IOT PROJECT: ROOM SELECTION DECISION SUPPORT SYSTEM

Tarik Tornes

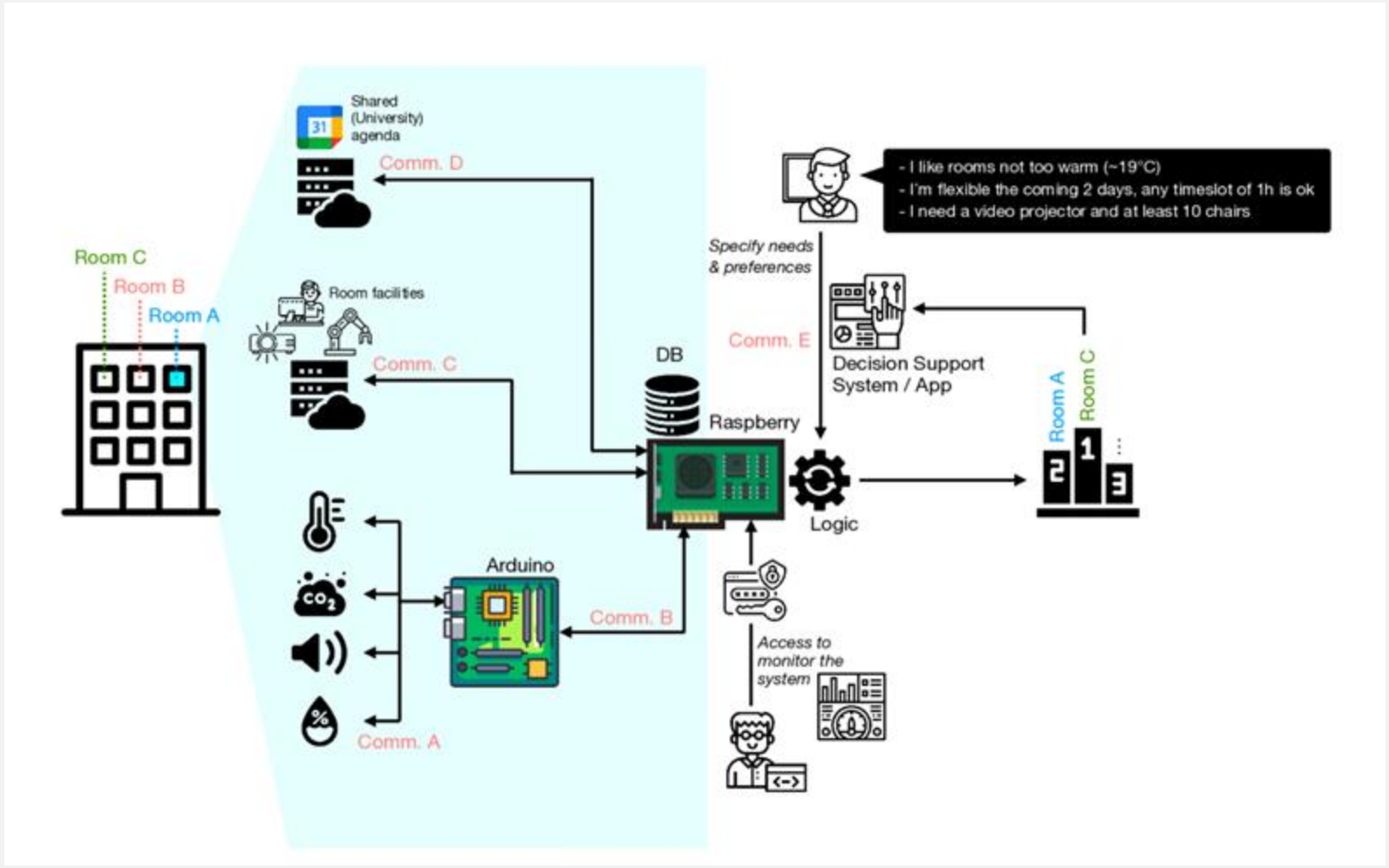
Thuc Kevin Nguyen

Vladyslav Siulhin

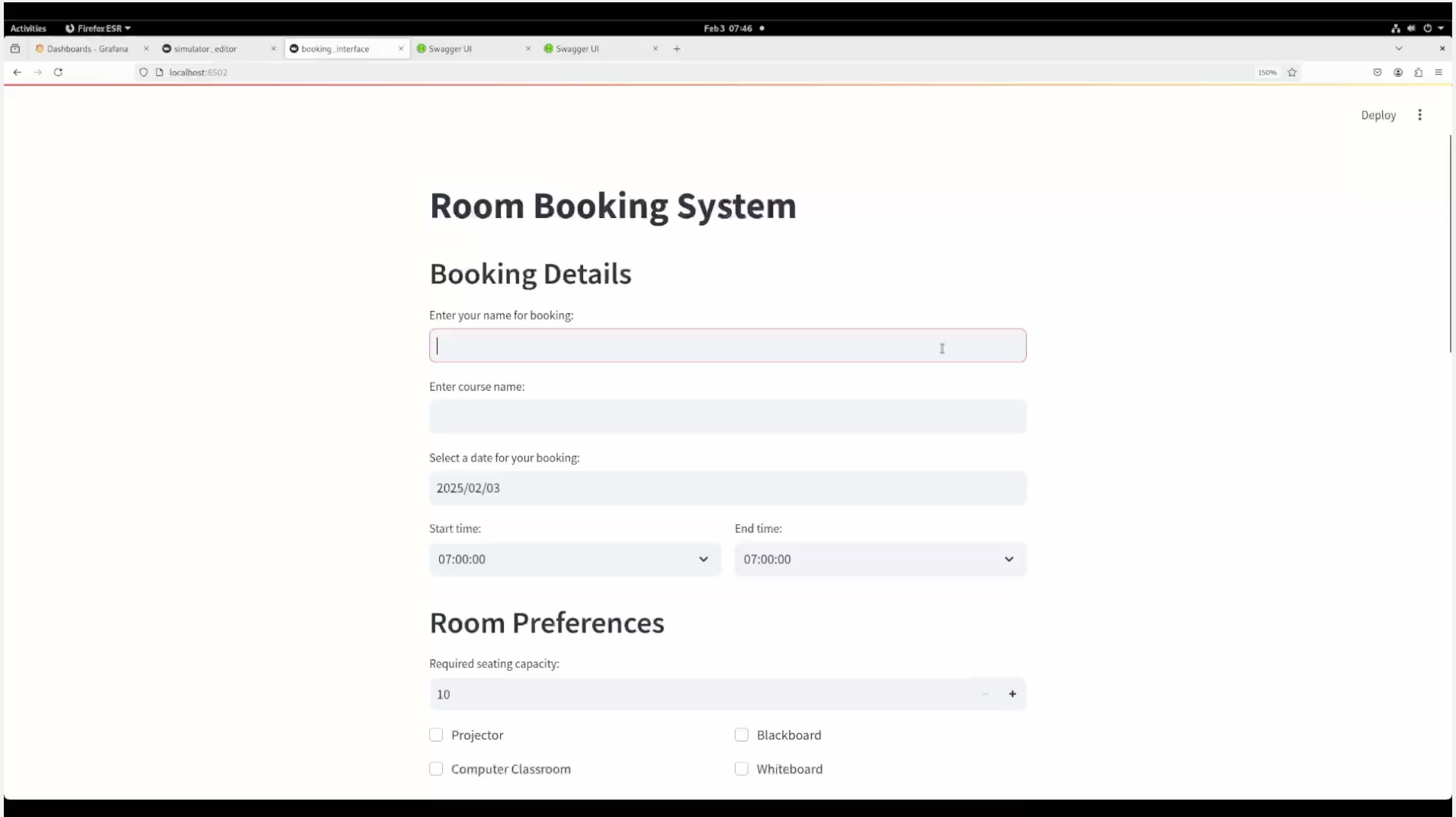
Fedor Chikhachev

University of Luxembourg, 2024/25

OVERVIEW


Shared (University) agenda
Comm. D
Room C
Room B
Room A
Room facilities
Comm. C
DB
Raspberry
Arduino
Comm. B
Comm. A
- I like rooms not too warm (~19°C)
- I'm flexible the coming 2 days, any timeslot of 1h is ok
- I need a video projector and at least 10 chairs
Specify needs & preferences
Comm. E
Decision Support System / App
Room A
Room C
Logic
Access to monitor the system

DEMO

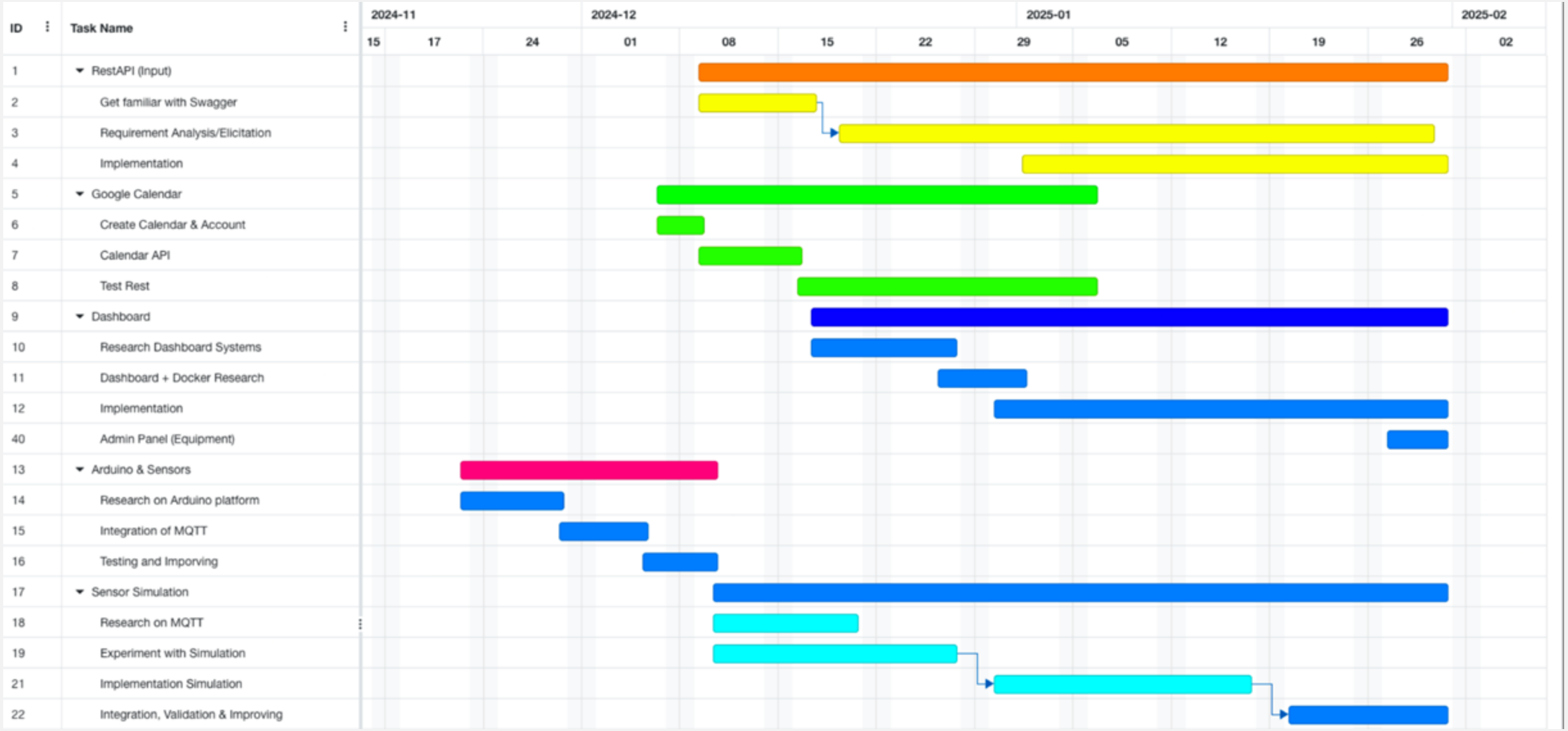

Activities
Firefox ESR
Feb 3 07:46
Dashboards - Grafana
simulator_editor
booking_interface
Swagger UI
Swagger UI
localhost:8502
150%
Deploy
Room Booking System
Booking Details
Enter your name for booking:
Enter course name:
Select a date for your booking:
2025/02/03
Start time:
07:00:00
End time:
07:00:00
Room Preferences
Required seating capacity:
10
Projector
Blackboard
Computer Classroom
Whiteboard

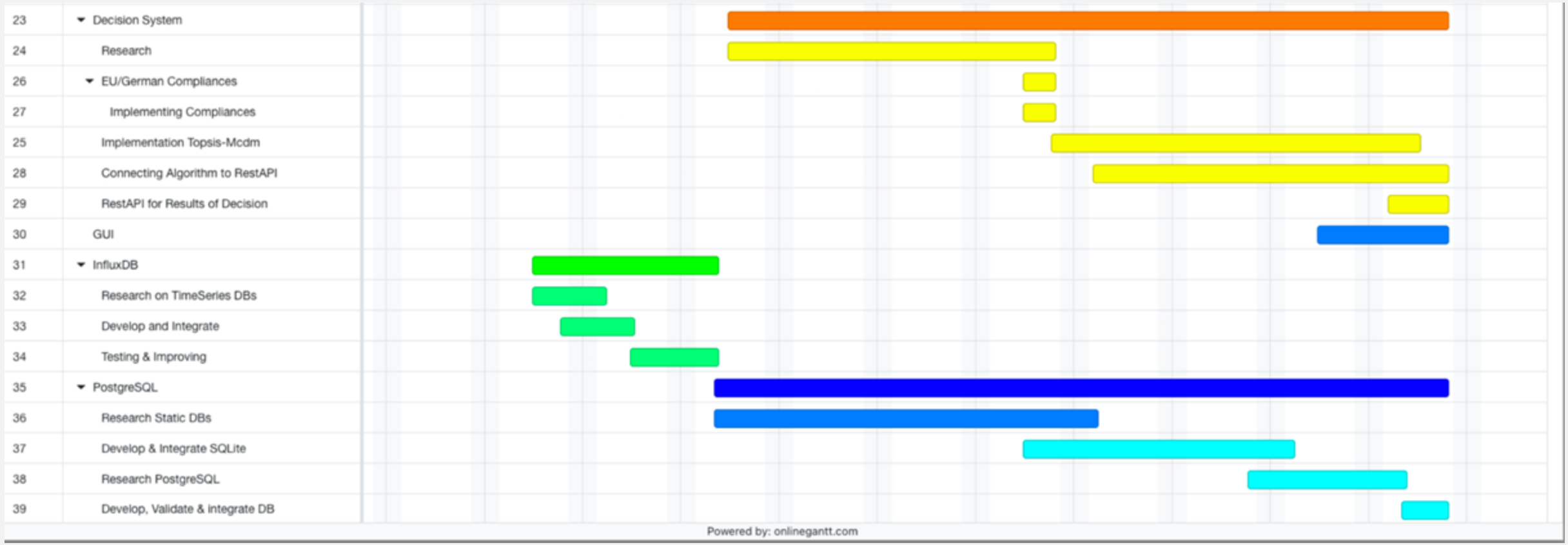
OVERVIEW

- 1 component = 1 Docker container
- This ensure that our project is modular, we can develop it (mostly) independently
- Stack of used technologies: Python, Arduino, Docker, Grafana, InfluxDB, PostgreSQL, Swagger API

MANAGEMENT & ORGANIZATION

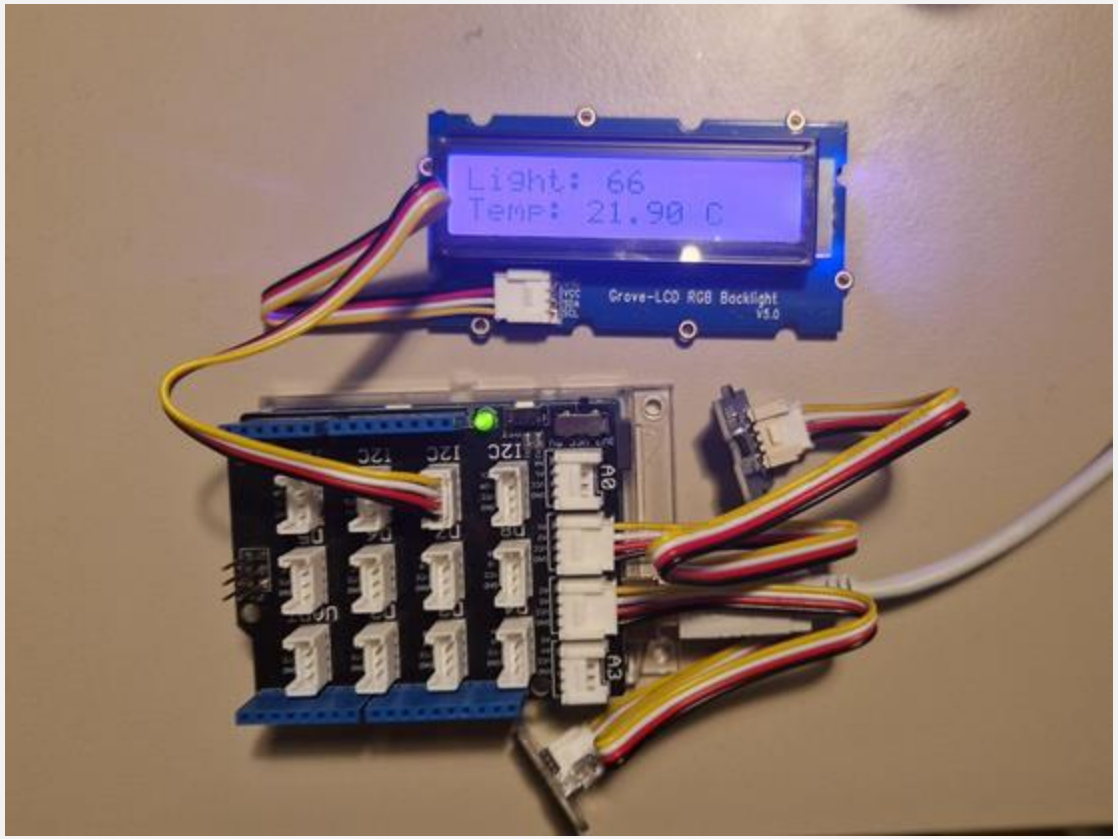
- TT – REST API for rooms data (sensors + equipment); Graphana dashboards, Google Calendars integration
- TN – Decision system: algorithm, REST API; Booking system UI
- VS – InfluxDB setup, static equipment database
- FC – Arduino integration; Simulation framework for sensors and equipment; MQTT linkage between sensor data and InfluxDB
- All – Docker setup & intercommunication of containers


ID
Task Name
2024-11
2024-12
2025-01
2025-02
15
17
24
01
08
15
22
29
05
12
19
26
02
1 RestAPI (Input)
2 Get familiar with Swagger
3 Requirement Analysis/Elicitation
4 Implementation
5 Google Calendar
6 Create Calendar & Account
7 Calendar API
8 Test Rest
9 Dashboard
10 Research Dashboard Systems
11 Dashboard + Docker Research
12 Implementation
40 Admin Panel (Equipment)
13 Arduino & Sensors
14 Research on Arduino platform
15 Integration of MQTT
16 Testing and Imporving
17 Sensor Simulation
18 Research on MQTT
19 Experiment with Simulation
21 Implementation Simulation
22 Integration, Validation & Improving

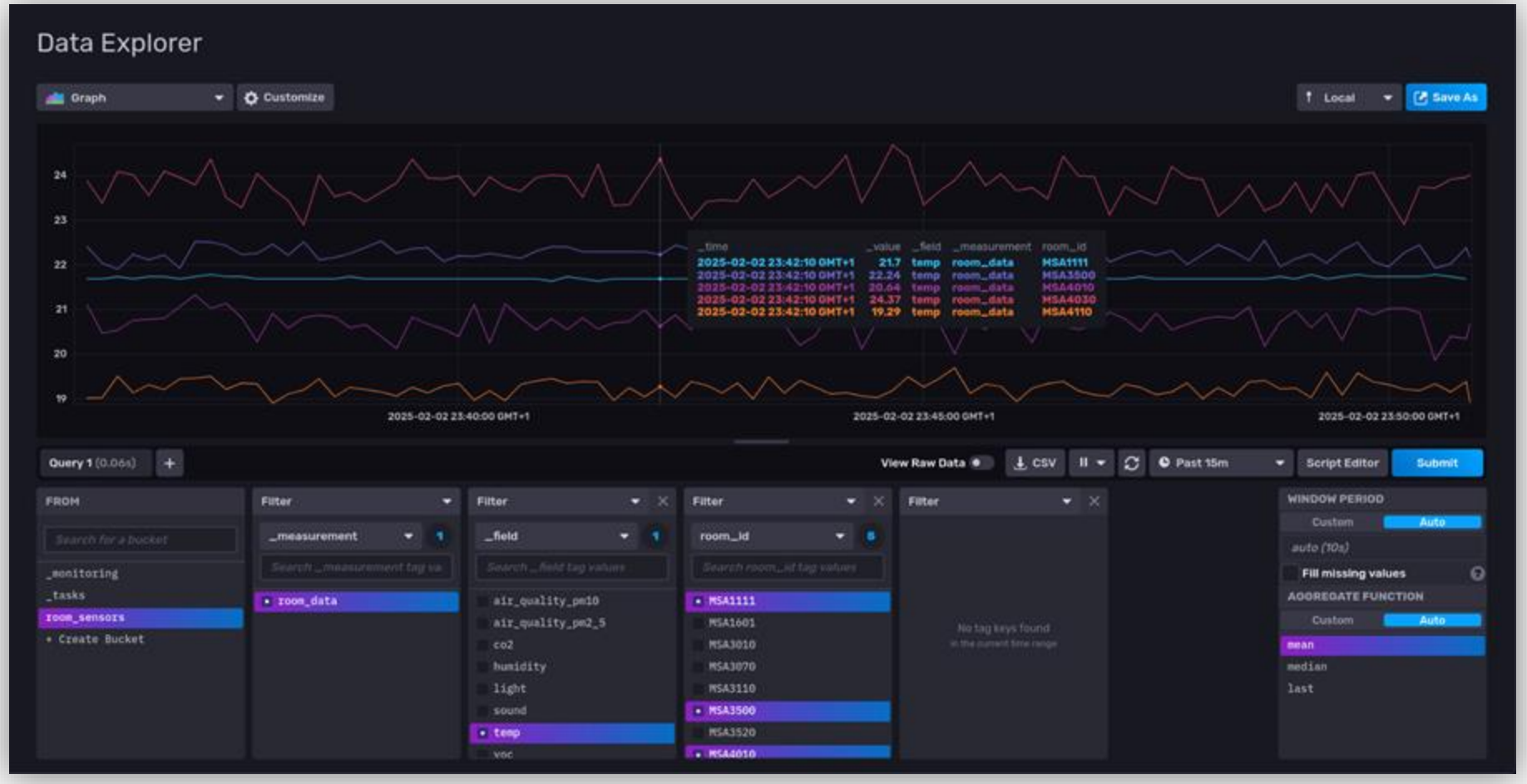

23 Decision System
24 Research
26 EU/German Compliances
27 Implementing Compliances
25 Implementation Topsis-Mcdm
28 Connecting Algorithm to RestAPI
29 RestAPI for Results of Decision
30 GUI
31 InfluxDB
32 Research on TimeSeries DBs
33 Develop and Integrate
34 Testing & Improving
35 PostgreSQL
36 Research Static DBs
37 Develop & Integrate SQLite
38 Research PostgreSQL
39 Develop, Validate & integrate DB
Powered by: onlinegantt.com

WHAT ABOUT ARDUINO?

- Used relevant sensors: light and temperature
- No Wi-Fi on UNO R3, cannot directly publish data to MQTT broker :(
- Solution: let’s use serial port to transfer data, then – MQTT (“bridge”)
- Limitations: each device needs to have room id assigned

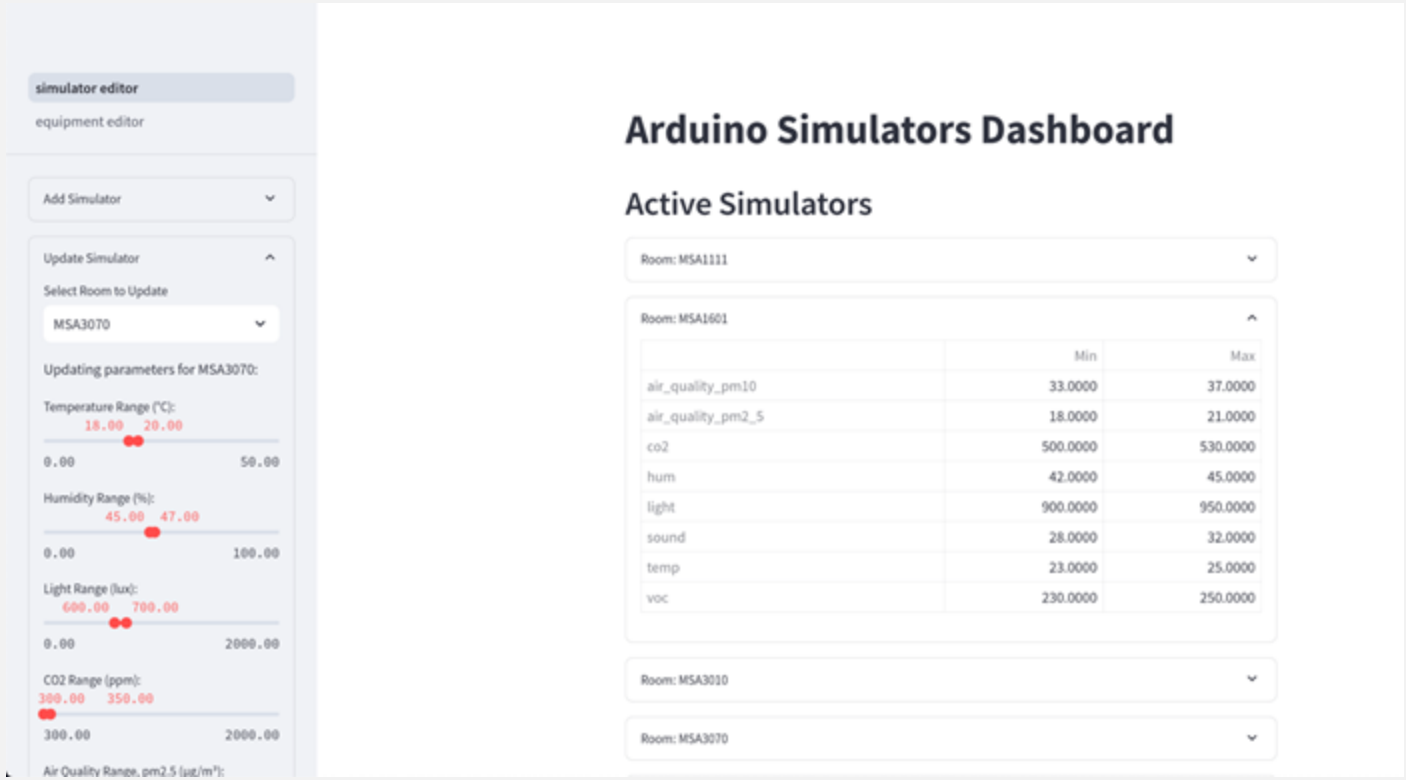


AND IT ACTUALLY WORKS!



SIMULATION FOR SENSORS

- We have only one Arduino with limited sensors: how to test our decision system?
- Answer: simulation framework!
- 1 room = 1 thread in Simulator
- Data is sampled from Gaussian distribution in certain range
- Fluctuations according to daytime, smoothed by cosine function
- Streamlit UI for convenient adjustment of parameters



simulator editor

equipment editor

Add Simulator

Update Simulator

Select Room to Update

MSA3070

Updating parameters for MSA3070:

Temperature Range (°C):

18.00 20.00

0.00 50.00

Humidity Range (%):

45.00 47.00

0.00 100.00

Light Range (lux):

600.00 700.00

0.00 2000.00

CO2 Range (ppm):

300.00 350.00

300.00 2000.00

Air Quality Range, pm2.5 (µg/m³):

Arduino Simulators Dashboard

Active Simulators

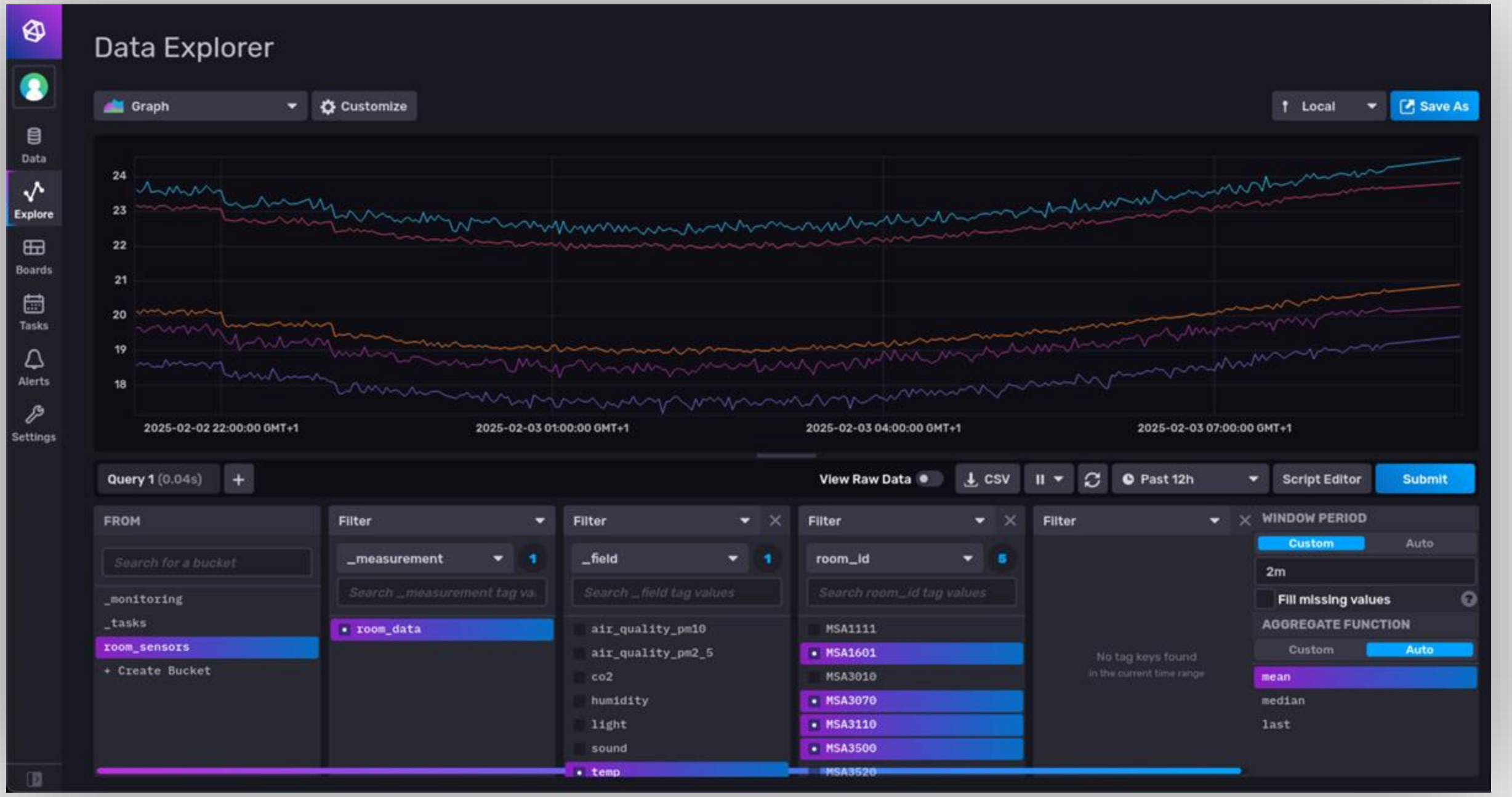
Room: MSA1111

Room: MSA1601

	Min	Max
air_quality_pm10	33.0000	37.0000
air_quality_pm2_5	18.0000	21.0000
co2	500.0000	530.0000
hum	42.0000	45.0000
light	900.0000	950.0000
sound	28.0000	32.0000
temp	23.0000	25.0000
voc	230.0000	250.0000

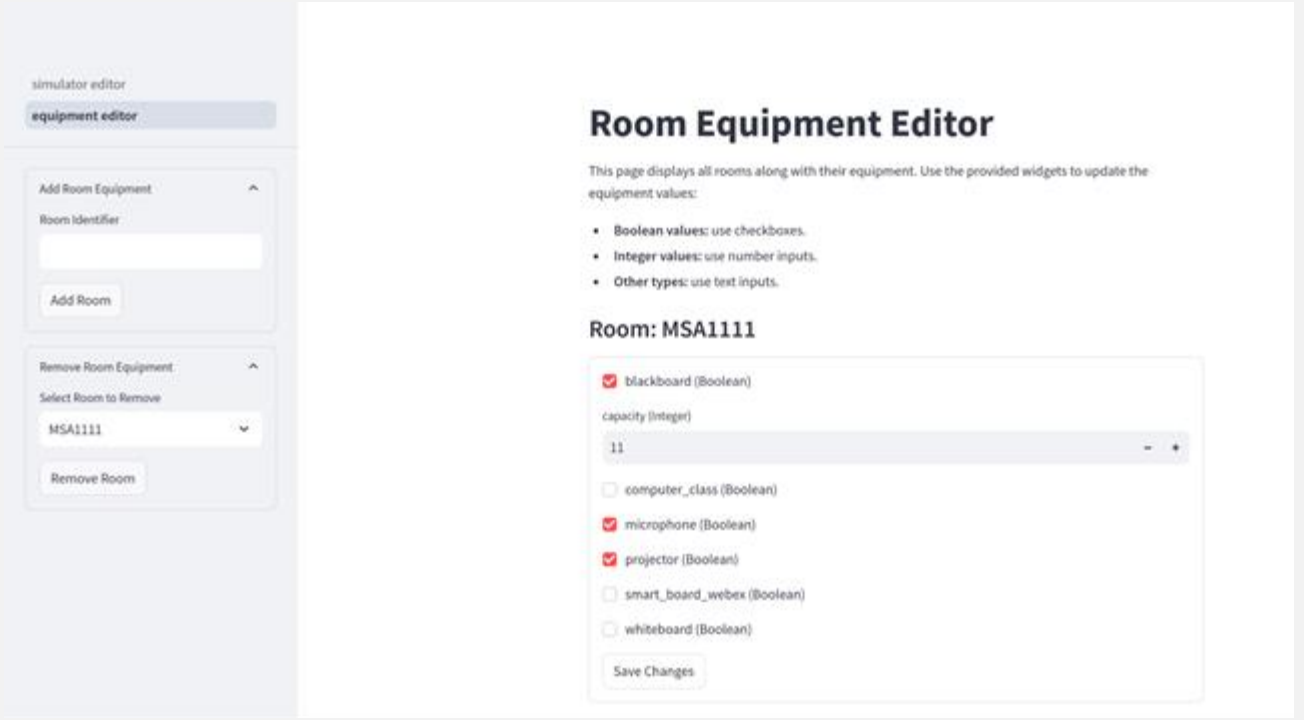
Room: MSA3010

Room: MSA3070


Data Explorer
Graph
Customize
Local
Save As
Data
Explore
Boards
Tasks
Alerts
Settings
24
23
22
21
20
19
18
2025-02-02 22:00:00 GMT+1
2025-02-03 01:00:00 GMT+1
2025-02-03 04:00:00 GMT+1
2025-02-03 07:00:00 GMT+1
Query 1 (0.04s)
View Raw Data
CSV
Past 12h
Script Editor
Submit
FROM
Search for a bucket
_monitoring
_tasks
room_sensors
+ Create Bucket
Filter
_measurement
Search _measurement tag va
room_data
Filter
_field
Search _field tag values
air_quality_pm10
air_quality_pm2_5
co2
humidity
light
sound
temp
Filter
room_id
Search room_id tag values
MSA1111
MSA1601
MSA3010
MSA3070
MSA3110
MSA3500
MSA3520
Filter
No tag keys found
in the current time range
WINDOW PERIOD
Custom
Auto
2m
Fill missing values
AGGREGATE FUNCTION
Custom
Auto
mean
median
last

EQUIPMENT EDITOR

- Information about static equipment is stored in PostgreSQL (e.g. capacity of room, projector, smartboard information, etc.)
- How to modify it conveniently? For tests & administration?
- Streamlit page + REST API for DB connection
- REST API: we leave the choice of UI for future admins of our system



simulator editor

equipment editor

Add Room Equipment

Room Identifier

Add Room

Remove Room Equipment

Select Room to Remove

MSA1111

Remove Room

Room Equipment Editor

This page displays all rooms along with their equipment. Use the provided widgets to update the equipment values:

- **Boolean values:** use checkboxes.
- **Integer values:** use number inputs.
- **Other types:** use text inputs.

Room: MSA1111

- [x] blackboard (Boolean)

capacity (Integer)

11 − +

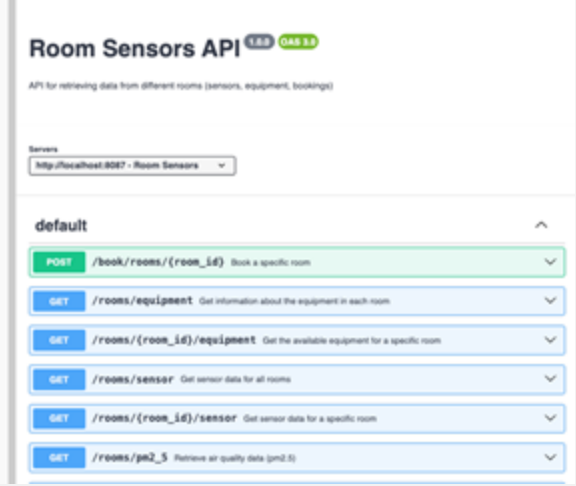
- [ ] computer_class (Boolean)
- [x] microphone (Boolean)
- [x] projector (Boolean)
- [ ] smart_board_webex (Boolean)
- [ ] whiteboard (Boolean)

Save Changes

ROOMS' DATA (REST API 1)

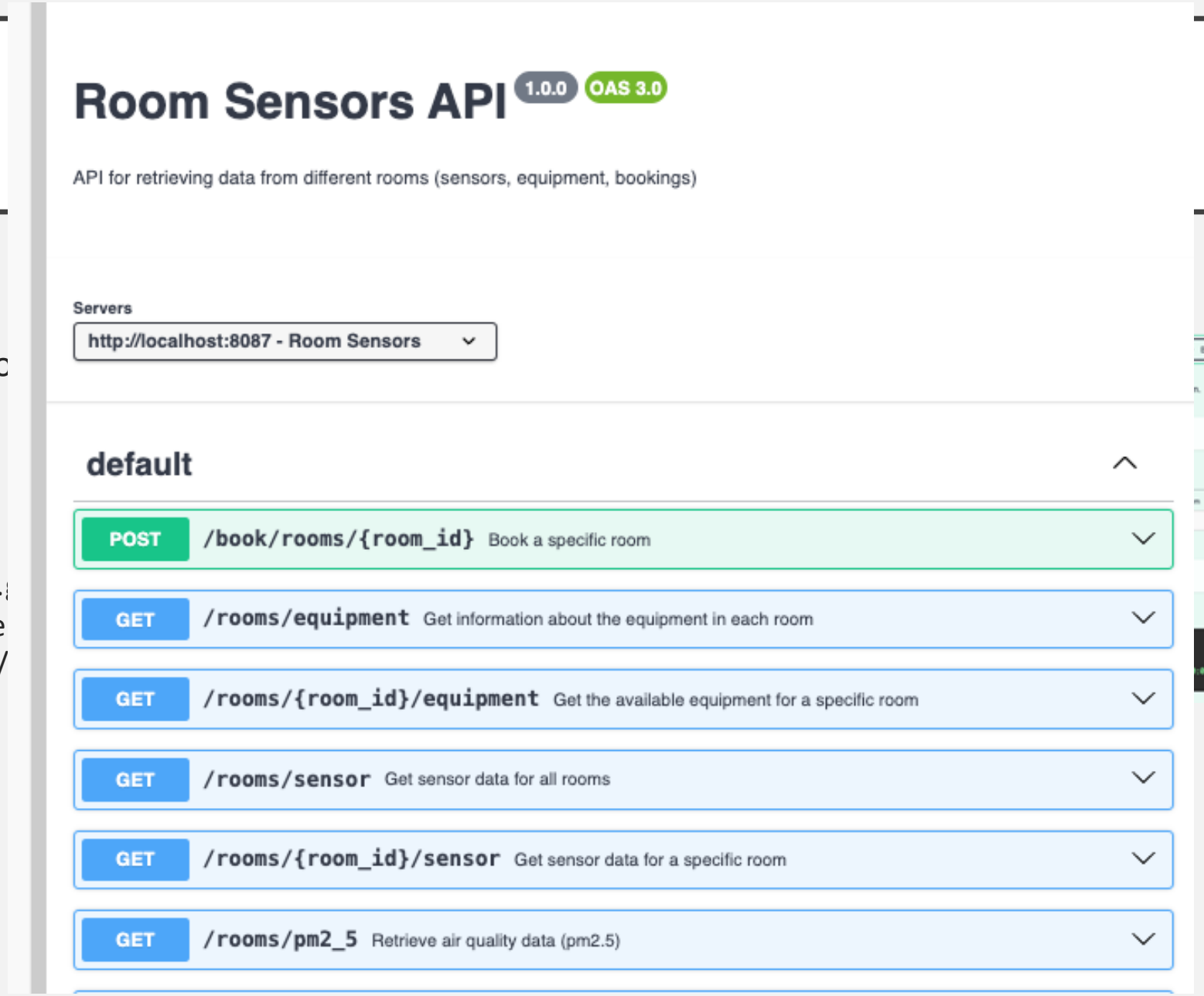
- Data-retrieval for Decision System
- Flask
- Swagger Editor

- GET
  - Sensor Data (e.g. /rooms/{room_id}/temperature)
  - Room Equipment
  - Bookings (e.g. /rooms/bookings/{rooms_id})
- POST
  - Book Rooms



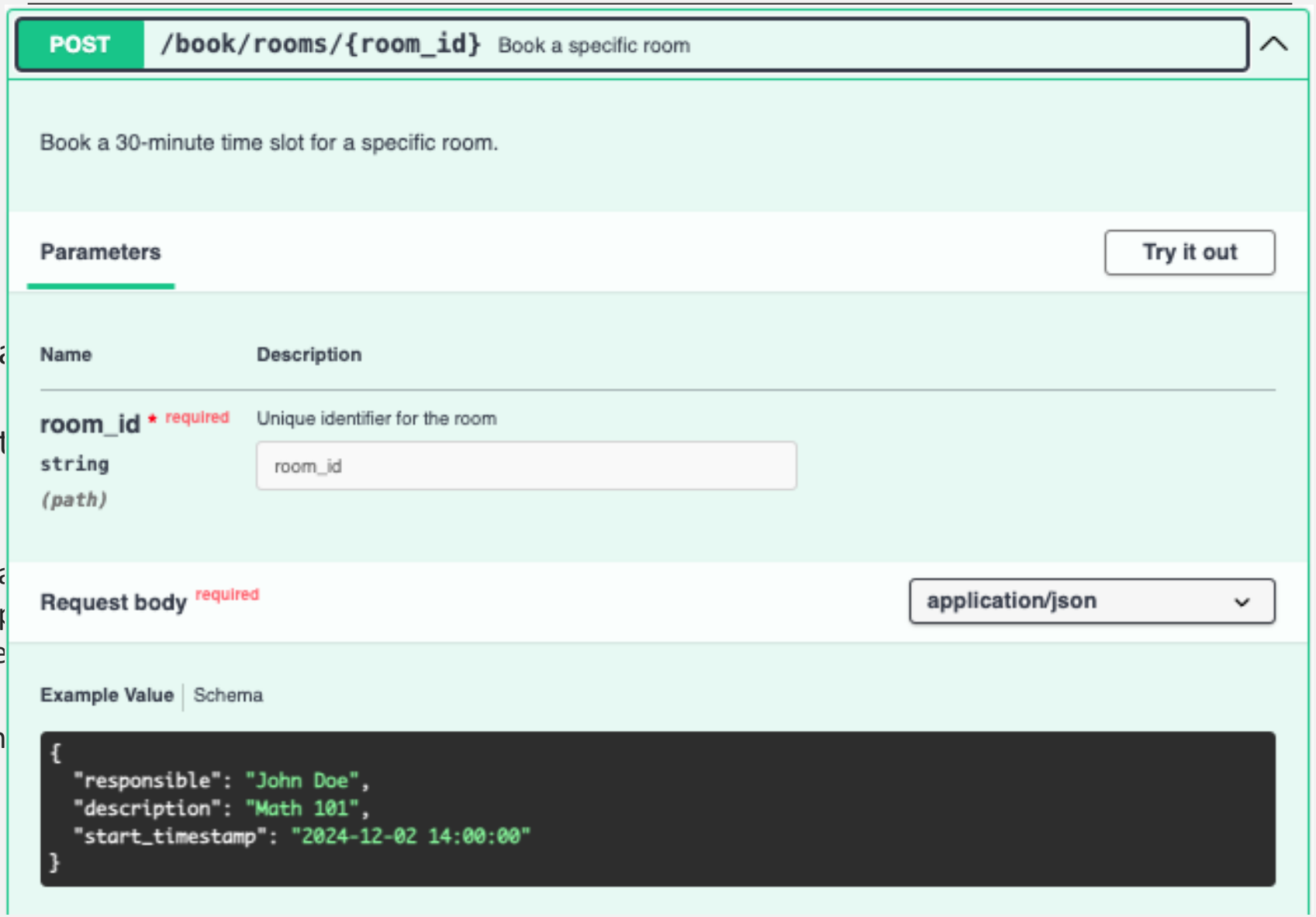
- Data-retrieval fo
- Flask
- Swagger Editor

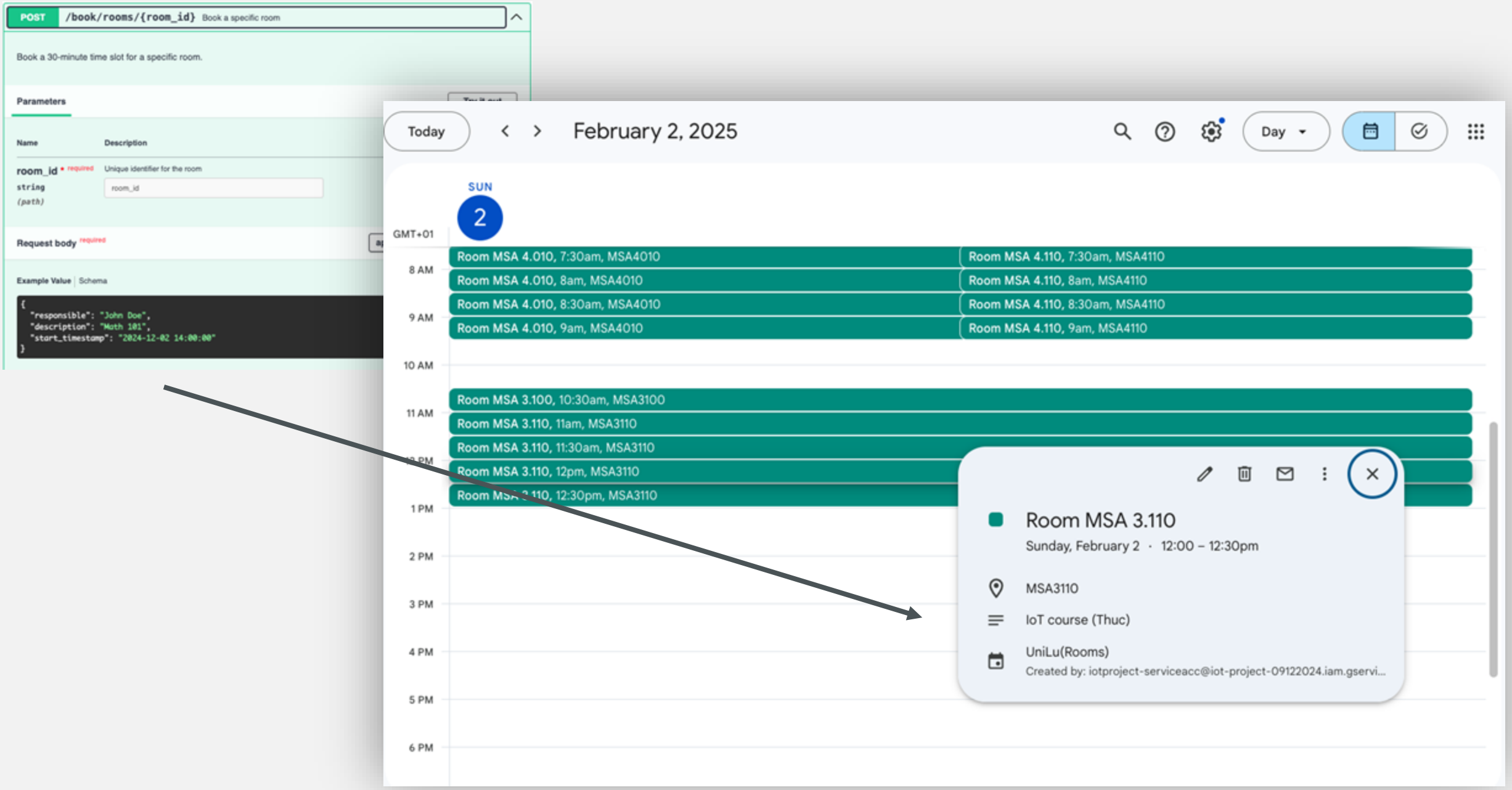
- GET
  - Sensor Data (e.
  - Room Equipme
  - Bookings (e.g. /
- POST
  - Book Rooms



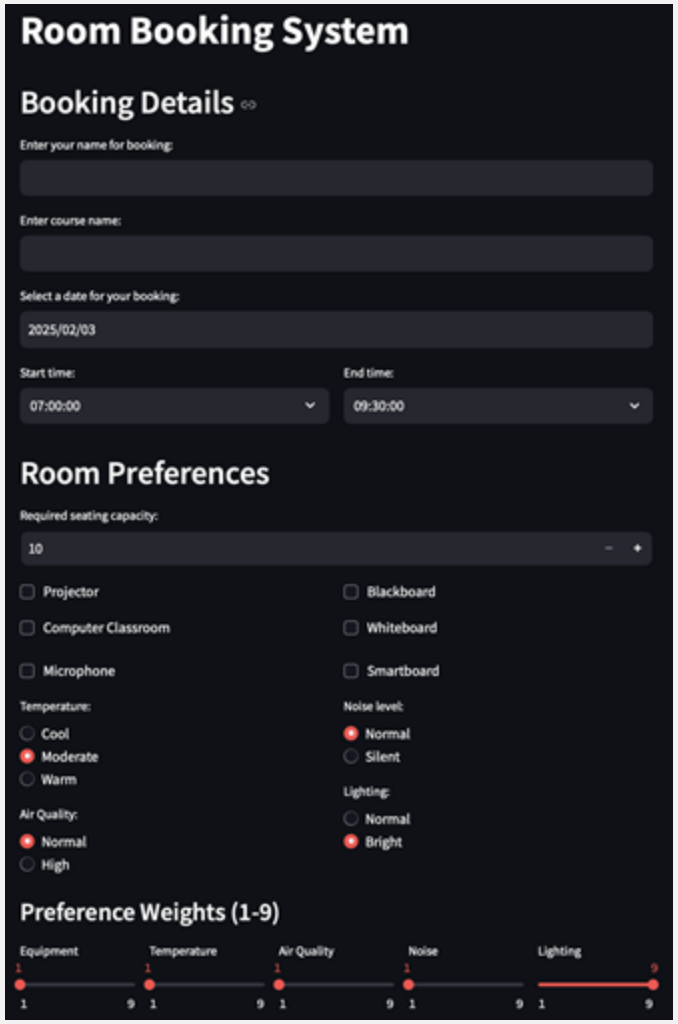
- Data-retrieva
- Flask
- Swagger Edit

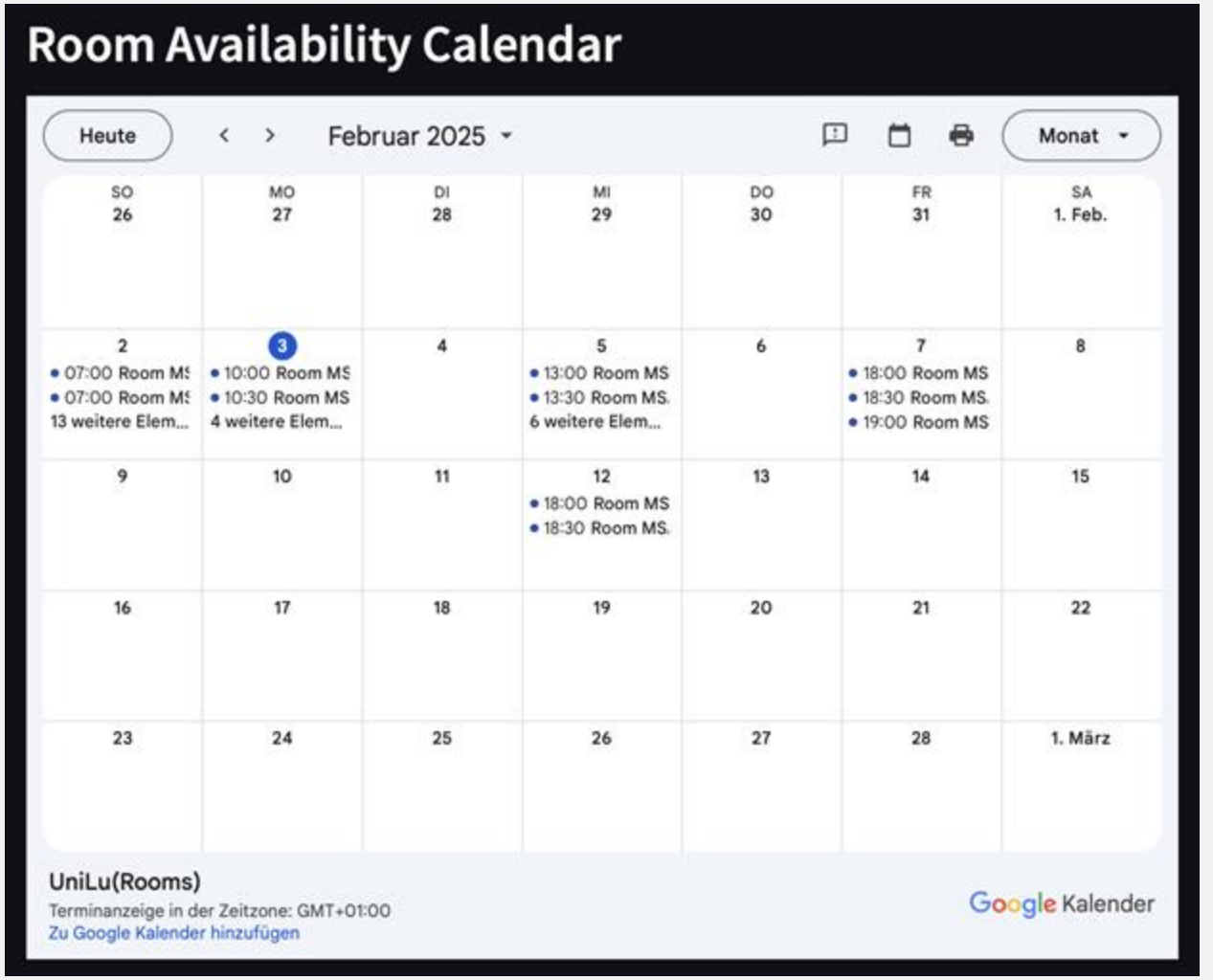
- GET
  - Sensor Data
  - Room Equip
  - Bookings (e
- POST
  - Book Room



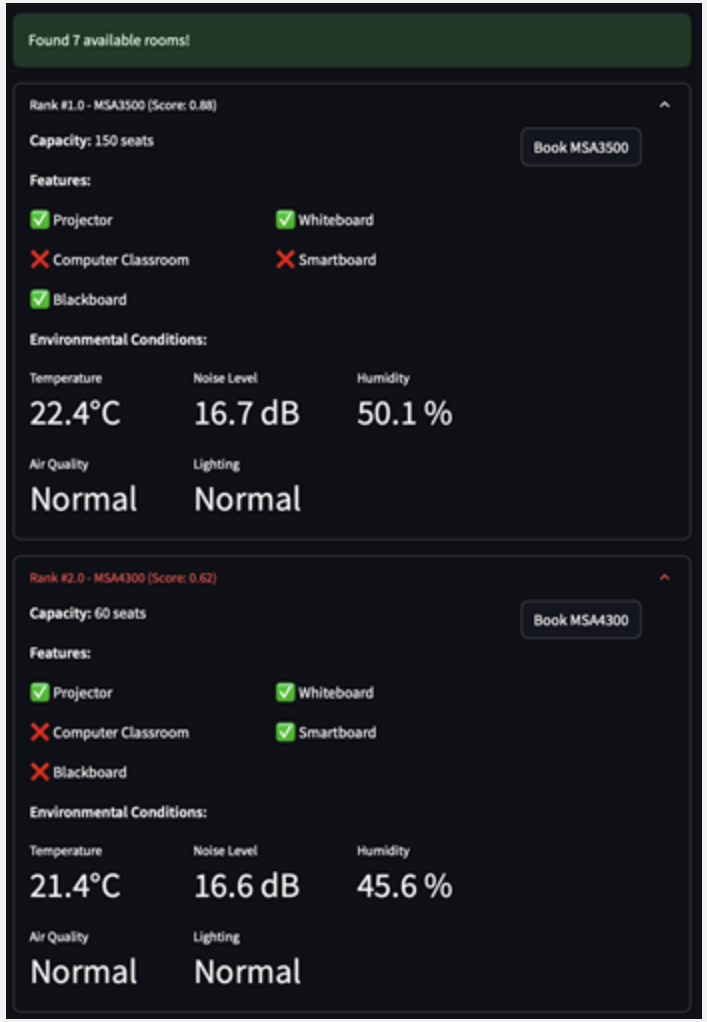

POST /book/rooms/{room_id} Book a specific room
Book a 30-minute time slot for a specific room.
Parameters
Name
Description
room_id • required
Unique identifier for the room
string
(path)
room_id
Request body required
Example Value | Schema
{
"responsible": "John Doe",
"description": "Math 101",
"start_timestamp": "2024-12-02 14:00:00"
}
Today
February 2, 2025
Day
SUN
2
GMT+01
8 AM
9 AM
10 AM
11 AM
1 PM
2 PM
3 PM
4 PM
5 PM
6 PM
Room MSA 4.010, 7:30am, MSA4010
Room MSA 4.110, 7:30am, MSA4110
Room MSA 4.010, 8am, MSA4010
Room MSA 4.110, 8am, MSA4110
Room MSA 4.010, 8:30am, MSA4010
Room MSA 4.110, 8:30am, MSA4110
Room MSA 4.010, 9am, MSA4010
Room MSA 4.110, 9am, MSA4110
Room MSA 3.100, 10:30am, MSA3100
Room MSA 3.110, 11am, MSA3110
Room MSA 3.110, 11:30am, MSA3110
Room MSA 3.110, 12pm, MSA3110
Room MSA 3.110, 12:30pm, MSA3110
Room MSA 3.110
Sunday, February 2 · 12:00 – 12:30pm
MSA3110
IoT course (Thuc)
UniLu(Rooms)
Created by: iotproject-serviceacc@iot-project-09122024.iam.gservi...

BOOKING INTERFACE

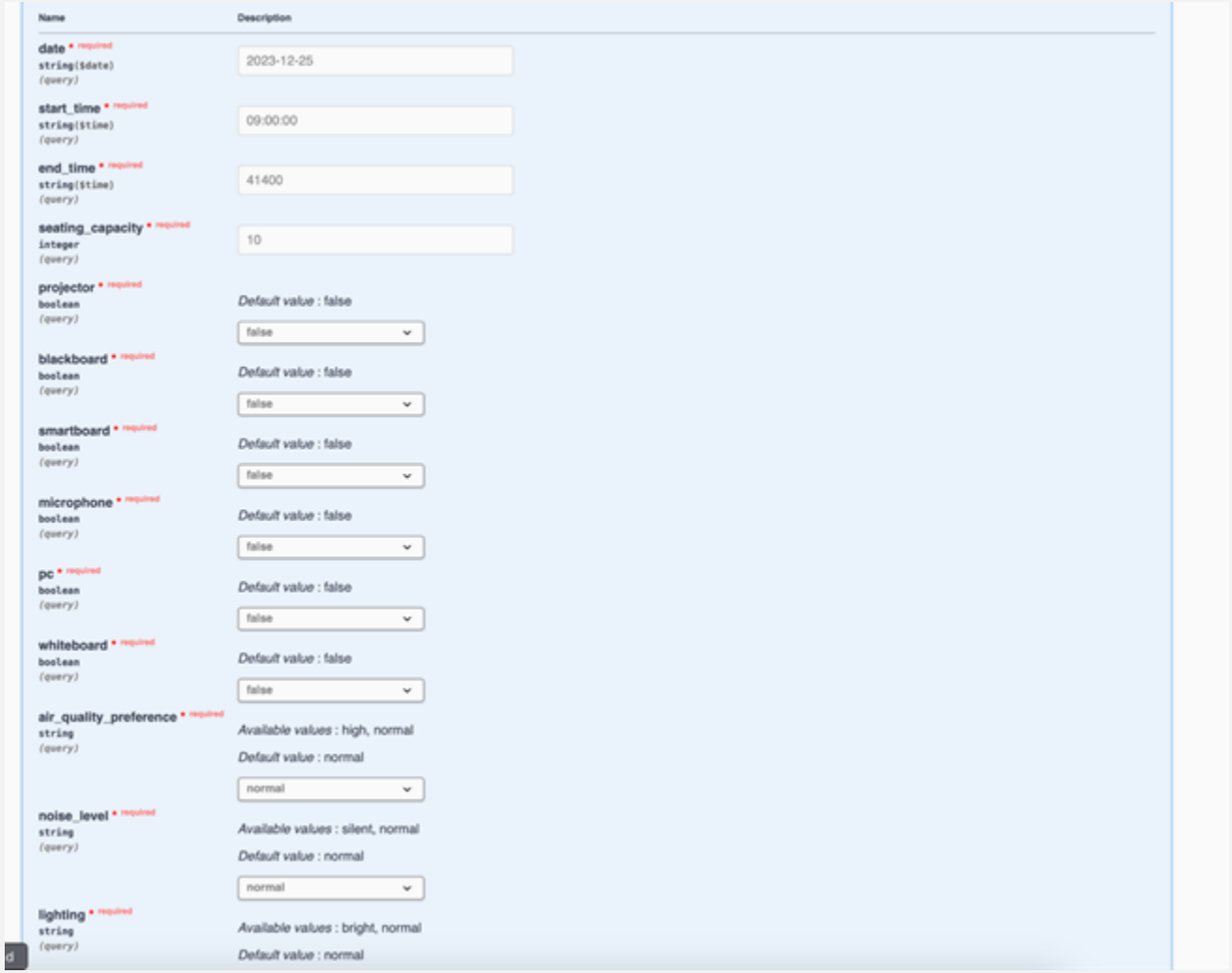




BOOKING INTERFACE



BOOKING SYSTEM (REST API 2)



BOOKING SYSTEM (REST API 2)

Curl

```
curl -X 'GET' \
  'http://localhost:8081/rank-rooms?date=2025-02-01&start_time=09%3A00%3A00&end_time=10%3A00%3A00&seating_capacity=10&projector=false&blackboard=false&smartboard=fa
  -H 'accept: application/json'
```

Request URL

```
http://localhost:8081/rank-rooms?date=2025-02-01&start_time=09%3A00%3A00&end_time=10%3A00%3A00&seating_capacity=10&projector=false&blackboard=false&smartboard=false&micro-
phone=false&pc=true&whiteboard=false&air_quality_preference=normal&noise_level=normal&lighting=normal
```

Server response

Code	Details
200	Response body


```
[
  {
    "blackboard": true,
    "capacity": 35,
    "co2": 650.1240646732953,
    "humidity": 54.82515311034776,
    "light": 700.6565296054714,
    "microphone": false,
    "noise": 33.49589176348788,
    "pc": false,
    "pm10": 23.976841216216215,
    "pm2_5": 9.994346846846847,
    "projector": true,
    "rank": 1,
    "room_id": "WSA4030",
    "score": 0.5777335224704666,
    "smartboard": true,
    "temperature": 24.240629260018675,
    "voc": 220.1343018018018,
    "whiteboard": true
  },
  {
    "blackboard": true,
    "capacity": 150,
```


Download

Response headers

```
connection: close
content-length: 2737
content-type: application/json
date: Sun,02 Feb 2025 23:26:15 GMT
server: Werkzeug/2.2.3 Python/3.11.11
```

Responses

BOOKING SYSTEM (REST API 2)

- GET Request executes decision logic for room ranking
- Fetching available rooms from REST API 1
- EU/German compliance regulation checks for sensor data
- Building Topsis MCDM matrix
- Ranking based on Topsis score

1. **Z-Score Transformation**: To standardize sensor readings:

$$\text{adjusted value} = -\left|\frac{x - \text{user\_pref}}{\sigma}\right|$$

where x is the room's sensor reading, user_pref is the user-defined target, and σ is the standard deviation across all rooms.

2. **Normalization:** Using Euclidean norm operations with a small constant:

$$V_{ij} = \frac{X_{ij}}{\sqrt{\sum X_{ij}^2} + 10^{-9}}$$

3. **Weighting:** User-defined weights are normalized and applied.

4. **Ideal and Negative-Ideal Solutions:** Computed for each attribute:

   - Ideal Best (PIS): Maximum values.
   - Ideal Worst (NIS): Minimum values.

   Attributes where lower values are preferable are adjusted accordingly.

5. **Distance Calculation and Ranking:** Using Euclidean distances:

$$D_i^+ = \sqrt{\sum (V_{ij} - A_j^+)^2}, \quad D_i^- = \sqrt{\sum (V_{ij} - A_j^-)^2}$$

6. **Final Ranking:** Closeness coefficient:

$$C_i = \frac{D_i^-}{D_i^+ + D_i^-}$$

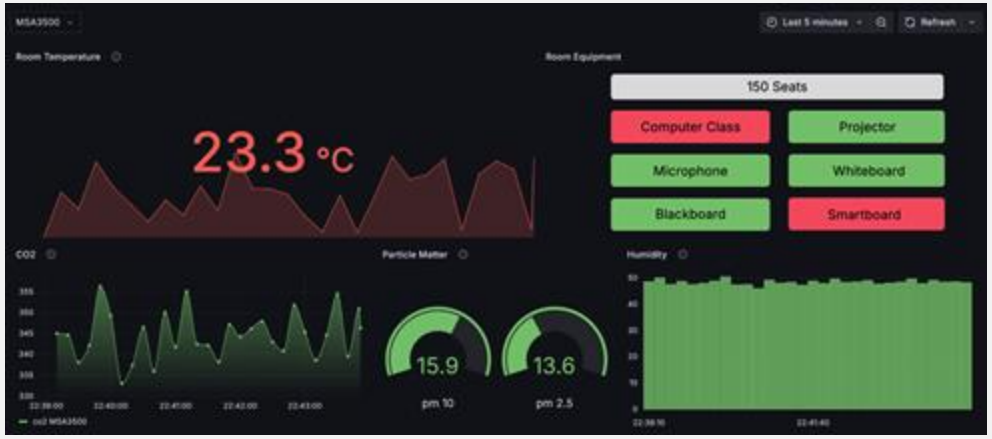
Higher C_i values indicate better room suitability.

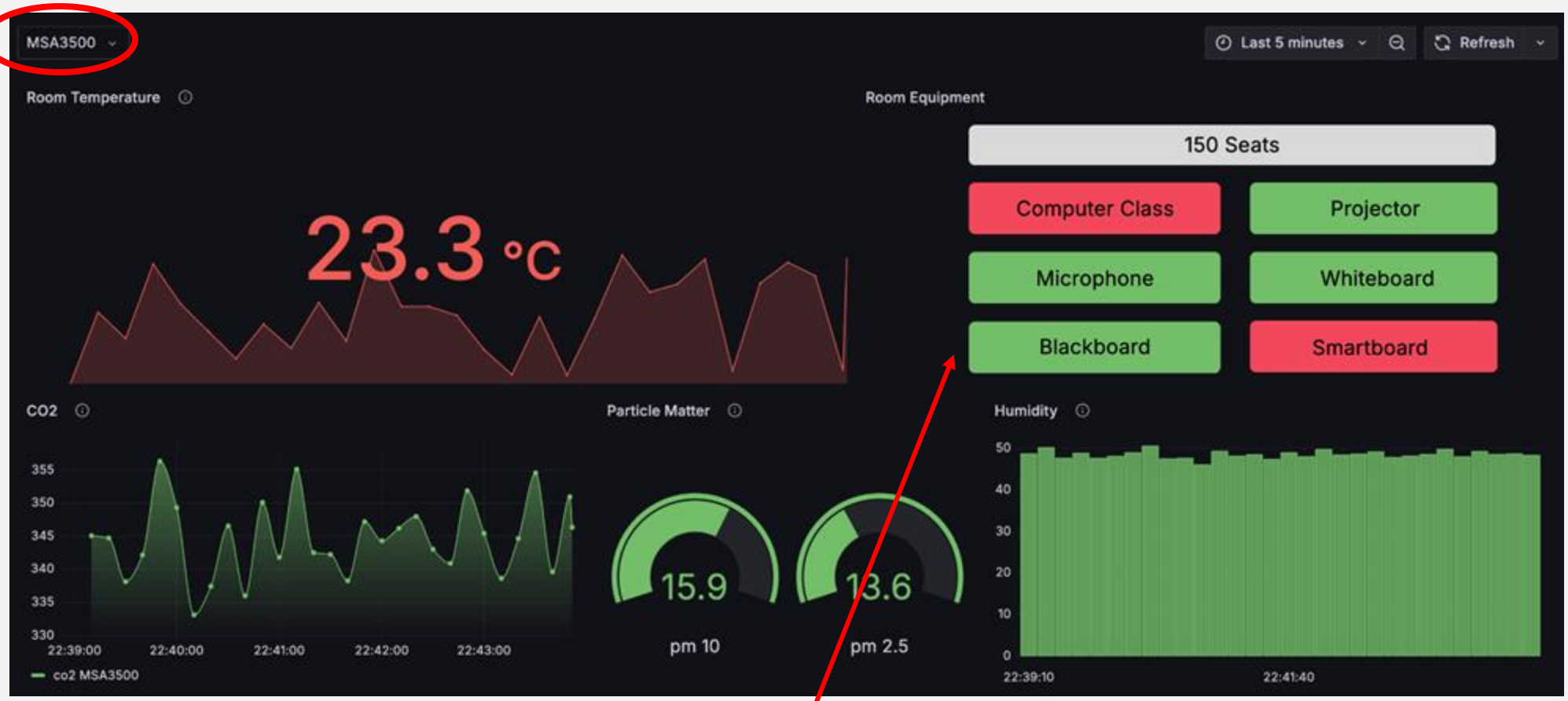
GRAFANA DASHBOARD

- Interface for Administrator
- Observe sensor data
- Detects Anomalies

- Queries
  - InfluxDB (for time series data)
  - Postgres (for static data e.g. equipment)

- Demo Dashboards
  - Overview of room measures
  - Sensor data for specific room





- Queries Postgres DB for available rooms and equipment

SENSOR DATABASE (INFLUX DB)

- Purpose of InfluxDB
- Why InfluxDB for IoT?
- Integration with IoT System
- Data Structure in InfluxDB
  - Measurement
  - Tags
  - Fields
  - Timestamps

SENSOR DATABASE (INFLUX DB)

- Purpose of the Equipment Database
- Why a Dedicated Database for Equipment?
- Integration with IoT System
- Data Structure in PostgreSQL
  - Tables
  - Columns
  - Relationships

THANK YOU FOR ATTENTION!