point** and automatically sends a recommendation to Daniel's team to slightly adjust the setpoint.
4. **Result:** Lucy gets an instant confirmation: "Adjustment requested and is being reviewed by Facilities." The room temperature stabilizes within 15 minutes, allowing Lucy to **maintain focus** on her class.

# User Flow

**Task Analysis**

This task analysis identifies the goals and the actions users (Lucy Lue - Lecturer; and Daniel Murphy - Facilities Manager) must take within the system to achieve them, based on their defined user roles and access levels.

**1. Persona:** Lucy Lue(Lecturer / Room Host)

**Access Level:** Basic User (Limited, feedback-only access via mobile/QR code)

|  |  |  |
| --- | --- | --- |
| User Type | Goal Description | Task Progression |
| Basic User | **Submit Comfort Feedback (Too Warm/Cool)** | Scan Room QR Code -> Mobile UI Opens -> View Live Temp/Status -> Tap "Too Warm" or "Too Cool" (Optional) Add Comment -> Tap Submit -> Receive Confirmation/ETA. |
| Basic User | **Check Status after Submission** | Scan Room QR Code (or stay on page) -> View status message (e.g., "Adjustment in Progress: ETA 5 min"). |
| Basic User | **Provide Feedback on Change** | Scan Room QR Code -> View prompt (e.g., "Did the change help?") -> Select "Better" / "Worse" / "No Change" -> Tap Submit. |
| Basic User | **View Current Room Conditions** | Scan Room QR Code -> Mobile UI Opens -> View Live Temperature and Comfort Indicator |

**2. Persona:** Daniel Murphy (Facilities Manager)

**Access Level:** Admin (Full Control)

|  |  |  |
| --- | --- | --- |
| User Type | Goal Description | Task Progression |
| Admin | **Review Energy Waste/Savings** | Log In -> Navigate to Dashboard -> Filter by "High Waste Zones" -> View Utilization vs. Condition Status -> Review Savings Metric. |
| Admin | **Set Occupancy Setback Policy** | Log In -> Navigate to **Rules Engine/Policies** ->Create New Policy Define Condition (e.g., "Occupancy = 0 for 15 min") -> Define Action (e.g., "Set Back Temp ") -> Apply to Zone/Building. |
| Admin | **Approve Lecturer's Comfort Nudge** | Log In -> View **Notification Panel** (or Alert Feed) -> See "Comfort Nudge from Lucy Lue in L-Hall 3" -> Review Room Data (Temp, Occupancy, Setpoint) -> Select "Approve Nudge" or "Dismiss." |
| Admin | **Integrate with BMS** | Log In -> Navigate to Settings -> Select **BMS Integration** -> Input BMS API/Protocol Details (e.g., BACnet IP) -> Map Thermo-Track Setpoints to BMS Registers -> Test Connection. |
| Admin | **Monitor System Health (Sensors/BMS)** | Log In -> Navigate to **Device Health** tab -> View Status of Sensors (Online/Offline/Battery) and BMS Connection -> Acknowledge/Assign Error Alerts. |

# The UI

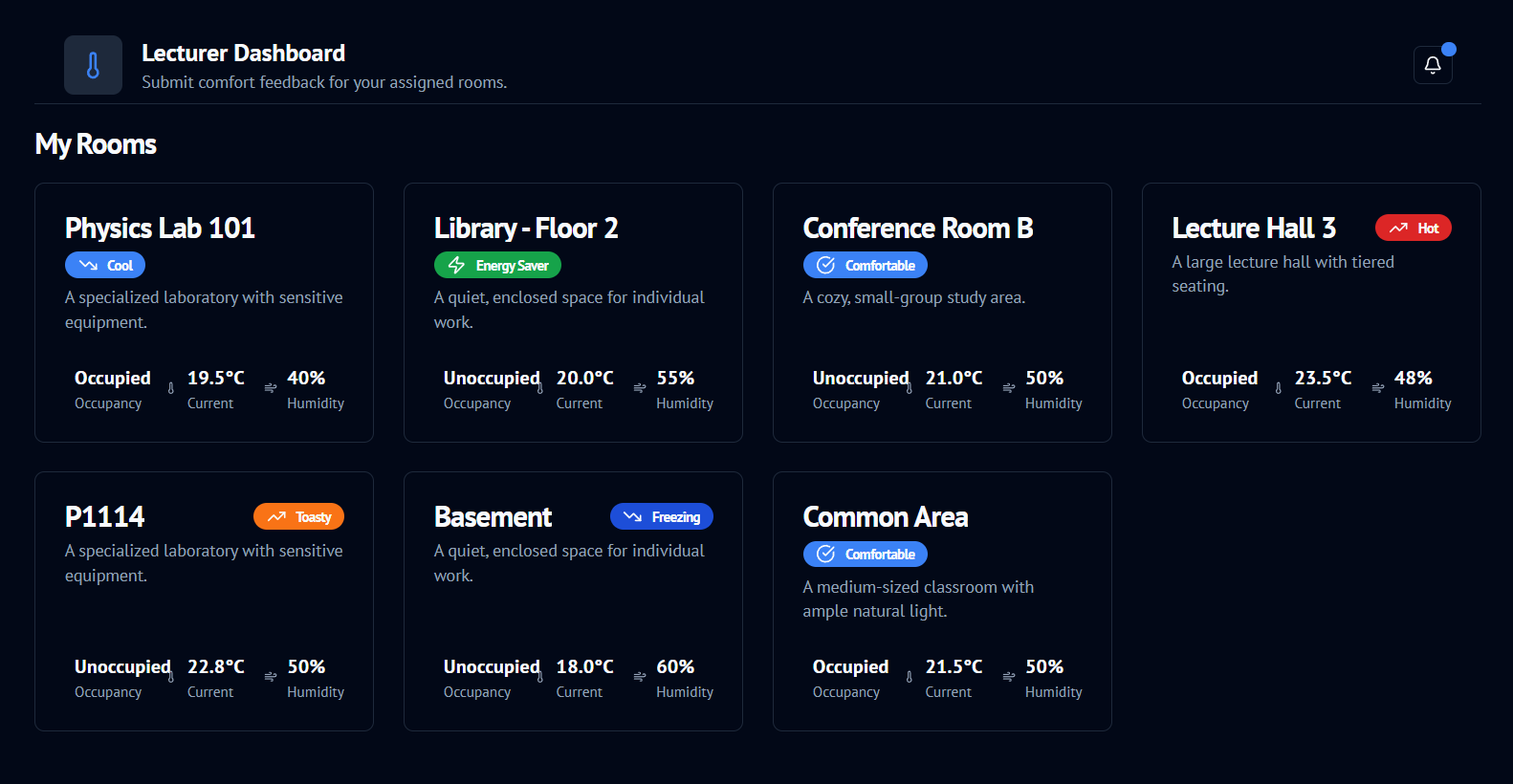
These mock ups show the flow of the system when being used by a restricted user and the by an administrator. The data they both have access to is that which is return by the sensors. The passive infrared sensor returns activity in the room which determines its occupation status. While the DHT22 returns the temperature and humidity. Knowing the room conditions allows the occupants to return feedback based on the current conditions e.g. request for lower the room temperature if the room is too warm. Occupancy and room conditions allows the administrator to set the appropriate conditions for the room according to its state e.g. Turn off the heating in an empty room. The admin deals with the user’s feedback, approving or dismissing room requests. This extra data helps them set favourable conditions for the occupants.

The distinction between admin and regular user will be made upon logging in as their credentials will limit or permit certain access after being validated.

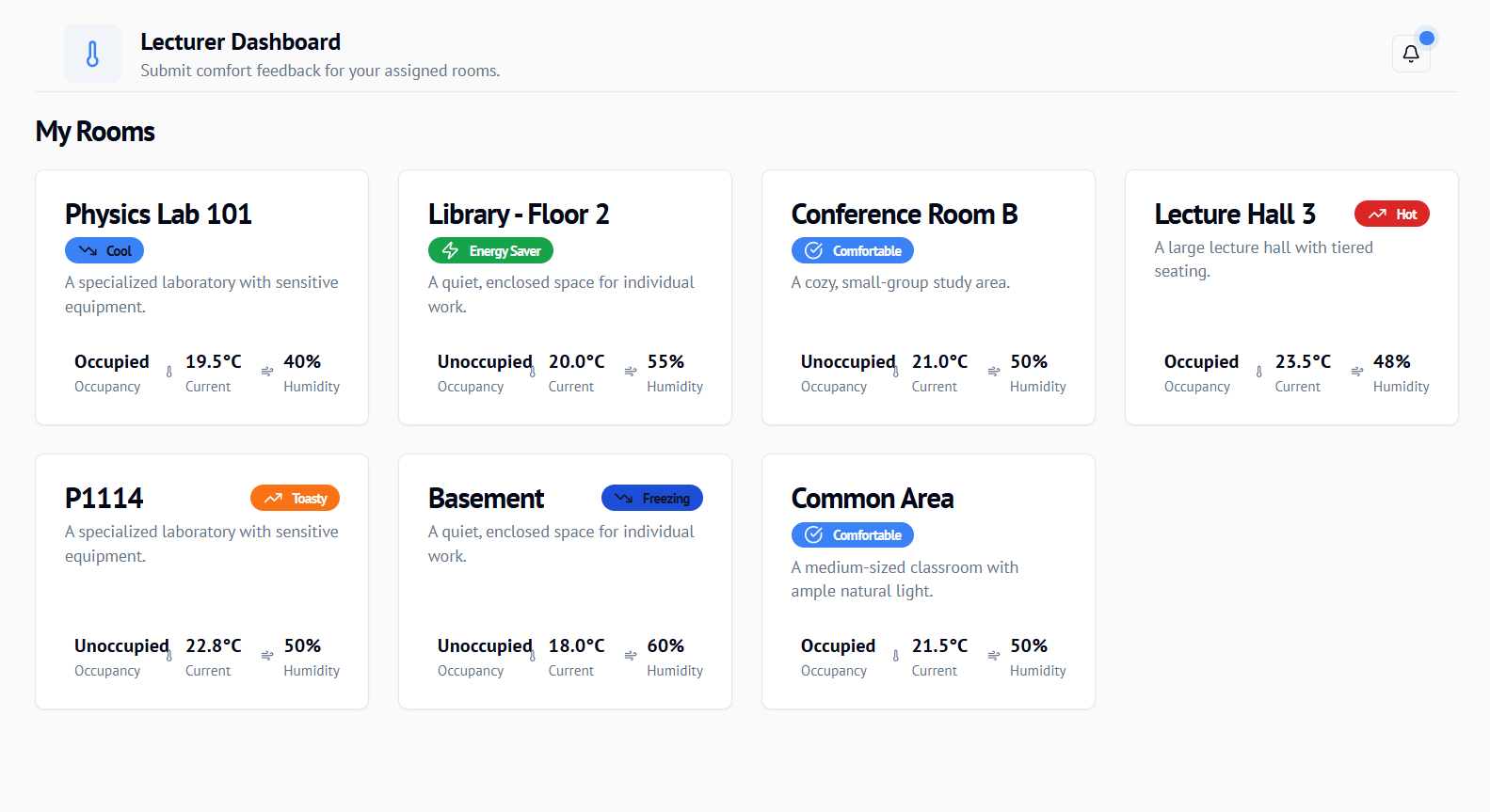
**Mock ups**

Main screens for the lecturer

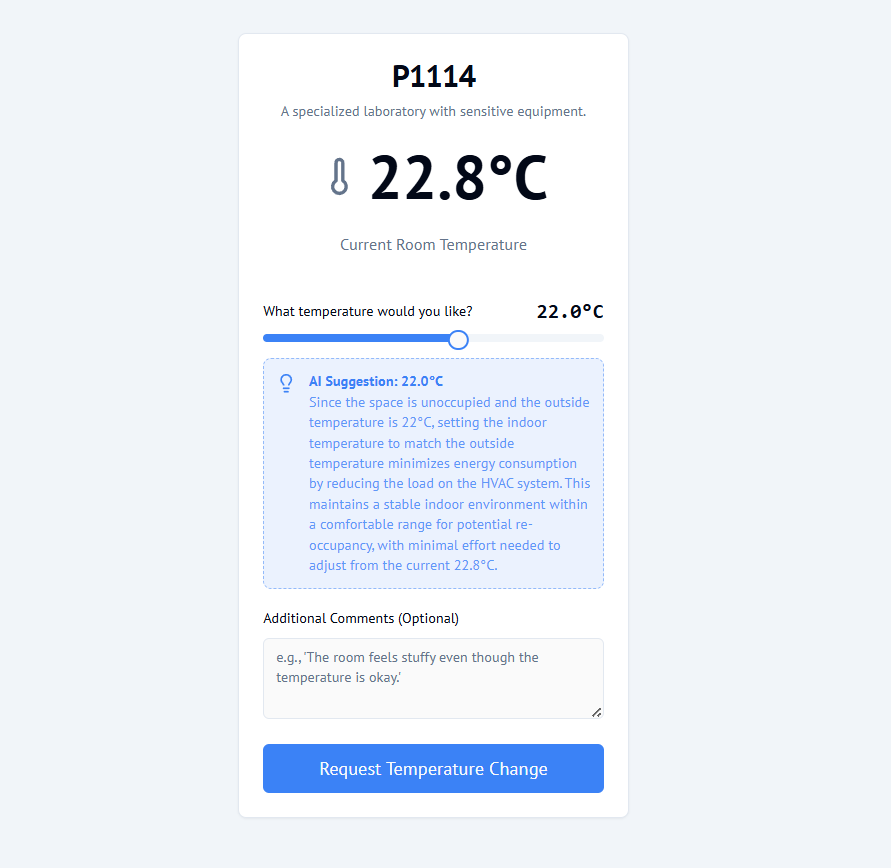
Dashboard showing rooms



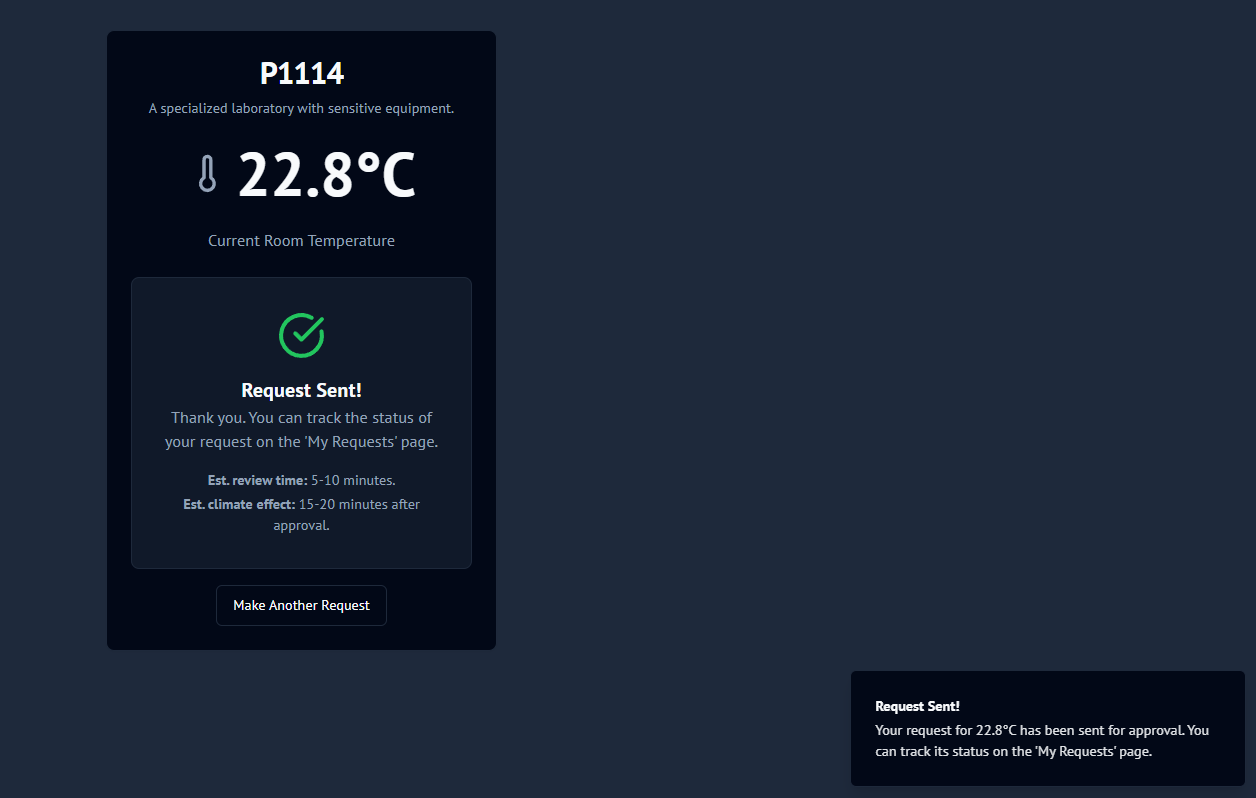
Lightmode:



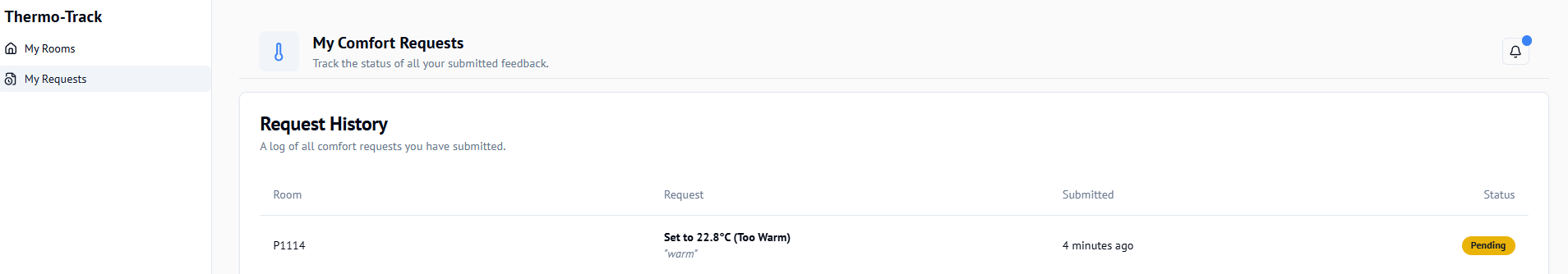
Room 114 was selected to give feedback on the room’s conditions. AI suggestions to help the user make a better decision since most will not know what temperature to input and what it will feel like when applied.



Dark mode view: The request is made for the room adjustments



Tracking the request



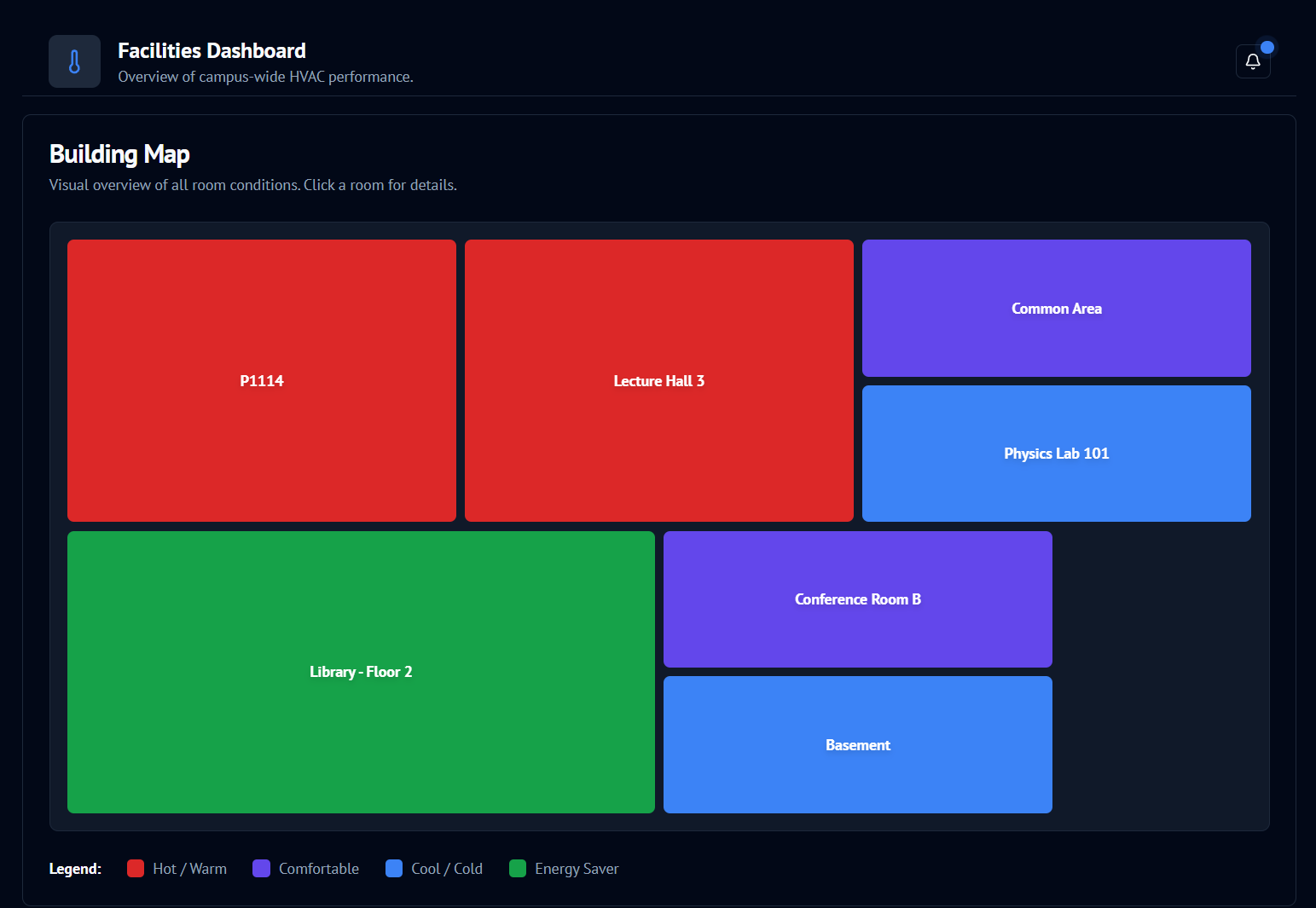
Main screens for the Facilities Manager (Admin)

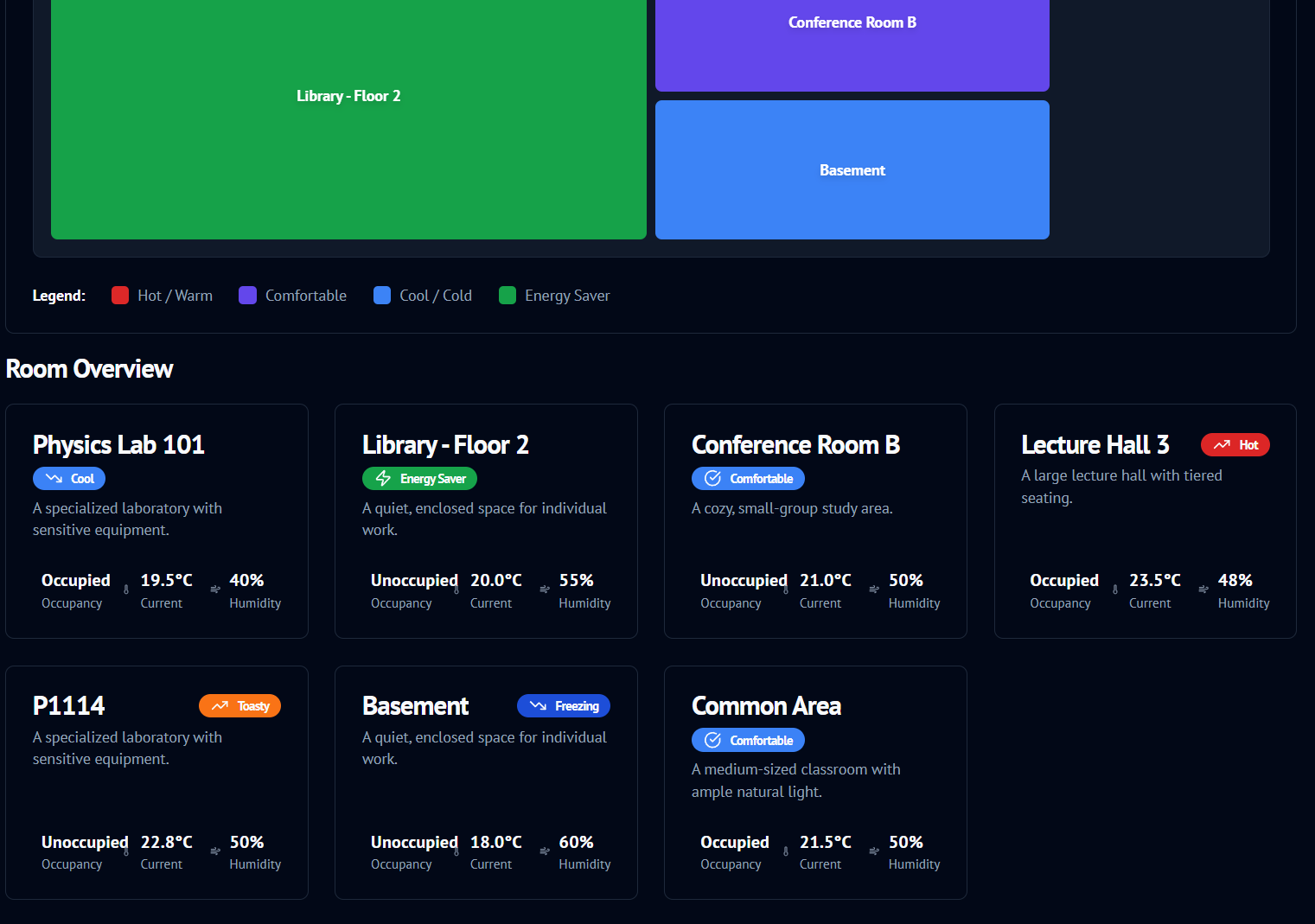
The dashboard for the admin shows all the rooms as well as other features that come with full access (setup, reports, policies, settings).

A screenshot of a computer

AI-generated content may be incorrect.

(slightly developed prototype that shows an overview of the building layout)

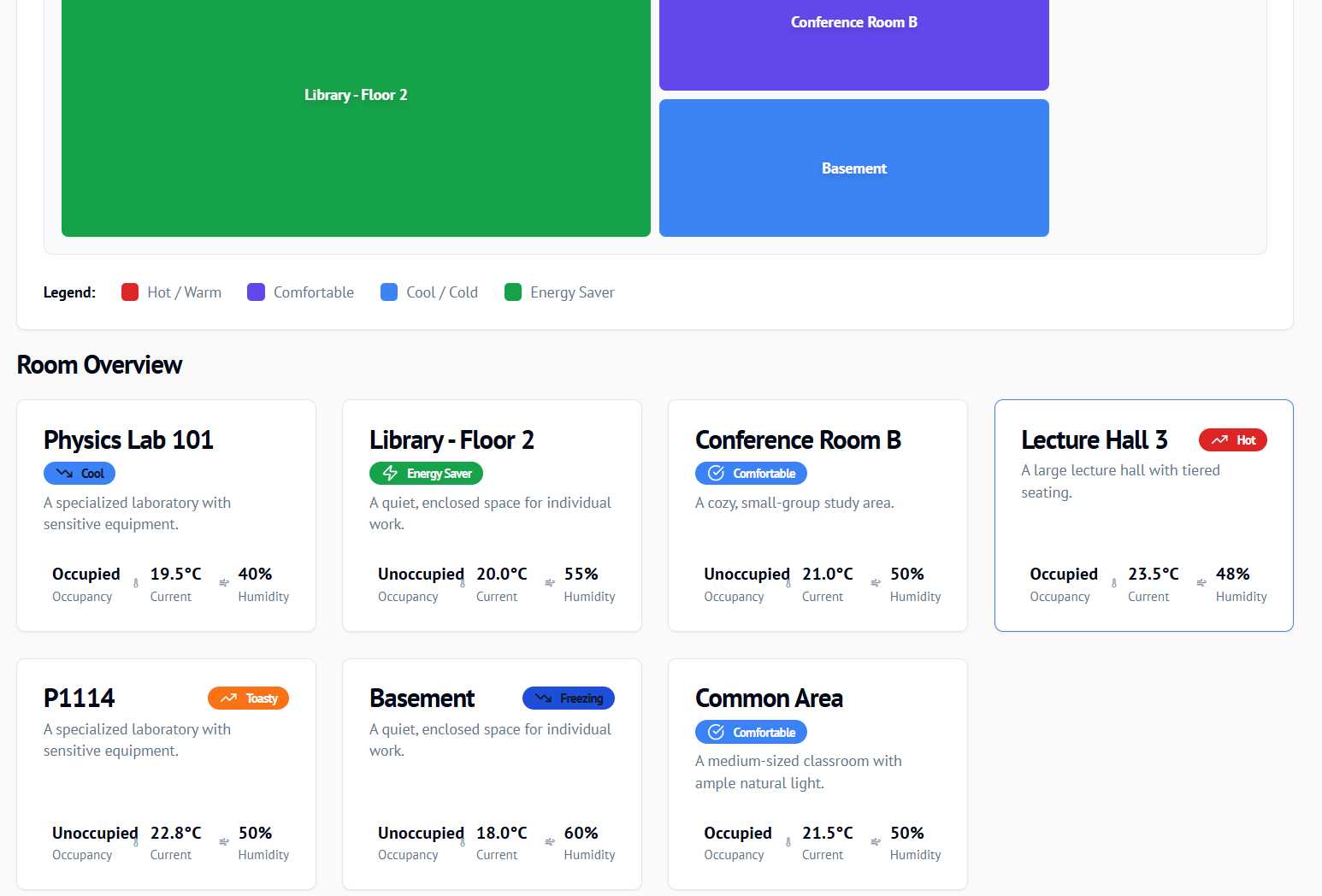




The dashboard in light modeA screenshot of a computer

AI-generated content may be incorrect.

Improved version showing more room detail (the building overview above)

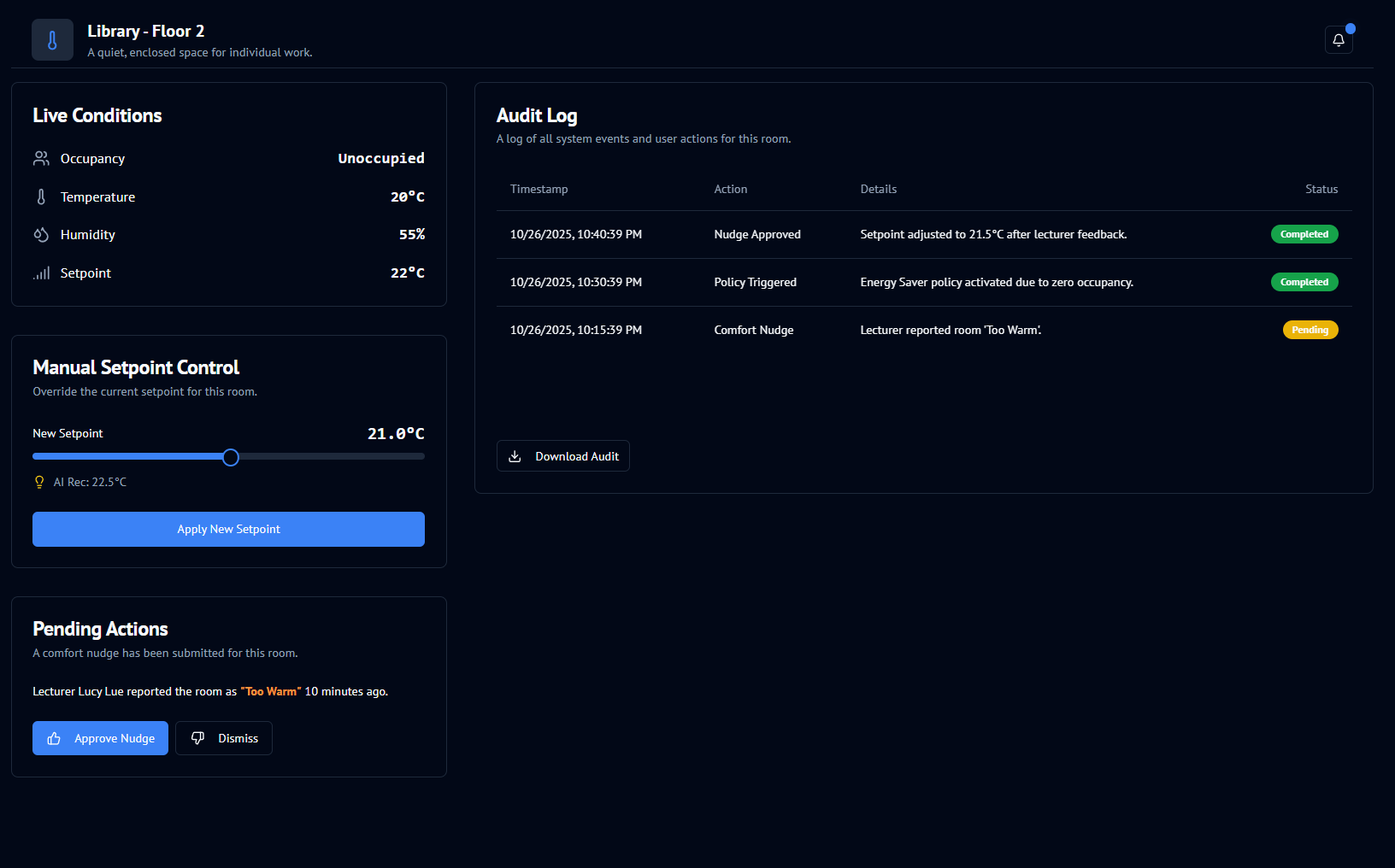


A single occupancy area selected then shown in detail (library floor 2)

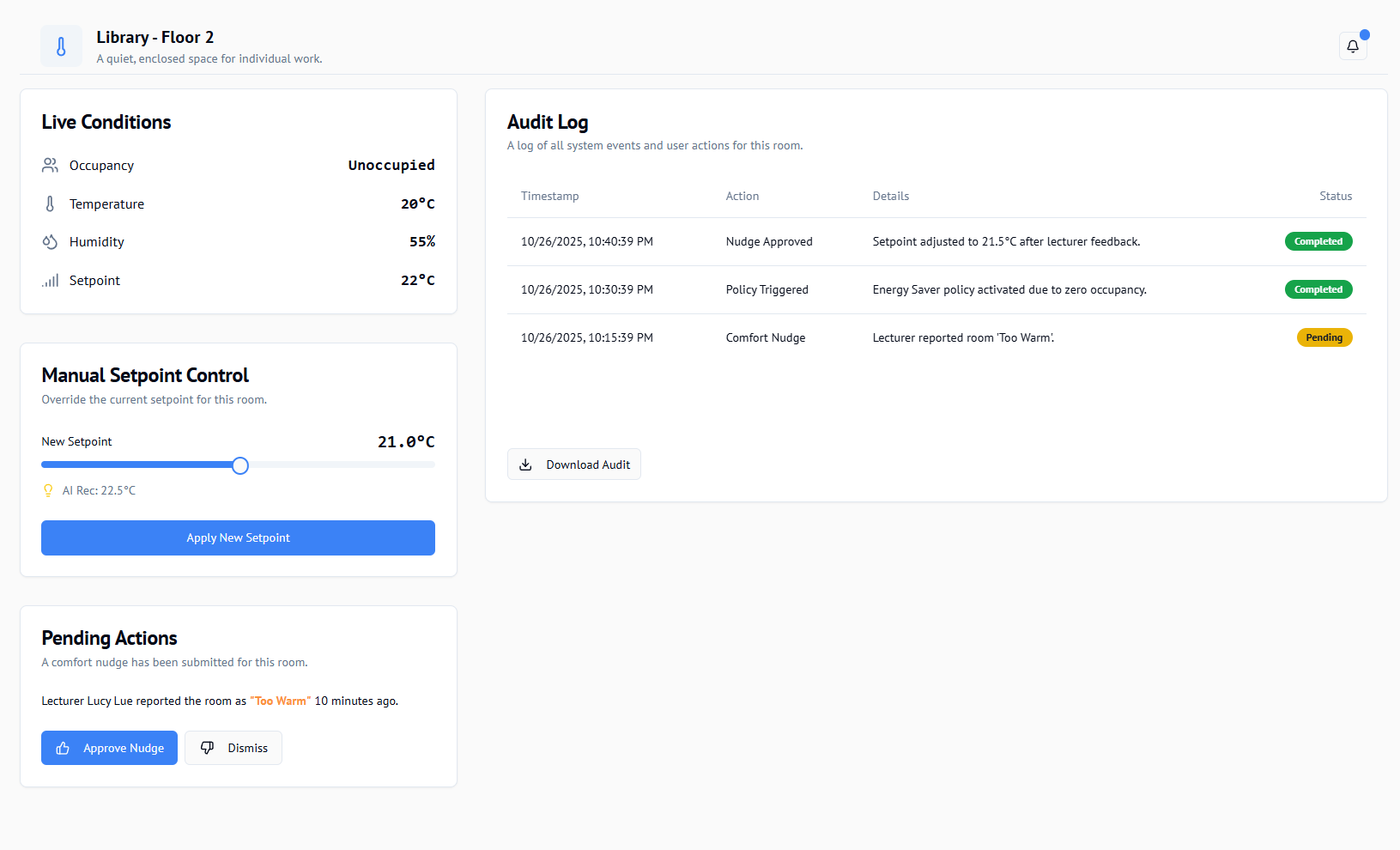
A screenshot of a computer

AI-generated content may be incorrect.

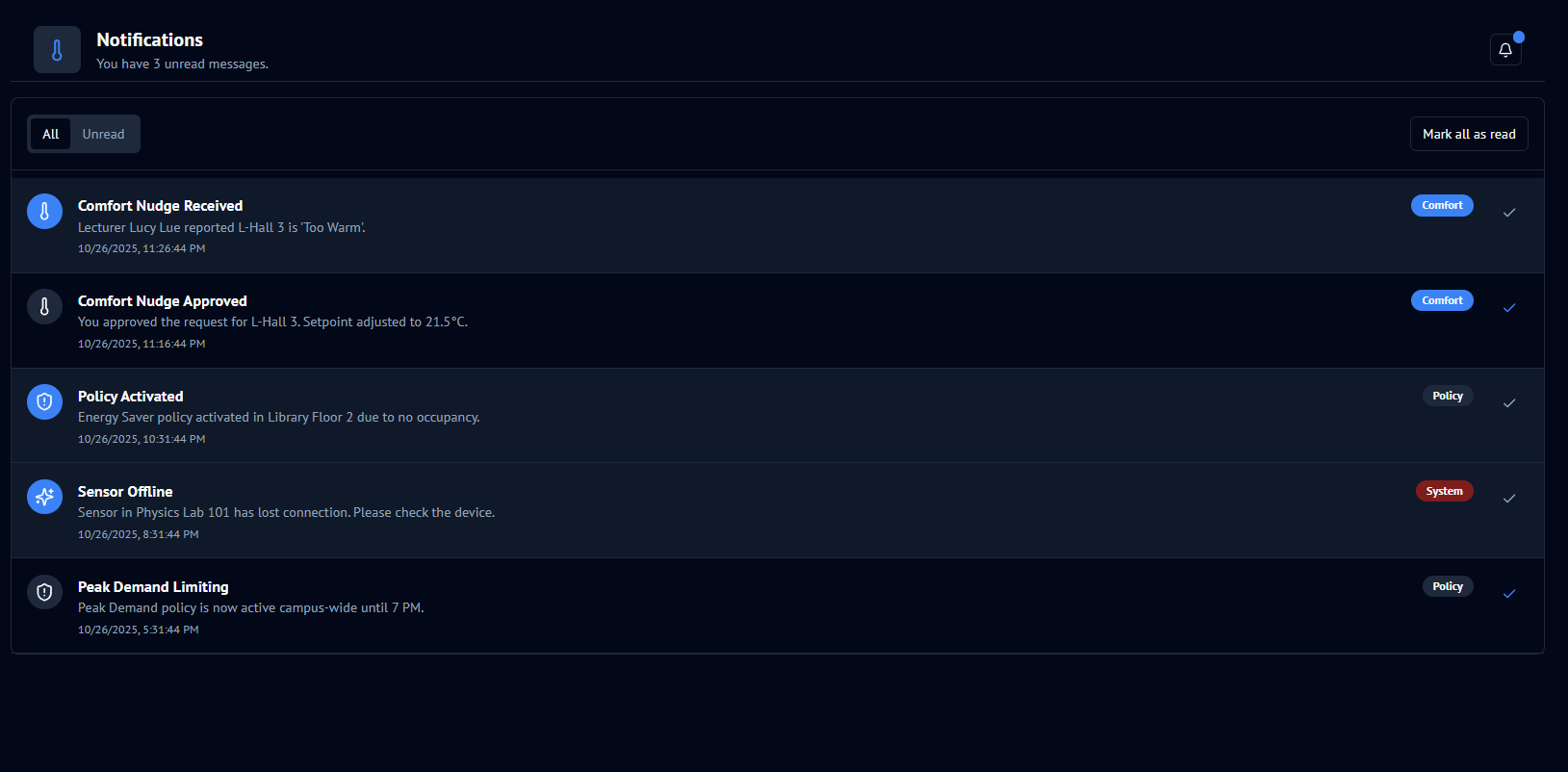
Improved version with a slider for quick manual configuration, accompanied by AI suggestion that recommends the best adjustment for the room



Display in light mode



Notifications for room requests



Screen for adding a new room

A screenshot of a computer

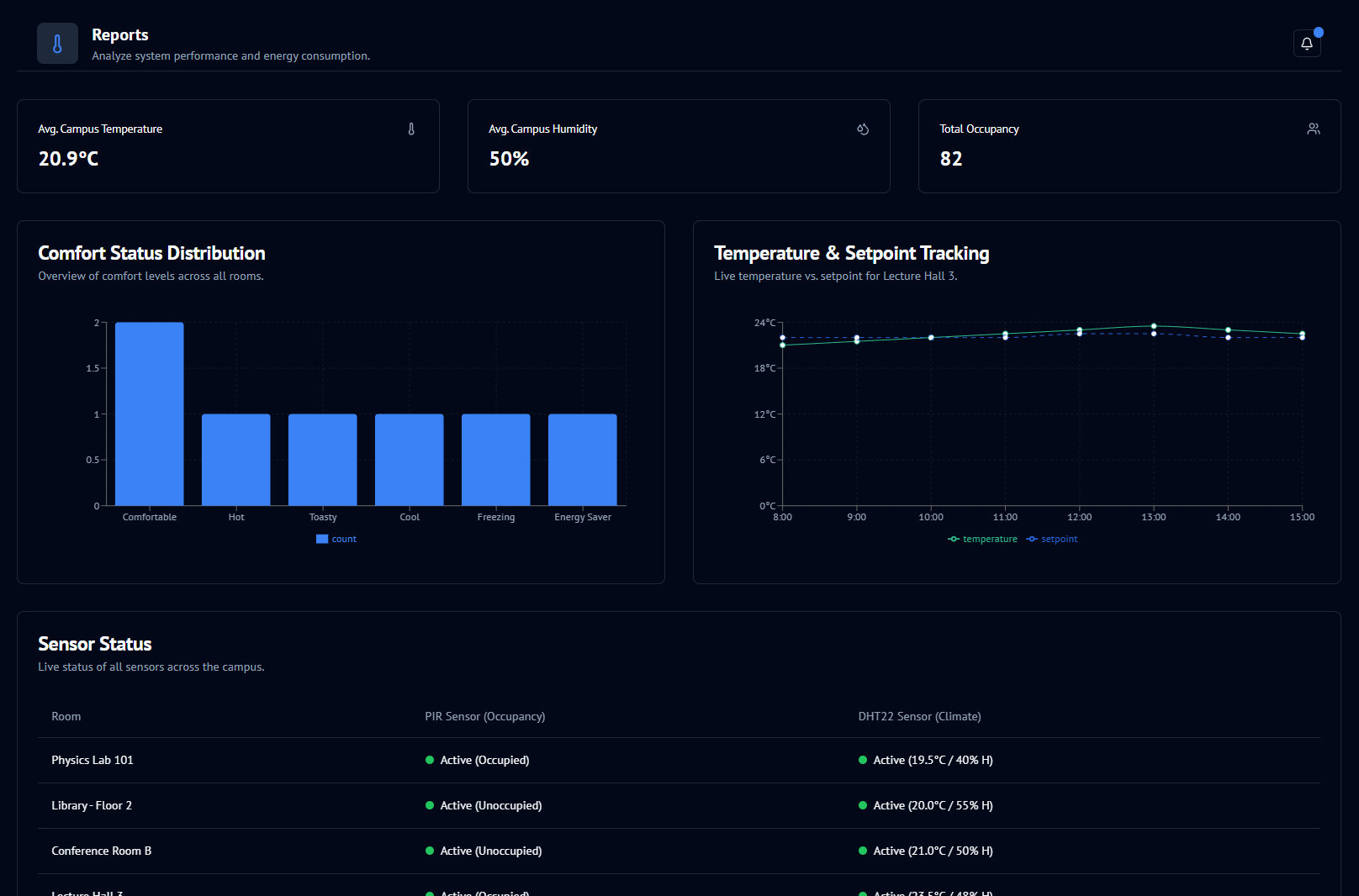
AI-generated content may be incorrect.

Screen to display reports (shows overview of the rooms being tracked + sensors)

A screenshot of a computer

AI-generated content may be incorrect.

V2 (shows the additional room descriptions + sensors’ status’



Screen showing policies

A screenshot of a computer program

AI-generated content may be incorrect.

Settings display

Screens screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

# Security Detail

**Data Security**

All data sent between the Thermo-Track device, web app and central server is protected.

* Data is encrypted so nobody can read it during transmission.
* PubNub is used for secure real time data transfer between IoT devices and the web app. PubNub built in end to end encryption, token based authentication and data integrity validation.
* Aws IotCore ensures devices are authenticated certificates before connecting to the cloud. Preventing fake or unauthorized devices from communicating.
* User’s logins on the web application are also protected HTTPS and secure cookies.

**Data Security During Storage**

When data reaches the server, it is stored safely in secured database (MariaDB)

* Sensitive information such as passwords is hashed and salted using algorithms like bcrypt so it cannot be read even if the database is compromised.
* All sensors and occupancy data stored AWS (MariaDB) is encrypted at rest using AWS manged keys.
* Data backups are encrypted and stored securely to prevent data loss.
* Access to the database is limited to authorized system components only.

All user and sensor data is stored following principle of the least privilege ensuring only required systems can access specific information.

**Application level Security**

* User’s roles determine what each user can view or modify such as Admin or standard users.
* Session management ensures that users are automatically logged out after inactivity to reduce unauthorized access.
* All web forms and API endpoints include server-side validation to ensure the data being sent is valid and secure.

**Privacy and Data governance**

* Only the minimum amount of data necessary for operation is collected.
* Access to personal or sensitive data is logged and reviewed periodically.
* All user actions and systems changes are logged for auditing and accountability.

# Thermo-Track Testing Plan

The Thermo-Track system will undergo both hardware and software testing to ensure the system performs accurately, securely and reliable.

The goal is to confirm that the device correctly detect occupancy, monitors environmental conditions and provides accurate smart recommendations for HVAC adjustments while maintaining user security and data privacy.

**Testing Methods**

Unit Testing (Software)

Each software component of the Thermo-Track web application and device logic will be tested individually to ensure it functions as expected.

Examples include:

* Testing login and registration forms for valid and invalid points.
* Verifying database connection and data retrieval

Success Criteria:

* All functions run without errors
* Input validation correctly blocks invalid data.
* Data is accurately stored and retrieved from the database

**Integration Testing**

Once individual parts work they will be tested together to ensure that smooth communication between:

* The device raspberry Pi and the web server.
* The database and the web interface.

Success Criteria:

* Data from the sensors is correctly sent to the server and displayed in the dashboard.
* User inputs correctly update the database.
* Communication remains stable with no data loss or timeouts.

**System Testing**

This will test the entire system Thermo-Track system from end to end. Combining hardware, network and software.

**User Testing**

This involves testing Thermo-Track with real users like students or staff to ensure it is easy to use, accessible and meets the intended goals.

* The web interface usability and accessibility.
* The accuracy of occupancy and comfort recommendations.

Success criteria:

* Feedback confirms that the system provides useful and understandable recommendations.
* Users can complete main task’s view data, adjust preferences, read recommendations without help.

# Thermo-Track Hardware detail

Thermo-track: A universally designed system that monitors room occupancy and conditions (temp adjustments) to provide **smart recommendations** for HVAC systems, instead of direct control. In a situation where adjustments are still desired, the system provides an easy and convenient way to change room HVAC settings or request alterations.

**Component Breakdown**

**Controller (Logic)**

* **Raspberry Pi 400:** This is the high-performance central processing unit. Since it's built into a keyboard, it offers a complete, robust platform for running the Python monitoring script, connecting to the internet via Wi-Fi or Ethernet, and managing the GPIO interfaces

**Sensors (Input Data)**

These components collect data about the environment and feed it back to the Thermo-Track application.

* **DHT22 (Temperature & Humidity Sensor):** Provides the core data for the application's primary function, tracking thermal conditions.
* **PIR Sensor (Passive Infrared):** Detects motion, useful for presence-based policy activation

**Actuators (Output Action)**

These components are controlled by the Thermo-Track policies to regulate or alert based on the sensor data.

* **DFR0332 Fan Module:** A dedicated **cooling actuator**.
* **LED:** A **binary actuator** for visual status confirmation (e.g., policy compliant/non-compliant). It can signal the status of the cooling fan (Green=Off, Red=On) or a compliance alert.
* **Piezo Buzzer:** The **audible alert actuator**, used to signal critical temperature breaches or policy violations.

**Local Interface (Control & Enclosure)**

The ABS Plastic Project Enclosures will house the entire system. The buttons, Fan, and Piezo Buzzer will be mounted directly into cutouts in the enclosure, providing a clean, finished appearance for the Thermo-Track monitoring station.

**Power and Internet Connectivity**

**Power Supply**

* **Controller (Pi 400):** The Raspberry Pi 400 is powered by a **USB-C power supply**. This is required for stable operation and is connected to the dedicated power port on the back. A power bank (<= 5.1v **× 3A**) can be used to power the pi 400 in the absence of a powered socket.
* **Hardware Power:**
  + **Low-Power Components (Sensors, LED, Buzzer):** These are powered safely and directly by the Pi's internal voltage regulators through the **3.3V GPIO Pin**.
  + **DFR0332 Fan Module:** The fan module requires external power that may be greater than the current limit of the 3.3V pin. We will use the Pi's **5V GPIO Pin** to power the module, and the Pi's **3.3V GPIO Pin** will safely send the low-voltage control signal to the module's dedicated signal pin. The module's internal circuitry handles the switching of the higher 5V current, protecting the Pi.

**Internet Connectivity**

* The Pi 400 is an extremely reliable network device, offering two options:
  + **Wi-Fi (Wireless):** The Pi is configured to join the local Wi-Fi network using the graphical interface or command line tools.
  + **Ethernet (Wired):** The Pi 400 also has a full-sized Ethernet port, providing the most stable and low-latency connection, which is ideal for a critical monitoring application like Thermo-Track.
* **Communication:** Once connected, the Python script running on the Pi uses standard network libraries to send sensor data (Temperature, Humidity, Motion, Light) as **HTTP POST requests** over the internet to the remote Flask application. The application then uses the same network connection to send **HTTP GET/POST control commands** back to a server running on the Pi when an actuator needs to be triggered.

# Thermo-Track Data detail

**Data captured, storage, and processing**

**Sensors & sampling**

* **DHT22 (temp/humidity)**
  + Sampling: **every 30s** (configurable 10–60s).
  + Payload fields: temperature\_c (float), humidity\_pct (float).
  + Local sanity checks: drop values outside −20 - 60 °C or 0 -100 % RH.
* **PIR motion**
  + Event-driven; sampled every **1s** to debounce.
  + A **motion window (e.g., 15s)** sets motion\_detected = 1 on the next reading row for the same device (cheap to store).
* **Device heartbeat**
  + Every **60s**; updates devices.last\_seen\_at.

**End-to-end data flow**

1. **Acquire (Edge)**
   1. DHT22 (temp/humidity) every **30s**; PIR motion debounced at **1s** and folded into the next reading.
   2. Edge validates ranges (−20…60 °C, 0–100 % RH) and adds device\_serial + UTC timestamp.
2. **Transmit (Edge → Server)**
   1. HTTPS POST /api/v1/ingest with JSON:  
       {"device\_serial":"TT-DHT-003","ts":"2025-10-20T10:38:00Z","temperature":22.8,"humidity":47.2,"motion":0}
3. **Ingest (Server)**
   1. Resolve device\_id by serial; reject unknown devices.
   2. Insert into readings(device\_id, temperature, humidity, motion\_detected, recorded\_at) using parameterized SQL.
   3. Update devices.last\_seen\_at = NOW().
4. **Process (Near-real-time + scheduled)**
   1. Lightweight rules evaluate recent data for alerts and occupancy (see cron).
   2. Dashboard queries recent windows (last 5–30 min) and hourly rollups for charts.
5. **Store & Retain**
   1. Raw readings: keep **90 days**, roll up hourly/day summaries, then archive or purge per policy.
   2. alerts kept **1 year**, audit\_log **6–12 months**.

**Cron jobs (Linux) / Task Scheduler (Windows)**

All jobs are **idempotent** (safe to retry) and log to audit\_log.

**1) Housekeeping / Device health (every 5 min)**

* **Purpose:** mark offline devices, backfill last\_seen, clear stale temp files.
* **Logic:** devices.status='inactive' if last\_seen\_at < NOW()-INTERVAL 15 MINUTE.
* **Cron:** \*/5 \* \* \* \* /usr/bin/python3 tasks/housekeeping.py

**2) Occupancy rollup (every 5 min)**

* **Purpose:** derive recent occupancy per room from PIR motion.
* **Logic:** if any readings.motion\_detected=1 in last 10–15 min → “occupied\_recently”.
* **Cron:** \*/5 \* \* \* \* /usr/bin/python3 tasks/occupancy\_rollup.py

**3) Alert engine (every 5 min)**

* **Purpose:** create alerts when comfort band is violated.
* **Logic (example):** if AVG(temperature) > 26°C for 10 min **and** occupied → insert alerts(room\_id, message, severity).
* **Cron:** \*/5 \* \* \* \* /usr/bin/python3 tasks/alert\_engine.py

**4) Hourly rollup (at HH:05)**

* **Purpose:** compress raw readings to hourly aggregates per room for charts.
* **Output table (example):** readings\_hourly(room\_id, hour\_start, avg\_temp, avg\_humidity, motion\_rate).
* **Cron:** 5 \* \* \* \* /usr/bin/python3 tasks/rollup\_hourly.py

**5) Daily retention & backup (02:15)**

* **Purpose:** archive or purge raw rows older than 90 days, verify backup.
* **Cron:** 15 2 \* \* \* /usr/bin/python3 tasks/retention.py

# Database Schema

CREATE DATABASE thermotrack;

USE thermotrack;

-- USERS: login + registration

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(100) NOT NULL UNIQUE,

email VARCHAR(255) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- ROOMS: physical spaces being monitored

CREATE TABLE rooms (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

location VARCHAR(255),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- DEVICES: sensors installed in each room

CREATE TABLE devices (

id INT AUTO\_INCREMENT PRIMARY KEY,

room\_id INT NOT NULL,

name VARCHAR(100) NOT NULL,

type VARCHAR(50), -- e.g., 'Temperature', 'Humidity', 'Motion'

status VARCHAR(50) DEFAULT 'active',

installed\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (room\_id) REFERENCES rooms(id) ON DELETE CASCADE

);

-- READINGS: temperature, humidity, motion data

CREATE TABLE readings (

id INT AUTO\_INCREMENT PRIMARY KEY,

device\_id INT NOT NULL,

temperature FLOAT,

humidity FLOAT,

motion\_detected BOOLEAN DEFAULT 0,

recorded\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (device\_id) REFERENCES devices(id) ON DELETE CASCADE

);

-- ALERTS: generated if readings exceed thresholds

CREATE TABLE alerts (

id INT AUTO\_INCREMENT PRIMARY KEY,

room\_id INT NOT NULL,

message VARCHAR(255) NOT NULL,

severity ENUM('info','warning','critical') DEFAULT 'info',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (room\_id) REFERENCES rooms(id) ON DELETE CASCADE

);

-- AUDIT LOGS: tracks actions like login, logout, or alerts cleared

CREATE TABLE audit\_log (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

action VARCHAR(100) NOT NULL,

details TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE SET NULL

);