**A close up of a logo

Description automatically generated**

**Braude College of Engineering  
Department of Software Engineering**

Capstone Project – Phase A

25-2-D-11

**An Integrated Internet of Things Monitoring System**

**Robomo 2.0**

[**GIT**](https://github.com/Mohammed19J/Robomo_2.0)

Collaboration with TAMK University

BY:  
Mohammed Jaber  
Amal Kandeel

Advisor:  
Dr. Naomi Unkelos Shpigel

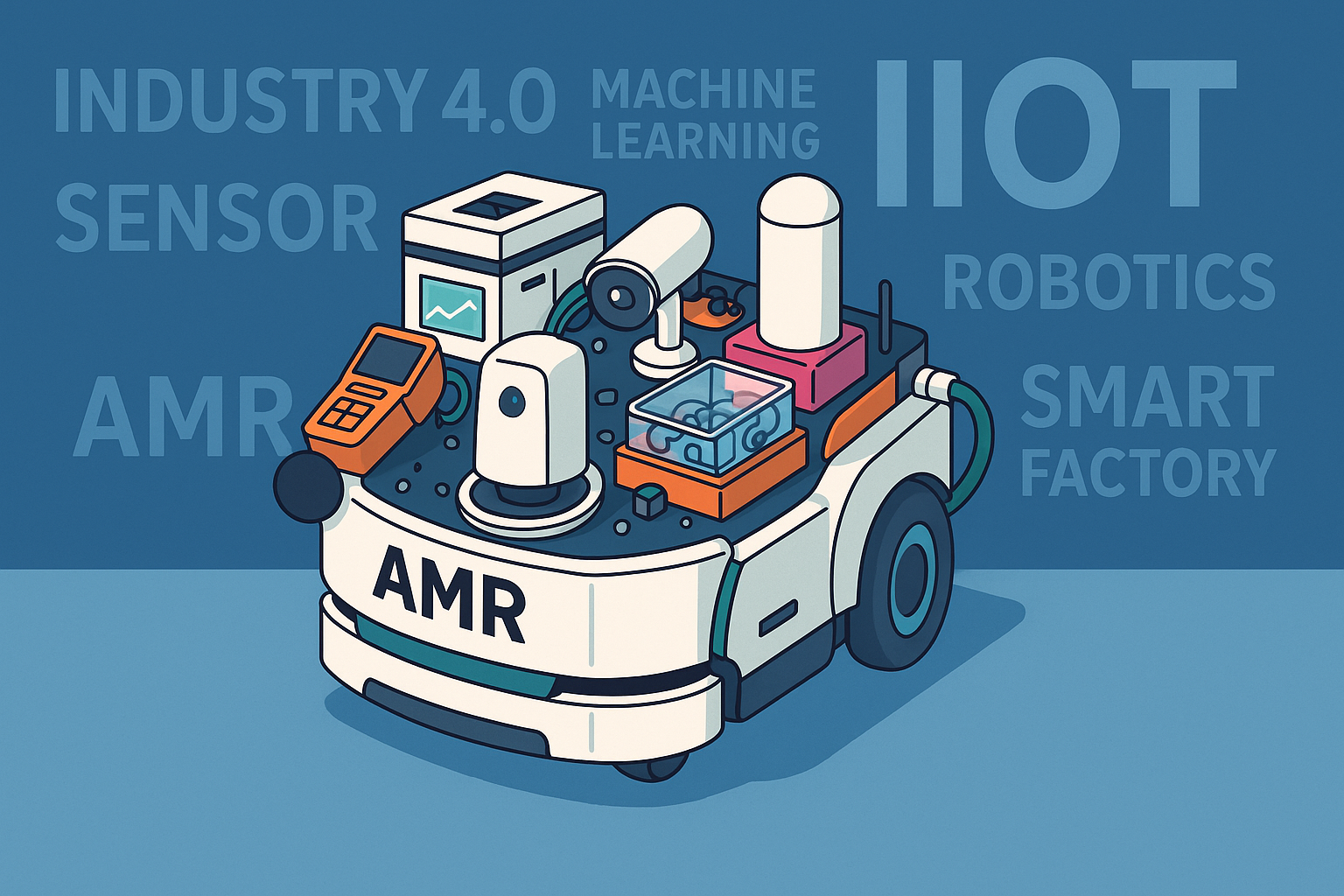


Table of Contents

[**Abstract** 3](#_Toc204812775)

[**1. Introduction** 4](#_Toc204812776)

[**2. Literature Review** 5](#_Toc204812777)

[**2.1. Smart Sensing for Indoor Monitoring** 5](#_Toc204812778)

[**2.2. Real-Time Occupancy Detection and Prediction** 5](#_Toc204812779)

[**2.3. Machine Learning Algorithms for Occupancy Estimation** 5](#_Toc204812780)

[**2.4. Real Time Monitoring in IoT** 6](#_Toc204812781)

[**2.5. Industrial Internet of Things (IIoT)** 6](#_Toc204812782)

[**2.6. Agile Development of Smart Sensor Systems** 6](#_Toc204812783)

[**3. Research** 6](#_Toc204812784)

[**3.1. Scheduled Meeting** 6](#_Toc204812785)

[**3.1.1. Meetings Summary** 7](#_Toc204812786)

[**3.1.1.1. Meeting Number 1 - 30.4.25** 7](#_Toc204812787)

[**3.1.1.2. Meeting Number 2 - 03.6.25** 7](#_Toc204812788)

[**4. Engineering Process** 8](#_Toc204812789)

[**4.1. Development Process** 8](#_Toc204812790)

[**4.2. User-Centered Design Approach** 8](#_Toc204812791)

[**4.2.1. Regular Customer Meetings** 8](#_Toc204812792)

[**4.2.2. Prototype-Driven** 8](#_Toc204812793)

[**4.2.3. Machine Learning Model Selection and Validation** 8](#_Toc204812794)

[**4.2.4. Conclusion** 12](#_Toc204812795)

[**4.3. Workflow** 12](#_Toc204812796)

[**4.3. Architecture Diagram** 13](#_Toc204812797)

[**4.4. Technologies Review** 14](#_Toc204812798)

[**4.4.1. Websocket** 14](#_Toc204812799)

[**4.4.2. Client-Side Technologies** 14](#_Toc204812800)

[**4.4.2.1. React** 14](#_Toc204812801)

[**4.4.2.2. Javascript** 14](#_Toc204812802)

[**4.4.2.3. Tailwind Css** 14](#_Toc204812803)

[**4.4.3. Server Technologies** 15](#_Toc204812804)

[**4.4.3.1. Node.js** 15](#_Toc204812805)

[**4.4.3.2. Fiware** 15](#_Toc204812806)

[**4.4.3.3. Firebase** 15](#_Toc204812807)

[**4.4.3.4. MQTT** 15](#_Toc204812808)

[**5. Work Artifacts** 15](#_Toc204812809)

[**5.1. Requirement** 15](#_Toc204812810)

[**5.1.1. Functional Requirements** 15](#_Toc204812811)

[**5.1.2. Non Functional Requirements** 17](#_Toc204812812)

[**5.2. Use Case** 18](#_Toc204812813)

[**5.2.1. User Functionalities (Figure 4)** 19](#_Toc204812814)

[**5.2.2. Agent Functionalities (Figure 5)** 20](#_Toc204812815)

[**5.3. Activity Diagram** 21](#_Toc204812816)

[**6. Expected Achievements** 24](#_Toc204812817)

[**6.1. Challenges** 24](#_Toc204812818)

[**6.2. Success Criteria** 25](#_Toc204812819)

[**6.3. Evaluation** 25](#_Toc204812820)

[**7. Testing Plan** 26](#_Toc204812821)

[**7.1. Scope** 26](#_Toc204812822)

[**7.2. Objectives** 26](#_Toc204812823)

[**7.3. Testing Approach** 27](#_Toc204812824)

[**8. AI Tools and Prompts** 29](#_Toc204812825)

[**8.1. AI Usage and Results** 29](#_Toc204812826)

[**8.2. AI Prompts and Responses** 29](#_Toc204812827)

[**9. References** 31](#_Toc204812828)

[**9.1 Academic sources (journals, conferences, datasets, standards)** 31](#_Toc204812829)

[**9.2 Web resources / developer documentation** 33](#_Toc204812830)

### **Abstract**

In recent years, smart sensor systems have been increasingly deployed in educational environments to enable real time environmental monitoring and space utilization analysis. In collaboration with TAMK University, our project builds upon an existing IIoT (Industrial Internet of Things) framework and extends it with intelligent capabilities. Specifically, we developed a system that uses environmental sensors measuring CO₂, temperature, humidity, and VOC (Volatile Organic Compounds) to estimate room occupancy using machine learning.

We inherited a working web application from a previous student team and expanded it by integrating data driven features, including a machine learning-based agent capable of analyzing air quality data to detect occupancy, predict health index values, and even trigger alerts for fire/smoke or HVAC adjustments. To achieve this, we trained and tested models like Random Forest, Support Vector Regression (SVR), and LSTM (Long Short-Term Memory) on both public and collected datasets.

The result is a real-time monitoring system that presents predictions through an intuitive dashboard and preserves user privacy, making it suitable for sensitive environments such as exam rooms. This project reflects the strength of interdisciplinary collaboration between Braude College and TAMK, showcasing the potential of AI-powered IIoT solutions for smart academic infrastructures.

### **1. Introduction**

Poor indoor air quality is a growing concern in modern environments such as classrooms, exam halls, and offices. Elevated levels of carbon dioxide (CO₂), particulate matter (PM1, PM2.5, PM10), and volatile organic compounds (VOCs) have been linked to health issues, cognitive impairment, and discomfort [2], [11], [21]. In addition, sudden spikes in these indicators may serve as early signs of smoke or fire hazards [3], [18]. Traditional occupancy detection methods, such as cameras or motion sensors, often raise privacy concerns or lack sufficient spatial resolution. These challenges highlight the need for non-intrusive, mobile, and intelligent monitoring systems that can support real-time analysis and control actions particularly in sensitive environments like exam rooms.

This project is part of a collaboration between Ort Braude College and TAMK University in Finland, aimed at advancing smart campus systems through sensor data and intelligent analytics. Our objective was to build upon an existing IIoT system and introduce a machine learning agent capable of enhancing the value of collected data. Specifically, we focused on predicting whether a room is occupied based on air quality measurements, rather than relying on intrusive methods such as cameras.

To do this, we used real time data from Robomo units mobile robots equipped with environmental sensors and centralized it using WebSocket and MQTT protocols. Our system tracks CO₂ levels, temperature, humidity, PM sensors, and VOCs, and uses these features to infer room occupancy. This is particularly relevant for spaces like exam rooms where privacy must be maintained, and traditional surveillance is not ideal.

Throughout development, we followed Agile principles, held regular meetings with our supervisor from TAMK (Mr. Kari Naakka), and iteratively improved the application. We tested different machine learning algorithms (including Random Forest, SVR, and LSTM) on both public and collected datasets. Our goal wasn’t to detect the exact number of people, but rather to classify whether the room is occupied or empty, enabling smarter HVAC (Heating,

### **2. Literature Review**

#### **2.1. Smart Sensing for Indoor Monitoring**

Indoor air quality (IAQ) sensing is central to our system. CO₂, temperature, humidity, VOC, and particulate matter (PM) are known to correlate with human presence and comfort, and they can also indicate potential hazards (e.g., smoke/PM spikes) [2], [11], [21]. Mounting these sensors on a mobile robot (Robomo) enables localized, flexible coverage that reduces blind spots found in static installations and improves the representativeness of measurements [6], [9], [16], [19]. In practice, CO₂ often emerges as a strong proxy for occupancy, while fusing multiple modalities (e.g., ΔCO₂/Δt, temperature–humidity trends, VOC and PM levels) improves robustness to environmental confounders [2], [11], [21].

#### **2.2. Real-Time Occupancy Detection and Prediction**

Occupancy estimation from environmental signals offers a privacy-respecting alternative to vision-based methods. As people enter or leave a room, characteristic patterns appear in CO₂, temperature, humidity, and sometimes VOC/PM, which statistical and machine-learning models can learn to classify rooms as occupied or vacant and, in some cases, estimate counts [2], [21]. To support low-latency inference and visualization, we stream time-synchronized data to the backend via MQTT and deliver updates to the dashboard via WebSockets [14], [15], [8]. For evaluation and bootstrapping, we used public datasets relevant to IAQ/occupancy and smoke (for hazard detection), while we continue collecting live data to refine feature engineering and model fit [1], [2], [3], [18], [21]. This pipeline enables downstream actions such as HVAC set-point adjustments and usage logging without cameras [2], [21].

#### **2.3. Machine Learning Algorithms for Occupancy Estimation**

Prior work reports a range of effective models using environmental features, including Random Forests, linear/polynomial models, SVM/SVR, and sequence models (e.g., LSTM) when temporal dependencies matter [2], [11], [21]. In our experiments we compared:  
• Random Forests: robust to non-linearities and feature interactions strong baselines on small/medium datasets and relatively interpretable via feature importance [2], [21].  
• SVR/SVM: effective when margins are clear used for binary occupancy or count regression [2], [21].  
• LSTM (and other RNNs): capture temporal patterns in sensor streams where lagged effects are meaningful [21].  
• Linear/Polynomial models: competitive in controlled settings with stable relationships and few variables [2].  
Common preprocessing: smoothing/denoising (e.g., Kalman-style filtering), windowed features, and rate-of-change indicators like ΔCO₂/Δt improves performance and stability [11], [21].

#### **2.4. Real Time Monitoring in IoT**

Our system operates as a real time IoT pipeline: Robomo publishes sensor frames over MQTT to a backend that processes and stores the data the dashboard receives live updates over WebSockets for immediate feedback to users [14], [15], [8]. In the robotics IoT literature, this architecture aligns with the “Internet/Robotic Things” direction that merges sensing, actuation, and cloud services for responsive applications [6], [9], [16], [19]. Compared to fixed wall sensors, the mobile platform adds spatial resolution, enabling detection of localized air-quality issues (e.g., hotspots, leakage, source proximity) that static layouts may miss [6], [16], [19].

#### **2.5. Industrial Internet of Things (IIoT)**

Positioning Robomo within the IIoT emphasizes interoperability, modularity, and secure data exchange among heterogeneous devices and services [4], [19]. Open protocols such as MQTT facilitate low-overhead telemetry from constrained devices, and platform ecosystems such as FIWARE support standardized context management and integration with analytics and actuation services [14], [15], [22]. This modular approach scales to additional rooms/robots and integrates with building systems (e.g., HVAC), aligning with emerging IoRT/IIoT practices in robotics-enabled monitoring [6], [16], [19]. Digital-twin concepts offer a natural extension for what-if analysis and fault detection based on the live data stream [5], [17], [19].

#### **2.6. Agile Development of Smart Sensor Systems**

We followed an iterative, sprint-based process with weekly planning, incrementally delivering sensor bring-up, occupancy modeling, smoke detection, and HVAC recommendations. Iterations were informed by tests and stakeholder feedback for example, early attempts at headcount regression proved unreliable, so we prioritized robust binary occupancy before expanding features [2], [21]. Configuration management and traceability supported stable evolution of models and data pipelines across sprints [7].

### **3. Research**

#### **3.1. Scheduled Meeting**

Throughout this project, we held scheduled meetings with Senior Lecturer Mr. Kari Naakka from TAMK University, alongside our advisor, Dr. Naomi Unkelos Shpigel. These meetings were crucial for guiding our work and refining our project goals. In each session, we presented our latest progress starting from taking over the system and setting it up, to sharing our research on machine learning approaches for occupancy detection using air quality sensors.

We used these meetings not only to report on what we had accomplished but also to discuss challenges we faced, such as the difficulty of accurately estimating the number of people in a room based on sensor data alone. These discussions led to valuable advice from Mr. Naakka, including the idea of focusing on presence detection and using the data to control HVAC systems, detect smoke and fire, and contribute to energy-saving efforts. Each meeting ended with clear action points and next steps, helping us maintain steady progress and align our work with both TAMK and Braude College expectations.

#### **3.1.1. Meetings Summary**

#### **3.1.1.1. Meeting Number 1 - 30.4.25**

During our first meeting, which was held via Microsoft Teams, the participants included Mohammed, Amal, Mr. Kari Naakka from TAMK University, and our advisor, Dr. Naomi Unkelos Shpigel. We discussed the direction of the project and evaluated the existing web application developed by a previous team. From there, we began shaping our own contribution. One of the key ideas we introduced was applying a machine learning algorithm to the sensor data we had collected. Mr. Naakka was impressed by this direction, and we had an in-depth discussion about which type of sensor would be most suitable for training the model. We ultimately chose the air quality sensor, as we already had a large dataset available and could continue collecting real-time data for training and prediction.

In addition, we briefly presented another project (see Figure 1) that we had developed in Unity using augmented reality. Mr. Naakka was enthusiastic about the concept but advised us to focus, for the time being, solely on the machine learning component and to expand the range of use cases supported by the web application related to the robotic systems.

Figure 1. First meeting with Kari

A collage of people wearing headphones

AI-generated content may be incorrect.

#### **3.1.1.2. Meeting Number 2 - 03.6.25**

During our second and final meeting with Mr. Kari Naakka from TAMK University, we presented our research findings and experimental results. Initially, our focus was on precise occupancy detection estimating the exact number of occupants in a room using available environmental sensors. However, our research concluded that accurate occupant counting required additional methods beyond our existing sensor capabilities, such as cameras or Bluetooth connectivity.

Consequently, we shifted our focus toward occupancy level detection (empty, low, medium, high), air quality indexing, and smoke and fire detection, utilizing environmental sensor data. Our extensive experimentation involved several machine learning models, including KNN, Random Forest, XGBoost, LightGBM, SVM, and Logistic Regression, evaluated on credible datasets from Kaggle and Mendeley Data. Random Forest emerged as the most effective model across all three use cases due to its accuracy, interpretability, and scalability.

Mr. Naakka expressed strong approval and support for these alternative approaches, affirming the project's direction. We discussed further implementation strategies and the integration of the selected ML model into our system architecture. This final meeting effectively defined and validated the next steps of our project's development.

### **4. Engineering Process**

#### **4.1. Development Process**

Our engineering process primarily focused on developing a robust machine learning (ML) agentic system, leveraging real-time sensor data from the TAMK API. This agent system was designed and initially tested using Google Colab, preparing it for future integration into the project's frontend and backend infrastructure in the upcoming semester. Our ML models address three critical use cases: occupancy prediction, fire and smoke detection, and computation of the Air Quality Index (AQI) on a standardized scale from 0 to 100, following established AQI calculation guidelines. Additionally, data utilized in the development and evaluation of these models was archived to support future enhancements and retraining.

Key challenges during this phase included ensuring reliable preprocessing of sensor data, addressing noise and synchronization issues, and maintaining efficient inference capabilities. We structured a preprocessing pipeline optimized for ML model performance and developed a model-selection strategy that dynamically chooses the most appropriate ML algorithm for specific analyses.

#### **4.2. User-Centered Design Approach**

Our project adopted a User-Centered Design (UCD) methodology, ensuring alignment with customer expectations and usability requirements. This approach significantly influenced our development decisions, prioritizing functionality, interpretability, and responsiveness in our system design.

#### **4.2.1. Regular Customer Meetings**

Regular meetings with Kari Naakka, the TAMK project manager and primary stakeholder, were pivotal in identifying and refining the project's objectives and use cases. Our initial meeting involved ideation and capturing Kari's vision and desired functionalities. Subsequent research and experimentation informed a detailed implementation plan, which we validated and confirmed during our second meeting. Feedback from these interactions directly influenced our ML model objectives and approach.

#### **4.2.2. Prototype-Driven**

Our prototype-driven approach facilitated iterative validation of ML model capabilities and user expectations:

* Initial prototypes validated basic model functionality and accuracy.
* Intermediate prototypes included refined predictions and anomaly detection capabilities.
* Final prototype iterations emphasized model scalability and interpretability, addressing stakeholder preferences for clarity and ease of use.

#### **4.2.3. Machine Learning Model Selection and Validation**

Extensive experimentation was conducted to determine optimal machine learning algorithms for each use case (occupancy prediction, fire/smoke detection, and AQI calculation). We evaluated at least six ML algorithms using credible datasets from Kaggle and Mendeley Data, carefully matching our project needs. Each algorithm was assessed using standard evaluation metrics, leading us to select Random Forest as the most consistently high-performing model across all scenarios due to its accuracy, scalability, and clear interpretability.

To identify the most suitable machine learning models for our system, we conducted systematic experimentation for three key use cases:

1. Occupancy Prediction
2. Smoke/Fire Detection
3. Indoor Air Quality Index (IAQI) Estimation

We used cleaned and labeled datasets collected from sensor-equipped environments (e.g., home, gym, lab) and public repositories (Kaggle, Mendeley). The selected features were derived from sensors measuring:

* Temperature
* Humidity
* CO₂
* VOC
* Particulate Matter (PM1, PM2.5, PM4, PM10)
* Nox
* CO (for IAQI)

Model Training and Evaluation

Each model was trained using an 80/20 train-test split with stratification (for classification tasks) or random sampling (for regression). Input features were standardized using StandardScaler. We evaluated several well known algorithms for each use case:

* Classification (Occupancy / Fire Detection):
  + Random Forest
  + XGBoost
  + LightGBM
  + Support Vector Machine (SVM)
  + Logistic Regression
  + K-Nearest Neighbors (KNN)
* Regression (IAQI estimation):
  + Random Forest Regressor
  + XGBoost Regressor
  + LightGBM Regressor
  + SVR
  + KNN Regressor
  + Linear Regression

Metrics Used

For classification tasks, we used:

* Accuracy
* Precision (weighted)
* Recall (weighted)
* F1 Score (weighted)

For regression tasks:

* Mean Absolute Error (MAE)
* Root Mean Squared Error (RMSE)
* R² Score

Summary of Results

After testing and comparing models, we found the following:

* Occupancy Prediction:

Random Forest Classifier outperformed all other models with an F1 Score above 0.95, offering both high accuracy and interpretability (via feature importance).

* Smoke/Fire Detection:

Again, Random Forest achieved the highest precision and F1 Score (both above 0.99), making it the most reliable for safety-critical alerts.

* IAQI Estimation:

Random Forest Regressor produced the best results with R² ≈ 0.99, low MAE, and excellent generalization.

The following pseudocode summarizes the evaluation logic used:

|  |
| --- |
| for name, model in models:  model.fit(X\_train, y\_train)  predictions = model.predict(X\_test)  metrics = evaluate(predictions, y\_test)  log\_results(name, metrics) |

Final Decision:

We selected Random Forest as the primary algorithm for all three tasks. Its consistent performance, low inference latency, resistance to overfitting, and clear feature importance made it the best fit for our real time IIoT system. Moreover, its transparency supports future integration with explainable AI (XAI) techniques.

**Room occupancy estimation**: multiple classification models were evaluated. As shown in the table below, the Random Forest and K-Nearest Neighbors (KNN) models demonstrated the highest performance, with Random Forest being selected for its balance of accuracy and interpretability.

Table 1: Room occupancy estimation models evaluation results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1 Balance of precision & recall |
| KNN | 0.952378 | 0.952371 | 0.952378 | 0.952366 |
| Random Forest | 0.951691 | 0.951580 | 0.951691 | 0.951629 |
| XGBoost | 0.935856 | 0.935431 | 0.935856 | 0.935511 |
| LightGBM | 0.932930 | 0.932297 | 0.932930 | 0.932503 |
| SVM | 0.780438 | 0.764408 | 0.780438 | 0.758536 |
| Logistic Regression | 0.719627 | 0.603513 | 0.719627 | 0.641736 |

**Air Quality Classification**: For calculating the Air Quality Index (AQI), a regression task, several models were tested. The Random Forest model significantly outperformed others, achieving the lowest Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) and the highest R-squared (R2) value, indicating a superior fit to the data.

Table 2: Air Quality Classification models evaluation results

|  |  |  |  |
| --- | --- | --- | --- |
| Model | MAE  (Average of how far predictions are from actual values) | RMSE  (Like MAE but gives more weight to big errors) | R2  (Measures how well the model explains the data) |
| RandomForest | 0.030005 | 0.680490 | 0.995966 |
| LightGBM | 0.251340 | 2.124422 | 0.960683 |
| XGBoost | 0.283394 | 2.189213 | 0.958249 |
| LinearReg | 5.291437 | 6.702015 | 0.608703 |
| KNN | 4.831589 | 6.768823 | 0.600863 |
| SVR | 5.817051 | 7.397784 | 0.523241 |

**Smoke Detection**: In the critical task of smoke detection, various models were assessed for their classification accuracy. While LightGBM and XGBoost showed marginally higher metrics, the Random Forest model also achieved exceptional performance (over 99.9% accuracy) and was chosen for consistency across all use cases.

Table 3: Smoke Detection models evaluation results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1 |
| LightGBM | 0.999760 | 0.999777 | 0.999888 | 0.999832 |
| XGBoost | 0.999760 | 0.999888 | 0.999777 | 0.999832 |
| Random Forest | 0.999441 | 0.999665 | 0.999553 | 0.999609 |
| KNN | 0.992496 | 0.991564 | 0.997989 | 0.994766 |
| SVM | 0.931981 | 0.950796 | 0.954195 | 0.952492 |
| Logistic Regression | 0.777503 | 0.771638 | 0.978103 | 0.862689 |

#### **4.2.4. Conclusion**

Our iterative, user-centered development and thorough validation processes resulted in the selection and preparation of robust ML models tailored to our project's requirements. The clear emphasis on usability, interpretability, and future scalability provides a strong foundation for the successful integration and deployment of these models in the subsequent development phases.

#### **4.3. Workflow**

At the start of the project, we held an introductory meeting with our fellow students, Or and Dima, from whom we inherited the initial system architecture and foundational infrastructure. During this meeting, essential information such as credentials and system access was exchanged, and the backend structure was transitioned to our management.

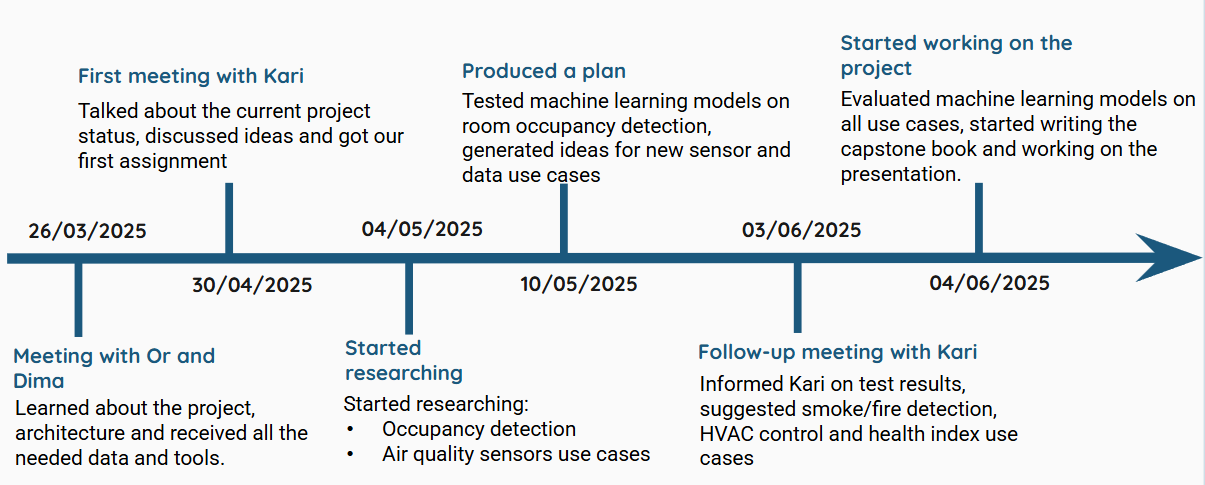
Following this, we initiated discussions with our project manager at TAMK University, Mr. Kari Naakka, to identify potential enhancements for the project. Kari initially suggested exploring precise occupancy detection estimating the exact number of people present using available environmental sensors. We conducted comprehensive research into this idea and discovered that accurate occupant counting was not feasible solely with our available sensors, as it would require additional methods like cameras or Bluetooth connectivity checks.

However, our research identified alternative valuable approaches, such as occupancy level detection (empty/low/medium/high), air quality indexing, and smoke and fire detection. We extensively tested these new concepts using publicly available datasets from platforms like Kaggle and Mendeley Data, applying various machine learning models including KNN, Random Forest, XGBoost, LightGBM, SVM, and Logistic Regression. These models were evaluated using metrics such as accuracy, precision, recall, F1 score, MAE, RMSE, and R².

Our results demonstrated that Random Forest consistently delivered outstanding performance across all three use cases, particularly noted for its accuracy, scalability, and interpretability. We presented these findings and suggested use cases to Kari during our second and final meeting, where he expressed strong approval and confirmed his interest in proceeding with these applications.

Following this validation, we focused on designing the system's architecture and workflow, clearly defining functional and non functional requirements, and preparing detailed documentation for our capstone presentation. This foundational work positions us effectively for integrating the ML driven agentic system into the existing project infrastructure during the upcoming development phase.

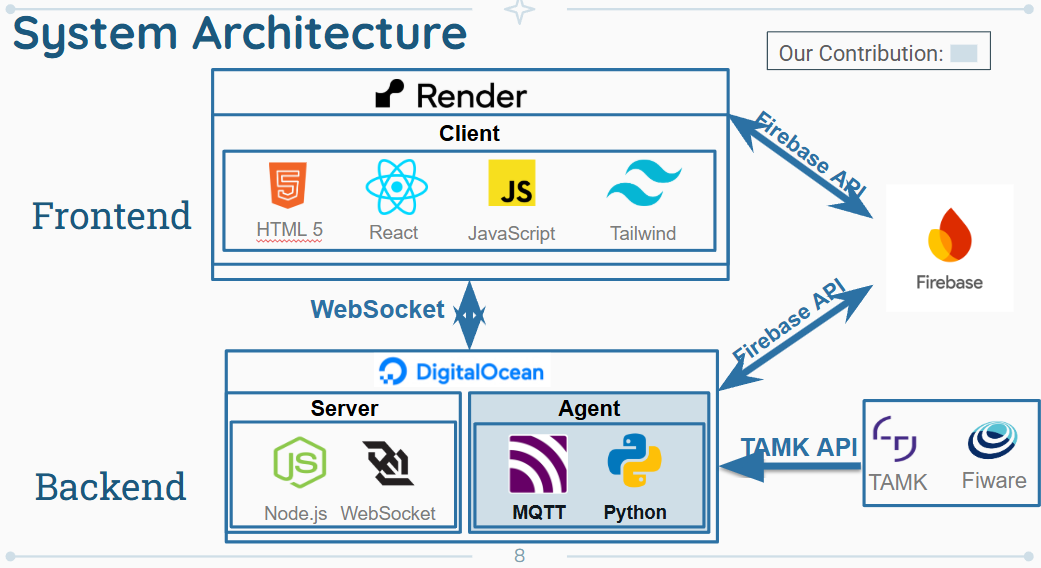
Figure 2:Workflow diagram



#### **4.3. Architecture Diagram**

Figure 6 illustrates the system architecture of our project, highlighting the integration of frontend and backend components to deliver a scalable, real time monitoring solution. The frontend, deployed on Render, is built using HTML5, React, JavaScript, and Tailwind CSS, providing users with an interactive interface to access real-time data and system insights. It communicates with the backend via WebSocket for continuous, low-latency data exchange. The backend, hosted on DigitalOcean, consists of a Node.js server that manages WebSocket communication and processes incoming data, alongside a Python based agent that handles MQTT messages and external API calls. The agent interacts with the TAMK API and Fiware to retrieve sensor data and context information, while Firebase is used not only for real-time data storage and notifications but also to log agent actions for future system improvements and analysis. This architecture supports seamless integration between user interfaces, sensor data collection, cloud storage, and external services, enabling a robust and privacy conscious smart monitoring system.

Figure 3:Architecture diagram

****

#### **4.4. Technologies Review**

Our project integrates several modern technologies to facilitate real-time data collection, processing, visualization, and machine learning model integration. These technologies provide a responsive and scalable system that supports continuous data flow and user interaction.

#### **4.4.1. Websocket**

WebSocket is employed as a real-time, bidirectional communication protocol, essential for seamless data exchange between the server and the client dashboard. Unlike traditional HTTP requests, WebSocket maintains an open connection, allowing instantaneous transmission of sensor data and ML model predictions, ensuring users receive timely updates reflecting the current environment.

#### **4.4.2. Client-Side Technologies**

#### **4.4.2.1. React**

React forms the foundation of our frontend interface, leveraging a component-based architecture to facilitate efficient management of complex user interfaces. React components dynamically update in response to new data, enabling real-time visualization of sensor data, predictions, and analytics without necessitating full page reloads.

#### **4.4.2.2. Javascript**

JavaScript is central to our frontend interactivity, enabling dynamic behavior and real-time responsiveness within React components. It facilitates data processing, visual updates, and user interaction handling, supporting a fluid and interactive user experience.

#### **4.4.2.3. Tailwind Css**

Tailwind CSS is utilized for precise and flexible styling, relying on utility-first CSS classes rather than predefined UI components.This approach allows us to build custom, responsive, and maintainable designs aligned specifically with our project's requirements, maintaining a cohesive and modern user interface.

#### **4.4.3. Server Technologies**

#### **4.4.3.1. Node.js**

Node.js serves as the backend's core technology, providing an efficient, lightweight, event-driven environment tailored for real time data processing. Its non blocking architecture effectively handles continuous data streams from sensors, supporting seamless WebSocket communication and interaction with machine learning algorithms. Additionally, Node.js powers our backend API endpoints and overall data handling processes.

#### **4.4.3.2. Fiware**

FIWARE supports the management and structuring of contextual information from sensor data. Its standardized APIs facilitate scalability and integration into smart environments or smart city infrastructures, enhancing our system's modularity and adaptability for future expansions.

#### **4.4.3.3. Firebase**

Firebase Realtime Database stores real-time sensor data, machine learning predictions, and logs of system activities. The automatic synchronization of Firebase across clients ensures consistent, up-to-date information for users, further supporting secure user authentication and role-based access control.

#### **4.4.3.4. MQTT**

MQTT (Message Queuing Telemetry Transport) is a lightweight publish subscribe messaging protocol ideally suited for IoT applications, particularly in environments with variable network conditions. In our project, MQTT facilitates efficient and reliable data transfer between the sensor equipped robotic units and the server, supporting real time streaming of environmental data essential for backend processing and machine learning predictions.

### **5. Work Artifacts**

#### **5.1. Requirement**

#### **5.1.1. Functional Requirements**

Table 4 outlines the functional requirements of the system. These requirements define the core behaviors and capabilities that the system must support, such as sensor data processing, real time visualization, alerting mechanisms, and user interactions on the dashboard.

Table 4: Functional Requirements

|  |  |
| --- | --- |
| **No.** | **Requirement** |
| 1 | The system shall detect room occupancy using CO₂, temperature, humidity, and VOC sensors. |
| 2 | The system shall display real-time sensor data and Health Index on the web dashboard. |
| 3 | The system shall compute and display a Health Index score that summarizes air quality conditions. |
| 4 | The system shall trigger alerts for abnormal air quality conditions (e.g., high CO₂ levels, smoke detection via VOC/temperature spikes). |
| 5 | The system shall recommend actions (e.g., increase ventilation) when air quality deteriorates or a room is unoccupied. |
| 6 | The system shall log all sensor data, predictions, alerts, and recommendations for review and analysis. |
| 7 | The system shall provide a live map showing robot location and sensor status. |
| 8 | The system shall allow users to generate graphs for selected attributes and view historical data. |
| 9 | The system shall send notifications when manual intervention is required (e.g., when HVAC cannot be automatically controlled). |
| 10 | The system shall support fire/smoke detection logic as part of safety features. |
| 11 | The system shall allow dynamic selection and comparison of multiple devices and sensors on the dashboard. |
| 12 | The system will allow the administrator to approve new sign-ups. |
| 13 | The system will allow users to select devices from a map. |
| 14 | The system will allow users to choose which devices are shown on the map. |
| 15 | The system will allow users to choose attributes for each device. |
| 16 | The system will allow users to copy selected data. |
| 17 | The system will allow users to mark an attribute as a favorite. |
| 18 | The system will allow users to hide a device's attributes. |
| 19 | The system will allow users to expand the information window. |
| 20 | The system will allow users to minimize the information window. |
| 21 | The system will allow users to add graphs of an attribute to the compare window. |
| 22 | The system will allow users to remove graphs from the compare window. |
| 23 | The system will allow users to view the device in 360 degrees. |
| 24 | The system will allow users to close the selected device. |
| 25 | The system will allow access to the site through mobile devices. |
| 26 | The system will support dark mode functionality. |
| 27 | The system will allow users to log in. |
| 28 | The system will allow users to sign up. |

#### **5.1.2. Non Functional Requirements**

Table 5 presents the non-functional requirements of the system. These describe the quality attributes and constraints such as performance, maintainability, security, and scalability, which ensure the system is reliable, efficient, and user friendly under operational conditions.

Table 5: Non Functional Reqguirements

|  |  |  |
| --- | --- | --- |
| **No.** | **Requirement** | **type** |
| 1 | Dashboard updates should occur within 1 second of receiving new sensor data. | Performance |
| 2 | Code modularity should support updates with minimal downtime (<5 minutes per month). | Maintainability |
| 3 | The site should be mobile friendly (WCAG 2.2 compliant). | Adaptability |
| 4 | The system should always provide up-to-date data. | Maintainability |
| 5 | The site should load within 3 seconds. | Efficiency, Performance |
| 6 | The site will save cookies. | Usability |
| 7 | The site will work properly even if none of the devices are available. | Fault tolerance |
| 8 | The system should support up to 5000 concurrent users without performance degradation. | Scalability |
| 9 | Only authorized users should be able to access data with MFA. | Security |
| 10 | The system should comply with data protection regulations (e.g., GDPR). | Compliance |
| 11 | New modules or features can be integrated with no more than 10% modification of existing code. | Extensibility |
| 12 | The system should provide interactive features with response time < 1 second for queries. | Interactivity |
| 13 | At least 80% of system components should be reusable in similar projects. | Modularity |

#### **5.2. Use Case**

The system’s core functionalities are defined through Use Case diagrams, which describe interactions between the system and its two primary actors: the User and the Agent.

The User Use Case diagram (Figure 4) was originally created by Or and Dima, the students who developed the first version of the system. It illustrates the set of manual actions available to users interacting with the web dashboard, including authentication, device interaction, attribute management, and data visualization.

As part of this year’s work, we extended the original diagram by adding several new high-level system monitoring and control features, marked in blue. These additions reflect the integration of machine learning capabilities into the user-facing interface and empower users to interact directly with the intelligent components of the system.

The Agent Use Case diagram (Figure 5), developed by our team, models the intelligent backend processes handled autonomously by the machine learning agent. These include sensor data analysis, occupancy and air quality prediction, emergency detection, HVAC control, and system logging.

Together, these diagrams illustrate how user-driven interactions and autonomous analytics work in tandem to form a robust, real time smart environment monitoring solution.

#### **5.2.1. User Functionalities (Figure 4)**

The User actor interacts primarily with the front end dashboard. Their responsibilities include:

* Authentication: Sign in and sign out to manage access and session control.
* Device Management: Sort and group devices, toggle their visibility on the map, and show or hide attributes.
* Attribute Interaction: Select individual sensor attributes, mark them as favorites, and pin them for comparison.
* Graphing and Analysis: Compare attributes across devices and generate visual graphs for selected time windows.
* System Monitoring: View real-time information such as current occupancy status, air quality index (AQI), and active alerts.
* Agent Control: Activate or deactivate the autonomous agent to enable or pause intelligent processing.
* Customization: Adjust the refresh interval to change how frequently the dashboard updates.

These user-centered functions support flexible and intuitive interaction with the system, especially in privacy-sensitive environments like exam halls.

(Originally created by Or and Dima, with additional blue use cases added in Robomo 2.0 for ML based monitoring and control)

Figure 4: User Use Case

A diagram of a company's work flow

AI-generated content may be incorrect.

#### **5.2.2. Agent Functionalities (Figure 5)**

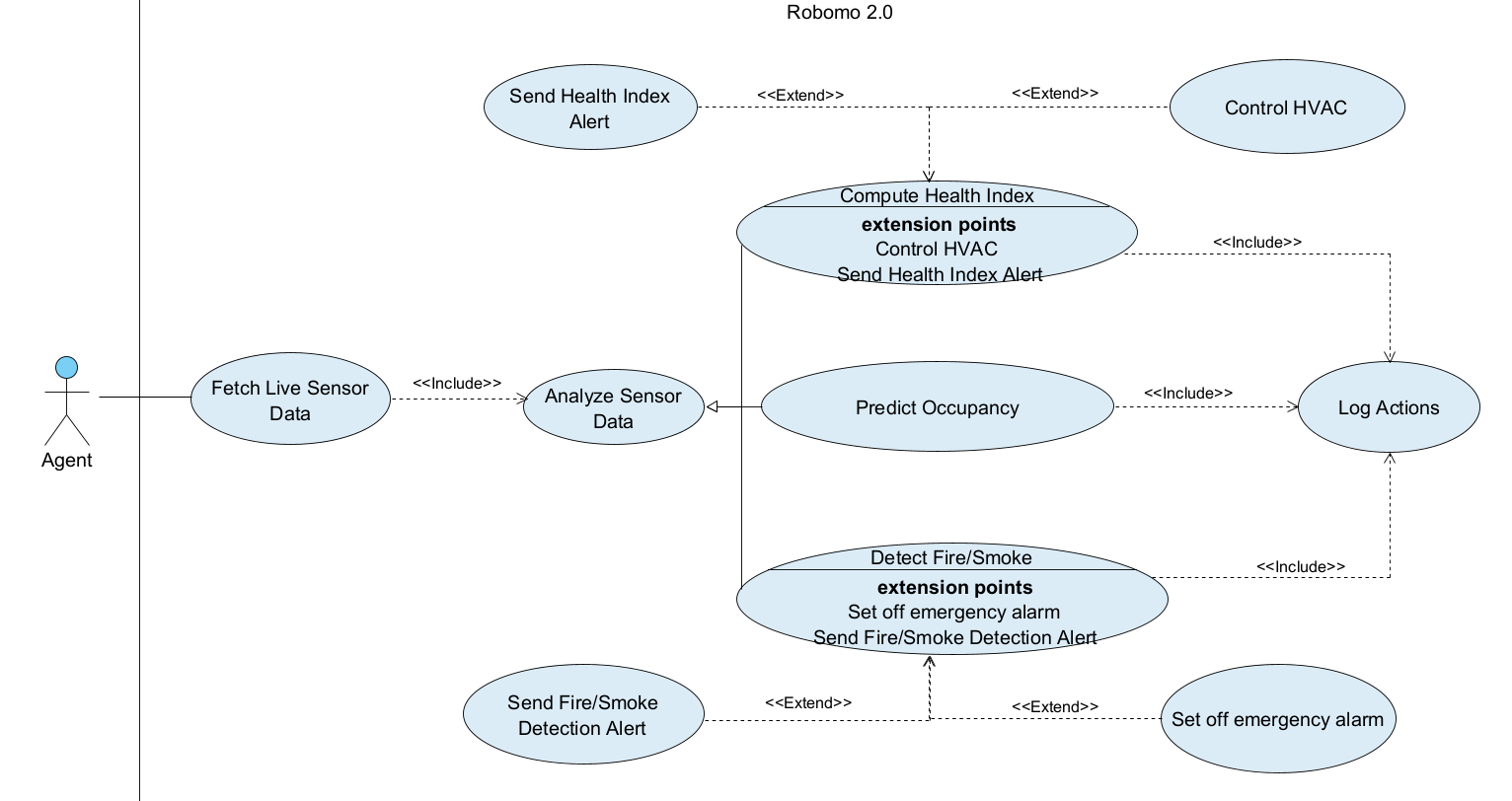
The Agent actor represents the intelligent component of the system. It is responsible for executing all real-time, automated analytics and decision-making processes, including:

* Data Collection: Fetch live environmental sensor data via WebSocket and MQTT from the Robomo robots and other sources.
* Data Analysis: Analyze incoming sensor data streams using trained machine learning models.
* Occupancy Prediction: Determine the current room occupancy status (e.g., empty, low, medium, high) based on air quality indicators.
* Health Index Computation: Calculate the IAQI (Indoor Air Quality Index) and extend actions to trigger HVAC control when necessary.
* Fire and Smoke Detection: Detect emergency conditions using thresholds on VOC and PM data and issue alerts and emergency alarms.
* System Logging: Log all key actions predictions, alerts, HVAC activations, and emergency triggers in Firebase for auditability and retraining purposes.
* External Response: Interface with third-party systems to push alerts, adjust ventilation, or notify staff.

This use case ensures the system operates continuously and reliably, enabling autonomous response to environmental hazards and efficiency improvements without compromising privacy.

(Developed by our team to model autonomous machine learning–driven backend behavior)

Figure 5: Agent Use Case



#### **5.3. Activity Diagram**

The following activity diagram (Figure 6) illustrates the complete flow of operations in the Robomo 2.0 system, showcasing the dynamic interactions between the User, the System, and the Autonomous Agent. This model reflects the finalized architecture and logic implemented in our project.

The diagram is divided into three vertical swimlanes:

User:

The user initiates actions through the web interface, including:

* **Login/Signup**: Authentication is verified through email confirmation and user info checks.
* **Device Interaction**: Users can show/hide devices, click on devices, and view detailed attributes, alerts, occupancy status, or air quality index.
* **Agent Control**: Users can activate or deactivate the machine learning agent manually as needed.
* **Data Visualization**: Users can generate graphs, extend/close windows, and export data for deeper insight.
* **Session Control**: Users can log out or close individual devices during interaction.

System:

The system acts as a middleware layer, facilitating:

* **Data Handling**: Refreshes and uploads sensor data from Robomo units, validating success and triggering preprocessing.
* **Logging and Error Handling**: Logs agent decisions and system errors for transparency and debugging.
* **Real-time Processing**: Fetches live data, validates its integrity, and communicates it to the ML agent.

Agent:

The machine learning agent operates autonomously to:

* **Initialize ML Models**: Prepares prediction models upon activation.
* **Analyze Sensor Data**: Continuously processes CO₂, temperature, humidity, VOC, and PM data.
* **Perform Inference**:
  + **Estimate Occupancy**: estimates the level of occupancy in a room.
  + **Compute Health Index**: Aggregates air quality indicators into a single interpretable score.
  + **Detect Fire/Smoke**: Identifies sudden hazardous changes such as PM or VOC spikes.
* **Decision Making**:
  + If an intervention is required (e.g., low health index or fire detected), the system determines:
    - Whether control actions can be taken automatically.
    - Whether emergency alerts or HVAC adjustments should be triggered.
    - Whether the user should be notified of conditions like poor air quality.

Figure 6: Activity diagram



**5.4. Website Screens**The following screenshots (Figures 7-8) are taken from the original system developed by Or and Dima, the previous student team. These interfaces formed the baseline of our inherited project, showcasing features such as device mapping, sensor selection, and real-time data visualization.

While these served as a valuable starting point, our current work proposes several enhancements and new features, particularly in areas related to machine learning-based predictions, health index display, and improved user interactivity.

Figure 7: Login screen

A screenshot of a computer

AI-generated content may be incorrect.

Figure 8: Home screen

A screenshot of a computer

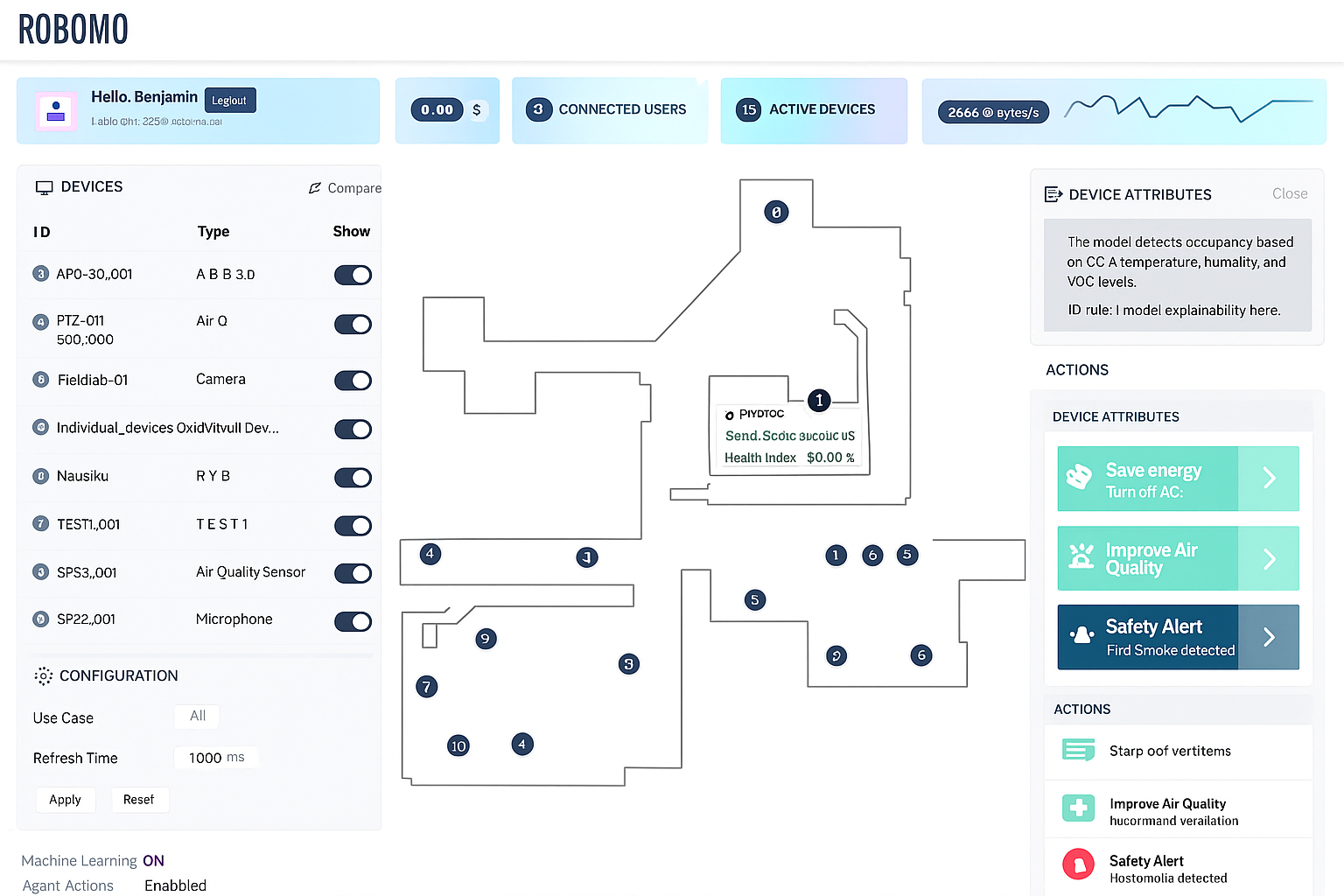
AI-generated content may be incorrect.

Figure 9 shows the updated system interface we developed during this project. It represents the evolution of the platform from a passive dashboard to a proactive and intelligent monitoring tool. The key enhancements introduced include:

* Machine Learning Status Panel: Displays the current state of the ML agent and whether it’s actively analyzing sensor data.
* Agent-Based Alerts and Actions:
  + Health Index Display for each room based on CO₂, VOCs, PM, temperature, and humidity
  + Safety Alert for detected smoke or fire risk
  + Energy-saving and air quality improvement suggestions
* Improved Device Attributes Panel with explainable model outputs
* Refined User Interface for better usability, responsiveness, and modern design aesthetics

These enhancements reflect our focus on real-time, privacy-respecting intelligence for smart indoor environments such as exam rooms, supporting automated responses while maintaining user oversight.

Figure 9: New Home screen



### **6. Expected Achievements**

* Develop a Smart Robot Prototype: Create a functional robot integrated with a variety of sensors for comprehensive environmental monitoring.
* Implement Machine Learning Models: Utilize machine learning to analyze sensor data for occupancy detection, health index calculation, and smoke/fire detection.
* Ensure Mobile Monitoring Capability: Enable the robot to autonomously navigate and collect data from different locations, providing more extensive coverage than stationary sensors.
* Design for Modularity and Scalability: Create a system that allows for the easy addition of new sensors and updated machine learning algorithms in the future.
* User-Friendly Interface: The system will feature an intuitive dashboard for users to easily monitor live sensor data and the calculated Health Index.

#### **6.1. Challenges**

* **Dataset Limitations:**
  + The available datasets did not perfectly match the specific lab environment.
  + There was no pre-labeled data for head-counting using the project's specific set of sensors.
  + Finding datasets that met all the specific requirements for the project proved to be difficult.
* **TAMK Misalignment:**
  + Coordination with the TAMK team was challenging due to their break coinciding with the project's semester.
  + There was a lack of precise details regarding the TAMK lab's dimensions and the locations of HVAC units.
* **Robot Availability:** 
  + The robot was not always operational when needed for data collection.

#### **6.2. Success Criteria**

* **Optimized Air Quality:** Maintain CO₂ levels below 1000 ppm and achieve a "good" rating for VOCs (under 500 μg/m³) for 90% of the time.
* **Enhanced Health Index:** Ensure the health index score remains above 80 for at least 90% of the day.
* **Accurate Occupancy and Automation:** Achieve over 99% accuracy in occupancy detection, leading to an estimated 10-30% in energy savings through HVAC automation.
* **Increased Productivity and Coverage:**
* Lead to a 5-10% increase in productivity and a 20-40% reduction in complaints regarding air quality.
* The robot's mobile monitoring should identify 20-40% more problem areas than fixed sensors.
* **System Performance:**
* The dashboard will update with sensor data within one second.
* The system will maintain at least 95% uptime during operational hours.

#### **6.3. Evaluation**

The project's success will be evaluated through a two-pronged approach: **technical performance validation** and **user-centric assessment**.

**Technical Performance Validation**

This phase will measure the system against the quantitative success criteria defined in the presentation.

* **Model Accuracy:** The performance of the machine learning models for occupancy, air quality classification , and smoke detection will be re-validated using real-world data collected by the robot in the TAMK lab environment.
* **Data Latency:** We will measure the time from a sensor detecting a change to the corresponding update appearing on the dashboard, ensuring it meets the sub-1-second goal.
* **System Uptime:** An automated monitoring service will be used to track the availability of the web dashboard and backend services to ensure the target of 95% uptime is met.

**User-Centric Assessment**

This phase focuses on the usability and effectiveness of the solution for its intended users (e.g., lab managers, facility operators).

* **Task-Based Usability Testing:** A small group of representative users will be given specific scenarios to complete using the dashboard. Example tasks include:
  + "Identify the current Health Index in the main lab and determine if any pollutants are at high levels."
  + "A smoke alert has been triggered. Locate the source of the alert on the map."
  + "Check if the system thinks the lab was occupied yesterday afternoon."
* **Qualitative Feedback:** Following the usability tests, users will be interviewed or asked to complete a short questionnaire to gather qualitative feedback on the interface's design, clarity, and overall usefulness in managing the indoor environment.

### **7. Testing Plan**

#### **7.1. Scope**

The testing scope covers the entire system architecture:

* **Frontend**: The React-based user dashboard, including all UI components, data visualizations, and WebSocket communications.
* **Backend**: The Node.js server, including API endpoints, data processing logic, and the MQTT message broker integration.
* **ML** **Agent**: The Python agent responsible for fetching live data, running predictions for occupancy, health index, and fire/smoke, and logging actions.
* **End**-**to**-**End**: The complete data flow from a sensor reading to a prediction and visualization on the dashboard.

#### **7.2. Objectives**

* To verify that all functional requirements, such as occupancy detection, alert generation, and data logging, operate as designed.
* To ensure the system meets its non-functional requirements for performance, scalability, and usability.
* To identify, document, and resolve bugs across the full technology stack.
* To validate the system's stability during continuous, long-term operation.

#### **7.3. Testing Approach**

* **Unit Testing**: Individual functions and components will be tested in isolation. We will use Jest for the React frontend components and PyTest for the Python ML agent's functions (e.g., the AQI calculation logic).
* **Integration Testing**: We will test the interfaces between system components, such as verifying that the backend correctly processes data received from the MQTT broker and forwards it to the ML agent.
* **End-to-End (E2E) Testing**: We will simulate a complete user scenario, from having the robot collect sensor data to verifying that the correct alerts and dashboard updates occur in the browser.
* **Manual Testing**: We will manually execute the test cases for the user interface to ensure the workflows are intuitive and the user experience is seamless.

To ensure the reliability and correctness of both the user facing dashboard and the backend intelligent agent, we defined a comprehensive set of test cases that align with the system's core functionalities. These test cases cover all major interactions, including sensor data acquisition, machine learning inference, alerting mechanisms, logging, and user interface operations. Each test is linked to a specific scenario with clear preconditions and expected outcomes. This structured approach helps validate not only technical correctness but also the user experience and the responsiveness of the system under real world conditions.

The test cases are summarized in Table 6 below:

Table 6: Test Cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Test ID | Precondition | Expected Result | Description |
| 1 | **FetchSensorData** | The agent is active and connected to the network. | Live sensor data is successfully received and displayed on the dashboard. | The agent initiates a request to fetch live data from the integrated sensors. |
| 2 | **PredictOccupancy** | The system has successfully received sensor data (CO₂, temp, humidity, etc.). | The system accurately predicts whether the room is occupied or empty. | The system analyzes the sensor data using the trained machine learning model to predict occupancy. |
| 3 | **ComputeHealthIndex** | The system has successfully received and analyzed sensor data. | A health index score is calculated and displayed on the dashboard. | The system aggregates various environmental parameters into a single health index score. |
| 4 | **DetectFire/Smoke** | The system has successfully received sensor data, including Particulate Matter (PM) readings. | The system correctly identifies the presence of smoke or fire. | The system analyzes sensor data to detect anomalies indicative of fire or smoke. |
| 5 | **SendHealthIndexAlert** | The computed Health Index falls below a predefined threshold. | An alert is sent to the user, and the system may trigger HVAC control. | The system sends a notification about poor air quality and may automatically adjust the HVAC system. |
| 6 | **SendFire/SmokeAlert** | The system detects a fire or smoke event. | An emergency alarm is set off, and a high-priority alert is sent. | The system triggers an immediate emergency alarm and notifies the user of the fire/smoke detection. |
| 7 | **LogActions** | The agent has performed an action (e.g., data fetching, sending an alert). | The action and relevant data are logged for historical analysis. | The system logs all agent decisions and data for future review. |
| 8 | **UserInteraction-ViewData** | A user is logged into the dashboard. | The user can view the live air quality index, occupancy status, and alerts. | A user navigates the interface to view different data visualizations. |
| 9 | **UserInteraction-ControlDevice** | A user is logged in and viewing device attributes. | The user can click on a device to see its detailed attributes. | A user selects a specific device on the map or list to view its data. |
| 10 | **UserInteraction-ActivateAgent** | A user is logged in with appropriate permissions. | The agent is successfully activated or deactivated. | A user toggles the agent's operational status through the UI. |

### **8. AI Tools and Prompts**

During our work on part a, we used AI tools such as: ChatGPT, Claude and Gemini. We used these different AI tools in a variety of areas such as checking syntax and spelling, conducting literature introduction research as well as in the practical part to check on various technologies and methods.

#### **8.1. AI Usage and Results**

* **Use**: We used the AI tools to look for datasets for our 3 use cases (Occupancy Detection, Fire/Smoke Detection and Air Quality Index).

**Result**: They provided us with many datasets from different websites like Kaggle, Mendeley Data, UCI Machine Learning Repository, Hugging Face.

* **Use**: We used AI tools to help us build the machine learning models testing code on google colab, and tested the datasets we choose there on different ML models and evaluated each model to check which is the best for our use cases.

**Result**: We solved bugs easily, tested the models on the datasets using only our sensors and got our results back in a table format with each evaluation value metric, thus we were able to find out which model was the best for us.

* **Use**: We used AI tools to learn which methods are used today to control HVAC systems and to learn more about the sensors we have and their capabilities.

**Result**: The AI tools explained two ways (MQTT, FIWARE), they taught us about each sensor we have and what is it useful for.

#### **8.2. AI Prompts and Responses**

**ChatGPT**

We used ChatGPT extensively for multiple stages of the project, including research, system design, and code generation.

**Prompts Used:**

* "Find datasets with CO₂, temperature, humidity, VOC, or PM values for indoor occupancy detection."
* "How can HVAC systems be controlled in a smart building using IoT technologies?"
* "Write a Python script that loads a CSV file, runs multiple machine learning models on selected features, evaluates their performance, and displays the results in a table. The code should run in Google Colab."

**Results:**

* ChatGPT provided a list of relevant datasets from **Kaggle**, **Mendeley Data**, and the **UCI Machine Learning Repository**. After reviewing them, we selected three datasets that best matched our use cases: **Occupancy Detection**, **Fire/Smoke Detection**, and **Air Quality Index Prediction**.
* In response to the HVAC control prompt, ChatGPT introduced us to **MQTT** and **FIWARE**:
  + **MQTT (Message Queuing Telemetry Transport):** A lightweight protocol ideal for sending sensor data from devices to servers with low overhead and high efficiency.
  + **FIWARE:** A platform offering standardized APIs to manage context data in smart systems, making it easier to integrate real-time sensor input with applications like HVAC control.
* The Python script generated by ChatGPT in Colab helped us test various machine learning models on our datasets using only the features our sensors provided. The script evaluated each model using metrics like accuracy, MAE, RMSE, and R², and summarized the results in a comparison table. This allowed us to select the best model (Random Forest) for each use case.

**Gemini**

Gemini was mainly used to supplement our dataset search.

**Prompts Used:**

* "Suggest datasets with environmental sensor data (CO₂, VOC, humidity) suitable for indoor occupancy estimation."

**Results:**

* Gemini returned a broad list of dataset sources across Kaggle, Mendeley Data, and Hugging Face. Some datasets were not relevant, but it helped us filter out and choose those that aligned best with our specific sensor types.

**Claude**

Claude was used specifically for debugging and optimizing the Python code generated in ChatGPT.

**Prompts Used:**

* "Fix this error in my Colab notebook when running Random Forest on my CSV data."
* "Why is my AQI calculation giving NaN values in some rows?"
* "How can I optimize this code block to loop through models and print results more cleanly?"

**Results:**

* Claude helped us quickly resolve syntax and logic issues in the Colab notebook, including handling NaN values in the dataset, fixing model training errors, and improving how results were formatted and printed.
* It also suggested efficient practices for code modularity, such as wrapping evaluation steps into functions and using loops to handle multiple models, improving maintainability for future testing.

### **9. References**

#### **9.1 Academic sources (journals, conferences, datasets, standards)**

1. Ahmed, K. P. (2022). IoT Indoor Air Quality Dataset [Data set]. Kaggle. <https://www.kaggle.com/datasets/khajaahmed1/iot-indoor-air-quality>
2. Candanedo, L. M., & Feldheim, V. (2016). Accurate occupancy detection of an office room from light, temperature, humidity and CO₂ measurements using statistical learning models. Energy and Buildings, 112, 28–39. <https://doi.org/10.1016/j.enbuild.2015.11.071> [ResearchGate](https://www.researchgate.net/publication/285627413_Accurate_occupancy_detection_of_an_office_room_from_light_temperature_humidity_and_CO2_measurements_using_statistical_learning_models?utm_source=chatgpt.com)
3. Contractor, D. (2021). Smoke Detection Dataset [Data set]. Kaggle. <https://www.kaggle.com/datasets/deepcontractor/smoke-detection-dataset>
4. Demertzi, V., Demertzis, S., & Demertzis, K. (2023). An overview of privacy dimensions on the Industrial Internet of Things (IIoT). Algorithms, 16(8), 378. <https://doi.org/10.3390/a16080378>
5. Fuller, A., Fan, Z., Day, C., & Barlow, C. (2020). Digital twin: Enabling technologies, challenges and open research. IEEE Access, 8, 108952–108971. <https://doi.org/10.1109/ACCESS.2020.2998358>
6. Grieco, L. A., Rizzo, A., Colucci, S., Sicari, S., Piro, G., Di Paola, D., & Boggia, G. (2014). IoT-aided robotics applications: Technological implications, target domains and open issues. Computer Communications, 54, 32–47. <https://doi.org/10.1016/j.comcom.2014.07.013>
7. IEEE Standards Association. (2012). IEEE Std 828-2012—IEEE Standard for Configuration Management in Systems and Software Engineering. <https://standards.ieee.org/standard/828-2012.html>
8. Internet Engineering Task Force (IETF). (2011). The WebSocket Protocol (RFC 6455). <https://www.rfc-editor.org/rfc/rfc6455>
9. Kamilaris, A., & Botteghi, N. (2020). The penetration of Internet of Things in robotics: Towards a web of robotic things. Journal of Ambient Intelligence and Smart Environments, 12(6), 491–512. <https://doi.org/10.3233/AIS-200582>
10. Khan, A., Aziz, S., Bashir, M., & Khan, M. U. (2020, March). IoT and wireless sensor network based autonomous farming robot. In 2020 International Conference on Emerging Trends in Smart Technologies (ICETST) (pp. 1–5). IEEE. <https://doi.org/10.1109/ICETST49965.2020.9080736>
11. Krishnamurthi, R., Kumar, A., Gopinathan, D., Nayyar, A., & Qureshi, B. (2020). An overview of IoT sensor data processing, fusion, and analysis techniques. Sensors, 20(21), 6076. <https://doi.org/10.3390/s20216076>
12. Li, P., & Liu, X. (2019, July). Common sensors in industrial robots: A review. Journal of Physics: Conference Series, 1267, 012036. <https://doi.org/10.1088/1742-6596/1267/1/012036>
13. Nayyar, A., Puri, V., Nguyen, N. G., & Le, D. N. (2018). Smart surveillance robot for real-time monitoring and control system in environment and industrial applications. In Information Systems Design and Intelligent Applications: Proceedings of Fourth International Conference INDIA 2017 (pp. 229–243). Springer. <https://doi.org/10.1007/978-981-10-7512-4_23>
14. OASIS. (2014). MQTT Version 3.1.1—OASIS Standard. <https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html>
15. OASIS. (2019). MQTT Version 5.0—OASIS Standard. <https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html>
16. Ray, P. P. (2016). Internet of robotic things: Concept, technologies, and challenges. IEEE Access, 4, 9489–9500. <https://doi.org/10.1109/ACCESS.2017.2647747>

1. Singh, M., Fuenmayor, E., Hinchy, E. P., Qiao, Y., Murray, N., & Devine, D. (2021). Digital twin: Origin to future. Applied System Innovation, 4(2), 36. <https://doi.org/10.3390/asi4020036>
2. Vela, A., Alvarado-Uribe, J., & Ceballos, H. G. (2021). Fitness-Gym and Living-room Occupancy Estimation Data (Version 3) [Data set]. <https://data.mendeley.com/datasets/kjgrct2yn3/3>
3. Vermesan, O., Bröring, A., Tragos, E., Serrano, M., Bacciu, D., Chessa, S., Gallicchio, C., Micheli, A., Dragone, M., Saffiotti, A., Simoens, P., Cavallo, F., & Bahr, R. (2022). Internet of robotic things—Converging sensing/actuating, hyperconnectivity, artificial intelligence and IoT platforms. In O. Vermesan & J. Bacquet (Eds.), Cognitive Hyperconnected Digital Transformation (pp. 97–155). Taylor & Francis / River Publishers. <https://doi.org/10.1201/9781003337584-4>
4. W3C (World Wide Web Consortium). (2024). Web Content Accessibility Guidelines (WCAG) 2.2 (W3C Recommendation, 12 December 2024). <https://www.w3.org/TR/WCAG22/>
5. Zhang, W., Wu, Y., & Calautit, J. K. (2022). A review on occupancy prediction through machine learning for enhancing energy efficiency, air quality and thermal comfort in the built environment. Renewable & Sustainable Energy Reviews, 167, 112704. <https://doi.org/10.1016/j.rser.2022.112704>

#### **9.2 Web resources / developer documentation**

1. FIWARE Foundation. (n.d.). About FIWARE. <https://www.fiware.org/about-us/>
2. Google. (n.d.). Colaboratory (Google Colab). <https://colab.research.google.com/>
3. Google. (n.d.). Firebase Realtime Database documentation. <https://firebase.google.com/docs/database>
4. HubSpot. (n.d.). Tailwind CSS: What it is, why use it & examples. <https://blog.hubspot.com/website/what-is-tailwind-css>
5. Meta Open Source. (2023). React documentation. <https://react.dev/>
6. Mozilla. (n.d.). JavaScript Guide. <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide>
7. Node.js Foundation (OpenJS Foundation). (n.d.). Node.js API documentation. <https://nodejs.org/docs/latest/api/>
8. Tailwind Labs. (n.d.). Tailwind CSS documentation. <https://tailwindcss.com/docs>
9. TM Forum. (2016). FIWARE, the standard that the IoT needs. <https://www.tmforum.org/press-and-news/fiware-standard-iot-needs/>