Concept Description

**Team: g1t2**

Member 1: (Christina Brückl, 01219643)

Member 2: (Leonie Katharina Geyr, 11778679)

Member 3: (Thomas Krabichler, 11906966)

Member 4: (Kevin Rimml, 12036261)

Member 5: (Ioannis Deligiannidis, 11913541)

**Proseminar Group: PS-703080-1, Team2**

**Date: 17.03.2022**

# System Overview

This software is a total solution for immediate and long-term indoor climate monitoring in office buildings. Based on the measurements the system not only improves the working conditions for Users in offices, but also offers the possibility of using and purchasing smart office applications based on the underlying long-term data.

For the analysis of the room climate, Arduinos with corresponding sensors for measuring temperature, humidity, air quality, noise pollution and light intensity are used. The devices are equipped with a light-emitting diode that provides live feedback on measurements and therefore are used to notify persons in a certain room of individually configurable measurement-violations.

The long-term monitoring of the indoor climate is ensured by data which is collected and displayed web-based under restriction of data-protection law.

Users have insight in daily, weekly and monthly measurements of their own office, conference-rooms and recreation rooms within their own department.

Head of Departments additionally are provided with cumulated data for all rooms of the corresponding department in the same manner as Users are.

Higher Management and Facility Administration have insight in cumulative data for the whole building to improve and react to general room-climate within the building.

For a better overview all data is graphically visualized in diagrams.

The features of the web-based monitoring application further include user management and sensor configuration as well as entering absences of Users in the system to take them into account in the evaluation of the collected data. As a convenient way for users to keep an overview of the conditions, users can subscribe to daily/weekly/monthly indoor climate reports via email.

# Use Cases

# Actors

**Users**

An Users has an account for the software and has access to the room climate data.

**Facility Manager**

The Facility Manager oversees configuring the limit values of the sensors and registering/deleting sensors of a specific room.

**Department Manager**

Has insights into cumulative room climate analysis and is responsible for the control of heating and air conditioning.

**RasberryPi/Backend**

The raspberry pi's are receiving the data from the sensors and are checking them for validity. Valid data is then stored in a database.

**Sensors**

The sensors capture various room climate informations such as temperature, humidity, airquality, etc.

# Use-Case Diagram

# Use Cases

# Employee

**Register absence**

* **Precondition:** User have to login and have the role “Users” or higher.
* **Procedure:** The user selects an interval of absence days (from – to) and register themselves for that interval as absent.
* **Success**: The user has been registered for that interval of days for its absence.
* **No success:** -
* **Involved Classes:** Users, Absence, AbsenceReason, SchedulerTaskConfig

**Unegister absence**

* **Precondition:** User have to login and have the role “Users” or higher and has already registered for absence.
* **Procedure:** User selects which days or an interval of days he/she wants to unregister.
* **Success**: Days or the interval of days have been unregistered.
* **No success:** -
* **Involved Classes:** Users

**View climate data of public rooms**

* **Precondition:** User have to login and have the role “Employee” or higher.
* **Procedure:** User selects from which room the data should be retrieved.
* **Success**: User sees the data from the selected room.
* **No success:** User gets an email with a failure report.
* **Involved Classes:** Users, Room, Department, Measurement

**View climate data of own office**

* **Precondition:** User have to login and have the role “Users” or higher and also the user has to be register as employee of that office.
* **Procedure:** User submits on the tab/button his own office to view the data.
* **Success**: User see’s the view of data of his/her own office.
* **No success:** User get a failure email with the report.
* **Involved Classes:** Users, Room, Department, Measurement

# Facility Manager

**Add sensor station**

* **Precondition:** User has to login and has the role “Facility Manager” or higher. The sensor station must not be damaged.
* **Procedure:** User registers a sensor station for a particular room (possible requirements e.g. room number, sensor id, etc.)
* **Success**: Sensor for the selected room has been added.
* **No success:** -
* **Involved Classes:** Facility Manager, Department, Room, Sensor

**Remove sensor station**

* **Precondition:** User has to login and has the role “Facility Manager” or higher. The sensor station must be already registered.
* **Procedure:** User selects the room and the sensor to be removed.
* **Success**: The sensor for that room has been successfully removed.
* **No success:** -
* **Involved Classes:** Facility Manager, Department, Room, Sensor

**View cumulative data of all departments**

* **Precondition:** User has to login and has the role “Facility Manager” or higher. Sensor stations must be registered. The connection between the sensors, raspberry pi and backend must be stable.
* **Procedure:** User can click on a tab/button to view all cumulative data of all departments.
* **Success**: User gets a view (graphical or numerical) of all departments.
* **No success:** User gets an email notification.
* **Involved Classes:** Facility Manager, Department, Sensor, Measurement

**Configure limits**

* **Precondition:** User has to login and has the role “Facility Manager” or higher. The connection between the sensors, raspberry pi and backend must be stable.
* **Procedure:** User selects the sensor station and then proceed to the configuration of limits in the shown dialog.
* **Success**: Limits have been changed and saved.
* **No success:** -
* **Involved Classes:** Facility Manager, Department, Room, Sensor

**Allocate Raspeberry Pi’s per room**

* **Precondition:** User has to login and has the role “Facility Manager” or higher.
* **Procedure:** User submits the tab/button to select the room and then allocates the raspberry pi to that room.
* **Success**: Rasberry Pi is allocated to that room and can exchange data with the sensor station and the backend.
* **No success:** -
* **Involved Classes:** Facility Manager, Department, Room

# Department Manager

**View climate data from all rooms of the department**

* **Precondition:** User has to login and has the role “Department Manager”.
* **Procedure:** User submits on the tab/button his department to view the data.
* **Success**:User can view the selected data.
* **No success:** User receives an error message
* **Involved Classes:** Department, Room. SensorStation
* **View cumulative data from all rooms of the department**
* **Precondition:** User has to login and has the role “Department Manager”.
* **Procedure:** User submits on the tab/button his department to view the data.
* **Success**:User sees the cumulative data from all rooms of his/her department,
* **No success:** User receives an error message,
* **Involved Classes:** Department, Room. SensorStation

# Facility Manager/Department Manager

**View audit-log**

* **Precondition:** User has to login and has the role “Facility Manager”, “Department Manager” or higher.
* **Procedure:** User submits on the tab/button.
* **Success**: User gets a view of the timestamp, the user and the event occurred.
* **No success:** -
* **Involved Classes:** Facility Manager, Department Manager, AuditLog

# Users/Facility Manager/Department Manager

**Register for email notifications**

* **Precondition:** User has to login and has the role “Employee” or higher and must be active.
* **Procedure:** User goes to his/her user settings and submits the daily, weekly, monthly or off button.
* **Success**: User gets email notification of the selected interval.
* **No success:** -
* **Involved Classes:** Users, MailInterval

**Unregister for email notifications**

* **Precondition:** User has to login and has the role “Users” or higher and must be active.
* **Procedure:** User goes to his/her profile and submits the “OFF” button to unregister hisself.
* **Success**: User does not get any further notifications as email.
* **No success:** -
* **Involved Classes:** Users, MailInterval

**View daily/weekly/monthly analysis report**

* **Precondition:** User has to login and has the role “Users” or higher.
* **Procedure:** User submits on tab/button to view the report.
* **Success**: A view with the daily, weekly and monthly report will appear to the user.
* **No success:** -
* **Involved Classes:** Users, Room, Department,Measurement,Sensor

# Rasberry Pi/Backend

**Check for data validity**

* **Precondition:** Data was successfully received from the Arduino sensors in intact form.
* **Procedure:** Every transmitted data gets checked by a predefined condition on validity.
* **Success**:Data was intact and will either be discarded or stored in the database.
* **No success:** -
* **Involved Classes:** SensorStation

**Discard invalid data**

* **Precondition:** Data was checked and doesn’t fulfil the predefined condition.
* **Procedure:** The invalid data will be discarded and therefore not stored in the database.
* **Success**:Data was not stored in the database.
* **No success: -**
* **Involved Classes:** SensorData

**Store valid data**

* **Precondition:** Data was checked and fulfils the predefined condition.
* **Procedure:** Makes a call to the specific REST method and stores it via POST in the database.
* **Success**:Data is stored in the database
* **No success:** Data won‘t be stored in the database
* **Involved Classes:** Sensor,Measurement,Room,Department

# Sensors

**Give visual feedback on limit exceed**

* **Precondition:** Limit has been individually setand is configured, Sensor is active and working
* **Procedure:** Sensor measures a certain value, System checks if the measurement exceeds the limit
* **Success**: The visual feedback on limit exceeded is provided by the defined color/flashing of the given LED
* **No success:** visual feedback is provided if limit not exceeded, no visual feedback provided if limit is exceeded
* **Involved Classes:**

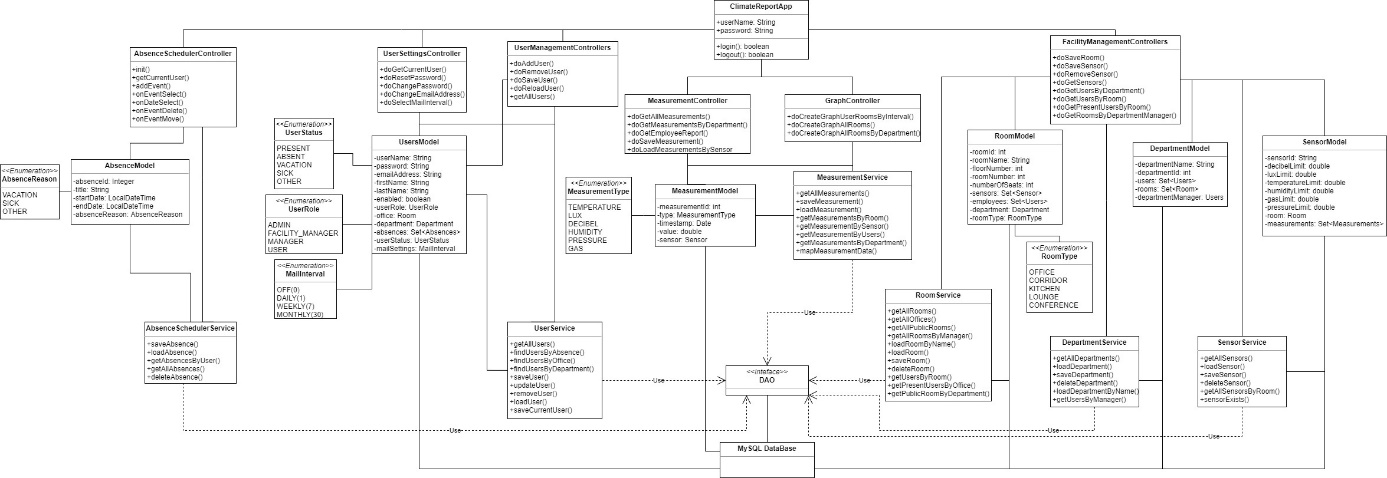
**Capture climate data**

* **Precondition:** Arduino isrunning and connected to the system, sensors are working
* **Procedure:** The sensors measure climate data permanently in given time intervals
* **Success**: data is captured
* **No success:** no data is captured
* **Involved Classes:**

**Transmission of the acquired data**

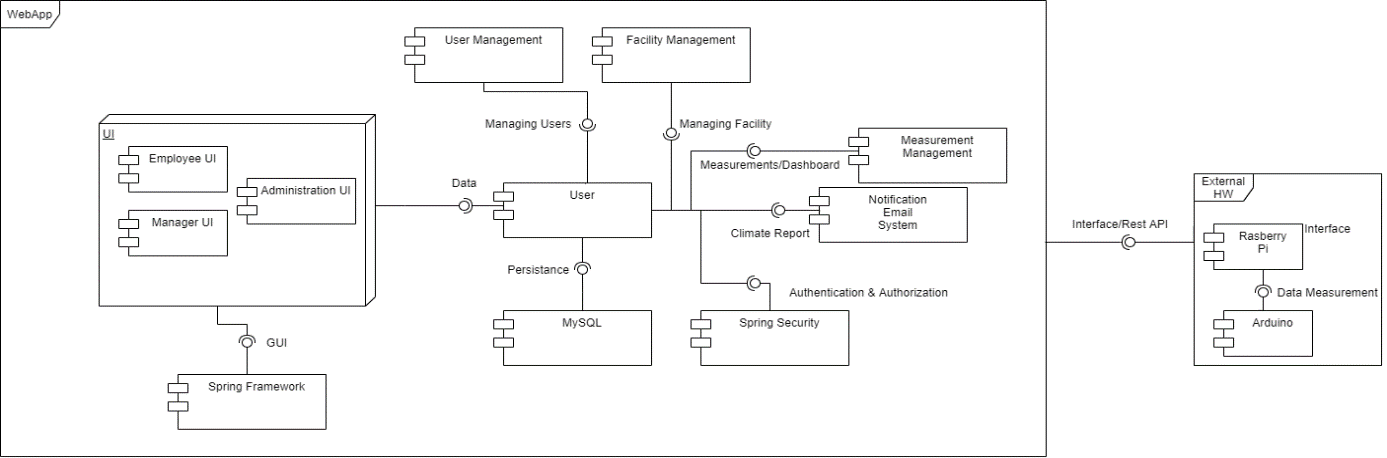
* **Precondition**: Data is captured by the sensors, Arduino is connected to the system
* **Procedure:** Data is transmitted via defined Bluetooth standard to the system where further processing takes place
* **Success**: All Data is transmitted correctly
* **No success:** Data is not transmitted, corrupt or only partially transmitted
* **Involved Classes:**

# Class Diagram



# SW-Architecture

# Component Diagram



# Selected Technologies

# Java

Java is an object-oriented programming language by Sun Microsystems (Oracle). Its benefit is

that it runs on a virtual machine and is therefore platform independent. The server part

this system is written in Java.

# Spring

Spring is an open-source framework for Java. It supports the programmer by providing the

Simplifies development with Java and promotes good programming practices. Additionally, Spring Boot used, which facilitates the rapid implementation of an independent, executable system.

# Java Server Faces (JSF)

JSF is a framework standard for developing graphical user interfaces for web applications.

# Prime Faces

A component framework like Prime Faces extends a JSF implementation. It enables that

easy generating of JavaScript and Ajax.

# Apache Maven

Maven is a build management tool based on Java and also is primarily intended to manage Java programs. Developers should be like that in all areas be supported, so that very little custom configuration is required.

# MySQL Database

H2 is a relational database management system written in Java and available as an open-source software for Java is available. SQL or JDBC can be used to address ORM is also possible in combination with Hibernate.

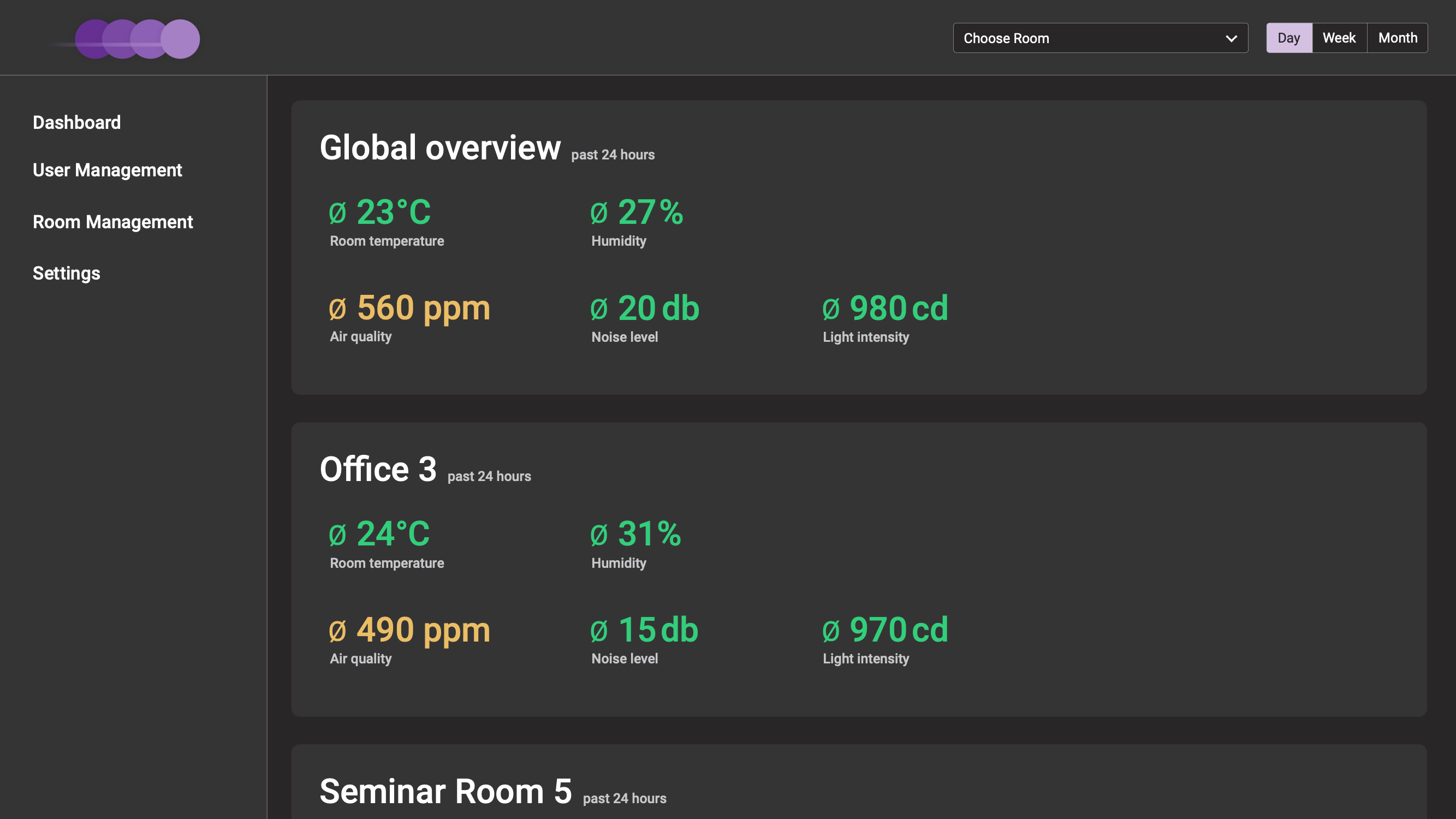
# JUnit

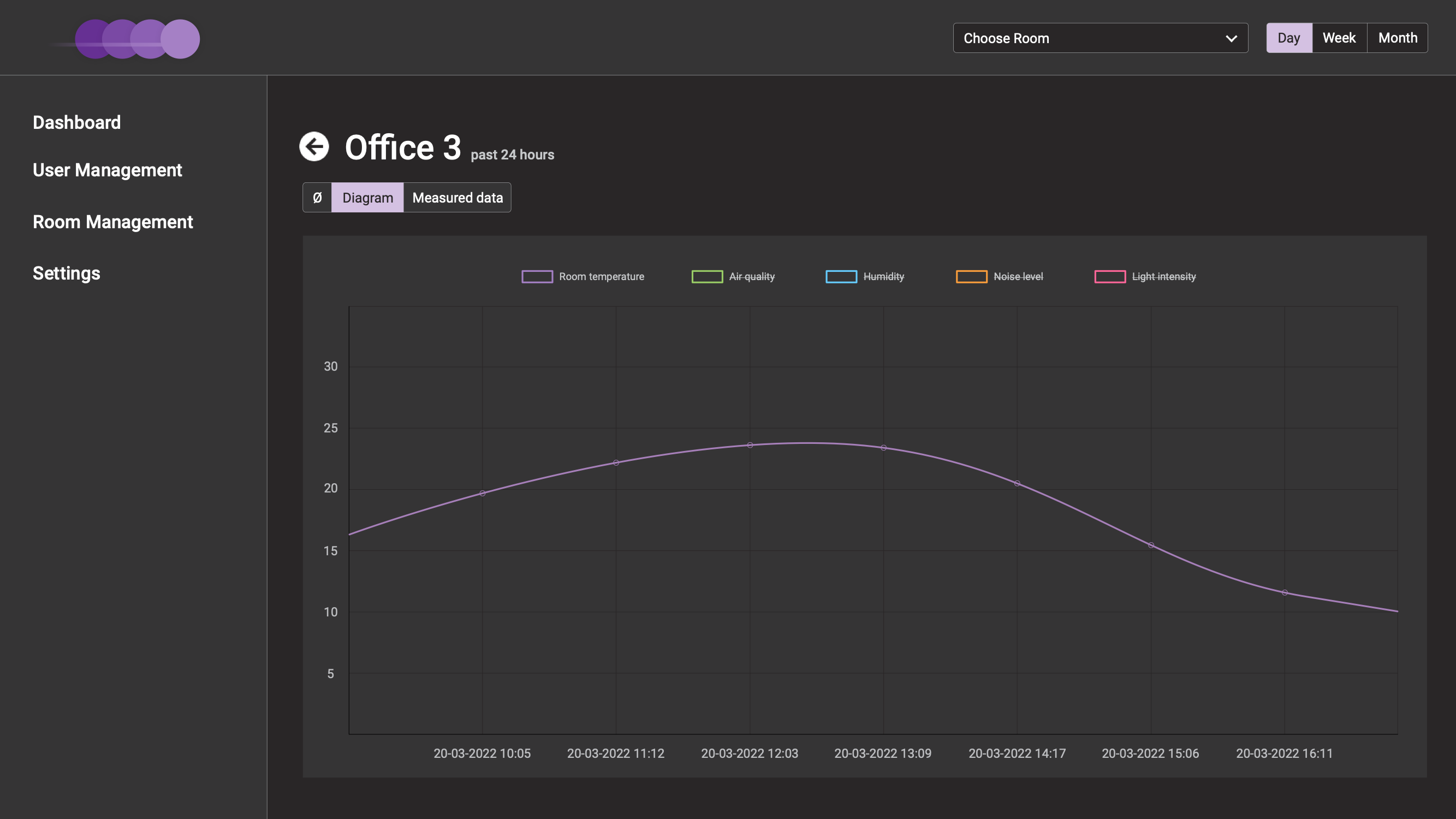
JUnit is a framework for testing Java programs, this project will use it written automated unit tests.

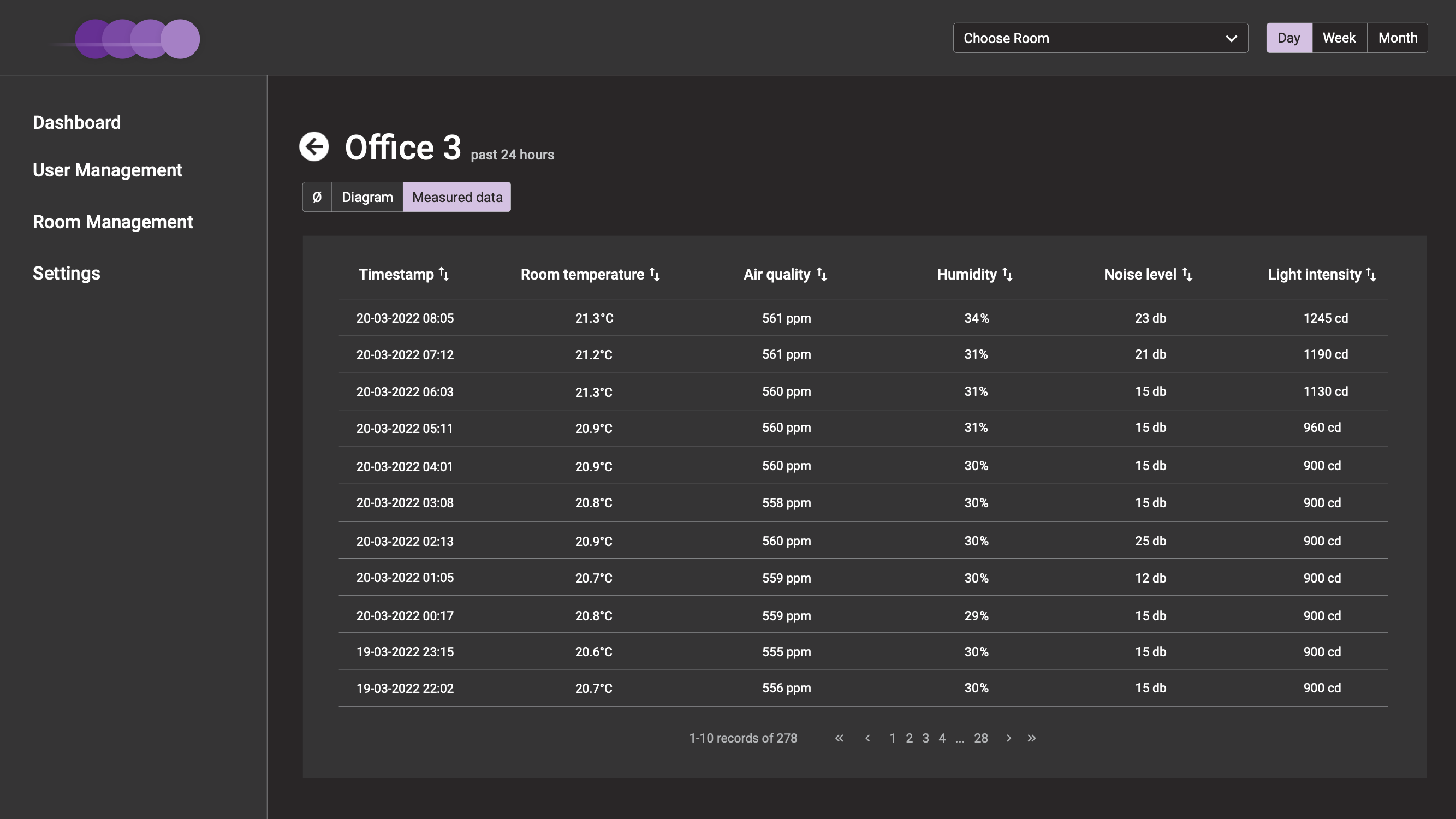
# Docker Compose

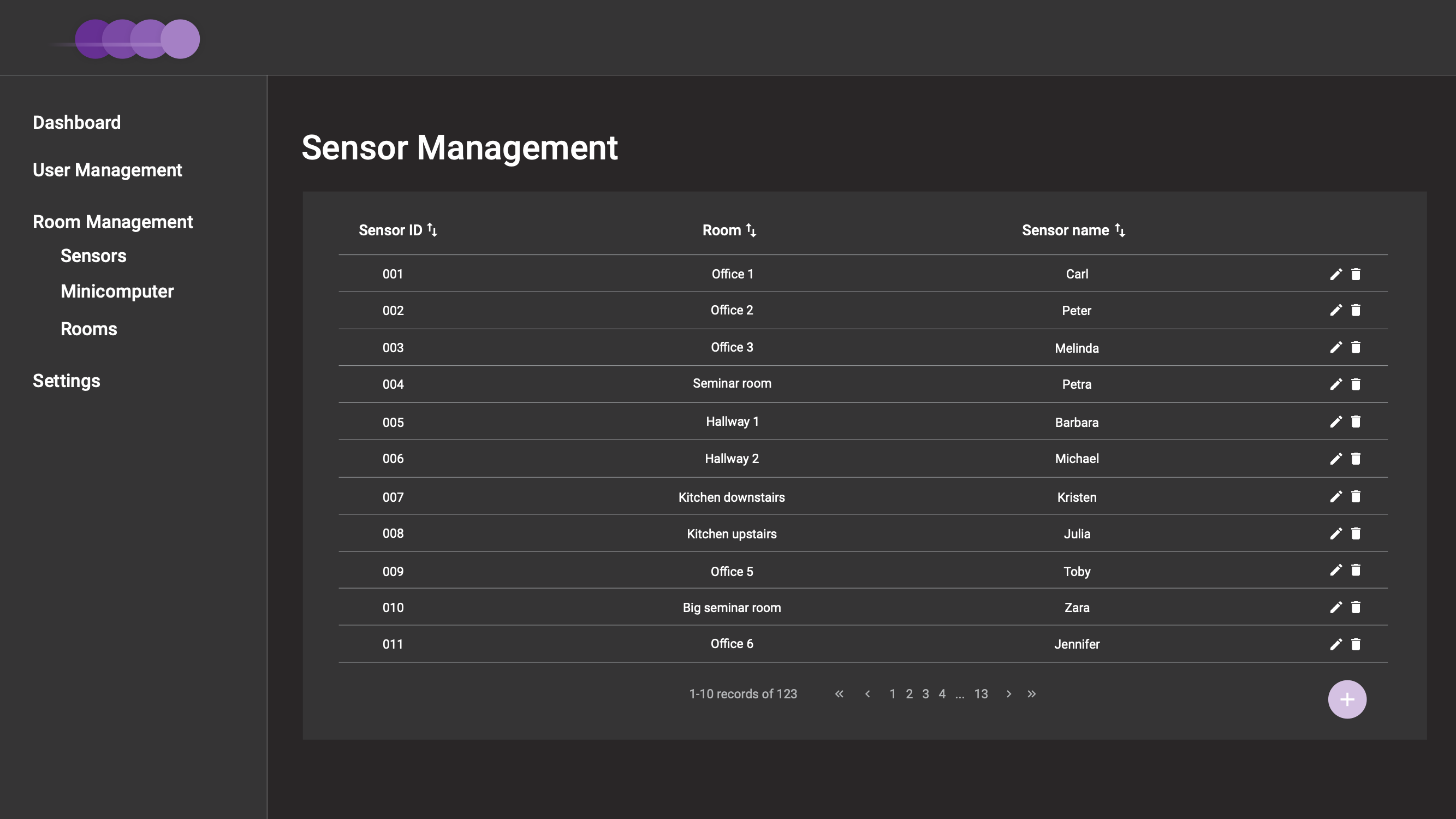
Docker compose is a framework to deploy the web application as well as the MySQL database.

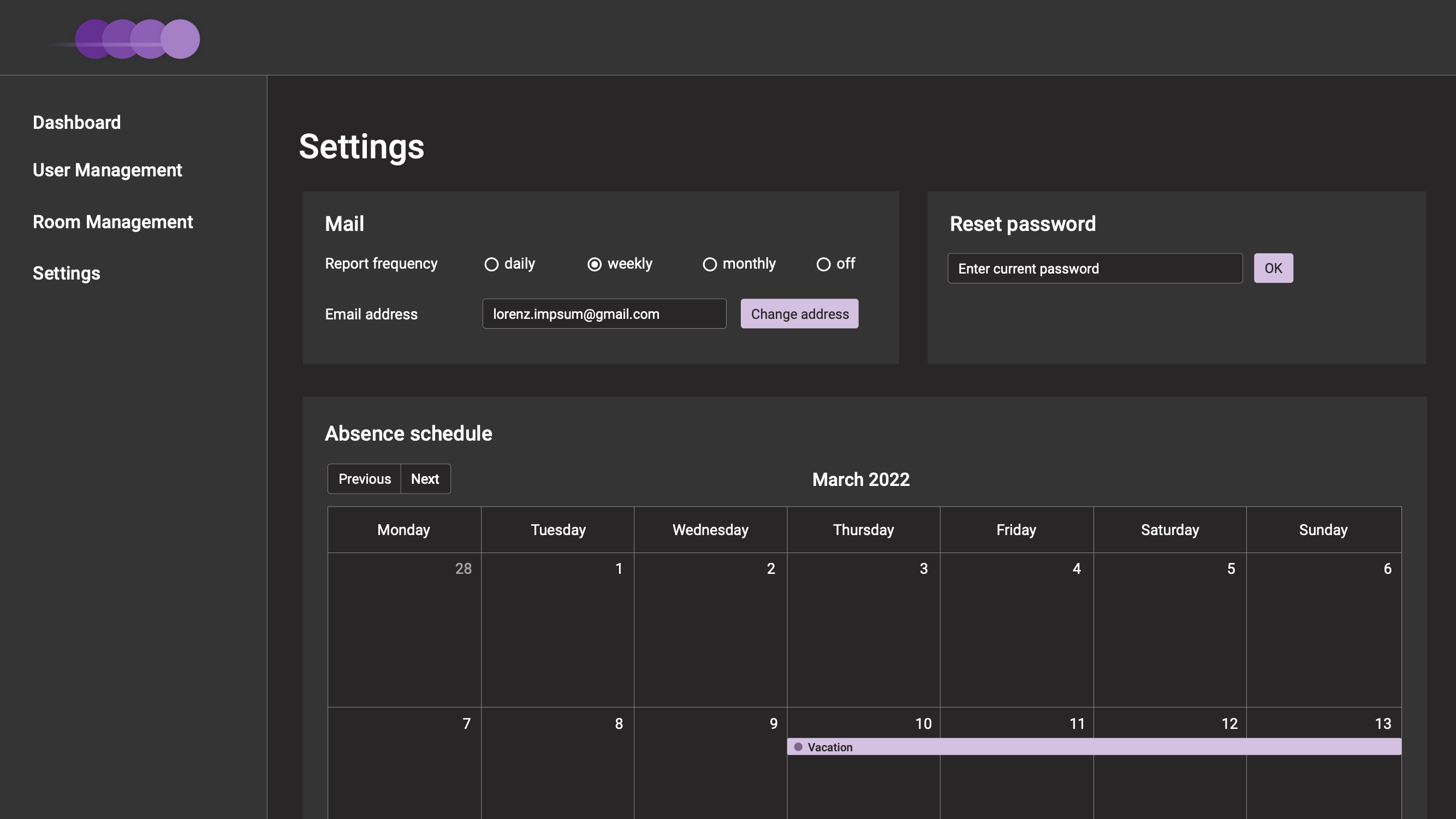
# GUI Prototype

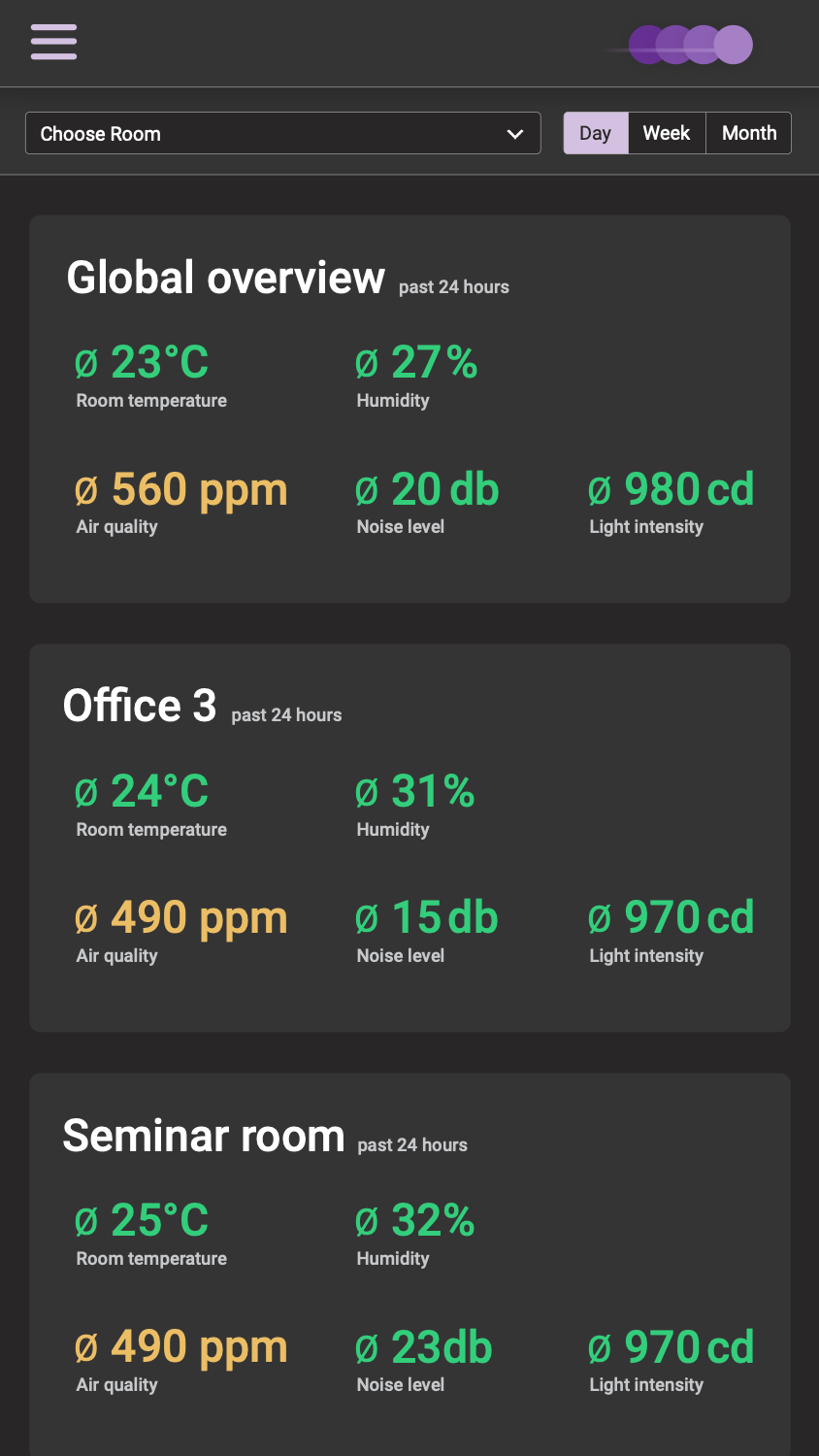
Dashboard / Starting page (view of Admin user)

Detail view: diagram

Detail view: table

Room management: sensor management (similar view for minicomputer, user and room management)

Settings



Dashboard (mobile view)

# Project Plan

|  |  |
| --- | --- |
| **Milestone** | **Deadline \*** |
| Concept Description | 17.03.2022 |
| Circuit Diagram | 24.03.2022 |
| ER-Diagram, Basic Framework and Basic Class implementation including Database setup | 04.04.2022 |
| Basic functionality (create/edit users, connection between raspberry and backend) | 24.04.2022 |
| Basic implementation of all required features | 02.05.2022 |
| Fine tuning of features; Deployment (Docker), final Tests | 16.05.2022 |
| End of Project | 19.05.2022 |
| Documentation – Acceptance-Test | 26.05.2022 |
| Systemtest and Report writing | 13.06.2022 |
| Final Project Results | 23.06.2022 |

\*Increment of two weeks

|  |  |
| --- | --- |
| **Name** | **Responsibility** |
| Ioannis Deligiannidis | Backend |
| Leonie Katharina Geyr | Frontend |
| Christina Bruckl | Hardware and Arduino Software |
| Thomas Krabichler | Backend |
| Kevin Rimml | Backend |